

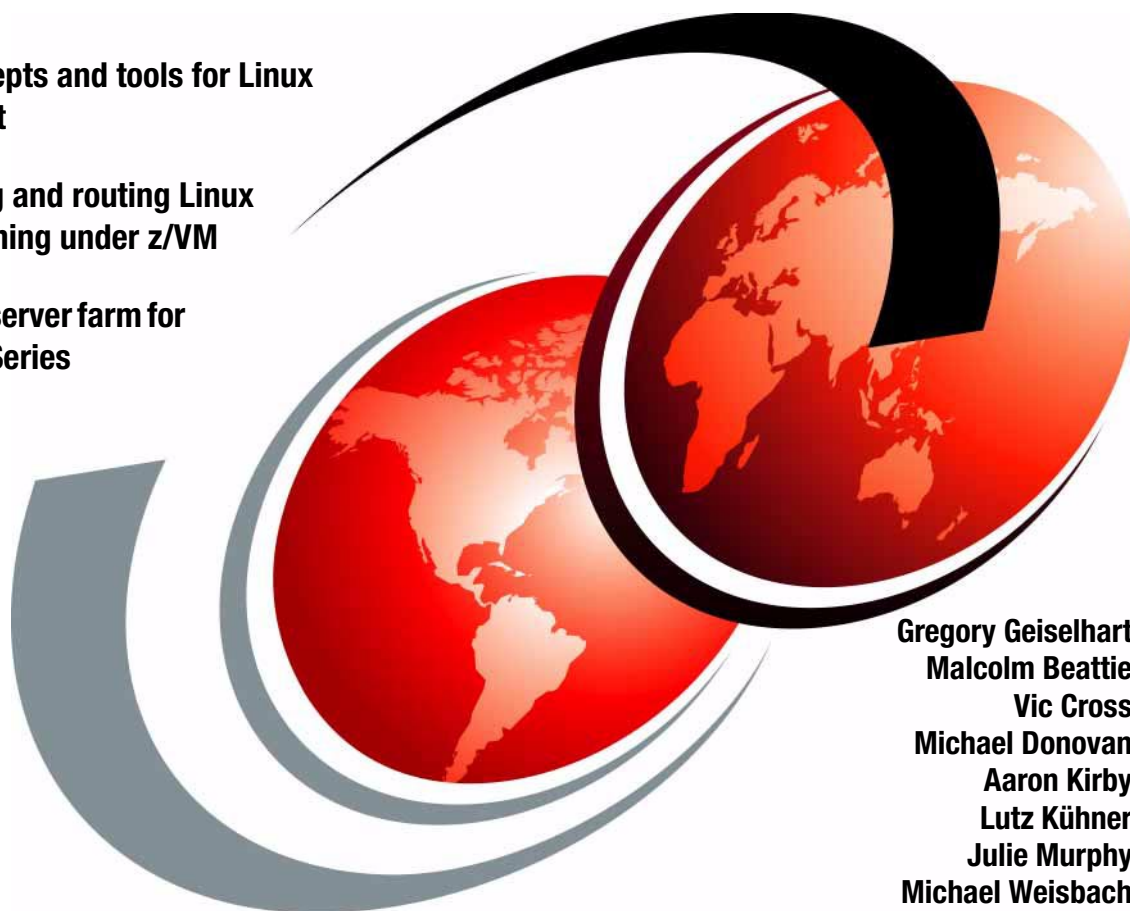


Linux on IBM **@**server[™] zSeries and S/390: Large Scale Linux Deployment

z/VM concepts and tools for Linux deployment

Networking and routing Linux guests running under z/VM

Building a server farm for Linux on zSeries



Gregory Geiselhart
Malcolm Beattie
Vic Cross
Michael Donovan
Aaron Kirby
Lutz Kühner
Julie Murphy
Michael Weisbach

ibm.com/redbooks

Redbooks



International Technical Support Organization

Large Scale Linux Deployment

October 2002

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

First Edition (October 2002)

This edition applies to z/VM 4.3 and many different Linux distributions. RedHat 7.1 for zSeries was used.

© Copyright International Business Machines Corporation 2002. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	xi
Trademarks	xii
Preface	xiii
The team that wrote this redbook	xiv
Become a published author	xv
Comments welcome	xv
Part 1. Running Linux under z/VM	1
Chapter 1. z/VM for beginners	3
1.1 The z/VM environment	4
1.2 Logging on to z/VM	4
1.3 General CP command structure	6
1.3.1 Command truncations and abbreviations	7
1.4 CP command privilege classes	7
1.4.1 How privilege class affects CP commands	9
1.5 The CP status indicator	11
1.6 Using Program Function keys	11
1.7 Disconnecting the 3270 session	13
1.8 Booting Linux in a virtual machine	13
1.8.1 Unattended startup of a Linux guest	14
1.8.2 Recovering from unattended startup	14
1.9 Communicating with CP from a Linux guest	15
1.9.1 Communicating with CP from the VM console	15
1.9.2 Communicating with CP from a Linux telnet session	15
1.10 Querying the virtual machine	16
1.10.1 Querying storage devices	16
1.10.2 Querying network devices	17
1.10.3 Querying the CPUs available to the virtual machine	18
1.10.4 Querying virtual storage	18
1.11 Using DDR to copy a minidisk	19
1.12 Virtual Machine Resource Manager	20
Chapter 2. Directory Maintenance Facility for z/VM	23
2.1 Managing VM using DirMaint	24
2.2 DirMaint service machines	24
2.2.1 DirMaint service machine	25
2.2.2 DATAMOVE service machine	25

2.3 DirMaint command syntax	25
2.3.1 Using prefix keywords	26
2.4 Some useful DirMaint commands	26
2.5 Defining a userid as a DirMaint administrator	27
2.5.1 Obtain the DirMaint AUTHFOR CONTROL file	27
2.5.2 Format of the AUTHFOR CONTROL file	27
2.5.3 Activating AUTHFOR CONTROL file changes	28
2.6 Adding a volume to a DirMaint group	28
2.6.1 Obtain the DirMaint EXTENT CONTROL file	28
2.6.2 Format of the EXTENT CONTROL file	28
2.6.3 Activating EXTENT CONTROL file changes	30
2.7 Adding directory entries	30
2.7.1 Defining a profile directory entry	30
2.7.2 Adding a profile directory entry	31
2.7.3 Defining a user directory entry	31
2.7.4 Adding a userid using a prototype file	31
2.8 Maintaining directory entries	32
2.8.1 Reviewing a directory entry	32
2.8.2 Adding a minidisk to a user directory entry	33
2.8.3 Adding access passwords to a minidisk	33
2.8.4 Dedicating a device to a userid	33
2.8.5 Deleting a new minidisk from a user directory entry	33
2.8.6 Changing virtual storage for VM users	34
2.8.7 Adding, deleting, and modifying CP options	34
2.8.8 Changing CP Privileges	34
2.8.9 Using the SPECIAL DirMaint command	34
2.8.10 Transferring a minidisk between userids	34
2.8.11 Adding shared logon access to a userid	35
Chapter 3. FCON/ESA for monitoring a penguin colony	37
3.1 Introducing FCON/ESA	38
3.2 FCON/ESA support for Linux on z/VM	38
3.3 The Distributed Data Server	39
3.3.1 Download DDS	39
3.3.2 Install DDS on a Linux guest	40
3.3.3 Starting DDS	40
3.3.4 Viewing monitored data	40
3.4 Customizing FCON/ESA for monitoring Linux guests	41
3.4.1 Preparing the control file	41
3.4.2 Updating the FCON/ESA profile	42
3.5 The FCON/ESA Linux systems option	42
3.6 FCON/ESA subcommands for Linux guests	43
3.6.1 The LINUX subcommand	44

3.6.2	The Linux systems selection menu	44
3.6.3	The Linux details selection menu	44
3.6.4	The LXCPU subcommand	46
3.6.5	The LXMEM subcommand	47
3.6.6	The LXNETWRK subcommand	48
3.6.7	The LXFILESYS subcommand	48
3.7	Monitoring overall z/VM performance	49
3.7.1	The CPU subcommand	49
3.7.2	The STORAGE subcommand	50
3.7.3	The DEVICE subcommand	51
3.7.4	The USER subcommand	52
Part 2.	Networking for Linux on zSeries	55
Chapter 4.	HiperSockets and z/VM Guest LAN	57
4.1	Introduction to HiperSockets	58
4.1.1	Operating system support	58
4.1.2	Capabilities	58
4.2	Configuring HiperSockets	58
4.2.1	Hardware tasks	59
4.2.2	z/VM tasks	59
4.2.3	Linux tasks	60
4.3	Introduction to the Guest LAN feature	62
4.3.1	Virtual HiperSockets	62
4.3.2	Virtual QDIO	63
4.4	VM Guest LAN configuration	63
4.5	Creating a VM Guest LAN segment	63
4.5.1	Establishing a VM Guest LAN owner	64
4.5.2	Establishing a VM Guest LAN lifetime	64
4.6	Creating a simulated NIC	65
4.7	Attaching the simulated NIC to the VM Guest LAN	66
4.8	A VM Guest LAN example	66
4.9	Restricted VM Guest LANs	67
4.9.1	Viewing VM Guest LAN attributes	68
4.9.2	Changing VM Guest LAN attributes	68
4.10	Defining a VM Guest LAN in the VM directory	70
4.10.1	Define the VM Guest LAN in the SYSTEM CONFIG file	70
4.10.2	Define and couple simulated NICs to the VM Guest LAN	70
4.10.3	Automating connections to a VM Guest LAN	71
4.11	Configuring a VM Guest LAN in a Linux guest	73
4.11.1	A word about network device drivers	73
4.11.2	Loading the Linux network interface device driver	73
4.11.3	Configuring the network interface	76

Chapter 5. TCP/IP direct connection	77
5.1 Introduction	78
5.1.1 Number of Linux guests	78
5.2 OSA port sharing	78
5.2.1 Hardware definition	80
5.2.2 Advantages sharing OSA-Express in QDIO mode	81
5.2.3 Issues sharing OSA-Express in QDIO mode	82
5.3 IEEE 802.1Q VLAN support	83
5.3.1 How VLANs work	83
5.3.2 VLANs on Linux for zSeries	85
5.3.3 Sharing an OSA-Express when using VLANs	87
5.3.4 Configuring VLANs in Linux	89
5.3.5 Infrastructure guests in a VLAN network	90
Chapter 6. TCP/IP routing	93
6.1 Planning for routing	94
6.1.1 Connectivity method	94
6.1.2 Isolation	95
6.1.3 Address allocation	95
6.1.4 Traffic shaping	96
6.1.5 Linux router or z/VM TCP/IP router	96
6.1.6 Routing considerations with OSAs	97
6.2 Linux routers	98
6.2.1 Device support	99
6.2.2 Routing function	99
6.2.3 Setting up a Linux router	99
6.2.4 Changing a running Linux router guest	100
6.3 z/VM TCP/IP routers	100
6.3.1 Device support	100
6.3.2 Routing function	100
6.3.3 Changing a running z/VM TCP/IP stack	102
6.3.4 z/VM TCP/IP support servers	104
6.4 z/OS routers	105
6.4.1 HiperSockets Accelerator	105
6.5 Traffic control	110
6.5.1 Components of traffic control	111
6.5.2 Configuring CBQ	112
6.5.3 CBQ usage example: bandwidth choke	114
6.5.4 CBQ usage example: differentiating interactive traffic	115
6.6 Dynamic routing	116
6.6.1 How dynamic routing works	116
6.6.2 Dynamic routing in a penguin colony	117
6.6.3 Controlling routing tables	119

Chapter 7. Network high availability	121
7.1 Planning virtual connectivity for high availability	122
7.1.1 Determine the level of redundancy you need	122
7.1.2 z/VM TCP/IP availability	122
7.2 Multiple network devices to Linux guests	123
7.2.1 Configuring multiple network interfaces	123
7.2.2 Virtual Router Redundancy Protocol (VRRP)	134
7.2.3 Virtual IP addresses	136
7.2.4 IP connections outbound from Linux guests	137
7.3 Redundancy outside the zSeries complex	143
7.3.1 Additional z/VM system	143
7.4 Linux high availability solutions	143
7.4.1 To cluster or not to cluster	144
7.4.2 Linux Virtual Server	145
Part 3. Creating and managing a penguin colony	147
Chapter 8. Shared Linux filesystems	149
8.1 Device filesystem mounts	150
8.2 Bind mount directories	150
8.3 Using bind mounts	152
8.3.1 Mounting writable directories on a read-only filesystem	153
8.3.2 Preserving access to the original read-only directories	154
8.4 The basevol filesystem	155
8.5 The guestvol filesystem	155
8.6 A basevol/guestvol Linux guest	156
8.7 The File Hierarchy Standard	156
8.8 RPM package management	156
8.9 Booting a basevol/guestvol Linux guest	159
8.9.1 The rc.guestvol script	160
8.9.2 Determining if the Linux guest uses a guestvol mount	160
8.9.3 The maintenance shell	161
8.9.4 Example basevol/guestvol Linux guest startup	161
8.9.5 Example basevol/guestvol Linux guest maintenance shell	162
8.10 Startup configuration	162
8.10.1 The rc.sysinit-guestvol script	163
8.11 Network configuration	164
8.11.1 The z/VM configuration server	164
8.11.2 Generating a CONFSEV response	165
8.11.3 Security considerations	166
8.11.4 The vmgetconf script	166
8.11.5 The itsonet script	168
8.11.6 Example of boot time configuration	169

8.12 Shutdown processing	169
8.12.1 The guestvol-start-halt script	170
8.12.2 The guestvol-final-halt script	170
8.12.3 Example of a basevol/guestvol Linux guest shutdown	170
8.13 Advantages of a basevol/guestvol Linux guest	172
Chapter 9. Building a basevol/guestvol penguin colony	173
9.1 Overview of the process	174
9.2 The BASEVOL virtual machine	174
9.3 The LDV01 virtual machine	175
9.4 Install Linux on the development image	176
9.4.1 Choosing the packages to install	176
9.5 Create the basevol and guestvol filesystem images	177
9.5.1 Prepare the LDV01 Linux guest	177
9.5.2 Create the golden basevol filesystem image	178
9.5.3 Prepare guestvol filesystem image	178
9.5.4 Booting the basevol/guestvol Linux guest	179
9.6 Guestvol package management	180
9.7 Cloning a basevol/guestvol Linux guest	181
9.7.1 The LNXCLONE prototype	181
9.7.2 Create the Linux clone virtual machine	183
9.7.3 Create the Linux clone guestvol	183
9.7.4 Define the Linux clone in the GUEST CONF configuration file	183
9.7.5 XAUTOLOG the Linux clone	183
9.8 Remote startup and shutdown of Linux clones	183
9.8.1 The ext_int kernel module	184
9.8.2 Handling a shutdown external interrupt	185
9.8.3 The management interface	186
9.8.4 PROP actions to manage Linux clones	187
9.8.5 The GUESTACT EXEC script	188
9.8.6 Security considerations	188
Chapter 10. Centralized management using LDAP	191
10.1 Using LDAP for centralized management	192
10.1.1 The OpenLDAP directory server	192
10.1.2 The penguin colony network topology	192
10.2 Configuring the LDAP server	194
10.3 LDAP tools	195
10.3.1 An LDAP browser	196
10.3.2 LDAP Data Interchange Format	196
10.3.3 LDAP migration tools	196
10.4 Network configuration and initialization	197
10.4.1 The redbook LDAP schema	197

10.4.2	Redbook LDAP object classes	198
10.4.3	Redbook LDAP attributes	198
10.4.4	The itsldap script	199
10.4.5	Configuring LDAP clients	200
10.5	UNIX authentication using LDAP	201
10.5.1	The nss-ldap and pam-ldap modules	201
10.5.2	Configuring PAM for LDAP authentication and authorization	202
10.5.3	Configuring NSS for LDAP user and group mapping	203
10.5.4	Migrating users and groups to LDAP	204
10.5.5	Adding users and groups to LDAP	204
10.5.6	Changing passwords stored in LDAP	205
10.6	Using LDAP with Domain Name System	207
10.6.1	The SDB LDAP backend for the ISC bind server	207
10.6.2	DNS resource records in LDAP	209
10.6.3	Configure the DNS server to use the LDAP backend	211
10.6.4	Adding indexes to speed lookups	211
10.7	A remote Web management interface to LDAP	212
10.7.1	Interface to reset passwords	212
10.7.2	Interface to IPL and shutdown Linux guests	213
Appendix A. The Unit Record device driver and utility		215
A.1	The UR device driver and utility	216
A.2	The UR device driver	216
A.2.1	Build the UR device driver	216
A.2.2	Install the UR device driver	216
A.2.3	Create the UR character devices	217
A.2.4	The addvmur command	217
A.3	The UR utility	218
A.3.1	Install the UR utility	218
A.3.2	The ur command syntax	218
A.3.3	The copy subcommand	218
A.3.4	The info subcommand	219
A.3.5	The list subcommand	220
A.3.6	The add subcommand	220
A.3.7	The remove subcommand	220
Appendix B. Installing Red Hat 7.1 with OCO modules		221
B.1	The Red Hat for zSeries distribution	222
B.2	Obtain the latest OCO drivers	222
B.3	Create a second initial ramdisk for OCO qeth drivers	223
B.4	Copy the installation images to the guest reader	223
B.4.1	Mount the installation CD-ROM image	224
B.4.2	Copy the installation images to the VM reader	224

B.5	Install the Linux guest	224
B.5.1	Beginning the installation	225
B.5.2	First stage configuration	225
B.5.3	Second stage configuration	226
Appendix C. Scripts and configuration files.		231
C.1	The basevol+guestvol-1.0.0-1.noarch.rpm package.	232
C.1.1	The /etc/rc.d/rc.guestvol script	232
C.1.2	The /etc/rc.d/rc.sysinit-guestvol script	234
C.1.3	The /etc/init.d/vmgetconf script.	250
C.1.4	The /etc/init.d/itsonet script.	252
C.1.5	The /etc/init.d/guestvol-start-halt script.	254
C.1.6	The /sbin/guestvol-final-halt script	257
C.1.7	The /usr/sbin/basevol-devel script	261
C.1.8	The /usr/sbin/mkguestvol script	261
C.2	The itsobasevol-1.0.0-1.s390x.rpm package	262
C.2.1	The /etc/init.d/itsoldap script.	263
C.2.2	The /usr/sbin/dasd script	268
C.3	The GVCOPY EXEC	268
C.4	The GUESTACT EXEC script.	269
C.5	The GETCONF EXEC script.	271
C.6	The PROP RTABLE configuration file	272
C.7	The redbook.schema file	273
C.8	The sample-ldap.php script	276
C.9	The ipl-shutdown.php script	277
Appendix D. Additional material		279
	Locating the Web material	279
	Using the Web material	279
	System requirements for downloading the Web material	280
	How to use the Web material	280
Abbreviations and acronyms		281
Related publications		283
	IBM Redbooks	283
	Other resources	284
	Referenced Web sites	285
	How to get IBM Redbooks	286
	IBM Redbooks collections.	286
Index		287

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:

DB2®
DirMaint™
@server™
IBM®
MORE™
MVS™
OS/2®

OS/390®
Parallel Sysplex®
Perform™
RACF®
Redbooks(logo)™ 
RMF™
S/390®

Sequent®
SP™
VM/ESA®
z/OS™
z/VM™
zSeries™

The following terms are trademarks of other companies:

ActionMedia, LANDesk, MMX, Pentium and ProShare 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.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

C-bus is a trademark of Corollary, Inc. in the United States, other countries, or both.

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

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

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

Preface

This IBM Redbook discusses tools and techniques you can utilize for large scale deployment on Linux on IBM @server zSeries and S/390. Its target audience is system administrators and I/T architects who are responsible for developing optimized solutions for large Linux systems installed on IBM zSeries and S/390 machines.

For large scale deployment of Linux on zSeries and S/390, we only consider running Linux guests under z/VM, for the following reasons:

- ▶ The hardware virtualization provided by z/VM allows many Linux to effectively share resources.
- ▶ z/VM features can be used to assist in centrally managing Linux guests.
- ▶ Linux clones can easily be added to the server farm using z/VM.

In this book, we refer to a large number of Linux server running under z/VM as a *penguin colony*.

This redbook is divided into three parts:

- ▶ In part one, we discuss basic VM concepts and commands. We examine tools available on z/VM to help in creating and monitoring a penguin colony.
- ▶ In part two, we discuss networking features available for Linux on zSeries and S/390. We describe how to use and configure HiperSockets and VM Guest LAN networking. Options available for OSA direct routing and for TCP/IP routing on zSeries and S/390 are covered in detail. In addition, we cover network high availability issues and solutions.
- ▶ In part three, we discuss techniques to manage and create a penguin colony. We cover a novel approach to sharing Linux filesystems among members of the colony, and consider an central LDAP management system for the colony.

The appendix includes sections on:

- ▶ Using a utility to transfer files between Linux guests and a VM virtual reader, punch, or printer: the UR device driver and utility
- ▶ An approach to installing Red Hat 7.1 with OCO module support
- ▶ A listing of important scripts covered in the redbook

Sample code discussed in the redbook is available online.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

Gregory Geiselhart is a project leader for Linux on zSeries at the International Technical Support Organization, Poughkeepsie Center.

Malcolm Beattie is a Linux Technical Consultant with IBM UK working in the EMEA IBM Enterprise Server Group (zSeries). He has 15 years of experience in computing (13 years of UNIX, 10 years of Linux, 3 years of VM). He holds the degrees of BA, MA and DPhil in mathematics from Oxford University. He has been active in the development of open source software, mostly of Perl and related software, but has also done minor work on Exim, PostgreSQL, the Linux kernel and other bits and pieces.

Vic Cross is the Linux for zSeries and S/390 Team Leader at Independent Systems Integrators, IBM's Large Systems Business Partner in Australia. He has more than 15 years of experience in general computing, seven of which has been spent working on S/390 and zSeries. He holds a Bachelor of Computing Science degree from Queensland University of Technology. His areas of expertise include networking and Linux. He is a co-author of the IBM Redbook *Linux on IBM @server zSeries and S/390: ISP/ASP Solutions*, SG24-6299.

Michael Donovan is a Senior Software Engineer with IBM in Endicott, NY. He joined VM Development in 1980 at the Glendale Programming Laboratory. He holds a bachelors degree in Computer Science from the State University of New York at Oswego. He has been involved in almost all aspects of the VM product, including testing, packaging, design, development, and service. Most recently, he was a developer on OpenExtensions for z/VM, Java for z/VM, and Language Environment for VM. He has also been involved in the z/VM support of Linux for zSeries.

Aaron Kirby is a Infrastructure Architect in New Zealand. He has five years of experience working on enterprise system infrastructure including Windows server and workstation, Linux, UNIX, ALCS Airline reservation emulation, Web platforms, J2EE environments, middleware and mid-range enterprise systems.

Lutz Kühner is a system programmer working for Deutsche Bank AG in Germany. He has 15 years of experience in z/OS, including UNIX System Services, Networking, and z/VM. He has written extensively on chapter networking, Linux guest LAN, and CP for beginners.

Julie Murphy is a senior I/T Specialist with IBM Global Services in Poughkeepsie, New York. She joined IBM in 1981 as a VM/MVS Computer

Operator. She has 16 years of experience with VM System Programming. Her current assignment is as Team Leader for IBM Server Group's TPF and VICOM systems. Her area of expertise focuses on VM.

Michael Weisbach is a Senior Systems Engineer working for IT Services and Solutions GmbH (an IBM Global Services company), and a active member of the IBM ITS Central Region Linux Community. He has ten years of experience in the Linux field on Intel architecture and is afflicted with a IT security paranoia. He holds a degree in Computing Science from Berufsakademie Sachsen, Staatliche Studienakademie Glauchau, Germany. Currently he works in the areas of Linux-based High Availability and High Performance Clustering at the EMEA Linux Center of Competence, IBM Laboratory Boeblingen, Germany.

Thanks to the following people for their contributions to this project:

Terry Barthel, Dave Bennin, Ella Buslovich, Alison Chandler, Rich Conway, Roy Costa, Al Schwab
International Technical Support Organization, Poughkeepsie Center

Bob Haimowitz
International Technical Support Organization, Raleigh Center

Team of the Technical Marketing Competence Center, Boeblingen, Germany

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



Part 1

Running Linux under z/VM

In this part we discuss aspects of running Linux as a guest operating system under z/VM. Topics include:

- ▶ An overview of CP commands intended for the z/VM novice.
- ▶ The Directory Maintenance Facility (DirMaint) - a tool to help manage the VM directory.
- ▶ FCON/ESA - a VM monitoring tool which integrates Linux guest performance monitoring into z/VM.



z/VM for beginners

In this chapter, we present basic concepts intended for the novice z/VM user and introduce some basic CP commands useful for Linux guests running under z/VM.

1.1 The z/VM environment

The z/VM Control Program (CP) controls the virtual machine environment. CP provides each z/VM user with a working environment referred to as a *virtual machine*. A virtual machine is the functional equivalent of a real system. Virtual machines share real resources—processors, storage, console, and input/output devices. Those real resources will be managed with a User Directory provided by z/VM. A guest operating system (such as Linux) running in the virtual machine “sees” the virtual resources as real hardware.

1.2 Logging on to z/VM

There are two ways to log on to VM:

1. Normal VM logon, using the LOGON command.
2. Delegated logon, using the LOGON BY command.

Example 1-1 illustrates normal VM logon; provide your userid and password in the appropriate fields.

Example 1-1 Regular VM logon

z/VM ONLINE

```

                / VV      VVV MM      MM
                / VV      VVV MMM     MMM
ZZZZZZ        / VV      VVV  MMMM   MMMM
  ZZ          / VV      VVV   MM MM  MM MM
  ZZ          / VV  VVV   MM  MMM  MM
  ZZ          / VVVVV   MM  M   MM
  ZZ          / VVV     MM     MM
ZZZZZZ /      V      MM      MM
```

built on IBM Virtualization Technology
Consolidating Linux guests for fun and profit since July 8

Fill in your USERID and PASSWORD and press ENTER
(Your password will not appear when you type it)

USERID ==> LNX4

PASSWORD ==>

COMMAND ==>

RUNNING VMLINUX

Example 1-2 illustrates a delegated logon using the BY option. Use the command line from the logon screen to issue the command:

```
LOGON LNX88 BY MAINT
```

After pressing the Enter key, you are prompted for the delegated user password (in this case, the MAINT user).

Note: There must be a LOGONBY statement in the LNX88 directory entry in order to log on in this manner. In Example 9-3 on page 181, we show an example user directory entry which utilizes the LOGONBY statement.

Example 1-2 LOGON BY logon

```
z/VM ONLINE
```

```
          / VV          VVV MM          MM
          / VV          VVV MMM          MMM
ZZZZZZ / VV          VVV MMMM          MMMM
  ZZ    / VV  VVV      MM MM MM MM
  ZZ    / VV  VVV      MM  MMM  MM
  ZZ    / VVVVV      MM  M  MM
  ZZ    / VVV          MM          MM
ZZZZZZ / V           MM          MM
```

```
built on IBM Virtualization Technology
Consolidating Linux guests for fun and profit since July 8
```

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

```
USERID  ==>
```

```
PASSWORD ==>
```

```
COMMAND ==> LOGON LNX88 BY MAINT
```

```
RUNNING  VMLINUX
```

Upon successful logon, you will be presented with a screen similar to Example 1-3 on page 6. This is the 3270 console for your virtual machine. The virtual machine is now ready to accept input.

Example 1-3 The z/VM 3270 console

```
LOGON LNX4
z/VM Version 4 Release 3.0, Service Level 0201 (64-bit),
built on IBM Virtualization Technology
There is no logmsg data
FILES: 0003 RDR, NO PRT, NO PUN
LOGON AT 14:29:31 EDT WEDNESDAY 08/07/02
z/VM V4.3.0 2002-01-14 14:17
```

RUNNING VMLINUX

At this point, the console is running under the Conversational Monitoring System (CMS) operating system. You interact with the virtual machine using CP commands.

1.3 General CP command structure

The general format for CP commands is a command name followed any operands applicable to that command. Commands may be prefixed by the option **CP** keyword to indicate the command is to be processed immediately by the Control Program.

Commands and operands are case-insensitive; they may be entered in upper case, or lower case, or mixed case. For example, this command will display the system clock time value as determined by CP:

```
CP QUERY TIME
```

Tip: For help on any VM topic, type the following from the 3270 console command line:

```
HELP
```

For help on a specific command, use:

```
HELP command
```

where *command* is the command in question.

1.3.1 Command truncations and abbreviations

You need not enter commands as full words; abbreviations and truncations may be supplied instead. Using truncation, you can drop one or more letters from the end of the command.

The syntax diagram for each command shows the command displayed in both upper case and lower case letters. The upper case letters indicate the minimum truncation available for the command. For example, the syntax diagram for the **CP QUERY** command shows the command syntax as:

Query

This indicates the command may be entered in any of the forms:

**QUERY
QUER
QUE
QU
Q**

Abbreviations allow you to use alternate, shorter forms of the command. For example, the syntax diagram for the **CP MESSAGE** command shows the syntax as:

**Message
Msg**

This indicates the command:

- ▶ Has a minimum truncation value of **M**
- ▶ Has an abbreviation of **MSG**
- ▶ The abbreviation may be truncated to the values **MS** or **M**

Thus, CP accepts all of the following forms of the **MESSAGE** command:

**MESSAGE
MESSAG
MESSA
MESS
MES
MES
MSG
MS
M**

1.4 CP command privilege classes

CP commands are grouped in privilege classes indicative of the authorization required to issue the command. User authorization is defined in the VM directory

by the system programmer, and this determines the set commands available to the user. Privilege classes and user types are summarized in Table 1-1.

Table 1-1 CP privilege classes and user types

Class	User	Description
A	System Operator	The class A user controls the z/VM system. The system operator is responsible for the availability of the z/VM system and its resources. In addition, the system operator controls system accounting, broadcast messages, virtual machine performance options, and other options that affect the overall performance of z/VM. Note: The class A user who is automatically logged on during CP initialization is designated as the primary system operator.
B	System Resource Operator	The class B user controls all the real resources of the z/VM system, except those controlled by the system operator and the spooling operator.
C	System Programmer	The class C user updates or changes system-wide parameters of the z/VM system.
D	Spooling Operator	The class D user controls spool files and the system's real reader, printer, and punch equipment allocated to spooling use.
E	System Analyst	The class E user examines and saves system operation data in specified z/VM storage areas.
F	Service Representative	The class F user obtains, and examines in detail, data about input and output devices connected to the z/VM system. This privilege class is reserved for IBM use only.
G	General User	The class G user controls functions associated with a particular virtual machine.
H		Reserved for IBM use.

Class	User	Description
Any		Commands belonging to class "Any" are available to any user, regardless of his privilege class. These commands are primarily those used to gain access to, or relinquish access from, the z/VM system.
I-Z		Classes I through Z are reserved for redefinition through user class restructure (UCR) by each installation for its own use.

Note: Linux guest users are typically defined with privilege class G (General User) authorization.

1.4.1 How privilege class affects CP commands

In addition to determining which commands a VM user has access to, CP command privilege class affects the output generated by some commands. For instance, Example 1-4 shows the output of the **CP QUERY DASD** command when issued by a class G user.

Note: For a class G user, **CP QUERY DASD** is actually the CP command:

CP QUERY VIRTUAL DASD

with the optional **VIRTUAL** keyword omitted.

Example 1-4 Output from CP QUERY DASD issued by class G user

```

DASD 0120 3390 430RES R/O          67 CYL ON DASD 3750 SUBCHANNEL = 000D
DASD 0190 3390 430RES R/O          107 CYL ON DASD 3750 SUBCHANNEL = 0007
DASD 0191 3390 LNXU1R R/W           25 CYL ON DASD 3731 SUBCHANNEL = 0002
DASD 019D 3390 430RES R/O          102 CYL ON DASD 3750 SUBCHANNEL = 0008
DASD 019E 3390 430RES R/O          175 CYL ON DASD 3750 SUBCHANNEL = 0009
DASD 0201 3390 LX3FA2 R/W           200 CYL ON DASD 3FA2 SUBCHANNEL = 0000
DASD 0202 3390 LX3FA2 R/W          3138 CYL ON DASD 3FA2 SUBCHANNEL = 0001
DASD 0401 3390 430RES R/O           102 CYL ON DASD 3750 SUBCHANNEL = 000B
DASD 0402 3390 430RES R/O           102 CYL ON DASD 3750 SUBCHANNEL = 000A
DASD 0405 3390 430RES R/O           102 CYL ON DASD 3750 SUBCHANNEL = 000C

```

From the first line in the example, we see:

- ▶ Virtual device 0120 resides on a 3390 DASD unit with volume label 430RES.
- ▶ The minidisk consists of 67 cylinders mounted read-only (R/O) by this virtual machine.

- ▶ The real device number for the DASD unit is 3750 connected on subchannel 000D.

Contrast this with the output of the command:

```
CP QUERY DASD
```

issued by a class B user as shown in Example 1-5.

Example 1-5 Output from CP QUERY DASD issued by class B user

DASD 3731	CP SYSTEM	LNXU1R	42
DASD 3750	CP OWNED	430RES	290
DASD 3751	CP OWNED	430W01	65
DASD 3753	CP OWNED	430PAG	2
DASD 3B44	CP OWNED	430PG2	0
DASD 3BA3	CP SYSTEM	LX3BA3	4
DASD 3BA4	CP SYSTEM	LX3BA4	6
DASD 3CA1	CP SYSTEM	LX3CA1	4
DASD 3CA3	CP SYSTEM	LX3CA3	4
DASD 3DA1	CP SYSTEM	LX3DA1	7
DASD 3DA2	CP SYSTEM	LX3DA2	0

In this example, we see:

- ▶ Real device 3731 is a DASD device allocated for user minidisk allocations (indicated by SYSTEM) with volume label LNXU1R. The volume is linked to 42 minidisks.
- ▶ Real device 3750 is a DASD device allocated for paging and spooling activity (indicated by OWNED) with volume label 430RES. The volume is linked to 290 minidisks.

To the class G user, the following command reports the attached virtual minidisks:

```
CP QUERY DASD
```

For the class B user however, the attached real DASD devices are reported.

Tip: To report attached virtual minidisks regardless of user privilege class, use this command:

```
CP QUERY VIRTUAL DASD
```

1.5 The CP status indicator

An indication of the state of the virtual machine and how it will respond to input can be found at the bottom right-hand corner of the 3270 VM console. The possible states and their respective meanings are listed in Table 1-2.

Table 1-2 CP status indicators meanings

Status indicator	Indicates
CP READ	CP expects input. Enter: Begin (abbreviated B) to continue processing.
VM READ	Similar to CP READ; however, input is expected by CMS. Press Enter to continue.
RUNNING	The virtual machine is working properly and is awaiting a command to process.
MORE...	The virtual machine display output exceeded the screen ranges. Press the CLEAR or PA2 key to proceed to the next screen (on modern PC keyboards, use the PAUSE/BREAK key). If no input is received after 10 seconds, the screen will be automatically cleared and the next page is displayed. A short beep is issued shortly before the screen changes. This time interval can be changed using the CP TERMINAL command.
HOLDING	Similar to MORE... but any highlighted messages are left outstanding on the screen.
NOT ACCEPTED	The virtual machine did not process the previous input.

Attention: If your virtual machine is in the CP READ state, the guest operating system is suspended. For a Linux guest, this means Linux will get no CPU usage and will not service any requests.

1.6 Using Program Function keys

To conserve keystrokes, use the CP SET PFxx command to assign a command to any of the 24 Program Function (PF) keys available on the keyboard. The command format is as follows:

```
CP SET PFxx [ options ] command
```

where:

xx Indicates the PF key number (1-24)
options Is an optional command modifier

command Is the command to bind to the PF key

Command modifiers are described in Table 1-3.

Table 1-3 CP SET PF command modifiers

Modifier	Description
DELAYED	The bound command is displayed in the input area when you press the PF key. The command may then be modified as necessary. Press Enter to process the command. DELAYED is the default.
IMMED	The bound command is executed immediately on pressing the PF key. The command is re-displayed on the command line upon completion.
NODISP	Similar to IMMED ; however, the command is not re-displayed.

Typically, PF key bindings are set in the user PROFILE EXEC to set definitions for the duration of the VM session. Example 1-6 illustrates some binding PF key bindings.

Example 1-6 Use of CP SET PF command in the PROFILE EXEC

```
CP SET PF1 NODISP HELP /* sets PF1 to HELP */
CP SET PF3 IMMED QUERY TIME /* sets PF3 to query system time */
CP SET PF12 RETRIEVE /* sets PF12 to command retrieval */
```

Note: When logging on to z/VM, the user will IPL a CMS guest operating system. As part of the IPL process, CMS will execute a REXX exec named PROFILE EXEC on the A-disk of logged on user.

By default, statements included in this file will be executed at logon. To bypass execution of the PROFILE EXEC at IPL, see 1.8.2, “Recovering from unattended startup” on page 14.

To find the current PF key assignments issue the command:

```
CP QUERY PF
```

The output is shown in Example 1-7.

Example 1-7 Output from the CP QUERY PF command

```
PF01 NODISP HELP
PF02 UNDEFINED
PF03 IMMED QUERY TIME
```

PF04 UNDEFINED
PF05 UNDEFINED
PF06 UNDEFINED
PF07 UNDEFINED
PF08 UNDEFINED
PF09 UNDEFINED
PF10 UNDEFINED
PF11 UNDEFINED
PF12 RETRIEVE

1.7 Disconnecting the 3270 session

To disconnect the 3270 VM console without stopping operation of the virtual machine, issue:

DISConnect

The virtual machine will remain disconnected until the next logon. To reconnect to the virtual machine, logon and issue:

Begin

Note: When you logon to a disconnected virtual machine, you are put into the CP READ state. The guest will be suspended as discussed in 1.5, “The CP status indicator” on page 11 until the **Begin** command is issued.

1.8 Booting Linux in a virtual machine

For a complete description of how to install Linux as a guest operating system running under z/VM, refer to the IBM Redbook *Linux on IBM @server zSeries and S/390: Distributions*, SG24-6264.

Typically, Linux is booted from a storage device defined to the Linux guest user virtual machine.

To boot the Linux guest, issue the following CP command, where *vdev* is the virtual device number containing the Linux guest operating system:

Ipl vdev CLear

On IPL, the VM console will display the familiar boot sequence messages generated by Linux startup.

Tip: The CLEAR operand to the IPL command is optional. If specified, the content of the virtual machine's storage is set to binary zeros before the guest operating system is loaded.

We suspect this option has no effect when IPLing a Linux guest, as we've seen no case in which omitting the parameter has had a detrimental effect.

In the event a fatal error occurs on Linux IPL, you can return to CMS using:

```
I CMS
```

Note that **I** is a truncation of the **IPL** command.

1.8.1 Unattended startup of a Linux guest

To automate the process of IPLing a Linux guest during logon, include the following statement in the PROFILE EXEC A of the Linux guest userid:

```
IPL vdev CLEAR
```

Important: Before including this statement in PROFILE EXEC, ensure that Linux has been successfully installed. Otherwise, you may get into a situation where z/VM will go into a continuous **CP READ** state. Refer to 1.8.2, "Recovering from unattended startup" on page 14 for a hint on how to recover.

1.8.2 Recovering from unattended startup

If you have included a IPL statement in the PROFILE EXEC of a Linux guest to automatically boot Linux on logon, there may be times you would like to forego execution of the PROFILE EXEC. For instance, you may need to modify the virtual machine of the Linux guest, or the Linux boot device may need repair.

To suppress the automatic invocation of the PROFILE EXEC, the first command you enter after the **IPL CMS** command is the **CMS ACCESS** command with the **NOPROF** option. For example, enter the following:

```
IPL CMS
```

The system response will be:

```
z/VM V4.4.0    2002-07-11 10:58
```

To suppress the execution of your PROFILE EXEC, enter:

```
ACCESS ( NOPROF
```


The system responds with:

```
Ready;
```

Now you have loaded CMS and accessed filemode A, without running your PROFILE EXEC.

1.9 Communicating with CP from a Linux guest

Commands can be passed from a Linux guest to the virtual machine for processing by CP. The communication mechanism depends on whether commands issued from the 3270 VM console, or from a Linux telnet session.

Attention: Be careful issuing the CP command **DETach** from the Linux guest. It can destroy the environment of the virtual machine.

If this happens, you will need to log off and then log on to z/VM to restore the virtual machine.

1.9.1 Communicating with CP from the VM console

From the 3270 VM console, you can issue CP commands by simply prefacing the command with the **#CP** control sequence. For instance, to query the virtual machine minidisk assignments, issue the command:

```
#CP QUERY VIRTUAL DASD
```

Note: This feature is enabled by z/VM and does not require installation of any additional program product or package.

1.9.2 Communicating with CP from a Linux telnet session

In order to issue CP commands from a Linux telnet session, you will need to install the **cpint** package. The **hcp** command can then be used to issue commands to CP from the telnet session. Using **hcp**, you can query the virtual minidisk assignments using:

```
hcp query virtual dasd
```

Tip: Remember to quote special characters (such as *****) to prevent expansion by the Bash shell.

Important: Do not enter the `hcp` command without any parameters will cause the virtual machine to enter the CP READ state (see 1.5, “The CP status indicator” on page 11).

We show an example of using the `hcp` command in 8.11.4, “The `vmgetconf` script” on page 166.

1.10 Querying the virtual machine

We now discuss some CP commands you can use to determine some of the attributes of the virtual machine. These commands are available to class G users, and they simply report resource definitions for the current virtual machine. To actually add or modify these resources, you need to have system programmer authority. See Chapter 2, “Directory Maintenance Facility for z/VM” on page 23 for an overview on how to define resources to a virtual machine.

1.10.1 Querying storage devices

The virtual machine defines disks (referred to as *minidisks*) for DASD storage. Minidisks are assigned to a user’s virtual machine by the z/VM system programmer. Minidisks are typically partitions on a real DASD device (similar to partitioning on other operating systems such as Linux, OS/2, Windows, and DOS). To view your virtual machine minidisk definition, use this command:

```
CP QUERY VIRTUAL DASD
```

Example 1-8 illustrates the output report.

Example 1-8 Output from the CP QUERY VIRTUAL DASD command

DASD 0120	3390	430RES	R/O	67	CYL	ON	DASD	3750	SUBCHANNEL = 000D
DASD 0190	3390	430RES	R/O	107	CYL	ON	DASD	3750	SUBCHANNEL = 0007
DASD 0191	3390	LN XU1R	R/W	25	CYL	ON	DASD	3731	SUBCHANNEL = 0002
DASD 019D	3390	430RES	R/O	102	CYL	ON	DASD	3750	SUBCHANNEL = 0008
DASD 019E	3390	430RES	R/O	175	CYL	ON	DASD	3750	SUBCHANNEL = 0009
DASD 0201	3390	LX3FA2	R/W	200	CYL	ON	DASD	3FA2	SUBCHANNEL = 0000
DASD 0202	3390	LX3FA2	R/W	3138	CYL	ON	DASD	3FA2	SUBCHANNEL = 0001
DASD 0401	3390	430RES	R/O	102	CYL	ON	DASD	3750	SUBCHANNEL = 000B
DASD 0402	3390	430RES	R/O	102	CYL	ON	DASD	3750	SUBCHANNEL = 000A
DASD 0405	3390	430RES	R/O	102	CYL	ON	DASD	3750	SUBCHANNEL = 000C

In the example, we see the minidisk allocation for virtual address 0120. It consists of 67 cylinders on a 3390 DASD unit with a physical volume name of 430RES. The virtual machine has read-only access to the minidisk.

1.10.2 Querying network devices

Your virtual machine will normally have at least one network interface to communicate with the outside world. If your virtual machine uses a simulated network card (referred to as a NIC), you can view the characteristics of that NIC by using this command:

```
CP QUERY NIC
```

In Example 1-9, we show the format of the command output:

Example 1-9 Output of the CP QUERY NIC command

```
Adapter 0500 Type: HIPERS Name: UNASSIGNED Devices: 3
Port 0 MAC: 00-04-AC-00-00-1D LAN: SYSTEM PRIVHIPE MFS: 16384
Adapter 0700 Type: QDIO Name: UNASSIGNED Devices: 3
Port 0 MAC: 00-04-AC-00-00-20 LAN: SYSTEM PRIVQDIO MFS: 8192
```

In this instance, the virtual machine has defined two NICs: a simulated hipersocket (HIPERS) attached to virtual address 0500, and a simulated ethernet (QDIO) attached to virtual address 0700.

Detailed NIC definition report

You can get more detailed information on NIC adapters by using this command, where *vdev* is the virtual address of the desired NIC:

```
CP QUERY NIC vdev DETAILS
```

In Example 1-10, we show the detailed report for the simulated ethernet interface generated by the command:

```
CP QUERY NIC 0700 DETAILS
```

Example 1-10 Output of the CP QUERY NIC 0700 DETAILS command

```
Adapter 0700 Type: QDIO Name: NIC0700 Devices: 3 1
Port 0 MAC: 00-04-AC-00-00-20 LAN: SYSTEM PRIVQDIO MFS: 8192 2
RX Packets: 259143 Discarded: 0 Errors: 0 3
TX Packets: 213788 Discarded: 12 Errors: 0
RX Bytes: 375747841 TX Bytes: 13044137
Connection Name: HALLOLE State: Session Established 4
Device: 0700 Unit: 000 Role: CTL-READ
Device: 0701 Unit: 001 Role: CTL-WRITE
Device: 0702 Unit: 002 Role: DATA
Unicast IP Addresses: 5
10.0.3.30
Multicast IP Addresses: 6
224.0.0.1 01-00-5E-00-00-01
```

Key points are discussed below:

1. The adapter assigned to virtual device number 0700 is of type QDIO (a hipersocket would be defined with `Type: HIPERS`). The portname of the adapter is `NIC0700`, and there are three devices which form the adapter (0700, 0701, and 0702).
2. The MAC address assigned to the adapter is `00-04-AC-00-00-20`. It is coupled to VM LAN `PRIVQDIO`, and has a Maximum Frame Size (MFS) of 8192.
3. The transmit and receive statistics for the adapter are reported.
4. The Connection Name for the adapter is `HALL0LE`, and it is enabled.
5. The IP address assigned for Unicast is `10.0.3.30`.
6. The IP address assigned for Multicast is `224.0.0.1`.

Tip: You can get a detailed report on all simulated NIC adapters using the command:

```
CP QUERY NIC ALL DETAILS
```

For details on simulated NIC adapters and how to define z/VM guest LANs, see Chapter 4, “HiperSockets and z/VM Guest LAN” on page 57.

1.10.3 Querying the CPUs available to the virtual machine

To find how many CPUs are defined to your virtual machine, use this command:

```
CP QUERY VIRTUAL CPUS
```

Example 1-11 Output of the CP QUERY VIRTUAL CPUS command

```
CPU 00 ID FF0C0ECB20640000 (BASE)
```

The command response, illustrated in Example 1-11, shows there is one CPU defined to the virtual machine.

1.10.4 Querying virtual storage

To find how much storage is defined to the virtual machine, use this command:

```
CP QUERY VIRTUAL STORAGE
```

Example 1-12 Output of the CP QUERY VIRTUAL STORAGE command

```
STORAGE = 128M
```

The response, illustrated in Example 1-12, shows there is 128 MEG of virtual storage defined to the virtual machine.

Note: Class G users may increase virtual storage in the virtual machine up to a limit set in the VM directory by using this command:

```
CP DEFINE STORAGE store
```

where *store* defines the amount of virtual storage to be defined. The value of *store* is limited by the VM directory entry for the user (see 2.7.3, “Defining a user directory entry” on page 31).

1.11 Using DDR to copy a minidisk

DASD Dump Restore (DDR) is a utility used to dump, copy, or print data that resides on z/VM user minidisks or dedicated DASD. This utility may also be used to restore or copy DASD data that resides on z/VM user tapes. The DDR utility has five functions:

1. Dumping part or all of the data from DASD to tape.
2. Restoring data from tapes created by the DDR dump function to DASD. The DASD must be the same type that originally contained the data.
3. Copying data from one device to another of the same type.
4. Printing selected parts of DASD or tape records to the virtual printer.
5. Displaying selected parts of DASD or tape records to the terminal.

DDR uses control and function statements to describe the intended processing and the needed I/O devices. The control statements are INput, OUTput, and SYsprint. The most useful function statements are COpy, DUmp, and REstore. We consider only the copy function of the DDR utility; refer to Example 1-3.

Example 1-13 Sample DDR copy

```
ddr 1  
z/VM DASD DUMP/RESTORE PROGRAM  
ENTER:  
sysprint cons 2  
ENTER:  
input 192 3390 3  
ENTER:  
output 193 3390 4  
ENTER:  
copy all 5  
HCPDDR711D VOLID READ IS TEMP 6  
DO YOU WISH TO CONTINUE? RESPOND YES, NO OR REREAD:  
yes  
HCPDDR711D VOLID READ IS TEMP
```

```

DO YOU WISH TO CONTINUE? RESPOND YES, NO OR REREAD:
yes
COPYING TEMP 7
COPYING DATA 08/02/02 AT 12.48.13 GMT FROM TEMP TO TEMP
INPUT CYLINDER EXTENTS OUTPUT CYLINDER EXTENTS
      START STOP START STOP
      00000000 00000009 00000000 00000009
END OF COPY 8
ENTER: 9

END OF JOB

```

Key details are discussed below:

1. Invoke the utility using the DDR command.
2. The SYSPRINT statement direct output the console.
3. The INPUT statement describes the input device; we specify a 3390 DASD device at address 192.
4. The OUTPUT statement describes the output device; we specify a 3390 DASD device at address 193.
5. The COPY statement instructs the utility to copy from the INPUT device (the 192 disk) to the OUTPUT device (the 193 disk).
6. DDR asks you to verify the volume id of each disk. If correct, answer YES.
7. The copy starts.
8. The copy ends - DDR reports the number of cylinders copied.
9. At this point you can either press Enter to end the utility, or enter more control and function statements.

For additional information on the DDR command, refer to *z/VM V4R3.0 CP Command and Utility Reference*, SC24-6008.

1.12 Virtual Machine Resource Manager

The Virtual Machine Resource Manager (VMRM) was introduced in z/VM Version 4 Release 3.0. VMRM provides a facility for monitoring and adjusting certain CPU and DASD goals. You, as the system administrator, can define workloads and create groups of users which will use those workloads. VMRM automatically adjusts CPU or DASD performance parameters when contention for a resource occurs between virtual machines. A complete description of VMRM can be found in *z/VM V4R3.0 Performance*, SC24-5999.

Important: VMRM uses the CP SET SHARE command to change a virtual machine's system-resource-access priority, and uses the CP SET IOPRIORITY command to change a virtual machine's I/O priority.

The use of VMRM and these CP commands can be very useful for adjusting the workload of your z/VM system and the ability of virtual machines to operate on your system.

However, they can also have an adverse effect on your overall z/VM performance if they are used incorrectly. For this reason, you need to review the overall performance of your z/VM system and carefully define the workloads and goals you want to achieve.



Directory Maintenance Facility for z/VM

This chapter introduces the z/VM Directory Maintenance Facility (DirMaint) for the maintenance of the VM directory. We describe some of the features available to help you maintain this directory, and provide example usage of DirMaint commands.

2.1 Managing VM using DirMaint

Directory Maintenance Facility (DirMaint) is an IBM program product used to manage the VM directory. DirMaint provides support for all the z/VM directory statements. For large scale deployment, DirMaint can ease management of the VM directory. You do not edit the directory; rather, you issue DirMaint commands to define VM resources. In addition, DirMaint provides automated validation and extent allocation routines to reduce the chance of operator error. DirMaint commands have a similar name and format as the VM directory statements they support.

Attention: When using the GA version of z/VM 4.3.0, APAR VM63033 must be applied to DirMaint to provide a variety of new functional enhancements.

We describe some of the more common functions provided by DirMaint. For a complete description of all DirMaint commands, see the *z/VM V4R3.0 Directory Maintenance Facility Function Level 410 Command Reference*, SC24-6025.

For details on how to administer DirMaint, see the *z/VM V4R3.0 Directory Maintenance Facility Function Level 410 Tailoring and Administration Guide*, SC24-6024.

2.2 DirMaint service machines

DirMaint uses several service machines for performing different tasks:

▶ 4VMDVH10

The DirMaint install and service user ID. By default, 4VMDVH10 owns the DASD space containing the product code, the customer tailored files, and any customized user exits.

▶ DIRMAINT

The primary DirMaint id. It handles everything to do with the source directory and controls the DATAMOVE and DIRSAT service machines.

▶ DATAMOVE

A service machine that handles minidisk manipulation on behalf of DirMaint.

▶ DIRMSAT

A service machine that is mainly used in a multiple system environment. It handles the object directory on other systems that are not running the DirMaint service machine.

2.2.1 DirMaint service machine

The directory maintenance service machine userid is by default DIRMAINT. Its tasks include:

- ▶ Owning the control program (CP) source directory
- ▶ Receiving transactions from authorized users
- ▶ Verifying the transactions are valid
- ▶ Making the appropriate updates to the source directory

If full Direct Access Storage Device (DASD) services are enabled, DirMaint also:

- ▶ Allocates work among one or more DATAMOVE service machines
- ▶ Monitors the progress of each machine

2.2.2 DATAMOVE service machine

If a command changes any DASD space allocation, DirMaint will delegate the task to the DATAMOVE service machine. By default, the DATAMOVE service machine userid is DATAMOVE. Its tasks include:

- ▶ Formatting newly allocated DASD space a virtual machine
- ▶ Copying Conversational Monitor System (CMS) files from an existing disk to newly formatted extents.
- ▶ Formatting old extents being deallocated to prevent exposure of any residual data to the next user.

2.3 DirMaint command syntax

The general format of a DirMaint command is:

```
DIRMaint [ prefix ] command [ cmd_options ]
```

where:

DIRMaint	Is the name of the DirMaint EXEC
<i>prefix</i>	Is an optional command prefix keyword and any operands required for that keyword.
<i>command</i>	Is the DirMaint command
<i>cmd_options</i>	Are any options to be passed to <i>command</i>

2.3.1 Using prefix keywords

Prefix keywords provide additional instructions to DirMaint regarding command processing. Prefixes are placed before the command on which they operate. As an example, the following prefix instructs DirMaint that *command* is to modify the directory entry for user *userid*:

```
FORuser userid
```

2.4 Some useful DirMaint commands

Some useful DirMaint commands are:

SEND	Request a copy of a DirMaint control file
FILE	Add or replace a DirMaint control file
RLDCode	Cause DirMaint to reload its resident operating procedures
RLDExtn	Cause DirMaint to reload its extent control file
Add	Add a new user or profile directory entry
REView	Review a user or profile directory entry
AMDisk	Adds a new minidisk
DEDicate	Add or delete an existing dedicate statements
DMDisk	Removes a minidisk
LOGONBY	Allows users to use their own password to logon to different IDs
MDisk	Change the access mode and passwords for minidisks
STorage	Change logon storage size
SETOptn	Add, change or delete CP options
CLAss	Change the CP class for a directory entry
SPEcial	Add or delete an existing special statement
TMDisk	Transfer ownership of a minidisk from one userid to another

2.5 Defining a userid as a DirMaint administrator

You can define additional VM users as DirMaint administrators. This gives you flexibility of having more than one administrator to manage the VM directory. To define a VM userid as a DirMaint administrator, you first need to:

1. Obtain the AUTHFOR CONTROL file from DirMaint
2. Edit the control file, adding the additional userid as a DirMaint administrator
3. Send the revised AUTHFOR CONTROL file back to DirMaint
4. Tell DirMaint to reload its resident operating procedures

2.5.1 Obtain the DirMaint AUTHFOR CONTROL file

To obtain the AUTHFOR CONTROL file, use:

```
DIRM SEND AUTHFOR CONTROL
```

DirMaint will send a file to your reader called AUTHFOR CONTROL. Receive this file using the replace option.

2.5.2 Format of the AUTHFOR CONTROL file

Example 2-1 illustrates the format of the AUTHFOR CONTROL file. The file consists of space delimited fields, as explained in Table 2-1 on page 27.

Example 2-1 AUTHFOR CONTROL file

ALL MAINT	*	140A	ADGHOPS
ALL MAINT	*	150A	ADGHOPS
ALL COSTA	*	140A	ADGHOPS
ALL COSTA	*	150A	ADGHOPS
ALL MBEATTIE	*	140A	ADGHOPS
ALL MBEATTIE	*	150A	ADGHOPS
ALL LXOPR	*	150A	ADGHOPS
ALL LXOPR	*	140A	ADGHOPS
ALL LXADMIN	*	150A	ADGHOPS
ALL LXADMIN	*	140A	ADGHOPS

Table 2-1 AUTHFOR CONTROL field definitions

Field number	Description
1	The target userid. When specified as the keyword ALL, the authorized userid may act on behalf of any userid in the VM directory.
2	The authorized userid. This userid may act on behalf of the target userid.

Field number	Description
3	The network node id. An asterisk allows the authorized userid to enter commands from any system within the cluster.
4	The command level. Valid values are 140A or 150A. A command level 140A allows the user to enter commands using DirMaint Release 4 compatibility syntax and 150A uses DirMaint Release 5 full function syntax
5	The command set. This identifies which commands the user may use. See <i>z/VM V4R3.0 Directory Maintenance Facility Function Level 410 Command Reference</i> , SC24-6025 for the IBM-defined default command sets.

2.5.3 Activating AUTHFOR CONTROL file changes

Once the AUTHFOR CONTROL file has been edited to define additional administrators, the changes are made to DirMaint using:

```
DIRM FILE AUTHFOR CONTROL
```

To tell DirMaint to reload its resident operating procedures, use:

```
DIRM RLDC
```

2.6 Adding a volume to a DirMaint group

The DirMaint EXTENT CONTROL file defines volumes used for minidisk allocation and provides the layout structure of how space on those volumes should be used. You should define volumes in the EXTENT CONTROL file before adding users. This comes into play especially when using a prototype.

2.6.1 Obtain the DirMaint EXTENT CONTROL file

To obtain the EXTENT CONTROL file, use:

```
DIRM SEND EXTENT CONTROL
```

DirMaint will send the file to your reader.

2.6.2 Format of the EXTENT CONTROL file

Example 2-2 shows a sample EXTENT CONTROL file.

Example 2-2 EXTENT CONTROL file

```
* ****
:REGIONS. 1
  *RegionId VolSer RegStart RegEnd Type
  430RES 430RES 001 3338 3390-03
  430W01 430W01 001 3338 3390-03
  LNXU1R LNXU1R 001 3338 3390-03
  LX651A LX651A 001 3338 3390-03
  LX660E LX660E 001 3338 3390-03
  LX660F LX660F 001 3338 3390-03
:END.
:GROUPS. 2
ANY LNXU1R
LXDEMO LX651A LX660E LX660F
:END.
:EXCLUDE. 3
* USERID ADDRESS
:END.
:AUTOBLOCK. 4
  * IBM supplied defaults are contained in the AUTOBLK DATADVH file.
  * The following are customer overrides and supplements.
  *
  *DASDType BlockSize Blocks/Unit Alloc_Unit Architecture
:END.
:DEFAULTS. 5
  * IBM supplied defaults are contained in the DEFAULTS DATADVH file.
  * The following are customer overrides and supplements.
  *
  *DASDType Max-Size
:END. 6
```

Following are explanations of the details shown in the example:

1. The :REGIONS. stanza defines the area on the DASD volume to be used by DirMaint for automatic allocation.
2. The :GROUPS. stanza defines a group of regions to be used by DirMaint for automatic allocation.
3. The :EXCLUDE. stanza defines users or user/device combination that should be excluded by DirMaint DASD subsystem.
4. The :AUTOBLOCK. stanza defines blocking factors for various device types.
5. The :DEFAULTS. stanza defines the default maximum size for various DASD devices.
6. The :END. tag defines the end of a stanza.

2.6.3 Activating EXTENT CONTROL file changes

Once the EXTENT CONTROL file has been edited, the changes are made to DirMaint using:

```
DIRM FILE EXTENT CONTROL
```

To tell DirMaint to use the updated file, use:

```
DIRM RLDE
```

2.7 Adding directory entries

Using DirMaint, you can add, modify, and review VM directory entries. To simplify the process of defining many similar userids, you can use directory *profiles*. Profiles allow you to define common user characteristics. Defined profiles can then be included in user definitions. More than one profile may exist in the VM directory.

2.7.1 Defining a profile directory entry

Directory profiles are defined text files with names of the form:

```
PROFNAME DIRECT
```

where *PROFNAME* is the profile name. Once defined, these can then be added to DirMaint. Example 2-3 illustrates a directory profile named MYSAMPLE DIRECT.

Example 2-3 MYSAMPLE DIRECT

```
PROFILE MYSAMPLE      1
  IPL CMS              2
  MACH XA              3
  SPOOL 000C 2540 READER * 4
  SPOOL 000D 2540 PUNCH A
  SPOOL 000E 1403 A
  CONSOLE 009 3215 T   5
  LINK MAINT 0190 0190 RR 6
  LINK MAINT 019D 019D RR
  LINK MAINT 019E 019E RR
```

Following are explanations of the details shown in the example:

1. Specifies the name of the profile - in this case, MYSAMPLE. Profile names must be one to eight-character alphanumeric strings.
2. Specifies that CMS is to be IPLed when the userid logs on.
3. Specifies the virtual machine type to be used. Valid types are ESA, XA, or XC.

4. Specifies the reader, punch, and printer address and device type.
5. Specifies the console address and device type.
6. Specifies links to the CMS system disks.

2.7.2 Adding a profile directory entry

To add the above directory profile, use:

```
DIRM ADD MYSAMPLE
```

2.7.3 Defining a user directory entry

Users are defined text files with names of the form:

```
USERNAME DIRECT
```

where *USERNAME* is the VM user name. Once defined, these can then added to DirMaint. Example 2-3 illustrates a user profile named LNX88 DIRECT.

Example 2-4 LNX88 DIRECT file

```
USER LNX88 LNX88PW 128M 1G G 1
INCLUDE MYSAMPLE 2
AMD 191 3390 AUTOG 5 LXDEMO MR 3
```

Following are explanations of the details shown in the example:

1. Defines a user named LNX88 with password LNX88PW. The user will run in virtual machine with 128 MEG of storage and 1 GIG storage maximum. The user will run with class G CP privileges.
2. The MYSAMPLE directory profile is to be included in this user's definition.
3. Allocates a five-cylinder 191 minidisk.

Adding a user directory entry

To add the above user, use:

```
DIRM ADD LNX88
```

2.7.4 Adding a userid using a prototype file

When you need to make several userids with the same specifications, you can use a prototype file. To create a VM user named LNX99 based on the LEAF prototype, use:

```
DIRM ADD LNX99 LIKE LEAF
```

In Example 2-5, we show the LEAF prototype definition.

Note: Prototype definitions are defined in a file named *NAME* PROTODIR, where *NAME* is prototype name.

Example 2-5 The LEAF PROTODIR definition

```
USER LEAF LEAFX 16M 64M G
    INCLUDE USRDFLT
    XAUTOLOG LEAFMSTR
    MDISK 0191 3390 AUTOG 1 LXDEMO MR
```

In Example 2-6, we show the directory entry for user LNX99.

Example 2-6 Directory entry for a user created using a prototype

```
USER LNX99 XXXXXXXX 16M 64M G
DVHRXV3355I The following records are included from profile: USRDFLT
    PROFILE USRDFLT
    IPL CMS
    MACH XA
    CONSOLE 0009 3215 T
    SPOOL 000C 2540 READER *
    SPOOL 000D 2540 PUNCH A
    SPOOL 000E 1403 A
    LINK MAINT 0190 0190 RR
    LINK MAINT 019D 019D RR
    LINK MAINT 019E 019E RR
*DVHOPT LNK0 LOG1 RCM1 SMSO NPW1 LNGAMENG PWC20020719 CRCâ"
DVHRXV3355I The preceding records are included from profile: USRDFLT
    XAUTOLOG LEAFMSTR
    MDISK 0191 3390 940 1 LX651A MR
```

2.8 Maintaining directory entries

You can use DirMaint to review and modify directory entries.

2.8.1 Reviewing a directory entry

To review a profile or user directory entry, issue the REView DirMaint command. For example, to view the directory entry for user LNX88, use:

```
DIRM FOR LNX88 REV
```

DirMaint will send the LNX88 DIRECT file to your reader.

2.8.2 Adding a minidisk to a user directory entry

You can add additional minidisks to a user with DirMaint. To create a new 192 minidisk of 10 cylinders allocated for userid LNX88, use:

```
DIRM FOR LNX88 AMD 192 3390 AUTOG 10 LXDEMO MR
```

The new disk will reside on one of the volumes under the DirMaint group called LXDEMO (see 2.6, “Adding a volume to a DirMaint group” on page 28 for a discussion of volume groups).

2.8.3 Adding access passwords to a minidisk

You can add a read, write, or multiple-write password to your minidisk. The following command will add these passwords:

```
DIRM FOR LNX88 MD 191 = ALL WRITE MULTI
```

The character string ALL will allow any userid to link read-only to that disk without knowing the correct password. It also set the write password to WRITE and the multiple-write password to MULTI,

2.8.4 Dedicating a device to a userid

The following command will dedicate real device 2320 as LNX88 virtual device 111:

```
DIRM FOR LNX88 DED 111 2320
```

The most common devices to dedicate are DASD, tape drives, and networking devices such as OSA cards and CTCs.

Note: If more than one virtual machine has a dedicate statement for a given real device, only the first virtual machine to log on receives control of the device.

2.8.5 Deleting a new minidisk from a user directory entry

To remove a minidisk entry, use:

```
DIRM FOR LNX88 DMD 192 CLEAN
```

The 192 minidisk for userid LNX88 will be removed from the directory entry, and the disk will be cleaned on removal.

2.8.6 Changing virtual storage for VM users

The following command will increase the storage for user LNX88 to 512 MEG:

```
DIRM FOR LNX88 STORAGE 512M
```

2.8.7 Adding, deleting, and modifying CP options

Use the SETOptn command to add, delete, or modify CP options. For example, the following command will add the QUICKDSP CP option to user LNX88:

```
DIRM FOR LNX88 SETO ADD QUICKDSP
```

Note: QUICKDSP causes a virtual machine to be added to the dispatch list immediately when it has work to do. This will increase the performance of that userid. This option should be used sparingly.

2.8.8 Changing CP Privileges

Use the CLAss command to change the CP privileges associated with a directory entry. For example, the following command will assign user LNX88 to CP class B:

```
DIRM FOR LNX88 CLASS ADD B
```

Note: A class B user can control all the real resources of the z/VM system. For a list of privileges, see 1.4, “CP command privilege classes” on page 7.

2.8.9 Using the SPECIAL DirMaint command

You can add or delete SPECIAL directory entry statements using the DirMaint SPECIAL command. For instance, the following command will create a simulated QDIO adapter for user LNX88:

```
DIRM FOR LNX88 SPECIAL 700 QDIO 3 SYSTEM PRIVQDIO
```

A simulated Network Interface Card (NIC) is defined at logon time with three devices starting at base address 700. The NIC will automatically be coupled to a VM LAN with the userid of SYSTEM and the LAN name of PRIVQDIO.

2.8.10 Transferring a minidisk between userids

You can transfer a minidisk to another userid; for example, using this command will transfer LNX88 193 minidisk to MAINT as its 999 minidisk:

```
DIRM FOR LNX88 TMD 193 to MAINT 999
```

2.8.11 Adding shared logon access to a userid

To be able to logon to another userid with your own password, use the following command:

```
DIRM FOR LNX88 LOGONBY MAINT
```

This will allow userid MAINT to logon to LNX88 using MAINT user password.

Note: A maximum of eight users can be specified on the LOGONBY directory statement.



FCON/ESA for monitoring a penguin colony

This chapter describes the VM/ESA Full Screen Operator Console and Graphics Realtime Performance Monitor (FCON/ESA) tool. Using FCON/ESA, you can monitor your Linux guests from a z/VM session.

3.1 Introducing FCON/ESA

VM/ESA Full Screen Operator Console and Graphical Realtime Performance Monitor (FCON/ESA) was developed by Eginhard Jaeger of IBM Switzerland. It provides performance monitoring capabilities with system console operations in full screen mode. FCON/ESA can give you a real time view of system performance. New performance monitoring and resource management functions will be made available in a future release of z/VM.

Using FCON/ESA as the base, this optional feature will include functional capabilities not provided today by the Performance Reporting Facility (PRF) and RealTime Monitor (RTM) features of z/VM. The existing PRF and RTM priced optional features will be withdrawn from marketing in a future release of z/VM.

Important: See the Statement of Direction at:

http://www.ibm.com/servers/eserver/zseries/library/specsheets/gm130075_more2.html#9

3.2 FCON/ESA support for Linux on z/VM

With release 3.2.03 of FCON/ESA, a Linux performance data interface has been added. Data retrieval and display of Linux internal performance data is based on the Linux Distributed Data Server (DDS) interface, originally written for use with Resource Measurement Facility (RMF) Performance Monitoring (PM).

The DDS interface must be installed and active on all Linux systems that are to be monitored. Performance data retrieval is based on Extensible Markup Language (XML) requests, sent to the Linux systems via TCP/IP. Only the data actually needed for building a specific Linux performance report is retrieved.

Performance data is collected only in the Linux systems, and performance history data is only available from the filesystems of the Linux systems. FCON/ESA does not collect or save any Linux internal performance data. Figure 3-1, “Linux performance data collection” on page 39 illustrates the components involved.

For additional information about FCON/ESA, see:

<http://www.vm.ibm.com/perf/perfprod.html>

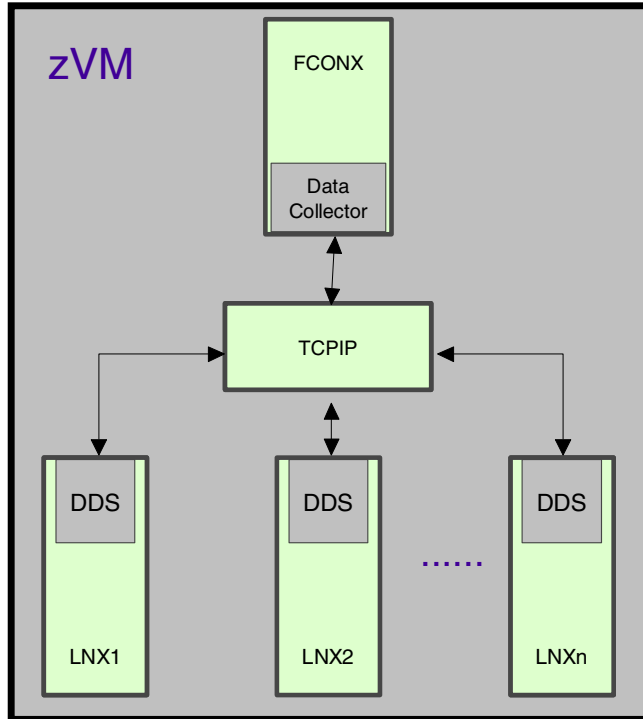


Figure 3-1 Linux performance data collection

3.3 The Distributed Data Server

The Distributed Data Server (DDS) acts to feed Linux monitor data to the Data Collector. DDS is distributed as the server component of the RMF PM for Linux. The RMF PM for Linux home page can be found at:

<http://www.ibm.com/servers/eserver/zseries/zos/rmf/rmfhtmls/pmweb/pm1in.htm>

Refer to this page for prerequisites and installation details.

3.3.1 Download DDS

Download the DDS server from the RMF PM for Linux home page. DDS is distributed in gzipped tar format (file `rmfpms_s390_bit64.tgz` for 64-bit Linux installations).

Important: Be sure to download the *server* component and not the client.

3.3.2 Install DDS on a Linux guest

Extract the `rmfpms_s390_bit64.tgz` tar file to a directory of your choice. The tar file packages all content under the relative directory named `rmfpms`. To extract to the `/opt` directory, issue these commands:

```
cd /opt
tar -zxf rmfpms_s390_bit64.tgz
```

3.3.3 Starting DDS

To start DDS, execute `rmfpms` (found in the `bin` directory of the RMF PM installation). Example 3-1 shows the output generated when starting RMF PM.

Example 3-1 Starting rmfpms

```
# cd /opt/rmfpms/bin/
#
# ./rmfpms start
Creating ~/rmfpms/.rmfpms ...
Starting performance gatherer backends ...
DDSRV: RMF-DDS-Server/Linux-Beta (Mar 1 2002) started.
DDSRV: Functionality Level=1.813
DDSRV: Reading exceptions from gpmexsys.ini and gpmexusr.ini.
DDSRV: Server will now run as a daemon process.
done!
```

3.3.4 Viewing monitored data

If you have installed the RMF PM client on a graphics-enabled workstation, you can view the monitoring data generated by the DDS. Example 3-2 on page 41 shows a sample of the monitoring output.

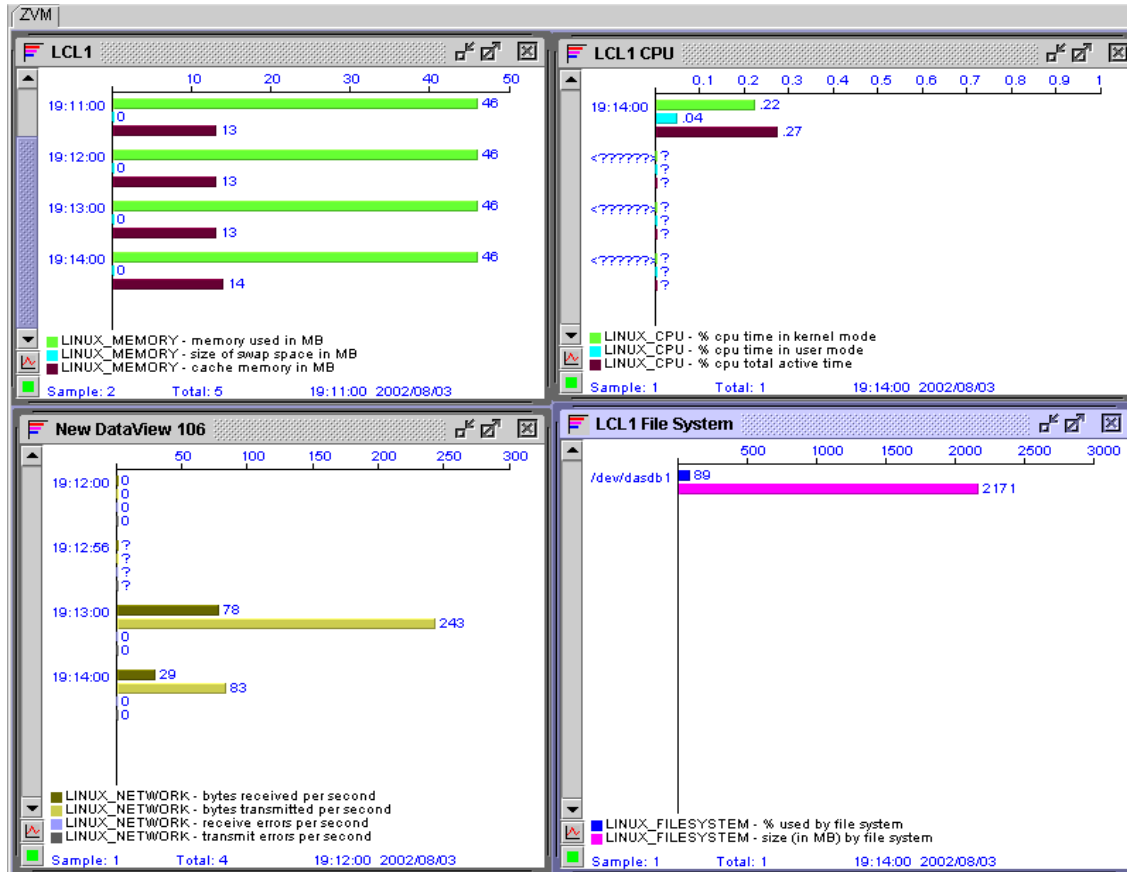


Figure 3-2 Monitoring a Linux guest using the RMF PM client

3.4 Customizing FCON/ESA for monitoring Linux guests

After installing DDS on the Linux guest, some customizing needs to be completed before monitoring can begin.

3.4.1 Preparing the control file

Create a file called FCONX LINUXUSR on the FCONX 201 disk. This control file must contain one record for each Linux system, with the Linux use rid and the IP address of its DDS interface. See Example 3-2, “The FCONX LINUXUSR control file” on page 42 for a sample.

Example 3-2 The FCONX LINUXUSR control file

```
*LINUX-ID  IP ADDRESS FOR DDS INTERFACE
*
LNX5      10.0.3.5:8803
LNX8      10.0.3.8:8803
```

3.4.2 Updating the FCON/ESA profile

To update the FCON/ESA profile, add the following command to the FCONX \$PROFILE, which resides on the FCONX 201 disk.

```
FC MONCOLL LINUXUSR ON
```

where:

FC	Specifies the FC subcommand that allows tailoring of FCON/ESA functions
MONCOLL	Allows control of data collection for performance monitoring
LINUXUSR	Controls the activation and deactivation of the TCP/IP request interface for performance data retrieval from Linux RMF DDS interfaces
ON	Activates the request interface

3.5 The FCON/ESA Linux systems option

In this section we look at monitoring of Linux guests. Performance monitor mode is activated by entering the MONITOR command. Example 3-3 shows the initial performance data selection menu.

Example 3-3 Performance data selection menu

FCX124	Performance Screen Selection	Perf. Monitor
General System Data	I/O Data	History Data (by Time)
1. CPU load and trans.	11. Channel load	31. Graphics selection
2. Storage utilization	12. Control units	32. History data files*
3. Storage subpools	13. I/O device load*	33. Benchmark displays*
4. Priv. operations	14. CP owned disks*	34. Correlation coeff.
5. System counters	15. Cache extend. func.*	35. System summary*
6. CP IUCV services	16. DASD I/O assist	36. Auxiliary storage
7. SPOOL file display*	17. DASD seek distance*	37. CP communications*
8. LPAR data	18. I/O prior. queueing*	38. DASD load
9. Shared segments	19. I/O configuration	39. Minidisk cache*

A. Shared data spaces	1A. I/O config. changes	3A. Paging activity
B. Virt. disks in stor.		3B. Proc. load & config*
C. Transact. statistics	User Data	3C. Logical part. load
D. Monitor data	21. User resource usage*	3D. Response time (all)*
E. Monitor settings	22. User paging load*	3E. RSK data menu*
F. System settings	23. User wait states*	3F. Scheduler queues
G. System configuration	24. User response time*	3G. Scheduler data
	25. Resources/transact.*	3H. SFS/BFS logs menu*
H. Exceptions	26. User communication*	3I. System log
	27. Multitasking users*	3K. TCP/IP data menu*
I. User defined data*	28. User configuration*	3L. User communication
	29. Linux systems*	3M. User wait states

Pointers to related or more detailed performance data can be found on displays marked with an asterisk (*).

Select performance screen with cursor and hit ENTER
Command ==>

Option 29 (**Linux systems***) is used to monitor Linux guests as discussed in 3.6, “FCON/ESA subcommands for Linux guests” on page 43.

Option D (**Monitor Data**) is used to monitor z/VM overall performance as discussed in 3.7, “Monitoring overall z/VM performance” on page 49.

3.6 FCON/ESA subcommands for Linux guests

We concentrate on the subcommands for Linux guests listed in Table 3-1.

Table 3-1 FCON/ESA LINUX subcommands

Subcommand	Description
LINux [<i>guest</i>]	Displays the Linux systems selection menu or a Linux details selection for <i>guest</i>
LXCPU <i>guest</i>	Displays the Linux CPU utilization screen for <i>guest</i>
LXMEM <i>guest</i>	Displays the Linux memory utilization screen for <i>guest</i>
LXNETwrk <i>guest</i>	Displays the Linux network activity screen for <i>guest</i>
LXFILsys <i>guest</i>	Displays the Linux filesystem usage screen for <i>guest</i>

3.6.1 The LINUX subcommand

Use the LINUX subcommand to get a list of the different Linux guests that can be monitored. This list is taken from the FCONX LINUXUSR file. To select a specific Linux guest for performance analysis, either:

- ▶ Place the cursor on the name of the Linux guest and press Enter.
This navigates to the systems selection menu described in 3.6.2, “The Linux systems selection menu” on page 44.
- ▶ Select the Linux guest by entering: LINUX *guest* on the command line.
This fastpath navigation takes you directly to the details selection described in 3.6.3, “The Linux details selection menu” on page 44.

3.6.2 The Linux systems selection menu

In Example 3-4, we show the Linux guest selection menu.

Example 3-4 Selecting a Linux system to monitor

```
FCX223      CPU 2064  SER COECB      Linux Systems      Perf. Monitor

Selectable Linux Systems
LNX1        LNX10     LNX11     LNX12     LNX13     LNX14
LNX15      LNX16     LNX17     LNX18     LNX19     LNX2
LNX20      LNX21     LNX22     LNX23     LNX24     LNX25
LNX26      LNX27     LNX28     LNX29     LNX3      LNX30
LNX4       LNX5      LNX6      LNX7      LNX8      LNX9

Select a system for Linux details
Command ===>
```

To navigate to the details selection menu for a specific guest, either:

- ▶ Place the cursor over the guest and press Enter.
- ▶ Provide the guest name on the command line and press Enter.

3.6.3 The Linux details selection menu

Choosing the LNX8 guest for detailed analysis, we are presented with the report shown in Example 3-5 on page 45.

Example 3-5 The Linux details selection screen

FCX224 CPU 2064 SER COECB Interval 10:10:00 - 10:11:00 Perf. Monitor

Linux Performance Data Selection for System LNX8

System Data

Processes created per second	0.066
Context switches per second	4.933
Apache: Requests per second	...
Bytes per request	...
Busy threads	...
Idle threads	...
404 Errors per minute	...

S	Perform. Reports	Description
_	LXCPU LNX8	CPU utilization details
_	LXMEM LNX8	Memory utilization & activity details
_	LXNETWRK LNX8	Network activity (overall & by device)
_	LXFILSYS LNX8	File system size and utilization

FCXLXD645E GPM0507I DDS could not retrieve valid data for the specified date
Command ==>

Note: If you get the error message: FCXLCD645E GPM0507I DDS could not retrieve valid data for the specified date when navigating to this menu, ensure that the Apache Web server is installed and running on the Linux guest.

This display shows general performance values for the specific Linux system. You can obtain more detailed information on:

► CPU utilization

To get the details on CPU utilization for guest LNX8 (discussed in 3.6.4, “The LXCPU subcommand” on page 46), either:

- Place the cursor on the LXCPU line and press Enter.
- On the command line, enter: LXCPU LNX8

► Memory utilization

To get the details on memory utilization for guest LNX8 (discussed in 3.6.5, “The LXMEM subcommand” on page 47), either:

- Place the cursor on the LXMEM line and press Enter.
- On the command line, enter: LXMEM LNX8

► Network activity

To get the details on network for guest LNX8 (discussed in 3.6.6, “The LXNETWRK subcommand” on page 48), either:

- Place the cursor on the LXNETWRK line and press Enter.
- On the command line, enter: LXNETWRK LNX8

► Filesystem size and activity

To get the details on filesystem size and activity for guest LNX8 (discussed in 3.6.7, “The LXFILESYS subcommand” on page 48), either:

- Place the cursor on the LXFILESYS line and press Enter.
- On the command line, enter: LXFILESYS LNX8

3.6.4 The LXCPU subcommand

Example 3-6 shows output from the command LXCPU LNX8. This shows overall CPU utilization, as perceived by the Linux guest, as well as utilization by processor and by process.

Example 3-6 The Linux CPU utilization screen

```

FCX230      CPU 2064  SER COECB  Interval 10:41:00 - 10:42:00  Perf. Monitor

Linux CPU Utilization for System LNX8

Processor          <--- Percent CPU Utilization --->  <-Accumulated (s)->
                  Total  User  Kernel  Nice  Idle  TotTm  UserTm  KernTm
>>Mean>>          1.23  0.36  0.86   0   98.76  ---    ---    ---
cpu0               1.24  0.36  0.88   0   98.75  ---    ---    ---

Process Name
gpmddsrv.906      0.33  0.26  0.06   ...  ---   761.8  554.6  207.2
procgat.891       0.01  0.01  0       0   ---   1340   80.12  1260
atd.634           0     0     0       0   ---    0.02   ...    0.02
automount.549     0     0     0       0   ---    ...    ...    ...
crond.604         0     0     0       0   ---    6.63   0.53   6.1
dasdgat.879       0     0     0       0   ---   385.5  10.18  375.3
filegat.881       0     0     0       0   ---   48.47  1.41   47.06
init.1            0     0     0       0   ---    3.3    0.27   3.03
keventd.3         0     0     0       0   ---    ...    0       0
kjournald.10      0     0     0       0   ---   36.9   0       36.9
klogd.461         0     0     0       0   ---    0.27   0.15   0.12
kmcheck.2         0     0     0       0   ---    0       0       0
kreclaimd.6       0     0     0       0   ---    ...    0       0
ksoftirqd_CPU0.4  0     0     0       19  ---    5.38   0       5.38
kswapd.5          0     0     0       0   ---   540.2  0       540.2
mingetty.649     0     0     0       0   ---    0.01   0.01   ...

```



```

qethsoftd0014.311      0      0      0      0    ---    ...    ...    ...
sshd.561                0      0      0      0    ---    2.26   2.23   0.03
syslogd.456            0      0      0      0    ---    5.1    1.04   4.06

```

Command ==>

3.6.5 The LXMEM subcommand

Example 3-7 shows the output from the command LXMEM LNX8. This shows overall Linux memory utilization and activity, as perceived by the Linux guest, as well as utilization and activity by process.

Example 3-7 The Linux memory utilization screen

```
FCX229      CPU 2064  SER COECB  Interval 11:15:00 - 11:16:00  Perf. Monitor
```

Linux Memory Util. & Activity Details for System LNX8

```

Total memory size      115MB      Swap space size      140MB
Total memory used      112MB      % Swap space used    4.08%
  Used for buffer      26MB      Swap-in rate         0/s
  Used for shared       0MB      Swap-out rate        0/s
  Used for cache       48MB      Page-in rate         7/s
Total free memory      2MB      Page-out rate        7/s

```

```

          .----- .
          |         |         |         |         |         |
          |<----- Size ----->|<----- Page Fault Rate/s ----->|
          | (Bytes)   (kB)      | Minor  Major  |<--Incl.Children-->|
          | VirtSize  ResidSet  | MinPgFlt MajPgFlt | MinPFltC MajPFltC |
Process Name
gpmddsrv.8755          128664k   1720      ....   ....   | ....   .... |
gpmddsrv.8756          128664k   1720      ....   0      | ....   .... |
gpmddsrv.8943          128664k   1720      0      0      | 0      0      |
gpmddsrv.897           128664k   1720      0      0      | 0      0      |
gpmddsrv.898           128664k   1720      0      0      | 0      0      |
gpmddsrv.899           128664k   1720      0      0      | 0      0      |
gpmddsrv.900           128664k   1720      ....   ....   | ....   .... |
gpmddsrv.901           128664k   1720      ....   ....   | ....   .... |
gpmddsrv.906           128664k   1720      8      ....   | 5      16     |
gpmddsrv.907           128664k   1720      ....   ....   | ....   .... |
gpmddsrv.9130          128664k   1720      0      0      | 0      0      |
gpmddsrv.9131          128664k   1720      0      0      | 0      0      |
gpmddsrv.924           128664k   1720      ....   ....   | ....   .... |
gpmddsrv.9442          128664k   1720      0      0      | 0      0      |

```

Command ==>

3.6.6 The LXNETWRK subcommand

Example 3-8 shows the output from the command LXNETWRK LNX8. This shows network activity, both as total activity and by network device.

Example 3-8 The Linux network activity screen

```
FCX227      CPU 2064  SER C0ECB  Interval 11:22:00 - 11:23:00  Perf. Monitor

Linux Network Activity for System LNX8

Network      <----- Received/s -----> <----- Transmitted/s ----->
Device       RcvPack  RcvByte  RcvError  SndPack  SndByte  SndError
>Total>      0.91     142      0          0.86     624      0
eth0         0.91     142      0          0.86     624      0
eth1         0         0         0          0         0         0
lo           0         0         0          0         0         0

Command ==>
F1=Help  F4=Top  F5=Bot  F7=Bkwd  F8=Fwd  F12=Return
```

3.6.7 The LXFILESYS subcommand

Example 3-9 shows the output from the command LXFILESYS LNX8. This shows overall DASD I/O activity and response time, and both overall and “by filesystem” information on filesystem size and usage. This shows if there are any DASD I/O problems, as well as if you are running out of space on the Linux filesystems.

Example 3-9 The LINUX filesystem usage screen

```
FCX228      CPU 2064  SER C0ECB  Interval 11:25:00 - 11:26:00  Perf. Monitor

Linux Filesystem Usage for System LNX8

DASD I/O Activity
I/O request rate per second          0
I/O response time/request (msec)     nan
I/O response time/sector (msec)     nan

Filesystem      <---- MBytes ----> <-Percent->
Name            Size    Free  %Used %Free
>Total>        2171    679   67.0 32.9
/dev/dasdb1     2171    679   67.0 32.9

Command ==>
```

3.7 Monitoring overall z/VM performance

Table 3-2 lists useful FCON/ESA system and user data subcommands. They can be executed from the MONITOR screen (shown in Example 3-3 on page 42).

Table 3-2 Useful FCON/ESA subcommands

Subcommand	Description
CPU	Shows CPU load, processor status, queue and user status, transactions, and extreme users
STORAGE	Shows central and expanded storage utilization, paging and spooling utilization, minidisk cache utilization and VDISKS
DEVICE or I/O	Shows I/O device rates
DEVICE <i>raddr</i>	Shows general device performance data, path data information, control unit cache, and minidisk load
USER	Shows users CPU load and virtual I/O
USER <i>userid</i>	Shows general performance information for that user and device activity and status

3.7.1 The CPU subcommand

Example 3-10 shows the output from the CPU subcommand.

Example 3-10 Layout of the general CPU screen

```

FCX100      CPU 2064  SER COECB  Interval 15:58:01 - 15:59:01  Perf. Monitor

CPU Load
PROC  %CPU  %CP  %EMU  %WT  %SYS  %SP  %SIC  %LOGLD  %VTOT  %VEMU  REST  ded. User
P00   6    3    2   94   2    0   97    6    not installed  Master
P01   5    2    3   95   1    0   98    5    not installed  Alternate

Total SSCH/RSCH    152/s    Page rate      .1/s    Priv. instruct.  349/s
Virtual I/O rate    2/s      XSTORE paging  .0/s    Diagnose instr.  343/s
Total rel. SHARE    11000    Tot. abs SHARE  3%

Queue Statistics:   Q0    Q1    Q2    Q3    User Status:
VMDBKs in queue    8     1     0    45    # of logged on users  73
VMDBKs loading     0     0     0     0    # of dialled users    0
Eligible VMDBKs    0     0     0     0    # of active users     58
El. VMDBKs loading 0     0     0     0    # of in-queue users   54
Tot. WS (pages)    128807  89    0 621991  % in-Q users in PGWAIT  0
Expansion factor    2     2     2     2    % in-Q users in IOWAIT  97
85% elapsed time   15.99  1.999  15.99  95.95  % elig. (resource wait)  0

```

Transactions	Q-Disp	trivial	non-trv	User Extremes:	
Average users	.7	.2	.2	Max. CPU %	LNX5 .4
Trans. per sec.	.1	.6	.1	Max. VECT %
Av. time (sec)	6.107	.436	1.780	Max. IO/sec	LNX5 .5
UP trans. time		.436	1.780	Max. PGS/s	LNX11 .1
MP trans. time		.000	.000	Max. RESPG	LXDISP 28078
System ITR (trans. per sec. tot. CPU)			17.4	Max. MDCIO
Emul. ITR (trans. per sec. emul. CPU)			.0	Max. XSTORE	LNX30 4841

Command ==>

3.7.2 The STORAGE subcommand

Example 3-11 shows the output from the STORAGE subcommand.

Example 3-11 Layout of the general storage utilization screen

```

FCX103      CPU 2064  SER COECB  Interval 16:10:01 - 16:11:01  Perf. Monitor

Main storage utilization:
Total available          3,072MB
Space for RI0370         0kB
CP resident nucleus     2,772kB
Shared storage          2,752kB
FREE storage pages      7,988kB
FREE stor. subpools     2,708kB
Subpool stor. utilization 92%
Total DPA size          2,033MB
Locked pages            38,344kB
Pageable (DPA - locked) 1,996MB
Storage utilization      147%
Tasks waiting for a frame 6
Tasks waiting for a page 0/s

V=R area:
Size defined             0kB
FREE storage            0kB
V=R recovery area in use ...%
V=R user                .....

Paging / spooling activity:
Page moves <2GB for trans. 0/s
Fast path page-in rate   0/s
Long path page-in rate   0/s
Long path page-out rate  0/s

XSTORE utilization:
Total available          262,144kB
Att. to virt. machines  0kB
Size of CP partition    262,144kB
CP XSTORE utilization    94%
Low threshold for migr. 6,904kB
XSTORE allocation rate  0/s
Average age of XSTORE blks 11939s
Average age at migration ...s

MDCACHE utilization:
Min. size in XSTORE     0kB
Max. size in XSTORE    262,144kB
Ideal size in XSTORE    260,048kB
Act. size in XSTORE     151,444kB
Bias for XSTORE         1.00
Min. size in main stor. 0kB
Max. size in main stor. 3,072MB
Ideal size in main stor. 2,265MB
Act. size in main stor. 402,704kB
Bias for main stor.     1.00
MDCACHE limit / user    322,400kB
Users with MDCACHE inserts 0
MDISK cache read rate   0/s
MDISK cache write rate  ....s
MDISK cache read hit rate 0/s

```

```

Page read rate          0/s      MDISK cache read hit ratio 100%
Page write rate        0/s
Page read blocking factor ...    VDISKs:
Page write blocking factor ...    System limit (blocks)      2092k
Migrate-out blocking factor ...    User limit (blocks)       524288
Paging SSCH rate      0/s      Main store page frames    0
SPOOL read rate       0/s      Expanded stor. pages      0
SPOOL write rate     0/s      Pages on DASD             1
Enter 'FREesub' command for Free Storage Subpool details
Command ==>

```

3.7.3 The DEVICE subcommand

Example 3-12 shows the output from the DEVICE or I/O subcommand.

Example 3-12 Layout of the general I/O device screen

```

FCX108      CPU 2064  SER COECB  Interval 16:14:01 - 16:15:12  Perf. Monitor
. . . . .
<-- Device Descr. --> Mdisk Pa- <-Rate/s-> <----- Time (msec) -----> Req.
Addr Type Label/ID Links ths I/O Avoid Pend Disc Conn Serv Resp CUWt Qued
>> All DASD <<
....      .0 .0 .2 .1 .4 .7 .0 .00
5090 CTCA >RSCS    ... 1 .2 ... .1 2000 .3 2001 2001 .0 .00
3C30 3390-3 TARHF7  0 3 .0 .0 14.3 .0 .3 14.6 14.6 .0 .00
3753 3390-3 430PAG CP 0 2 .1 .0 .2 6.5 2.3 9.0 9.0 .0 .00
099C 3390-3 VMPSL2  0 4 .0 .0 5.6 .0 .5 6.1 6.1 .0 .00
64AD ->641B PAS305  0 4 .0 .0 5.5 .0 .3 5.8 5.8 .0 .00
25EA 3390-3 037RC1  0 4 .0 .0 5.2 .1 .4 5.7 5.7 .0 .00
64AC ->641B PAS305  0 4 .0 .0 5.2 .1 .4 5.7 5.7 .0 .00
3D1C 3390-3 TOTCI4  0 3 .0 .0 5.2 .1 .3 5.6 5.6 .0 .00
64AE ->641B PAS305  0 4 .0 .0 5.2 .0 .3 5.5 5.5 .0 .00
2567 3390-3 TC7CKT  0 4 .0 .0 5.0 .1 .3 5.4 5.4 .0 .00
3BA3 3390-9 LX3BA3  4 3 .9 .1 .6 2.5 2.2 5.3 5.3 .0 .00
3CA1 3390-9 LX3CA1  4 3 .5 .0 .2 1.6 3.2 5.0 5.0 .0 .00
3B6B 3390-3 NW3B6B  0 3 .0 .0 4.5 .1 .3 4.9 4.9 .0 .00
3BA4 3390-9 LX3BA4  6 3 .4 .0 .2 3.1 1.6 4.9 4.9 .0 .00
3B44 3390-3 430PG2 CP 0 3 .1 .0 .2 3.2 .7 4.1 4.1 .0 .00
OCEE 3390-3 TOTDSO  0 4 .0 .0 2.9 .1 .5 3.5 3.5 .0 .00
3731 3390-3 LNXU1R  22 2 .1 .0 .2 .0 3.2 3.4 3.4 .0 .00
64AA ->641C FK641C  0 4 .0 .0 3.1 .0 .3 3.4 3.4 .0 .00
099D 3390-3 VMTDK2  0 4 .0 .0 .3 1.2 1.0 2.5 2.5 .0 .00
256B 3390-3 TRNRSB  0 4 .0 .0 2.1 .0 .4 2.5 2.5 .0 .00
253C 3390-3 037RA1  0 4 .0 .0 1.6 .1 .4 2.1 2.1 .0 .00
6000 3390-3 SAPS01  0 4 .0 .0 .3 .1 1.7 2.1 2.1 .0 .00
6100 3390-3 SAPS06  0 4 .0 .0 .3 .0 1.8 2.1 2.1 .0 .00
67FE ->6700 HG6700  0 2 .0 .0 .3 .1 1.7 2.1 2.1 .0 .00
Select a device for I/O device details
Command ==>

```

Specifying a device

Example 3-13 shows the output from the DEVICE raddr subcommand, where raddr is the real address of some device (in this case, we used 3731).

Example 3-13 Layout of the detailed I/O device screen

```
FCX110      CPU 2064  SER COECB  Interval 16:19:51 - 16:19:52  Perf. Monitor

Detailed Analysis for Device 3731 ( SYSTEM )
Device type : 3390-3      Function pend.: ...ms      Device busy : ...%
VOLSER      : LNXUIR     Disconnected : ...ms      I/O contention: ...%
Nr. of LINKs: 22        Connected    : ...ms      Reserved    : ...%
Last SEEK   : 516       Service time : ...ms      SENSE SSCH  : ...
SSCH rate/s : .0        Response time: ...ms      Recovery SSCH: ...
Avoided/s   : .0        CU queue time: ...ms     Throttle del/s: ...
Status: MDCACHE USED

Path(s) to device 3731:  1B  27  32  3D
Channel path status   :  ON  ON  OFF  OFF

Device          Overall CU-Cache Performance          Split
DIR ADDR VOLSER IO/S %READ %RDHIT %WRHIT ICL/S BYP/S IO/S %READ %RDHIT

  MDISK Extent  Userid  Addr IO/s VSEEK Status  LINK  VIO/s %MDC MDIO/s
+-----+-----+
C   1 - 20     LNX2   1777 .0    0 WR      1   .0 ...   .0 C
C  21 - 25     TCPIP111 0191 .0    0 WR      1   .0 ...   .0 C
C   51 - 75     LNX16  0191 .0    0 WR      1   .0 ...   .0 C
C  276 - 300    LNX24  0191 .0    0 WR      1   .0 ...   .0 C
C  426 - 450    LNX30  0191 .0    0 WR      1   .0 ...   .0 C
C  451 - 475    LNX23  0191 .0    0 WR      1   .0 ...   .0 C
C  476 - 495    MBEATTIE 0191 .0    0 WR      1   .0 ...   .0 C
C  516 - 575    FCONX  0191 .0    0 WR      2   .0 ...   .0 C
C  576 - 587    FCONX  0200 .0    0 WR      2   .0 ...   .0 C
C  588 - 596    TCPMAINT 0298 .0          owner          C

Command ==>
```

3.7.4 The USER subcommand

Example 3-14 on page 53 shows the output from the USER subcommand.

Example 3-14 Layout of the user resource usage screen

```

FCX112      CPU 2064  SER COECB  Interval 16:37:01 - 16:38:01  Perf. Monitor
-----
          . . . . .
          <----- CPU Load -----> Vect <-- Virtual IO/s ---->
          <-Seconds->  T/V  Fac
Userid    %CPU  TCPU  VCPU  Ratio %Vec  Total  DASD  Avoid  UR  Pg/s  User Status
>System<  .1    .1    .1    1.1  .0    .0    .0    .0  .0  .0  ---,---,---
CONFSERV  0    0    0    ...  0    0    0    0  0  0  ESA,---,DORM
DATAMOVE  0    0    0    ...  0    0    0    0  0  0  ESA,---,DORM
DIRMAINT  0    0    0    ...  0    0    0    0  0  0  ESA,---,DORM
DISKACNT  0    0    0    ...  0    0    0    0  0  0  ESA,---,DORM
EREP      0    0    0    ...  0    0    0    0  0  0  ESA,---,DORM
FCONX     .0    .0    .0    ...  .0    .1    .1    .0  .0  .0  ESA,---,DORM
GCS       0    0    0    ...  0    0    0    0  0  0  ESA,---,DORM
LCL101    .2    .1    .1    1.0  .0    .0    .0    .0  .0  .0  EME,CL3,DISP
LCL102    .2    .1    .1    1.0  .0    .0    .0    .0  .0  .0  EME,CL3,DISP
LCL103    .2    .1    .1    1.0  .0    .0    .0    .0  .0  .0  EME,CL3,DISP
LCL104    .2    .1    .1    1.0  .0    .0    .0    .0  .0  .0  EME,CL3,DISP
LCL105    .2    .1    .1    1.0  .0    .0    .0    .0  .0  .0  EME,CL3,DISP
LCL106    .2    .1    .1    1.0  .0    .0    .0    .0  .0  .0  EME,CL3,DISP
LCL107    .2    .1    .1    1.0  .0    .0    .0    .0  .0  .0  EME,CL3,DISP
LCL108    .2    .1    .1    1.0  .0    .0    .0    .0  .0  .0  EME,CL3,DISP
LCL109    .2    .1    .1    1.0  .0    .0    .0    .0  .0  .0  EME,CL3,DISP
LCL110    .2    .1    .1    1.0  .0    .0    .0    .0  .0  .0  EME,CL3,DISP
LCL111    .2    .1    .1    1.0  .0    .0    .0    .0  .0  .0  EME,CL3,DISP
LCL112    .2    .1    .1    1.0  .0    .0    .0    .0  .0  .0  EME,CL3,DISP
LCL113    .2    .1    .1    1.0  .0    .0    .0    .0  .0  .0  EME,CL3,DISP
LCL114    .2    .1    .1    1.0  .0    .0    .0    .0  .0  .0  EME,CL3,DISP
LCL115    .2    .1    .1    1.0  .0    .0    .0    .0  .0  .0  EME,CL3,DISP
LCL116    .2    .1    .1    1.0  .0    .0    .0    .0  .0  .0  EME,CL3,DISP
Select a user for user details or IDLEUSER for a list of idle users
Command ==>

```

Specifying a specific user

Example 3-15 on page 54 shows the output from the USER userid subcommand, where userid is the id you want to look at (we used CONFSERV id).

Example 3-15 Layout of the user resource details screen

FCX115 CPU 2064 SER COECB Interval 16:35:00 - 16:35:57 Perf. Monitor

Detailed data for user LCL101 (sec. user: CONFSERV)

Total CPU :	.1%	Storage def. :	64MB	Page fault rate:	.0/s
Superv. CPU :	.1%	Resident <2GB:	3499	Page read rate :	.0/s
Emulat. CPU :	.0%	Resident >2GB:	5268	Page write rate:	.0/s
VF total :%	Proj. WSET :	10038	Pgs moved >2GB>:	.0/s
VF overhead :%	Reserved pgs :	0	Main > XSTORE :	.0/s
VF emulation:%	Locked pages :	22	XSTORE > main :	.0/s
VF load rate:/s	XSTORE dedic.:	OMB	XSTORE > DASD :	.0/s
I/O rate :	.0/s	XSTORE pages :	302	SPOOL pg reads :	.0/s
DASD IO rate:	.0/s	DASD slots :	3030	SPOOL pg writes:	.0/s
UR I/O rate :	.0/s	IUCV X-fer/s :	.0/s	MDC insert rate:	.0/s
Diag. X'98' :	.0/s	Share :	100	MDC I/O avoided:	.0/s
*BLOCKIO :	.0/s	Max. share :	...		

#I/O active :	4	Active :	100%	PSW wait :	0%	I/O act. :	0%
Stacked blk :	..	Page wait :	0%	CF wait :	0%	Eligible :	0%
Stat.: EME,DSC,IOWT		I/O wait :	100%	Sim. wait:	0%	Runnable :	0%

Data Space Name	Size	Mode	PgRd/s	PgWr/s	XRd/s	XWr/s	Migr/s	Steal/s
BASE	64MB	Priv	.0	.0	.0	.0	.0	.0

Device activity and status:

0009 3215 .0		000C 254R		CL *, EOF	NOH NCNT
000D 254P	CL A, CO 01, NOH NCNT	000E 1403		CL A, CO 01, NOH NCNT	
0190 3390 .0	3750,RR, 107Cyl,--->0	019D 3390	.0	3750,RR, 102Cyl,--->0	
019E 3390 .0	3750,RR, 175Cyl,--->0	0202 3390	.0	6711,RR,3338Cyl,>1927	
0700 OSA .0	INT.MISS	0701 OSA	.0		

Enter 'STOrage Display' for storage details

Command ==>

Networking for Linux on zSeries

In this part we describe networking the penguin colony. We discuss using HiperSockets and z/VM Guest LAN, routing, and traffic shaping. Topics discussed include:

- ▶ An overview of zSeries HiperSockets
- ▶ The Guest LAN feature of z/VM version 4
- ▶ Connectivity in the server farm without routing
- ▶ TCP/IP routing configurations available to the server farm
- ▶ Network redundancy issues for high availability



HiperSockets and z/VM Guest LAN

This chapter describes the Guest LAN feature of z/VM Version 4. It describes the basic emulated-HiperSockets function, as well as the new virtual-QDIO mode introduced with z/VM Version 4.3.

In addition, we look at the HiperSockets function of the zSeries 800 and 900 processors.

4.1 Introduction to HiperSockets

HiperSockets is a feature of the zSeries 800 and 900 processors. It provides a high-speed network connectivity method, which can be used to link operating systems running in logical partitions on a single zSeries processor.

Note: For complete information about HiperSockets, including how to define HiperSockets devices on your zSeries processor, refer to *zSeries Hipersockets*, SG24-6816.

4.1.1 Operating system support

HiperSockets is supported by the following zSeries operating systems:

- ▶ Linux (kernel 2.4, using QDIO network driver)
- ▶ z/VM (Version 4 Release 2 and higher)
- ▶ z/OS (Version 1 Release 2 and higher)
- ▶ z/OS.e

4.1.2 Capabilities

HiperSockets uses the Self-Timed Interconnect (STI) bus of the zSeries processors to perform high-speed data transfer.

Up to four HiperSockets networks can be defined on one zSeries processor. This means that traffic between particular systems can be separated onto different networks.

One reason you might do this is for security considerations: you can define a DMZ HiperSockets network, separate from a private HiperSockets network used for application database access. You could also use separate HiperSockets networks to separate test LPARs from production LPARs.

4.2 Configuring HiperSockets

In order to use HiperSockets, you must perform hardware definition work, as well as configuration in your operating system.

4.2.1 Hardware tasks

You must add definitions for the HiperSockets networks you want to use to the IODF for your zSeries processor. The device type to be used is IQD (for internal-queued-direct).

The HiperSockets devices do not consume real physical channel paths, but do require virtual channel path numbers to be allocated. There are no restrictions on the channel path numbers that can be used for HiperSockets; however, it is recommended that high channel path numbers are used. This means you can keep the numbers used for virtual channels away from the “real” channel numbers allocated to physical channels.

Once the hardware definition work is complete, you can configure your operating systems for access to the HiperSockets network.

4.2.2 z/VM tasks

To use your HiperSockets connection in your z/VM TCP/IP stack, perform these steps:

- ▶ Attach the correct device addresses to the TCPIP user
- ▶ Update the PROFILE TCPIP file
- ▶ Update the configuration of MPROUTE (if used)
- ▶ Restart the TCPIP machine, or use an obeyfile to activate the interface

Attaching the devices

Update the SYSTEM DTCPARMS file to ensure that the devices for your HiperSockets interface are attached to your TCPIP service machine when it is logged on. The system directory can be used for this also, but adding the statements to the SYSTEM DTCPARMS file allows the TCPIP administrator to control the configuration more directly.

To attach the devices while the TCPIP machine is up, you can issue the ATTACH command (from a user with class B privilege) as shown in Example 4-1.

Example 4-1 Attaching HiperSockets devices to a guest

```
attach 7100 to tcpip 7100  
OSA 7100 ATTACHED TO TCPIP 7100  
Ready; T=0.01/0.01 18:26:44  
attach 7101 to tcpip 7101  
OSA 7101 ATTACHED TO TCPIP 7101  
Ready; T=0.01/0.01 18:26:52  
attach 7102 to tcpip 7102  
OSA 7102 ATTACHED TO TCPIP 7102  
Ready; T=0.01/0.01 18:26:58
```

Configure the TCP/IP stack

Once the device addresses are available to TCPIP, you can configure the device to the stack. You use DEVICE and LINK statements for this purpose. An example is shown in Example 4-2.

Example 4-2 TCPIP DEVICE and LINK statements for HiperSockets

```
DEVICE HIPERDEV HIPERS 7100 PORTNAME HIPERP
LINK HIPERLNK QDIOIP HIPERDEV
```

Update MPROUTE

If you are using the MPROUTE service machine to provide dynamic routing, you will need to add your new interface to the configuration. If you do not define it, MPROUTE may take default settings which you do not want. Refer to *z/VM V4R3.0 TCP/IP Level 430 Planning and Customization*, SC24-6019, for information on how to configure MPROUTE.

Activate your new interface

You can reconfigure TCP/IP dynamically, to make use of your new device without having to schedule a restart of TCP/IP. Refer to 6.3.3, “Changing a running z/VM TCP/IP stack” on page 102 for more information on how to do this.

4.2.3 Linux tasks

To add a HiperSockets connection to your Linux guest, perform these steps:

- ▶ Attach the correct device addresses to the Linux guest
- ▶ Update the system configuration
- ▶ Update the configuration of your dynamic routing daemon (if used)
- ▶ Restart the guest, or unload and reload the qeth.o module

Attaching the devices

From a privilege class B user, you can use the ATTACH command to add the devices to your Linux guest dynamically.

Example 4-3 Attaching HiperSockets devices to a guest

```
attach 7104 to lnx6 7100
OSA 7104 ATTACHED TO LNX6 7100
Ready; T=0.01/0.01 18:27:14
attach 7105 to lnx6 7101
OSA 7105 ATTACHED TO LNX6 7101
Ready; T=0.01/0.01 18:27:17
attach 7106 to lnx6 7102
```

Tip: These commands can be issued while a Linux guest is active. However, you will need to reconfigure the channel device layer in order for the network device to be usable.

Also, if you already have the qeth.o driver loaded for other OSA or HiperSockets devices, you will need to schedule a time for the module to be reloaded before your newly added HiperSockets interface can be configured.

To make these changes permanent, you can add the DEDICATE statement to the directory entry for the guest, as shown in Example 4-4.

Example 4-4 Directory statements for Linux guest HiperSockets

```
DEDICATE 7104 7100  
DEDICATE 7105 7101  
DEDICATE 7106 7102
```

If you use DirMaint to manage your user directory, refer to 2.8.4, “Dedicating a device to a userid” on page 33 for instructions on how to perform this operation using DirMaint.

Updating the system configuration

With Linux, you define the connection to HiperSockets interfaces using entries in the /etc/chandev.conf file, which specifies the device addresses to be used for each HiperSockets device you wish to use. Refer to “The /etc/chandev.conf file” on page 74 for more information about how to correctly set up /etc/chandev.conf.

Updating the dynamic routing daemon

Depending on the daemon you use and the configuration method it employs, you will have to add the new interface to its configuration. Refer to the documentation for your chosen routing daemon for more information.

Note: Even if you do not plan on using dynamic routing through this new interface, you may still have to define it to the daemon. You may have to change the way that the interface is advertised to the dynamic routing domain, or ensure that the correct subnet mask or MTU value is used.

Configuring the interface to TCP/IP

Once the channel device layer is correctly configured, an hsiX device is available for configuration to Linux TCP/IP.

Restriction: If you are already using interfaces supported by the `qeth.o` module, a newly created or attached interface will be available for use immediately.

However, if you are changing existing devices, the module will have to be unloaded and reloaded after the channel device layer is reconfigured. This will require all `qeth`-supported network interfaces to be brought down.

You can use the `ifconfig` command or the `ip` command to assign an IP address to the interface and bring it up. If your distribution provides a network configuration program (such as the “Network configuration” component of YaST in a SuSE distribution), you can use that.

The following `ip` command will assign the address 10.0.6.6 to interface `hsi1`.

```
ip addr add 10.0.6.6/24 dev hsi1
```

Reminder: TCP/IP configurations performed on the command line like this are not persistent (that is, they will be lost at the next reboot). Various distributions keep IP configuration information in different places (for example: in Red Hat, the `/etc/sysconfig/network-scripts/ifcfg-*` files; in SuSE, the `/etc/rc.config` file followed by the `SuSEconfig` command).

Update the appropriate files, or use your distribution configuration utility, to make changes permanent.

4.3 Introduction to the Guest LAN feature

The Guest LAN feature provides support for network connection of guest machines. Guest LAN can be used for TCP/IP communication between guests, or between guests and a z/VM TCP/IP service machine.

Important: A VM Guest LAN can only support connections to virtual machines on the same VM system.

4.3.1 Virtual HiperSockets

The original VM Guest LAN feature, introduced with z/VM Version 4.2, is based on the HiperSockets microcode feature provided on zSeries processors.

VM Guest LAN may be used on any processor supported by z/VM; it does not utilize HiperSocket microcode. VM Guest LAN provides support for TCP/IP traffic. With z/VM Version 4.3, IP multicast is supported, as well.

4.3.2 Virtual QDIO

Introduced with z/VM Version 4.3, the QDIO VM Guest LAN simulates a Shared Access Transport Facility (SATF) network interfaced via a QDIO mode adapter such as the OSA-Express Fast Ethernet. As far as the guest operating system is concerned, the QDIO VM Guest LAN acts as an Ethernet network, and functions such as broadcast and multicast are supported.

Important: The name 'DIO Guest LAN is somewhat of a misnomer, since the QDIO interface layer is used for both HiperSockets and the simulated Ethernet Guest LAN.

4.4 VM Guest LAN configuration

To use a VM Guest LAN, you need to do the following:

1. Create a VM Guest LAN segment to act as the network between virtual machines.
2. On each virtual machine to be connected to the VM Guest LAN, create a simulated network interface card (NIC).
3. Attach each virtual machine's simulated NIC to VM Guest LAN segment.
4. For each Linux guest to be connected to the VM Guest LAN segment:
 - a. Start the Linux guest.
 - a. Configure the Linux network interface for the VM Guest LAN.

4.5 Creating a VM Guest LAN segment

Use the **DEFINE LAN** command to define a VM Guest LAN segment. The command syntax is:

```
DEFine LAN lanname [ operands ]
```

where:

<i>lanname</i>	Is a 1- to 8-character alphanumeric name for the VM Guest LAN segment
<i>operands</i>	Define the characteristics of the VM Guest LAN

Operands accepted by the DEFINE LAN command are summarized in Table 4-1.

Table 4-1 Operands to the CP DEFINE LAN command

Operand	Description
OWNERid <i>ownerid</i>	Establishes the owner of the LAN.
TYPE <i>lantype</i>	Specifies the LAN type. Valid <i>lantype</i> values are HIPERsocket for by simulated HiperSockets adapters (the default), or QDIO for use by simulated QDIO adapters.
MAXCONN <i>maxconn</i>	Sets the maximum number of simultaneous adapters allowed to connect to the LAN at <i>maxconn</i> . If specified as INFINITE (the default), no limit is set.
MFS <i>size</i>	Sets the Maximum Frame Size for adapters on the network. This option is only valid for HiperSocket LAN (the default value in this case is 16 K).
UNRESTRICTed	Defines a LAN with no access control - any user may connect to the LAN. The default is to create an unrestricted LAN.
RESTRICTed	Defines a LAN with access control - only authorized users may connect to the LAN.
ACCOUNTing <i>value</i>	Determines if accounting records are created for the LAN - <i>value</i> ON enables accounting, OFF disables accounting.

4.5.1 Establishing a VM Guest LAN owner

The OWNERid operand determines both the owner of the LAN and the lifetime of the LAN. The OWNERid may assume values:

- * The owner is invoker of the DEFINE LAN command (this is the default).
- ownerid* A VM user that owns the LAN.
- SYSTEM The LAN is to be a persistent LAN.

Both the LAN name and its owner uniquely identify a VM Guest LAN (this means you may define two distinct LANs, provided they are owned by different VM users).

4.5.2 Establishing a VM Guest LAN lifetime

The lifetime of a VM Guest LAN is determined by its owner. When assigned an owner VM user at creation, the LAN is defined to be *transient*. If assigned the special owner SYSTEM at creation, the LAN is defined to be *persistent*. If no owner

is specified for the LAN, it will be created as a transient LAN owned by the invoker.

Transient VM Guest LANs

A transient LAN exists as long as the owner VM user is logged on. Once the owner logs off VM, a transient guest LAN will be destroyed, provided no simulated NICs are coupled to the LAN (once all NICs are decoupled, however, the LAN will be destroyed).

Note: Class G users may create a VM Guest LAN segment. However, the owner must be specified as the invoker (thus, all LANs created by a class G user are necessarily transient).

Persistent VM Guest LAN

A persistent LAN can only be destroyed using the DETACH LAN command.

Note: The CP DETACH LAN command can also be used to destroy transient LANs.

Attention: A persistent VM Guest LAN will not persist across a VM IPL. To ensure the definition of the LAN is maintained across IPL, see the procedure outlined in 4.10, “Defining a VM Guest LAN in the VM directory” on page 70.

4.6 Creating a simulated NIC

Use the DEFINE NIC command to create a simulated NIC. The command syntax is:

```
DEFine NIC vdev [ operands ]
```

where:

<i>vdev</i>	Specifies the base virtual device address for the adapter
<i>operands</i>	Define the characteristics of the simulated NIC

Table 4-2 on page 66 lists operands accepted by the DEFINE NIC command.

Table 4-2 Operands to the CP DEFINE LAN command

Operand	Description
HIPERsockets	Creates a simulated HiperSockets NIC.
QDIO	Creates a simulated QDIO NIC
DEVices <i>num</i>	Sets the number of virtual devices associated to the adapter. If omitted, the number of devices will default to 3.

Tip: When creating an interface for a Linux guest, you normally need three devices: one for the read channel; one for the write channel; one for the data channel. In this case, do not specify the DEVices operand; CP will allocate three sequential devices starting with number *vdev*.

4.7 Attaching the simulated NIC to the VM Guest LAN

Use the COUPLE command to attach a simulated NIC to a VM Guest LAN. The syntax of COUPLE command when used to attach a simulated NIC to a VM Guest LAN is:

```
COUPLE vdev TO ownerid lanname
```

where:

vdev Specifies the base virtual device address for the adapter
ownerid Specifies the owner of the VM Guest LAN
lanname The name for the VM Guest LAN

Important: You can only couple a simulated NIC to a VM Guest LAN of the same type. For instance, if the VM Guest LAN is defined to be of type QDIO, you can only couple simulated NICs of type QDIO to that LAN.

4.8 A VM Guest LAN example

We illustrate the process of creating and explicitly destroying a QDIO VM Guest LAN in Example 4-5.

Example 4-5 Steps to create and activate a VM Guest LAN

```
DEFINE LAN QDIOSAMP TYPE QDIO 1
LAN LNX88 QDIOSAMP is created
Ready; T=0.01/0.01 10:00:49
DEFINE NIC 0543 QDIO 2
NIC 0543 is created; devices 0543-0545 defined
```

```
Ready; T=0.01/0.01 10:01:44
COUPLE 0543 TO * QDIOSAMP 3
NIC 0543 is connected to LAN LNX88 QDIOSAMP
Ready; T=0.01/0.01 10:02:45
DETACH LAN QDIOSAMP 4
NIC 0543 is disconnected from LAN LNX88 QDIOSAMP
LAN LNX88 QDIOSAMP is destroyed
Ready; T=0.01/0.01 10:10:59
```

The notes highlighted in Example 4-5 refer to the following points:

1. Define a transient VM Guest LAN of type QDIO. The owner is the command invoker (LNX88).
2. Define a simulated NIC of type QDIO. The virtual device number is 0543.
3. Couple the simulated NIC to the VM Guest LAN.
4. Explicitly destroy the VM Guest LAN.

At the end of this process, the simulated NIC at devices 0543-0545 still exists, and can be COUPLED to another VM Guest LAN. The command DETACH NIC is used to destroy the simulated NIC, as shown in Example 4-6.

Example 4-6 Destroying a simulated NIC

```
DETACH NIC 0543
NIC 0543 is destroyed; devices 0543-0545 detached
Ready; T=0.01/0.01 10:11:09
```

Important: The command DETACH 0543 cannot be used to detach a simulated NIC. This is what happens if you try:

```
DETACH 0543
HCPDTV2793E Device 0543 not detached; DETACH NIC 0543 to remove network
devices 0543-0545
```

Remember, save yourself some effort by issuing DETACH NIC when you want to remove a simulated NIC.

4.9 Restricted VM Guest LANs

The RESTRicted operand on the DEFINE LAN command allows you to restrict VM users permitted to attach simulated NICs to the LAN. This security feature is enabled when the LAN is created. Use the SET LAN command to authorize VM users to attach to the LAN.

4.9.1 Viewing VM Guest LAN attributes

You can view the attributes of a VM Guest LAN using the `QUERY LAN` command. The syntax to get a detailed view of a specific VM Guest LAN is

```
Query LAN lanname DETails
```

Tip: You can get a detailed report on all VM Guest LANs using the variation:

```
Query LAN ALL DETails
```

If *lanname* is omitted, this is the default.

In Example 4-7, we request a detailed report on the VM Guest LAN named QDIOSAMP.

Example 4-7 Using the QUERY LAN command to view VM Guest LAN attributes

QUERY LAN QDIOSAMP DETAILS

```
LAN LNX88 QDIOSAMP      Type: QDIO      Active: 1      MAXCONN: INFINITE
  TRANSIENT  UNRESTRICTED  MFS: 8192     ACCOUNTING: OFF
  Adapter Owner: LNX88  NIC: 0543     Name: UNASSIGNED
Ready; T=0.01/0.01 10:19:42
```

1. The VM Guest LAN is uniquely identified by both the owner and the LAN name (LNX88 QDIOSAMP). It is of type QDIO with one active attached simulated NIC. No limit on the number of connection to the LAN is defined.
2. The VM Guest LAN is transient and unrestricted. Accounting is turned off.

Note: Although the MFS operand is only valid for type HiperSocket LANs, type QDIO LANs have an effective MFS value of 8 K.

3. An adapter owned by user LNX88 is attached to the LAN. The simulated NIC virtual device number is 0543, and the it has no assigned port name (Name: UNASSIGNED).

4.9.2 Changing VM Guest LAN attributes

You can modify the attributes of a VM Guest LAN using the command `SET LAN`. Attributes that may be modified include:

- ▶ The LAN owner

Note: Changing the owner from a VM user to special owner `SYSTEM` will change the LAN from transient to persistent.

- ▶ Accounting for the LAN (the ACCOUNTING operand of DEFINE LAN)
- ▶ The access list of authorized users for a restricted VM Guest LAN

The syntax to authorize a VM user to attach to a restricted LAN is:

```
SET LAN lanname GRANT userid
```

To revoke a VM users authorization, use:

```
SET LAN lanname REVOKE userid
```

Important: Revoking a VM users authorization *will not* break any existing connections by that user to the LAN.

In Example 4-8, we illustrate the creation of a restricted VM Guest LAN of type HiperSockets, and the modification of its access list.

Example 4-8 Authorizations for a restricted VM Guest LAN

```

DEFINE LAN HIPRSAMP OWNER * MAXCONN 4 MFS 64K REST 1
LAN LNX88 HIPRSAMP is created
Ready; T=0.01/0.01 13:05:35
QUERY LAN HIPRSAMP DETAILS 2
LAN LNX88 HIPRSAMP Type: HIPERS Active: 0 MAXCONN: 4
  TRANSIENT RESTRICTED MFS: 65536 ACCOUNTING: OFF
  Authorized userids:
    LNX88
Ready; T=0.01/0.01 13:06:31
SET LAN HIPRSAMP GRANT LNX4 3
Command complete
Ready; T=0.01/0.01 13:07:20
QUERY LAN HIPRSAMP DETAILS 4
LAN LNX88 HIPRSAMP Type: HIPERS Active: 0 MAXCONN: 4
  TRANSIENT RESTRICTED MFS: 65536 ACCOUNTING: OFF
  Authorized userids:
    LNX4 LNX88
Ready; T=0.01/0.01 13:08:10
SET LAN HIPRSAMP REVOKE LNX4 5
Command complete
Ready; T=0.01/0.01 13:48:30
QUERY LAN HIPRSAMP DETAILS 6
LAN LNX88 HIPRSAMP Type: HIPERS Active: 0 MAXCONN: 4
  TRANSIENT RESTRICTED MFS: 65536 ACCOUNTING: OFF
  Authorized userids:
    LNX88
Ready; T=0.01/0.01 13:48:34

```

The notes highlighted in Example 4-8 on page 69 refer to the following points:

1. Create a restricted HiperSockets VM Guest LAN named HIPRSAMP.
2. Query the LAN attributes. Note that the owner is the only authorized VM user.
3. Grant authorization to VM user LNX4.
4. Query the LAN attributes. Note that LNX4 is added to the access list.
5. Revoke authorization for user LNX4.
6. LNX4 is no longer in the access list.

4.10 Defining a VM Guest LAN in the VM directory

Because persistent VM Guest LANs are not maintained across VM IPL, we describe a procedure to automate configuration of the LAN on IPL.

4.10.1 Define the VM Guest LAN in the SYSTEM CONFIG file

Add the DEFINE LAN command to the SYSTEM CONFIG to ensure the LAN is defined on IPL (see *z/VM V4R3.0 CP Planning and Administration*, SC24-6043, for details on the SYSTEM CONFIG file).

4.10.2 Define and couple simulated NICs to the VM Guest LAN

Using the SPECIAL statement in the user directory entry, you can define and couple a simulated NIC to a defined VM Guest LAN using the format:

```
SPEcial vdev type devs owner lanname
```

where:

<i>vdev</i>	Specifies the base virtual device address for the adapter
<i>type</i>	Specifies type of NIC to create (HIPERs or QDIO)
<i>devs</i>	Specifies the number of virtual devices in the NIC
<i>owner</i>	Specifies the owner of the VM Guest LAN
<i>lanname</i>	The name for the VM Guest LAN

Use the SPECIAL statement in the directory entry of all users intended to connect to the VM Guest LAN.

Note: The SPECIAL statement has the effect of executing both the DEFINE NIC and COUPLE commands consecutively.

In Example 4-9, we show a VM directory entry for user LNX4 which defines a QDIO simulated NIC and connects that NIC to a persistent QDIO LAN named PRIVQDIO.

Example 4-9 VM directory entry to define and couple an NIC to a VM Guest LAN

```
*
USER LNX4      xxxx 126M 512M G
  INCLUDE IBMDFLT
  MACHINE XA
  SPECIAL 700 QDIO 3 SYSTEM PRIVQDIO
  IPL 190 PARM AUTOOCR
  MDISK 191 3390 3075 025 430W01 MR READ      WRITE      MULTIPLE
  MDISK 201 3390 0001 0200 LX3755 MR READ      WRITE      MULTIPLE
  MDISK 202 3390 0201 1469 LX3755 MR READ      WRITE      MULTIPLE
  MDISK 203 3390 1870 1469 LX3730 MR READ      WRITE      MULTIPLE
*
```

Attention: Remember to create a simulated NIC of the same type (HiperSocket or QDIO) as the VM Guest LAN to which it is to be coupled.

Using DirMaint to define a simulated NIC

If you use DirMaint to manage the VM directory, you can add the SPECIAL statements to a profile or user directory entry; see 2.7, “Adding directory entries” on page 30 for details. Alternately, you can modify an existing profile or user entry using the DIRM SPECIAL command; see 2.8.9, “Using the SPECIAL DirMaint command” on page 34 for details.

4.10.3 Automating connections to a VM Guest LAN

As an alternative to using a SPECIAL statement in the VM directory, you can have the VM user who intends to connect to the VM Guest LAN manage its own network connections.

In Example 4-10, we show a REXX exec which automates connecting to a VM Guest LAN and IPLing a Linux guest. The main features of the script are:

- ▶ The VM creates a simulated QDIO NIC using virtual devices 700, 701, and 702.
- ▶ The NIC is coupled to a persistent VM Guest LAN named PRIVQDIO.
- ▶ The Linux boot filesystem resides on the user’s 202 minidisk.

Example 4-10 Automating connections to a VM Guest LAN from the VM user

```
/*-----
sample startup procedure for linux guests under z/VM
-----*/
/*-----
clear environment and define simulated NIC first
-----*/
'pipe CP DETACH NIC 700          ! stem det_result.'
'pipe CP DEFINE NIC 700 QDIO DEVICES 3 ! stem nic_result.'

if subword( nic_result.1, 1, 1) <> 'NIC'
then do
  /*-----
  any error condition are ocured, inform any other components
  -----*/
  say 'ERROR: defining simulated NIC'
  exit( 1 )
end
else say 'NIC 700 successfully defined'
/*-----
connect to desired VLAN
-----*/
'pipe CP COUPLE 700 TO SYSTEM PRIVQDIO ! stem couple_result.'

if subword( couple_result.1, 1, 1) <> 'NIC'
then do
  /*-----
  any error condition are ocured, inform any other components
  -----*/
  say 'ERROR: coupling NIC to VLAN'
  exit( 2 )
end
else say 'NIC 700 successfully connected to PRIVQDIO'
/*-----
linux startup
-----*/
say 'unattended start for linux/390 will executed in 5 seconds'
say '      if not desired press <ENTER> to abort ...'

address cms 'sleep 5 sec attn' /* wait for five seconds for manual
                                startup or something like that */

if RC = 0
then do
  address cms 'cp i 202 clear'
end
else say 'start of linux/390 aborted'

exit
```

4.11 Configuring a VM Guest LAN in a Linux guest

Once the VM Guest LAN segment is defined and a simulated NIC is coupled to that LAN in the virtual machine of a Linux guest, the Linux network interface can then be configured and enabled. The steps to configure the network interface are:

1. Load the network interface device driver.
2. Provide the TCP/IP configuration for the interface.
3. Provide TCP/IP routing information for the device.

Important: The 2.4 Linux kernel is required in order to use a VM Guest LAN; the HiperSocket interface is not supported in the 2.2 Linux kernel.

4.11.1 A word about network device drivers

The device drivers required to support VM Guest LAN are distributed by IBM in object code only (OCO) form. These drivers may be included as part of your Linux for zSeries distribution (if you use SuSE, for example).

However, these drivers are not included as part of the Red Hat Linux for zSeries distribution. In this case, you will need to download and install the device drivers from the IBM Developerworks Web site:

<http://www.ibm.com/developerworks/opensource/linux390/index.shtml>

Follow the OCOs for RedHat link.

4.11.2 Loading the Linux network interface device driver

In order to configure the network interface for a VM Guest LAN in a Linux guest, you first need to load the interface device drivers. The drivers are normally found in the `/lib/modules/kernel-version/net` directory (where *kernel-version* is the identifier of the booted Linux kernel).

To install the device drivers for a VM Guest LAN:

1. Update the `/etc/chandev.conf` file to add definitions for the physical devices.
2. Update the `/etc/modules.conf` file associate a driver to an interface.
3. Load the device driver using the **modprobe** command.

Note: Details on installation of device drivers on Linux for zSeries can be found in *Linux for zSeries and S/390 Device Drivers and Installation Commands*, LNUX-1303 available at:

<http://www.ibm.com/developerworks/opensource/linux390/docu/lnuxdd01.pdf>

Interface naming conventions

Interfaces names formed from a base name (indicative of the type of network device) and a device number (indicative of the number of devices of that type). For VM Guest LAN interfaces, Linux on zSeries will generate a base name for the interface according to the convention:

eth	QDIO VM Guest LAN interfaces
hsi	HiperSockets VM Guest LAN interfaces

The `/etc/chandev.conf` file

Network interface devices must be described in the `/etc/chandev.conf` file. In Example 4-11, we show the `chandev` configuration necessary to configure a QDIO VM Guest LAN (`eth0`) and a HiperSockets VM Guest LAN (`hsi0`).

Important: When configuring the channel device layer, the device names that the interfaces will be known to IP and to the `qeth` driver are not specified. Instead, names used internally by the channel device layer (derived from the values defined for the channel type) are used. You can display the `/proc/chandev` pseudo-file for information that will tell you which `chandev` name refers to which device name.

Example 4-11 The `/etc/chandev.conf` file describing two VM Guest LAN interfaces

```
noauto;qeth0,0x0700,0x0701,0x0702;addparms,0x10,0x0700,0x0702,portname:NIC0700
noauto;qeth1,0x0500,0x0501,0x0502
```

The fields and the meanings listed in Example 4-11 are described here:

- ▶ `noauto`

No auto-detection will occur when probing for the device:

- ▶ `qeth0,0x0700,0x0701,0x0702`

The first interface to be handled by the `qeth` driver, `qeth0`, uses read-control subchannel address 0700, write-control subchannel address 0701, and data subchannel address 0702.

Important: Specify the three sequentially numbered virtual device addresses in VM user's simulated NIC created for this LAN (see 4.6, "Creating a simulated NIC" on page 65).

- ▶ `addparms,0x10,0x0700,0x0702,portname:NIC0700`

Identifies the device as a ‘qeth’ device (the 0x10 code) using devices ranging from 0x0700 (the low device number) to 0x0702 (the high device number). The portname used to identify the network the device will connect to is NIC0700.

Important: For the portname parameter, care is required. On a real OSA-Express adapter, you would need to use the real portname that is specified for use on all the systems that share the OSA port. For VM Guest LAN, you can use the VM Guest LAN segment name used when creating the LAN with the `DEFINE LAN` command (see 4.5, “Creating a VM Guest LAN segment” on page 63), or any name that suits your system conventions.

For HiperSockets (real or VM simulated), we found you could safely leave the portname off. However, doing so caused problems if the interface had to be reconfigured later. We recommend following the same procedure as for VM Guest LAN.

- ▶ `qeth1,0x0500,0x0501,0x0502`

The second interface handled by the qeth driver, qeth1, uses read-control subchannel address 0500, write-control subchannel address 0501, and data subchannel address 0502.

For this interface, we decided (against our own advice) to not specify a portname. Therefore, we did not specify the `add_parms` section on this configuration line.

Update the `/etc/modules.conf` file

In the `/etc/modules.conf` file, add a line for each interface instructing the kernel to load the appropriate qdio device driver for that interface. In Example 4-12, we show the entries which apply to the interfaces described in “The `/etc/chandev.conf` file” on page 74.

Example 4-12 The `/etc/modules.conf` file to describe VM Guest LAN interfaces

```
alias eth0 qeth
alias hsi0 qeth
```

Adding these alias entries to `/etc/modules.conf` makes it unnecessary to manually install the device driver.

When the kernel is instructed to activate a network device called eth0, it will first check to see if any already-active device drivers can handle the device being requested. If not, it will probe for a module called eth0.

The alias entry basically says “if you’re looking for module eth0, it’s really called qeth”. The kernel will load qeth.o instead. Internally, a dependency is known between qeth.o and qdio.o, so the kernel will first load qdio.o (if it’s not already loaded), and then load qeth.o.

4.11.3 Configuring the network interface

Once the interface is available to Linux, it can be configured to IP just like any other interface. In “Configuring the interface to TCP/IP” on page 61, we provide information about configuring IP on a HiperSockets interface; the same applies to VM Guest LAN.



TCP/IP direct connection

This chapter describes how network connectivity can be implemented in Large Scale implementations on zSeries without the use of internal routing facilities.

Many large scale Linux installations will employ some type of “virtual routing” configuration. This type of configuration is discussed in Chapter 6, “TCP/IP routing” on page 93.

However, depending on the size of your configuration and the type of connectivity you require, you may be able to deploy your Linux guests without using routing at all.

5.1 Introduction

Before you decide to build a routing infrastructure in your large scale Linux installation, you need to determine whether you need to route at all. For smaller numbers of Linux guests, you may be able to set up your guests with direct connection to the network.

You will need to keep several issues in mind, including:

- ▶ The total number of Linux guests you expect to run in your z/VM system
- ▶ The type of network interface your installation uses between your zSeries host and your network “at-large”
- ▶ Capabilities of the network interface that might work well with configuration or technologies used in your network
- ▶ Availability of “routable” IP addresses (IP addresses that can be directly accessible from the rest of your network, or the public Internet, if required)

5.1.1 Number of Linux guests

The network interface your installation uses will, to a large extent, determine whether you can support all your Linux guests using direct connection. For example, an OSA-2 interface can only support 16 entries in its OSA Address Table per port, which means that each port can be shared by only 8 TCP/IP stacks. The OSA-Express interface provides a very high level of “shareability”, with up to 4096 IP addresses per port available¹.

Recommendation: If you expect that the number of guests you will support under z/VM will grow beyond 50 per OSA-Express port, or you are not using the OSA-Express for network connectivity, we recommend that you build your connectivity using routing rather than direct access.

5.2 OSA port sharing

The easiest way to directly connect Linux guests to the network is by sharing a port (or ports) on an OSA-Express interface. Each stack sharing the OSA port is directly attached to the network, with an individual LAN IP address.

¹ On z800/z900 servers only, up to 512 addresses on G5/G6 servers.

Restriction: If you have OSA-2 interfaces instead of OSA-Express, be aware that the OSA-2 can only support 16 IP addresses per port. This limits the number of guests (or LPARs) that can share the port. We recommend that you use a routing infrastructure instead of direct access when your network interfaces are OSA-2.

For the rest of this section, we will discuss OSA-Express only, so where we refer to OSA, we mean OSA-Express (unless otherwise specified).

Under z/VM you can dedicate the required sets of device addresses to your Linux guests, giving them direct access to the OSA hardware.

A possible port sharing configuration would look like the diagram shown in Figure 5-1.

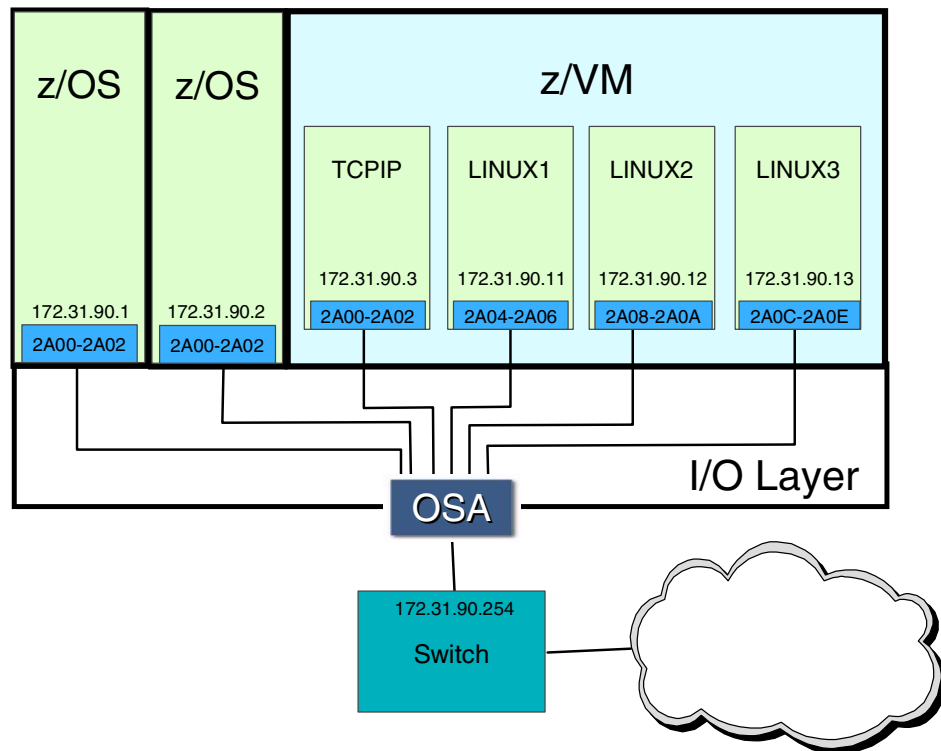


Figure 5-1 OSA-Express port sharing

In this example, two z/OS LPARs, a z/VM TCP/IP stack, and three Linux guests under z/VM are all sharing an OSA port. The network to which the OSA is

attached is the 172.31.90.0/24 network, and all of the IP stacks have addresses on that network. No routing within the zSeries complex is required to reach other LAN-attached services on the same LAN.

The device addresses in this diagram show another feature of a sharing configuration. The device addresses used by the two z/OS LPARs, and by z/VM TCP/IP, are all the same. Using the Enhanced Multiple Image Facility (EMIF), a device can be shared across multiple LPARs using the same device address². Within a single LPAR, however, device addresses must be unique. This is shown in the device addresses used by the Linux guests.

5.2.1 Hardware definition

In the IODF for your system, ensure that there are sufficient device addresses defined against your OSA CHPID for the number of guests you want to share the OSA. These device addresses will all be defined to the LPAR in which your z/VM system will run.

You will then need to keep track of the device numbers you allocate to each of your guests. Your system naming convention and resource allocation standards will provide for this.

Note: The number of guests you can support will vary depending on whether you use LCS or QDIO mode for accessing the OSA. LCS requires only two device addresses per guest, while QDIO requires three, so in theory you can support more guests using LCS mode than QDIO.

However, in LCS mode, the advanced features of the OSA-Express such as IP Assist and dynamic IP addressing are not available, seriously affecting the dynamic configurability of your Linux installation. Also, LCS mode is not available on the OSA-Express Gigabit Ethernet feature.

We recommend that you run OSA-Express interfaces in QDIO mode in large-scale Linux installations under z/VM. If this does not give you the number of direct connections you require, change to a routing design rather than running your OSA-Express adapter in LCS mode.

Defining the OSA as reconfigurable

An OSA CHPID can be defined so that it can be used by only one LPAR out of a number of different LPARs at a time. This can be used for an OSA that is used normally by a test system, but you need to ability to move it to a production system if required. Such a CHPID is defined as *reconfigurable*. The maximum

² IOCP uses the tuple of unit address and LPAR number to obtain uniqueness.

device address capacity of the OSA is available to each LPAR, since only one LPAR can use the OSA at a time.

Defining the OSA as shareable

If you plan to share your OSA with other operating systems in other LPARs, you would need to define the OSA CHPID as “shareable”. This has great impact on how the device allocation is done.

The candidate list is the list of LPARs that can have access to the CHPID. LPARs not defined in the candidate list cannot configure the CHPID online.

A limit of 240 device addresses can be configured on an OSA CHPID. When you define the CHPID as shareable, this limit applies to the total number of devices defined across all LPARs in the candidate list for the CHPID.

You will not be able to configure maximum capacity on any one LPAR, since you must configure devices against all LPARs you place in the candidate list. The 240 devices for the OSA CHPID must be distributed across all systems in the candidate list.

If you want maximum capacity on the LPAR used for your z/VM system, you will need to take care when defining such a shareable CHPID. For other LPARs, define only the actual number of device addresses you need (on a z/OS LPAR, for instance, define only three devices). Define the rest of the devices on your z/VM LPAR if you want to make them available for Linux guests.

Important: If you want the ability to move an OSA CHPID from LPAR to LPAR for recovery purposes, defining the CHPID as shareable is not a good idea. Doing so would reduce the maximum usable capacity of the OSA by at least half, since you would need to provide the same definitions on both the production and backup LPARs (to provide access for all the Linux guests you are recovering). Defining the CHPID as reconfigurable is better in this case.

If you must share the OSA with another LPAR as well as using it for recovery, then defining the CHPID shareable is the only solution. You would need to take into account the reduced capacity when designing your Linux guest connectivity.

5.2.2 Advantages sharing OSA-Express in QDIO mode

Apart from an administrative saving in not having to maintain virtual routing guests, there are a number of advantages that OSA port sharing can provide.

IP Assist function

The OSA-Express IP Assist function offloads the following protocol functions from the IP stack for processing in the adapter:

- ▶ Broadcast filtering
- ▶ ARP processing
- ▶ Building MAC and LLC headers

Offloaded processing can reduce the cycles consumed by your Linux guests for network traffic, giving a slight performance improvement. In a single guest the effect might not be significant, but in a z/VM LPAR with Linux guests generating a moderate-to-high volume of network traffic, we would expect an overall saving.

Stack-to-stack routing

When IP stacks are sharing an OSA, and one stack sends traffic destined to another stack sharing the same OSA port, the OSA sends the traffic directly to the destination stack. Network traffic processed in this way does not appear on the LAN. This action takes place transparently to the sending stack.

This can provide a performance benefit when a number of guests send network traffic to each other, as it will be sent directly between the stacks rather than out across the network.

5.2.3 Issues sharing OSA-Express in QDIO mode

If you are considering an OSA-sharing configuration for your large scale Linux installation, you need to be aware of some points regarding this connection method.

How many stacks can share an OSA

A Washington Systems Center Flash (document number Flash10144) describes some considerations to be observed when sharing an OSA Express port defined in QDIO mode. It details the number of LPARs or guests that can share an OSA (depending on the hardware type), and some zSeries code level dependencies.

Note: The full text of the WSC Flash can be found at:

<http://www-1.ibm.com/support/techdocs/atsmastr.nsf/PubAllNum/Flash10144>

In summary, a z900 with up-to-date microcode, or any z800, can support up to 80 IP stacks on an OSA-Express port in QDIO mode. Other combinations can support considerably fewer than that.

What is not clear is the level of performance that can be expected when the OSA is configured to its peak. Performance testing to date has focussed on throughput from a single stack, rather than saturation across a fully-configured OSA.

Broadcast traffic

Having your Linux guests visible directly on your LAN may expose them to unwanted LAN broadcast traffic. This will create CPU load, preventing your idle Linux guests from reaching a storage usage pattern that reflects their actual useful work (rather than processing of spurious network traffic).

The broadcast filtering component of the IP Assist function in the OSA-Express might help in this regard, but it is still almost guaranteed that your Linux guests will be presented with more LAN traffic than they would behind a router.

5.3 IEEE 802.1Q VLAN support

Introduced with the IBM *@server* zSeries 900 Turbo, Linux support for IEEE 802.1Q VLANs allows Linux guests to participate in VLAN infrastructure. You can extend your VLAN configuration to include your Linux guests, so that no routing is necessary to reach workstations on the LAN.

Important: You may have noticed that in previous discussions on HiperSockets and VM Guest LAN, we avoided the use of the term Virtual LAN to describe these. This is because, in general networking terms, Virtual LAN (or VLAN) refers specifically to IEEE 802.1Q VLANs.

Therefore, to avoid confusion (especially when talking with your friendly networking folks), do not refer to zSeries and z/VM “virtual” networking technologies as VLANs.

5.3.1 How VLANs work

IEEE 802.1Q VLANs operate by defining switch ports as members of virtual LANs. Ports used to attach non-VLAN-aware equipment such as end-stations are called “access” ports, while ports used to connect to other switches are known as “trunk” ports. Network frames generated by VLAN-aware equipment are marked with a “tag”, which identifies the VLAN that the frame belongs to.

Access ports are defined to belong to a single VLAN. All frames received from stations attached to access ports are ‘tagged’ with the same VLAN ID (VID). Trunk ports, on the other hand, can be defined to multiple VIDs, because the VLAN-aware equipment knows how to handle frames with different VIDs. Figure 5-2 on page 84 shows a logical diagram of a VLAN environment, showing two switches and a number of VLANs.

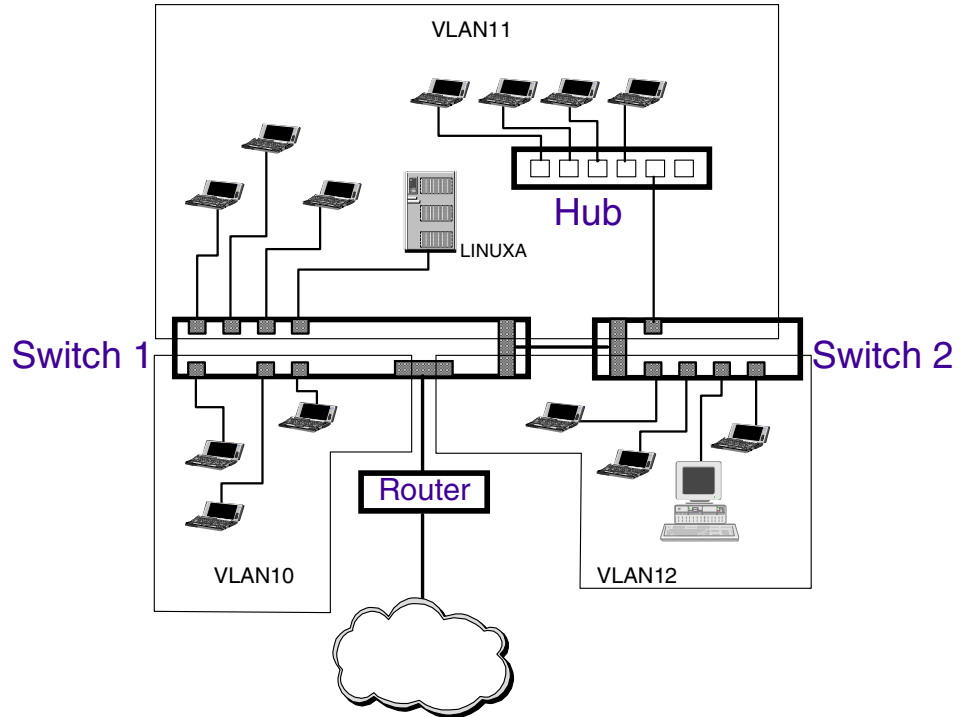


Figure 5-2 VLAN example

In this example network, VLAN 10 only exists in switch 1, because the trunk port to switch 2 is not a member of VLAN 10. VLANs 11 and 12 span the two switches, because the trunk ports in each switch are members of both VLANs.

There is a single VLAN-aware router connected to switch 1, which provides access to an external network to users in VLANs 10 and 12. (The trunk port to which the router is attached is defined to VLANs 10 and 12, not to VLAN 11. Therefore, no stations in VLAN 11 can reach the router). Hub 1 is a non-VLAN-aware device, so the port in switch 2 that it connects to is an access port.

In this example, we can see two of the reasons that VLANs are generally used:

- ▶ Staff in different physical locations retain common access to resources.

The server LINUXA is used by staff in both buildings. By defining these end stations to the same VLAN, no additional configuration or equipment is required for both locations to access the server, while at the same time ensuring that other staff do not get access.

- ▶ Consolidation of resource access.

The external network has to be accessed by different staff in both buildings. Extending VLAN 12 across to the router port in switch 1 (and configuring the router correctly) saves having to provide an additional link to the external network or the router from building 2.

Broadcast in VLANs

All ports which are members of the same VLAN, including trunk ports, will operate as if they were part of the same physical network. When a multicast or broadcast frame is received from a station in a VLAN environment, the switch transmits the packet on all ports (both trunk and access ports) belonging to the same VLAN.

The only difference between the trunk and access port in this case is that the frame transmitted onto the trunk port will have the VLAN tag intact (so that the VLAN-aware equipment at the other end of the link will know how to handle it).

VLAN isolation

An important point about VLANs in general is that they provide isolation. VLANs behave like separate physical networks, even though they may be contained within the same switch.

In order for devices in different VLANs to communicate, TCP/IP routing must occur. In the network shown in Figure 5-2 on page 84, stations in VLAN 10 and VLAN 11 cannot communicate, because there is no routing path between the two VLANs. Stations in VLANs 10 and 12 can communicate, as long as the router they are attached to is configured appropriately.

5.3.2 VLANs on Linux for zSeries

Linux VLAN support will be of most use when you have servers in your Linux environment that are providing service to particular parts of your network that are already configured into different VLANs.

Note: VLAN support was added to the Linux kernel at version 2.4.14. If your Linux guests are not running this level or higher, you will require an updated kernel if you want to use VLANs. Contact your Linux vendor for more information.

A Linux under z/VM network using VLANs might look like the diagram shown in Figure 5-3 on page 86.

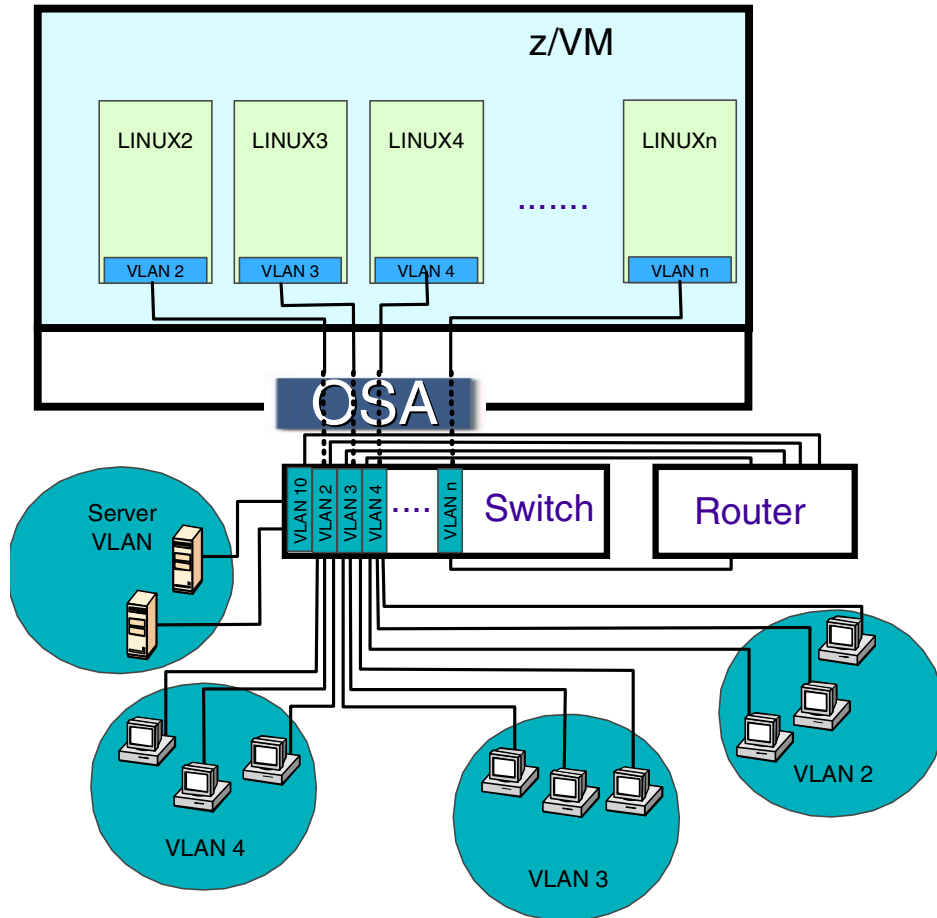


Figure 5-3 Linux guests using VLAN support.

In this diagram, the Linux guests have a logical association with certain groups of stations in the network (they might be Samba file-and-print servers for certain areas of the organization, for instance). Using VLANs allows those Linux guests to be connected to their client networks without the overhead of routing, and with the advantage of behaving as if they were on the same physical network.

In “VLAN isolation” on page 85, we stated that VLANs provide isolation between networks. Figure 5-3 shows a router with connections to the different VLANs to provide access between the client PCs and the servers in VLAN 10. This can also be done with a router which is VLAN-aware, in which case only one physical connection to the switch is needed.

Note: Often, a separate router is not required either. Most switch vendors provide ways to include routing function in their switches to allow traffic between VLANs. For example, the Cisco Catalyst 3550 series switches run Cisco Internetwork Operating System (IOS) routing code as well as Catalyst switch code to provide routing function in addition to switching. The server and core grade Cisco Catalyst switches provide optional modules that can be added to the switch to provide routing function.

Logically, the routing function is still separate from the switching function, even if a separate physical device is not used.

5.3.3 Sharing an OSA-Express when using VLANs

At the time of writing, Linux is the only zSeries operating system to support VLAN tagging using the OSA-Express interface. It is possible, however, to share an OSA between Linux and other zSeries operating systems even though they do not provide VLAN support.

Figure 5-4 on page 88 shows a network with z/VM and z/OS systems sharing an OSA with Linux guests using VLANs.

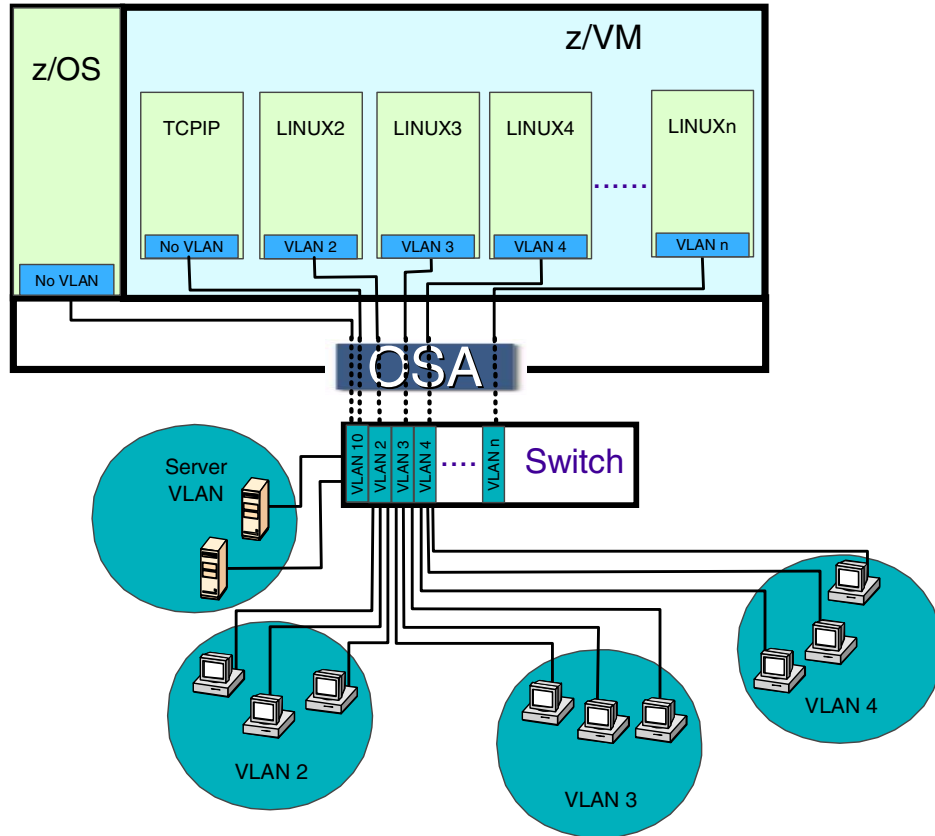


Figure 5-4 Sharing OSA between VLAN-capable and non-VLAN capable OSes

Usually, all frames sent on a trunk link are VLAN-tagged. In Figure 5-4, however, we have a z/VM TCP/IP stack and a z/OS Communications Server stack sharing the OSA with the Linux guests using VLANs. Since neither z/VM nor z/OS support IEEE 802.1Q VLANs, the traffic generated by these stacks will not have VLAN tagging.

According to the 802.1Q standard, this is a valid configuration. Rather than a trunk port, the switch port the OSA connects to is called a *hybrid port*, since it must behave like both an access port and a trunk port. A hybrid port acts like a trunk port for frames that carry a VLAN tag, but the switch tags frames without a VLAN tag with a port VID (like it would with an access port).

Attention: Before designing a solution based a configuration like this, you should consult your switch vendor to verify that their equipment will support VLAN-unaware equipment on a trunk port (in other words, they understand a hybrid port). If they do not, you will not be able to share an OSA port between Linux guests using VLAN trunking and any other guests or LPARs.

5.3.4 Configuring VLANs in Linux

The `vconfig` program is used to add a VLAN configuration to an existing defined adapter. Example 5-1 shows the full syntax of the `vconfig` command.

Example 5-1 The syntax of the vconfig program

```
Usage: add          [interface-name] [vlan_id]
       rem          [vlan-name]
       set_flag     [interface-name] [flag-num]      [0 | 1]
       set_egress_map [vlan-name]      [skb_priority] [vlan_qos]
       set_ingress_map [vlan-name]      [skb_priority] [vlan_qos]
       set_name_type [name-type]
```

- * The `[interface-name]` is the name of the ethernet card that hosts the VLAN you are talking about.
 - * The `vlan_id` is the identifier (0-4095) of the VLAN you are operating on.
 - * `skb_priority` is the priority in the socket buffer (`sk_buff`).
 - * `vlan_qos` is the 3 bit priority in the VLAN header
 - * `name-type`: `VLAN_PLUS_VID` (`vlan0005`), `VLAN_PLUS_VID_NO_PAD` (`vlan5`), `DEV_PLUS_VID` (`eth0.0005`), `DEV_PLUS_VID_NO_PAD` (`eth0.5`)
 - * `bind-type`: `PER_DEVICE` # Allows vlan 5 on `eth0` and `eth1` to be unique.
`PER_KERNEL` # Forces vlan 5 to be unique across all devices.
 - * `FLAGS`: `1 REORDER_HDR` When this is set, the VLAN device will move the ethernet header around to make it look exactly like a real ethernet device. This may help programs such as DHCPd which read the raw ethernet packet and make assumptions about the location of bytes. If you don't need it, don't turn it on, because there will be at least a small performance degradation. Default is OFF.
-

Attention: Something else to confirm with your Linux vendor is the availability of the `vconfig` program. On our SuSE system (based on the 2.4.17 kernel), we had the `8021q.o` module available, but no `vconfig`. We experimented with building the program from source, but hopefully our Linux distributors will include `vconfig` soon.

As an example, to add a connection to VLAN 100 via the OSA Express at eth0, enter the following:

```
vconfig add eth0 100
```

This creates a new VLAN interface called eth0.100 which you can now configure as normal using **ifconfig**.

```
ifconfig eth0.100 192.168.100.1 netmask 255.255.255.0 up
```

Note: This example uses the DEV_PLUS_VID_NO_PAD format of the VLAN name, which is implied by the examples in *Linux for zSeries and S/390 Device Drivers and Installation Commands*, LNUX-1303. Other examples you may see in different documentation may show different formats of the VLAN name.

Startup configuration

At this time, configuring VLANs is a manual process that must be scripted to take place at your Linux guests' bootup. Expect that as VLAN usage grows, distributors will include VLAN boot-time configuration in their network scripts.

5.3.5 Infrastructure guests in a VLAN network

If you have guests that are part of your network infrastructure (such as DNS or LDAP servers), you have two choices for connecting these in a VLAN environment:

- ▶ Single interface on a “backbone” or infrastructure VLAN.
This makes the configuration of the server Linux guest easier, however, your clients attached to VLANs will have to pass through some kind of router (either a physical router in the network, or a virtual router in your penguin colony) to reach it.
- ▶ Multiple interfaces, one for each VLAN.
This allows your clients to access your server without routing. However, the configuration on the Linux server guest will be more complex, and some services will require careful configuration (Samba using WINS, for example, due to limitations with WINS name resolution).

The multiple interface option will allow you to provide the most direct path to all of your clients, but it will require some careful design and configuration. If you choose this option, we suggest using a virtual IP addressing method like that introduced in 7.2.3, “Virtual IP addresses” on page 136.

If the amount of traffic is low, the overhead of routing to these Linux server guests will not be a burden, and the “infrastructure VLAN” approach will provide the best solution.

Tip: If you do use multi-homed infrastructure servers, turn off IP forwarding. Your network people will probably demand it anyway, and unless you really want these guests to act as backup routers (and configure the network to suit), it's one less thing to worry about if you have routing problems in your network.



TCP/IP routing

This chapter describes how routing can be implemented in Large Scale implementations on zSeries.

When you deploy a large number of Linux guests under z/VM, you have choices as to how the network connectivity to those guests will be arranged. zSeries gives you a number of options, each of which can provide benefits in certain configurations or for certain applications.

6.1 Planning for routing

It is critical to have a design for your routing environment planned in advance of implementation. If your design is not well planned, you will encounter problems later in the life of your project through growth and capacity problems. In fact, poor planning can lead to problems during initial implementation.

Some of the issues that your plan should cover include:

- ▶ Connectivity method

The technology and topology you will use to connect the Linux guests to each other, and to infrastructure services.

- ▶ Isolation

Ideally, the design will ensure that Linux guests have connectivity only to sections of the environment that they need to. For example, testing guests should not be able to get connectivity to production guests.

- ▶ Address allocation

The method you use to allocate IP addresses will be dependent upon the connection methods you use, but should be flexible enough to provide some opportunity for growth in the environment.

- ▶ Traffic shaping

Depending on your requirements, heavily utilized guests should not be able to dominate the network and prevent other guests from providing service.

- ▶ Dynamic routing

Using a dynamic routing protocol is largely unnecessary if you have simple connectivity (Linux guests with single network interfaces and no virtual IP addressing, for instance). As the complexity of your configuration increases, however, dynamic routing can provide benefits for redundancy and failover. However, there can be considerable system overhead in running dynamic routing.

In the following sections, we examine these issues in more detail.

6.1.1 Connectivity method

zSeries provides a number of technologies for network connectivity. Generally, there are exterior connection methods (which provide connection to servers and users outside the zSeries complex) and interior connection methods (which are used to connect systems inside the zSeries complex).

Exterior connection methods supported by zSeries include:

- ▶ OSA-Express adapters (Fast Ethernet, Gigabit Ethernet, and Token Ring¹)
- ▶ Routers (Cisco Channel Interface Processor, IBM 2216)

Interior connection methods are:

- ▶ HiperSockets
- ▶ z/VM Guest LAN
- ▶ CTC (physical)
- ▶ vCTC (CTC simulation within z/VM)
- ▶ IUCV

Note: The Parallel Sysplex Coupling Facility (CF) can also be used for TCP/IP communication within a zSeries complex. However, only z/OS supports Parallel Sysplex and the CF, so we will not discuss this further here since it is not applicable to running Linux guests under z/VM.

6.1.2 Isolation

Keeping certain parts of your environment separate from each other may be critical to the successful operation of your large scale Linux environment. As already mentioned, it might be simply to keep test and development systems away from production, but there are other reasons this might be required:

- ▶ Separating guests belonging to different customers
- ▶ Keeping Internet traffic away from internal applications and systems
- ▶ Implementing a “Demilitarized Zone” (DMZ)-style firewall within the z/VM system
- ▶ Separating secure and insecure traffic (application data paths, for example)
- ▶ Creating separate internal networks for infrastructure services

6.1.3 Address allocation

TCP/IP addresses can be a scarce resource, particularly when your guests are Internet-facing servers which cannot utilize RFC1918 private addresses. Effectively utilizing the addresses you have can help you reduce costs by only purchasing from your ISP an address range that fits your requirements.

¹ OSA-Express Token Ring is available only on the z900 processor.

Did you know? RFC1918 describes several IP address ranges that are free to be used within an organization's private network without having to be registered. You can learn more about private addressing by reading the RFC:

<http://www.faqs.org/rfcs/rfc1918.html>

Even if you can utilize RFC1918 addressing, your addressing plan will ideally allocate subnets and addresses in a way that allows for easy expansion with minimal need to change existing allocations.

There are many ways you can assign addresses in your large scale Linux installation. We strongly recommend working with the network staff at your site to arrive at an address allocation strategy that fits the rest of your organization's networking configuration.

6.1.4 Traffic shaping

The connectivity options available on zSeries today can provide extremely high bandwidth for servers running in the zSeries processor.

However, with such high amounts of bandwidth available, it is critical to ensure that the bandwidth is used properly. This is important for two reasons:

- ▶ A heavily utilized service on one guest should not be allowed to dominate the network, preventing other guests from providing service,
- ▶ A rogue application on a guest should not be able to flood the network with traffic, potentially causing problems in other parts of the network.

This is discussed further in 6.5, "Traffic control" on page 110.

6.1.5 Linux router or z/VM TCP/IP router

In the IBM Redbook *Linux on IBM @server zSeries and S/390: ISP/ASP Solutions*, SG24-6299, the authors raise the topic of which system to use as a router. The information in this book is still relevant, but needs to be updated a little.

Limited device support in Linux

With the advent of HiperSockets and Guest LAN, the problem of limited interface numbers supported in z/VM TCP/IP is relieved. HiperSockets and Guest LAN allow a z/VM TCP/IP stack to support a very large number of guests through a single interface (rather than individual IUCV or vCTC devices for each guest).

Linux reconfigurability

The channel device layer introduced with the Linux 2.4 kernel allows for new devices to be added to a Linux guest without an IPL, which is an improvement over the Linux 2.2 kernel. This means that you can use z/VM commands to add new devices such as Guest LAN NICs or real HiperSockets connections, and have these available to Linux without an IPL. In most cases, you can also use these new devices in TCP/IP immediately.

Device support in z/VM and Linux

z/VM still has greater support for unusual devices and connection methods, but Linux is catching up. The Linux CLAW driver is no longer considered experimental, for example.

Extra routing features

While Linux still has the advantage in terms of firewall functionality (through its use of facilities such as iptables), z/VM TCP/IP has added intrusion detection facilities into the stack. This intrusion detection function can handle many common denial-of-service (DoS) attacks. Beware, however, that this support will not provide comprehensive protection for an Internet-facing system due to the rate at which new attacks are developed.

The authors of the ISP/ASP book state:

As it is necessary to set up at least one VM TCP/IP stack simply to support the VM system itself, the VM TCP/IP stack is a good choice for general routing use.

This is still a true comment. However, it is likely that the additional features being added to Linux will favor the use of Linux routers. An example of such new function is the support for IEEE 802.1Q Virtual LAN trunking provided in the OSA-Express adapters in the z900 Turbo processor, which at this time is supported only in Linux.

6.1.6 Routing considerations with OSAs

If you are using an OSA to provide connectivity to your network environment, you must be aware of dependencies in the configuration of your OSA devices.

The ISP/ASP Redbook discusses the issues of using an OSA with Linux (refer to 4.4.3, “Using the OSA with Linux” in that publication).

In summary, the router guest or stack that “owns” the OSA must open the device as the primary router in order to provide connectivity for a large-scale Linux installation behind the router.

- ▶ For an OSA-Express attached to a VM TCP/IP stack in QDIO mode, this is done by adding the keyword “PRIRouter” to the OSA’s DEVICE statement in the PROFILE TCPIP file. This is documented in *TCP/IP Level 420 Planning and Customization*, SC24-6019.
- ▶ For a Linux TCP/IP router guest, this is done by adding the keyword “primary_router” (with its preceding comma separator) to the add_parms line of the OSA’s chandev configuration. This is documented in *Linux for zSeries: Device Drivers and Installation Commands*, LNUX-1103.

Note: This requirement applies no matter what type of router (Linux guest, z/VM TCP/IP stack, or z/OS system) you use.

The OSA will not forward IP packets to the router if the destination address is not known to the OSA, *unless* the guest or LPAR is defined as primary router.

The alternative to defining primary router is to not share the OSA between guests or LPARs. This may be a more workable solution for your installation; however, it may increase the number of physical OSA ports you need to use.

Also, keep in mind that a shared OSA can have only one primary router specified. If you have multiple router images that must have simultaneous connectivity, each router must have its own OSA. Refer to 7.2, “Multiple network devices to Linux guests” on page 123 for information about a configuration that uses multiple routers and OSAs to provide high network availability for Linux guests.

6.2 Linux routers

Using a Linux guest as a router provides a lot of flexibility, good routing performance, and all of the advanced function available to Linux as a router (including firewalling and traffic shaping). Most zSeries connectivity methods are supported by Linux, particularly advanced methods such as OSA-Express, HiperSockets and Guest LAN.

It is important, however, to make sure that all unnecessary services have been removed from your Linux router guest to ensure that it occupies the smallest possible footprint in your system. Minimizing services is good for security as well, as it reduces (or eliminates) paths for compromise of the system.

6.2.1 Device support

Linux on zSeries supports a subset of the connection methods available on zSeries, including OSA-Express (QDIO and LCS), OSA-2 (LCS), CTC, IUCV, HiperSockets, z/VM Guest LAN, and CLAW.

6.2.2 Routing function

The TCP/IP code in Linux started out as an implementation of classical UNIX TCP/IP. It provides all basic routing and protocol functions. The Linux 2.2 kernel introduced many new advanced routing features, including traffic control and high performance routing. Linux 2.4 introduced stateful firewall features, in addition to further routing enhancements.

All of the capabilities of Linux routing are functional on zSeries, and can be added to the routing function performed by a Linux router guest. The only restrictions are imposed by the capabilities of the network devices that provide connectivity. A simple example of this is that CTC devices cannot handle broadcast frames.

Using user-space daemons such as **routed**, **mtr** and **zebra**, Linux can participate in dynamic routing networks using protocols such as RIP and OSPF.

6.2.3 Setting up a Linux router

A Linux router guest will have very little additional software installed. In fact, even the “minimal” installation set provided by Linux distributors is likely to have much more software running than required for a router.

The only components required for a Linux router guest will be:

- ▶ The kernel itself
- ▶ If a modularized kernel, appropriate modules for network drivers² and additional functions such as firewall
- ▶ A remote access daemon (sshd or telnetd)
- ▶ If you choose to do SNMP management of your router, an appropriate SNMP daemon (such as ucd-snmp, now called net-snmp)
- ▶ If a dynamic routing protocol will be used, the appropriate dynamic routing daemon (gated, zebra, etc.).

² If you are using any network connectivity supported using the qeth.o driver, which is shipped object-code-only from IBM, your kernel must be built with loadable module support.

6.2.4 Changing a running Linux router guest

It is fairly easy to change a Linux router guest during operation. With root access to the console or a Telnet or SSH session, you can use the **ifconfig**, **route**, **ip** and other commands to change the operation of the running stack.

If you are adding new network interfaces, the channel device layer (introduced with the Linux 2.4 kernel) can be reconfigured without rebooting Linux. For devices supported by the qeth.o driver, most changes in the channel device layer configuration take effect immediately (“The /etc/chandev.conf file” on page 74 has more information about this).

Routing table updates can be made on the fly using the **route** or **ip** commands. The route command is the traditional command, while the ip command is part of iproute2, a new IP command suite that allows more functions and options to be provided.

6.3 z/VM TCP/IP routers

As discussed in “Linux router or z/VM TCP/IP router” on page 96, the z/VM TCP/IP stack can provide a very effective routing platform. Unlike using a Linux router, there are no additional concerns with trimming the installation down to minimum. z/VM also supports dynamic routing (through the MPROUTE service machine).

6.3.1 Device support

z/VM TCP/IP supports a wide range of TCP/IP devices, including OSA, CTC, IUCV, HiperSockets, and z/VM Guest LAN. It also supports other connection methods that are less frequently used, including ATM, HIPERchannel, SNALINK and X.25 NPSI devices.

If your network includes host connectivity devices that are not supported by Linux, then your only choice may be to use z/VM TCP/IP as a router.

6.3.2 Routing function

z/VM TCP/IP provides the same basic routing function as a Linux guest running as a router. Compared to Linux, however, z/VM TCP/IP does not support advanced routing features such as filtering and traffic shaping.

Configuring routing in z/VM TCP/IP

The way that routes are configured in z/VM TCP/IP is quite different from TCP/IP stacks on other platforms. The z/VM TCP/IP stack was originally written in the days before Classless Inter-Domain Routing (CIDR) and variable-length subnet masks, and routing configuration on z/VM TCP/IP reflects that. Once you think in terms of network classes again, it becomes much easier to configure routing in z/VM TCP/IP.

As an example, let's configure a default route on an interface on the 10.10.10.0/24 network, to the router 10.10.10.201. Example 6-1 shows the GATEWAY statement that would be coded in TCPIP PROFILE.

Example 6-1 z/VM TCP/IP GATEWAY statement

```
GATEWAY
10      =          ETH1  1500  0.255.255.0  0.10.10.0
DEFAULT 10.10.10.201 ETH1  1500  0
```

The key to understanding the first entry in the GATEWAY statement is realizing that the 10.10.10.0/24 network can also be referred to as a subnetwork of the 10.0.0.0 Class A network. z/VM TCP/IP requires the subnetwork portion to be separated from the network portion, which is why the first part of the GATEWAY statement shows 10 and the last part shows 0.10.10.0.

Put the two together, combined with the subnet mask of 0.255.255.0 (not 255.255.255.0, because the Class A 10.0.0.0 network already mandates 255.0.0.0, so we do not specify that again), and you get the network 10.10.10.0/24.

Another example is shown here.

Example 6-2 z/VM TCP/IP GATEWAY statement

```
GATEWAY
172.16  =          ETH2  1500  0.0.255.0    0.0.10.0
DEFAULT 172.16.10.201 ETH2  1500  0
```

This interface is on the 172.16.10.0/24 network, which is a subnet of the 172.16.0.0 Class B network.

172.16.0.0 mandates a 255.255.0.0 netmask, so to get a /24 mask, we specify 0.0.255.0 in the netmask portion, and the subnet value is 0.0.10.0, giving us the desired 172.16.10.0/24 network.

Note: Starting with z/OS Version 1 Release 2, a new method of configuring routing was introduced to z/OS Communications Server TCP/IP Services. To date, this new process is not supported on z/VM TCP/IP.

Hopefully, this new syntax will be adopted in z/VM TCP/IP, so that the “unique” method discussed here can finally be retired.

6.3.3 Changing a running z/VM TCP/IP stack

Using a process called OBEYFILE, a running z/VM TCP/IP stack can be changed without a restart. Almost all parts of the stack can be updated, including:

- ▶ Adding devices and links
- ▶ Changing interface IP addresses
- ▶ Updating the stack routing table
- ▶ Changing operational parameters of the stack (IP forwarding, etc.)

To dynamically update the running stack, you make your changes with appropriate syntax and use the OBEYFILE command to direct the TCP/IP stack to read the statements in the file. The stack will be reconfigured accordingly.

Note: As a security measure, the OBEY statement in the TCPIP PROFILE file contains a list of userids that will be “obeyed” by the TCPIP stack. This is required because the disk that contains the OBEYFILE module also contains other user TCP programs such as FTP. This means that almost every user can run the OBEYFILE program.

Specifying privileged userids on the OBEY statement in the TCPIP PROFILE prevents malicious or inadvertent changes to the TCP/IP stack by unauthorized users.

Changes made using the OBEYFILE process are temporary; they are only available until the TCP/IP machine is logged off. To make the changes permanent, the PROFILE file for the stack must be updated with the changes.

OBEYFILE “gotcha”

When using OBEYFILE to update a running z/VM stack, you must be careful that you do not affect the stack’s current parameters. Careful consideration must be given to statements that contain many parameters, such as GATEWAY, HOME, and IPCONFIG. When making a change to a statement, the entire statement must be reproduced with the changes added.

Statements like DEVICE and LINK are what we might call “discrete” statements, where a single set of parameters are given that have effect only on the specified attribute of the stack. HOME, GATEWAY and IPCONFIG are some of the statements that could be described as compound statements, where a single statement contains multiple sets of parameters affecting many attributes of the stack.

Because each set of parameters is usually specified on a separate line in the PROFILE file, it seems like they are separate statements when in fact they are not. When changing a compound statement, the entire statement must be given in the OBEYFILE.

Consider the fragment of a TCPIP PROFILE file shown in Example 6-3:

Example 6-3 TCPIP PROFILE example

```
; Device and Link for OSA-Express
DEVICE OSA2324D OSD 2324 PORTNAME OSA2320
LINK OSA2324L QDIOETHERNET OSA2324D
;
HOME
9.12.6.66 OSA2324L
;
GATEWAY
9 = OSA2324L 1500 0.255.255.0 0.12.6.0
DEFAULTNET 9.12.6.75 OSA2324L 1500 0
```

We wish to add a HiperSockets network to this TCP/IP stack, so we code the OBEYFILE shown in Example 6-4:

Example 6-4 TCPIP OBEYFILE example: in error

```
; Device and Link for HiperSockets
DEVICE IUTIQDED HIPERS 7100
LINK HIPER1 QDIOIP IUTIQDED
;
HOME
10.0.6.88 HIPER1
;
GATEWAY
10 = HIPER1 1500 0.255.255.0 0.0.6.0
```

If we were to run this OBEYFILE, we would find that the device and link for our HiperSockets connection would be added as we expected. However, TCP/IP communication with our z/VM system via the OSA-Express would no longer function. This is because the HOME address for the OSA was removed by our

OBEYFILE. Likewise, the interface route and default gateway were also removed.

The correct OBEYFILE, with all the required parameters to HOME and GATEWAY, would look like Example 6-5:

Example 6-5 TCP/IP OBEYFILE example (correct)

```
; Device and Link for HiperSockets
DEVICE IUTIQDED HIPERS 7100
LINK HIPER1 QDIOIP IUTIQDED
;
HOME
9.12.6.66 OSA2324L
10.0.6.88 HIPER1
;
GATEWAY
9 = OSA2324L 1500 0.255.255.0 0.12.6.0
10 = HIPER1 1500 0.255.255.0 0.0.6.0
DEFAULTNET 9.12.6.75 OSA2324L 1500 0
```

Notice that the new parameters being added to HOME and GATEWAY are specified, in addition to the existing parameters from the PROFILE file.

6.3.4 z/VM TCP/IP support servers

z/VM TCP/IP provides other services that support a TCP/IP network. You can use z/VM to provide DNS, BOOTP and/or DHCP, IMAP mail, Kerberos, and other TCP/IP server applications.

Note: Refer to *z/VM V4R3.0 TCP/IP Level 430 Planning and Customization* for more information on how these servers are set up.

DNS server

You can configure a DNS server on z/VM. However, this DNS server uses DB2 as its backend database, which will generally preclude it from being used in a large scale Linux installation. Therefore, we recommend that in a large scale Linux installation, you use an alternative DNS server.

6.4 z/OS routers

In general, the issues associated with using z/OS as a router are similar to z/VM, with the obvious exception that your site may not be using z/OS. We would not recommend that a new installation license z/OS simply to perform TCP/IP routing. However, if your installation does currently use z/OS, there are a couple of reasons why a z/OS router system may be beneficial.

Note: In this section, when we refer to “z/OS”, this includes the new z/OS.e available on the zSeries 800 processor.

6.4.1 HiperSockets Accelerator

The HiperSockets Accelerator function (HSA) is currently supported only on z/OS. Using HSA, network packets can be efficiently routed to systems attached to a HiperSockets network via an OSA-Express adapter.

When a router stack is used, packets arriving must be processed by the IP layer in the LPAR or guest running the router. This may not be a large amount of processing, but if a large number of guests are being routed via the stack or a large traffic flow is experienced, the processing performed by the router stack becomes significant.

HSA provides a means of increasing the efficiency and reducing the overhead of routing by transferring packets from the OSA-Express adapter to a HiperSockets network (and vice versa) at the lowest possible layer in z/OS Communications Server. Processing in the z/OS TCP/IP stack is almost eliminated.

Note: Refer to the Redbook *zSeries HiperSockets* for a more complete explanation of HiperSockets Accelerator, including how it works and how it is set up.

To see the effect of HiperSockets Accelerator, we added a z/OS Version 1 Release 2 system to our network configuration according to the diagram shown in Figure 6-1 on page 106.

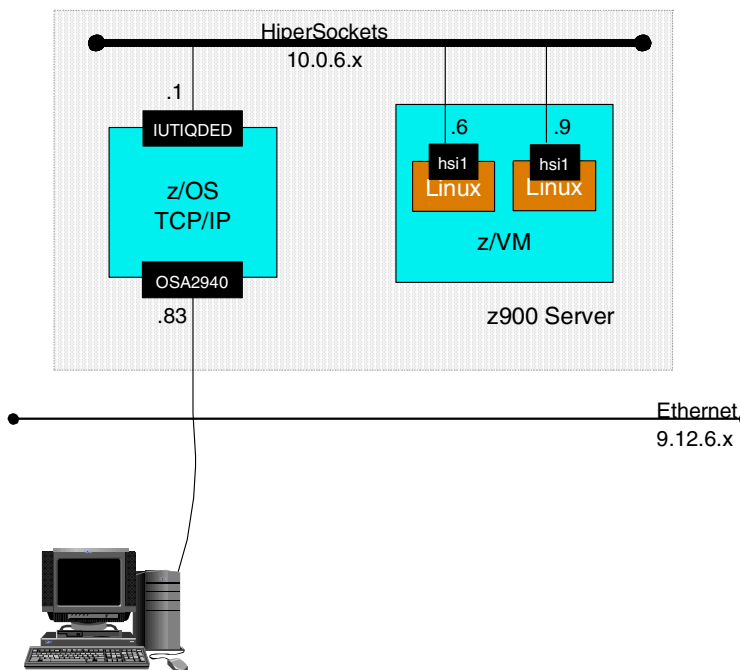


Figure 6-1 HiperSockets Accelerator test configuration

Setting up HiperSockets Accelerator in z/OS TCP/IP

The TCPIP PROFILE defined for the z/OS TCP/IP stack contained the DEVICE and LINK statements listed in Example 6-6 to define the OSA-Express and HiperSockets interfaces.

Example 6-6 HiperSockets Accelerator: z/OS TCP/IP DEVICE and LINK

```

DEVICE OSA2940 MPCIPA PRIROUTER
LINK OSA2940LNK IPAQENET OSA2940
;
DEVICE IUTIQDED MPCIPA
LINK HIPERLED IPAQIDIO IUTIQDED

```

Notice that the OSA-Express device statement contains the keyword **PRIROUTER**. This is an essential component of the HiperSockets Accelerator configuration (refer to 6.1.6, “Routing considerations with OSAs” on page 97 for a reminder on this).

Also required is a new keyword on the IPCONFIG statement. The HiperSockets Accelerator feature is enabled using the IQDIORouting keyword. We added the following to our TCPIP PROFILE:

```
IPCONFIG IQDIOR
```

When we start TCP/IP on our z/OS system, the messages shown in Example 6-6 tell us the status of TCP/IP and HiperSockets Accelerator on our system.

Example 6-7 HiperSockets Accelerator: TCP/IP startup messages

```
EZZ0300I OPENED PROFILE FILE DD:PROFILE  
EZZ0309I PROFILE PROCESSING BEGINNING FOR DD:PROFILE  
EZZ0316I PROFILE PROCESSING COMPLETE FOR FILE DD:PROFILE  
EZZ0688I IQDIO ROUTING IS ENABLED  
EZZ4202I OPENEDITION-TCP/IP CONNECTION ESTABLISHED FOR TCPIP  
EZZ4313I INITIALIZATION COMPLETE FOR DEVICE OSA2940  
EZZ4313I INITIALIZATION COMPLETE FOR DEVICE IUTIQDED  
EZB6473I TCP/IP STACK FUNCTIONS INITIALIZATION COMPLETE.  
EZAIN11I ALL TCPIP INTERFACES FOR PROC TCPIP ARE ACTIVE.
```

Linux configuration for HiperSockets Accelerator

There is no specific configuration task required for HiperSockets Accelerator in any guest operating system, including Linux. The z/OS TCP/IP stack handles all of the work. The way that the guest systems receive traffic from the HiperSockets network is no different, whether HiperSockets Accelerator is used or not.

Tip: If your Linux guests have multiple interfaces, and you are using HiperSockets Accelerator, ensure that your Linux guests specify the z/OS system as their default router. This will ensure that you get maximum benefit from the HiperSockets Accelerator configuration.

Demonstrating HiperSockets Accelerator

To verify the effect of HiperSockets Accelerator, we used a Linux system which was attached to both the “accelerated” HiperSockets network, and a QDIO guest LAN. The diagram shown in Figure 6-2 on page 108 illustrates the configuration.

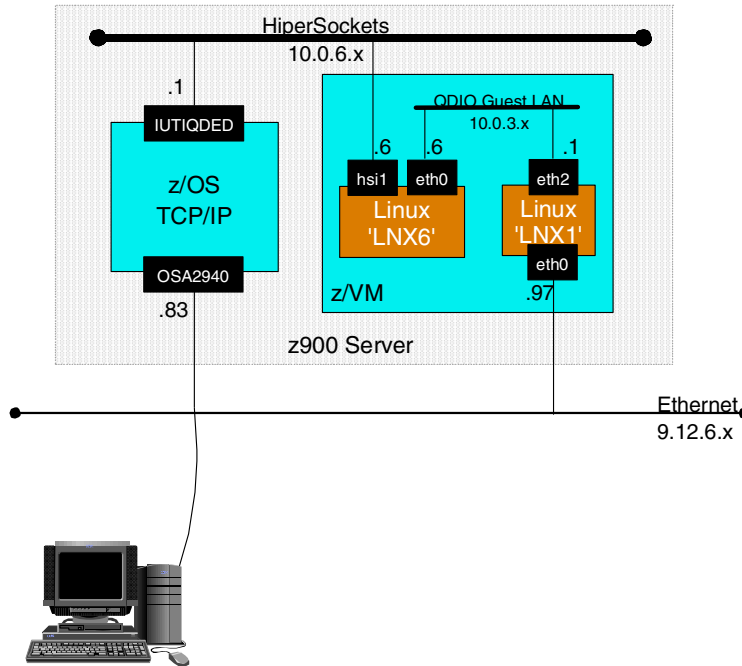


Figure 6-2 HiperSockets Accelerator: verification environment

In this diagram, you can see that there are two paths to LNX6:

- ▶ To 10.0.3.6, via LNX1 and the QDIO Guest LAN
- ▶ To 10.0.6.6, via z/OS and the HiperSockets LAN

Both of these paths are two-hop paths to LNX6 from the 9.12.6.x Ethernet network, so we would expect that the output of the **tracert** command (an IP diagnosis tool that shows the path from one host to another) would differ only in the IP address of the router used to reach the destination.

We issued the **tracert** command from the LAN PC, to trace the IP path to the host LNX6. Example 6-8 on page 109 shows the output from the tracert commands we issued.

Example 6-8 HiperSockets Accelerator: traceroute commands

```
C:\>tracert -d 10.0.3.6
Tracing route to 10.0.3.6 over a maximum of 30 hops
  1    10 ms  <10 ms  <10 ms  9.12.6.97
  2   <10 ms  <10 ms  <10 ms  10.0.3.6
Trace complete.
```

```
C:\>tracert -d 10.0.6.6
Tracing route to 10.0.6.6 over a maximum of 30 hops
  1    20 ms  <10 ms  <10 ms  10.0.6.6
Trace complete.
```

```
C:\>
```

In the output of the first **tracert** command, the hop through the Linux router is clearly shown. The output of the second **tracert** command tells us that there was no intervening router in the path, and that the traffic arrived directly at the destination host. This is HiperSockets Accelerator at work: the packets were not processed by the TCP/IP stack of z/OS, but forwarded directly via low-level device drivers across the HiperSockets to the destination.

Note: When we issued the **tracert** command after a period of network inactivity, we got the following output:

```
C:\>tracert -d 10.0.6.6
Tracing route to 10.0.6.6 over a maximum of 30 hops
  1    20 ms   10 ms   20 ms  9.12.6.83
  2    10 ms   20 ms  <10 ms  10.0.6.6
Trace complete.
```

This would seem to indicate that HSA was not functioning, but is actually normal. The first packet to a destination on the HiperSockets network is routed by z/OS TCIP as normal. In doing this, z/OS TCP/IP learns that the destination can be reached through HiperSockets and is eligible for “acceleration”. Subsequent packets are then sent via the HiperSockets Accelerator “fast path”, and after some inactivity, this learned information is dropped.

When we issued the **tracert** command a second time, we got the expected result (as shown in Example 6-8).

When to use HiperSockets Accelerator

As mentioned earlier, a customer who does not currently use z/OS should not licence z/OS simply to get HiperSockets Accelerator. However, there is at least

one configuration where HiperSockets Accelerator can (and should) be used to improve the performance of Linux routing.

Some high availability TCP/IP features of Communications Server for z/OS require the OSA-Express port to be configured as primary router. This prohibits the port from being shared with a Linux or z/VM router. In the past, in this configuration you would have used a CTC or outboard router connection to link between your z/OS system and your Linux or z/VM router, incurring a performance penalty in doing so.

However, if you define your Linux or z/VM router on a HiperSockets LAN with the z/OS TCP/IP, HiperSockets Accelerator will route traffic to and from the Linux or z/VM router as if they were sharing the OSA port. This means your Linux guests will enjoy greater performance than before, the routing load on z/OS TCP/IP will be greatly reduced, and the z/OS high availability features will not be affected.

6.5 Traffic control

Traffic Control (TC) provides a way to interface with the TCP/IP prioritization features of the Linux kernel. These functions operate on the IP output queue, shown in the diagram in Figure 6-3.

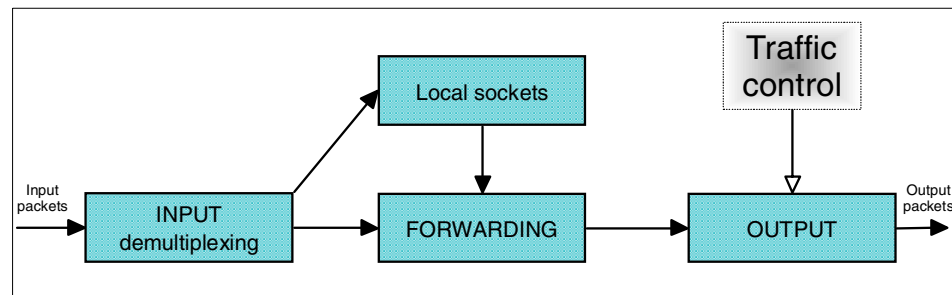


Figure 6-3 Packet path in the Linux 2.2 kernel

Traffic control is implemented at the output stage in Linux, because there's no point in trying to control traffic that has already arrived in your input queue.

Note: A complete discussion of traffic control in Linux is beyond the scope of this book. For additional information about this topic, you can refer to the paper we used while researching this topic entitled "*Linux - Advanced Networking Overview*" by Saravanan Radhakrishnan, available at:

<http://qos.ittc.ukans.edu/howto/howto.html>

Most of the first half of the paper is technical material about how traffic control is implemented, but later there are examples of using the Linux tools to configure it (although we did spot some errors in the examples; see our comment box on Page 116).

Another excellent reference is *Linux 2.4 Advanced Routing HOWTO* by Netherlabs BV et al., at:

<http://www.linuxguruz.org/iptables/howto/2.4routing.html>

6.5.1 Components of traffic control

There are a number of complementary features that come together to provide traffic control in Linux. Traffic control requires configuration of the following:

- ▶ Queue discipline

This component implements the traffic control queue associated with a particular network interface. It defines what attributes can be assigned to traffic, and the way that the prioritization attributes will affect traffic flow.

- ▶ Traffic class

The traffic class assigns values such as priority and bandwidth share to traffic. The traffic classes configured for a particular traffic control queue set the points that traffic can enter the queue. Some queue disciplines operate without traffic classes because of the nature of the algorithm used.

- ▶ Filter

A traffic filter determines how traffic will be allocated to the classes. Filters are defined using certain classification methods. Packets passing through the kernel are checked against the defined filters, and processed accordingly (in traffic control, this processing would be assignment to the appropriate class).

Depending on the type of classifier used in your filter, additional configuration may be required. For example, a route classifier requires information to be added to the route table to trigger the filter.

Queue disciplines

The Linux kernel includes support for several queuing disciplines, which define the ways that traffic arriving at an traffic control queue will be processed. Some of

the queuing disciplines are First In, First Out (FIFO), Class-Based Queue (CBQ) and Token Bucket Flow (TBF).

The traffic control queue is separated from the interface queue because the interface queue is concerned with scheduling packets onto the physical medium and not with prioritization. Separating the interface queue from the traffic control queue makes it easier for different traffic control protocols to be used.

CBQ is the traffic control method most commonly used. Traffic is controlled using classes. For each class, you can configure the priority of the traffic in that class relative to other classes, and the bandwidth that class is allowed to consume.

Note: Whether CBQ is actually most commonly used is probably debatable; we are only using the amount of documentation available on Linux traffic control as a guide. Differential Service (diffserv), another queue discipline, is likely to gain popularity as it is the principle upon which traffic control protocols like Reservation Protocol (RSVP) are based.

A conceptual view of the interaction between the CBQ queue discipline, traffic classes and filters is shown in Example 6-4.

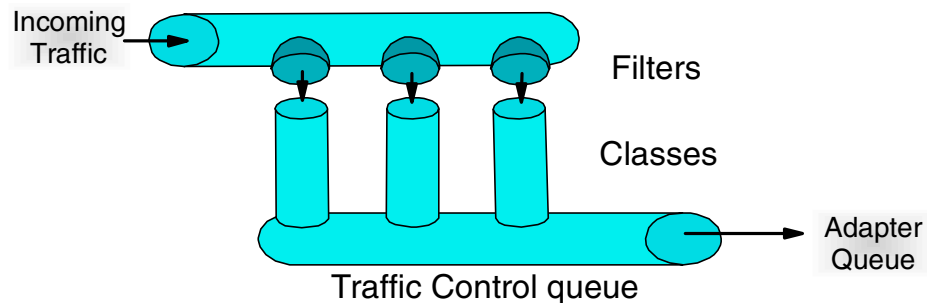


Figure 6-4 Traffic control using class-based queueing

In basic terms, filters classify traffic into different classes. The queue decides how the classified traffic will be sent to the adapter queue for transmission.

6.5.2 Configuring CBQ

To use class-based queueing, you need to define a CBQ at each of the interfaces you wish to control. The following command will define a new CBQ attached to interface eth0.

```
tc qdisc add dev eth0 root handle 1: cbq bandwidth 10Mbit allot 1514 \  
cell 8 avpkt 1000 mpu 64
```

Some of the features of this queue are:

- ▶ Its handle (the name by which it is referenced) is 1 (handle 1:).
- ▶ The queueing discipline used is CBQ (cbq).
- ▶ The total bandwidth to be managed by this queue is 10 megabits per second (bandwidth 10Mbit).
- ▶ Packet transmission time will be measured in terms of 8bytes (cell 8).
- ▶ The average size of a packet is 1000 bytes (avpkt 1000).
- ▶ The minimum packet size is 64 bytes (mpu 64).

Once a queue is defined, you can define a class which will utilize the queue. The following command will define a new class associated with the queue defined in the last command.

```
tc class add dev eth1 parent 1:1 classid 1:2 cbq bandwidth 10Mbit \  
rate 1Mbit allot 1514 cell 8 weight 100Kbit prio 3 maxburst 20 \  
avpkt 1000
```

This class has the following attributes:

- ▶ Its ID is 1:2 (classid 1:2), and its parent class is 1:1 (parent 1:1).
- ▶ The total bandwidth allocated to the queueing discipline owned by this class is 10 megabits per second (bandwidth 10Mbit).
- ▶ The bandwidth allocated to this class is 1 megabit per second (rate 1Mbit).
- ▶ The weight of this class is 100 kilobits per second (weight 100Kbit).
- ▶ The priority allocated to this class is 3 (prio 3).
- ▶ No more than 20 packets can be sent in a burst (maxburst 20).

Filters are used to allocate traffic to classes, and are defined using classifiers. The simplest classifier is a route classifier, which allows you to allocate traffic to classes based on entries in the route table. A more complex classifier is u32, which you can use to allocate traffic based on source or destination port and other attributes. This would allow you to control traffic on a per-application basis.

The following command will install a route classifier on a CBQ device.

```
tc filter add dev eth0 parent 1:0 protocol ip prio 100 route \  
to 1 classid 1:2
```

The parameter route designates this filter as a route classifier. It applies to IP packets (protocol ip) and will give packets classified by it a default priority of 100. This filter will pick up packets assigned to route realm 1, and assign them to traffic class 1:2.

You can now make entries in the routing table to classify traffic. Here is an example of classifying traffic to the host 10.0.3.4 into the realm 1, which we linked to the traffic class using the previous command:

```
ip route add 10.0.3.4 dev eth0 realm 1
```

6.5.3 CBQ usage example: bandwidth choke

We used a Linux guest with two network interfaces to demonstrate traffic control. We configured a CBQ on one of the interfaces to see if the configuration was effective in limiting bandwidth usage by the Linux guest.

We used the FTP server on a common machine to transfer a large file via two network paths, one using CBQ and the other using a direct network path. The effect we were looking for was to have the FTP transfer throttled by traffic control to the bandwidth we specify.

Figure 6-9 shows the log of the session.

Example 6-9 Setup and use of a CBQ traffic control configuration

```
# tc qdisc add dev hsi1 root handle 1: cbq bandwidth 100Mbit allot 1514 cell 8
avpkt 1000 mpu 64
# tc class add dev hsi1 parent 1:0 classid 1:1 cbq bandwidth 100Mbit rate
100Kbit allot 1514 cell 8 weight 10Kbit prio 3 maxburst 20 avpkt 1000
# tc filter add dev hsi1 parent 1:0 protocol ip prio 100 route to 1 classid 1:1
# ip route change 9.12.6.0/23 via 10.0.6.1 dev hsi1 realm 1
# tc qdisc ls dev hsi1
qdisc cbq 1: rate 100Mbit (bounded,isolated) prio no-transmit
# tc class ls dev hsi1
class cbq 1: root rate 100Mbit (bounded,isolated) prio no-transmit
class cbq 1:1 parent 1: rate 100Kbit prio 3
#
```

The following notes further explain points in this example:

- ▶ The first command, **tc qdisc**, added the CBQ to our HiperSockets interface hsi1. We allocated a manageable bandwidth of 100 Mbps for this interface.
- ▶ The second command, **tc class**, defined our traffic class. In order to plainly demonstrate the effect, we have set the rate limit to 100 kbps.
- ▶ The third command, **tc filter**, creates a filter using route classification. A priority of 100 is assigned to these packets, and a routing realm of 1 is used to select this filter.
- ▶ The fourth command, **ip route**, changes the route we had defined to the destination network. The route realm is added, which invokes the route classification filter. At this point, our traffic control configuration is controlling packets.
- ▶ The fifth and sixth commands show the queue disciplines and classes configured.

From a remote machine on the 9.12.6.0 network, we commenced an FTP transfer of a large file.

We noted that the transfer rate for the file was between 1 and 2 megabytes per second, so it did not appear that the configuration had worked. After checking the configuration, we discovered that we had omitted a crucial part of the configuration that would be required to achieve our desired effect.

In our `tc class` command, we specified our desired rate as 100 kbps. However, by default, traffic controlled by a class is allowed to use any excess bandwidth in the parent class. Our class was working, but because excess capacity was available in the parent class there was no visible bandwidth limiting taking place.

Example 6-10 shows the command we issued to change the class, followed by a class list command to show the difference.

Example 6-10 CBQ test: fixing our configuration error

```
# tc class change dev hsi1 parent 1:0 classid 1:1 cbq bandwidth 100Mbit rate
100Kbit allot 1514 cell 8 weight 10Kbit prio 3 maxburst 20 avpkt 1000 bounded
# tc class ls dev hsi1
class cbq 1: root rate 100Mbit (bounded,isolated) prio no-transmit
class cbq 1:1 parent 1: rate 100Kbit (bounded) prio 3
#
```

When we retried the FTP transfer, we had successfully limited the transfer to approximately 12 kilobytes per second.

6.5.4 CBQ usage example: differentiating interactive traffic

We noticed that once our traffic control configuration was in place and the FTP transfer was in progress, interactive traffic (SSH, telnet) was extremely slow. Since we were using route classification, all traffic to the destination network was being throttled down by our class.

To attempt to get interactive traffic back to normal, we referred to some `iproute2+tc` examples available on the Web. These examples imply that you can set up sub-classes of the original class, and use the `defmap` attribute to classify your traffic. However, this approach did not work for us.

Other examples on this topic use the `u32` classifier to set up port-specific filters. This looked like it would be the only effective way to prioritize interactive traffic. However, that meant setting individual filters for each prioritized traffic type, and defining a bitmap to link traffic types with the appropriate class.

Comment: Arguably the most significant barrier to a deeper understanding of Linux's advanced routing features is the lack of good documentation. Very little documentation exists, and what does exist relies on the reader already having a solid understanding of the features of the kernel and just needing help with the syntax of the `iproute2` commands. Some of the examples circulating also have errors.

Don't let this discourage you, however! This chapter has only scratched the surface of the capabilities of traffic control and advanced routing. If you take the challenge and decide to explore further, we make one request: document!

6.6 Dynamic routing

It is possible to use dynamic routing protocols inside a penguin colony. However, the processing overhead involved in managing routing domains means that the amount of dynamic routing should be kept to a minimum. We suggest using dynamic routing protocols only between the router images and the exterior network.

Within the penguin colony, you can use a static routing structure such as that suggested in *Linux on IBM @server zSeries and S/390: ISP/ASP Solutions*, SG24-6299, to provide connectivity to all your networks with the least amount of routing configuration.

If you do need to use dynamic routing within your penguin colony, be aware that your z/VM system may become slightly harder to tune because of the additional overhead caused by dynamic routing updates. You may need to experiment with increasing the interval between routing update messages (for example, increasing the 'Hello interval' and 'Dead router interval' in the case of OSPF) to reduce the amount of traffic in the network. Be aware, however, that increasing these values will make your network less responsive to changes, since more time is likely to pass between routing updates.

6.6.1 How dynamic routing works

Dynamic routing protocols generally fall into two categories: *distance-vector* protocols and *link-state* protocols.

Distance-vector protocols like Routing Information Protocol (RIP) and Interior Gateway Routing Protocol (IGRP) maintain lists of the cost to reach all of the known networks (the "distance" to other networks). At regular intervals, routers send a copy of their route table to all other routers in their network. This information is used by routers to keep their own information up to date.

Distance-vector protocols introduce a lot of unnecessary traffic onto the network, and take a long time to respond to network changes and failures.

Link-state protocols such as Open Shortest Path First (OSPF) and Intermediate System-to-Intermediate System (IS-IS) maintain a database of the links in the network. The routing table is built from the information contained in this database.

Changes in the network cause routers to flood Link State Advertisements (LSAs) through the network, advising all routers of the change, which means that link-state protocols generally learn about changes and failures much quicker than distance-vector protocols.

However, link-state protocols consume large amounts of memory (compared to distance vector protocols), and the complexity of the protocol processing consumes more processor resource.

Note: For more information about TCP/IP dynamic routing and how it works, refer to *Designing Large-Scale IP Internetworks* by Cisco Systems, at:

<http://www.cisco.com/univercd/cc/td/doc/cisintwk/idg4/nd2003.htm>

6.6.2 Dynamic routing in a penguin colony

There are three circumstances under which you might want to use dynamic routing in a penguin colony:

- ▶ Advertising a virtual IP address to the network

If you are using a virtual IP addressing method, you may want to use dynamic routing to advertise this address to the network. This is the way z/OS customers often make use of the VIPA function provided by z/OS Communications Server, to allow them to use additional features like Dynamic VIPA and VIPA Takeover.

In a penguin colony, however, it is much less likely that you will move an IP address from one Linux guest to another. Therefore, you are more able to use static routing mechanisms (like the route summarization method described in the ISP/ASP redbook) to reach your virtual addresses.

- ▶ Advertising changes and reconfigurations in the penguin colony

If the routing configuration inside the penguin colony is subject to frequent change, using dynamic routing would update the network rapidly and automatically. However, with broadcast-style network connectivity like z/VM Guest LAN being available inside the penguin colony, it is much less likely that frequent changes will occur in the z/VM system.

- ▶ Making use of multiple interfaces for redundancy

When you configure multiple interfaces to Linux guests, you have to ensure that the Linux guest can make use of any interface for outgoing traffic, and the network can use any available interface for incoming traffic. Duplicate routes can take care of the requirement for the outgoing traffic, and for incoming traffic, static routing can be used.

In any of these situations (particularly the last two), using dynamic routing between your router images and the router network outside the zSeries environment can be beneficial. This allows the router images to advise the network about changes within the z/VM system.

The growth of your “neighborhood”

Perhaps the biggest reason *not* to use dynamic routing inside your penguin colony becomes apparent when you have large numbers of guests attached to z/VM Guest LANs and HiperSockets.

Dynamic routing protocols rely on the transmission of routing updates between routing “neighbors”. Neighbors are routers attached by a common network transport. Depending upon the nature of the routing protocol being used, large amounts of network traffic can be generated, or large link-state databases can be set up.

In a penguin colony, this is undesirable: with virtual networking (Guest LAN, virtual CTCs, etc.), all network traffic results in processing, handling routing updates requires processing, and managing dynamic routing domains requires storage.

In a penguin colony that uses IUCV or vCTC connections, a dynamic routing daemon on a Linux guest would be a neighbor to only one or two others: the router (or routers) that provide their connection to the “outside world”. In a Guest LAN or HiperSockets network, however, a dynamic routing daemon would become a neighbor to every other guest on that same network.

In the case of a distance-vector protocol, all Linux guests would be sending their whole routing table to each other at regular intervals, and a link-state protocol would keep a record of every other Linux guest and its links and link status. Clearly, this is not a desirable situation for our penguin colony.

Recommendation: We do not think that running dynamic routing *inside* a large-scale Linux installation is worthwhile. Unless you really have a strong need to use dynamic routing, consider alternatives.

6.6.3 Controlling routing tables

You can use dynamic routing at your router systems to advertise the networks within your penguin colony to the rest of the network. In this case, you get some of the benefits of dynamic routing while minimizing the adverse effects.

Even so, you should ensure that the routing tables of the router images do not become overpopulated with useless route entries. This might occur if your router images are considered to be part of the backbone network, from a dynamic routing perspective.

There are methods to control routing table updates that vary between routing protocols and the dynamic routing daemons used. One such process that can be used in an OSPF configuration, for example, is the *stub area*.

An OSPF stub area is an area with certain attributes, the main one being that it is not a through-path for network traffic (traffic is routed into or out of the area, but never through it to another area). The routers at the border of a stub area advertise a summarized route to the routers inside the stub area (usually just a default route). The rest of the routing information that exists in the rest of the OSPF domain is hidden from the routers in the stub area.



Network high availability

In this chapter, we discuss methods you can use to make the network connectivity in your large scale Linux environment more reliable and less susceptible to outage.

We also touch briefly on clustering and server high availability.

7.1 Planning virtual connectivity for high availability

There are a number of tools and utilities in the Linux world that allow you to build highly available application hosting facilities. Many papers and HOWTOs have been written on high availability Linux, and projects such as Linux-HA specialize in providing information and user-space tools for making highly available Linux systems.

Some of the technologies that work well in discrete server environments do not necessarily map well to a large scale Linux on zSeries installation. Since one of the design objectives of the penguin colony is to minimize unnecessary CPU consumption in the Linux guests, high availability solutions that rely on heartbeat polling or dynamic routing protocols are less desirable, since they increase the “idle” CPU utilization of the Linux guests. Also, the relatively cheap server network cards with dual ports and automatic switchover do not compare to the capabilities of an interface like the OSA-Express.

7.1.1 Determine the level of redundancy you need

Some of the options that will be described in this chapter introduce considerable complexity to the configuration of the z/VM and Linux installations you will have to maintain. For this reason, it is important not to overengineer your solution.

There are plenty of things that can be done in case something fails. In 7.2.1, “Configuring multiple network interfaces” on page 123 we describe a way of giving a Linux guest access to multiple router guests attached to multiple OSA-Express adapters. This configuration is intended only for guests requiring the highest levels of availability. If you have guests that do not have high availability requirements, give consideration to using a basic configuration.

7.1.2 z/VM TCP/IP availability

A critical component which may be overlooked when designing connectivity in a large scale Linux installation is the z/VM TCP/IP stack itself. There can be many times when, as a result of misconfiguration or failure in a Linux system, you cannot get access via the network to a Linux guest. On these occasions, being able to obtain a connection via z/VM TCP/IP to the Linux console lets us do enough to get network connectivity again, log on, and fix the problem. Being able to get reliable access to the z/VM TCP/IP stack is therefore very important.

The z/VM TCP/IP stack supports the VIPA function. We suggest that you look at using multiple interfaces to the z/VM stack, and use VIPA to address z/VM.

Console access

An alternative to TCP/IP access would be to use console devices. zSeries systems require console devices to support the operation of the system. A console access controller such as the IBM 2074 Console Support Controller can provide local console access via TN3270, allowing you to use a TN3270 emulator over your network to get a console connection to your zSeries system.

Using consoles would relieve some of the dependencies in the z/VM system (you no longer require a functional TCP/IP stack, for instance). However, we do not believe that TCP/IP access to the 2074 can be set up as reliably as to a z/VM TCP/IP stack using VIPA. Also, since they are mainly used for system consoles and require a high level of security, it is generally not recommended to put a 2074 on an open network (like a corporate backbone). This means that you may only be able to connect to your 2074 from the systems operation area or other secured locations.

7.2 Multiple network devices to Linux guests

You can provide multiple network connections to your Linux guests. This will allow you to connect to multiple routing paths, removing the single point of failure and making your Linux guests less susceptible to network outage.

7.2.1 Configuring multiple network interfaces

An IP host with more than one interface is called a “multi-homed” host.

You can use your system administration tool (YaST on SuSE, for example) to configure a second network interface in the same way you configure the first. Remember that you will have to do more than just configure the Linux network device; you will also have to do z/VM device definition and `/etc/chandev.conf` updates as well, since these are not usually handled by the network configuration tool. Refer to Chapter 4, “HiperSockets and z/VM Guest LAN” on page 57 for information about configuring HiperSockets and Guest LAN networks.

We suggest that when configuring multiple interfaces on your Linux guest, you use the same type of device for all connections. This avoids the possibility of intermittent problems caused by certain network features (broadcast and multicast, for example) not being present on one adapter or another.

The need for a virtual IP address

Having multiple network interfaces is, by itself, not sufficient to provide efficient redundancy, because of the way that IP routing works. Figure 7-1 on page 124 will help to explain this.

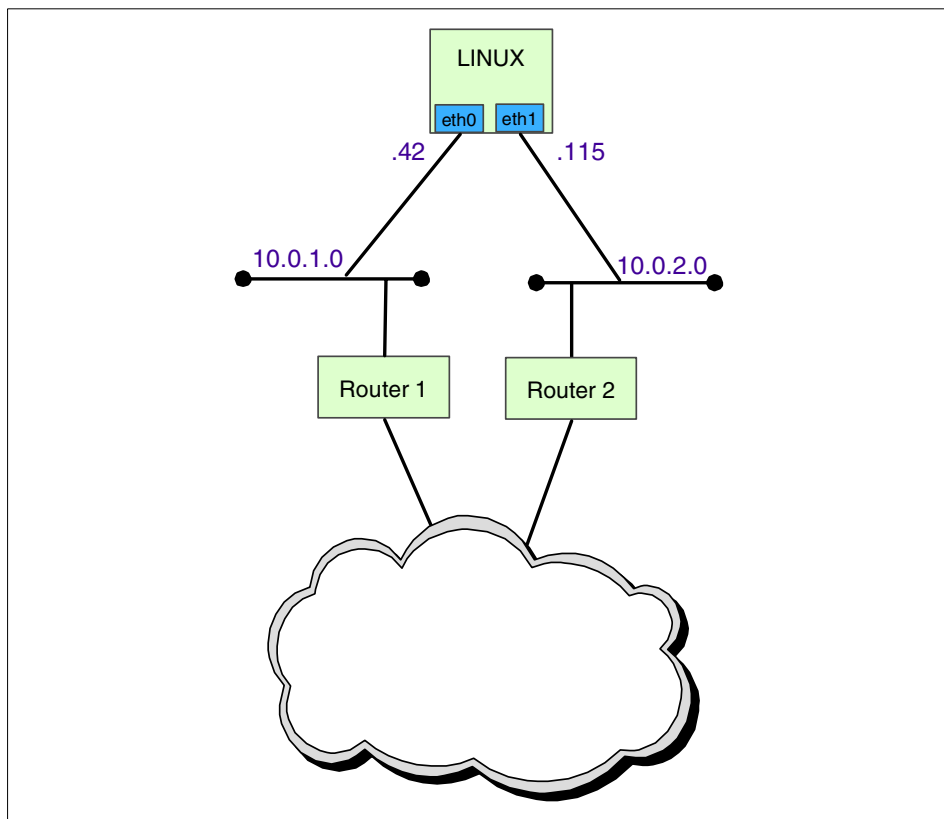


Figure 7-1 IP addressing issue in multi-homed configuration

In Figure 7-1, consider traffic inbound to the Linux host from the network, bound for the IP address 10.0.2.115. Linux, by default, will respond to traffic for any of its configured IP addresses on any of its interfaces, so if the network directed the traffic via Router 1 and the eth0 interface, then all would still work. Normal IP routing, however, will direct this traffic to Router 2, because that's how the network learns to get traffic to the 10.0.2.0 network. If the eth1 interface was down, there is no way (without complex configuration in the routers) to get this traffic redirected to the other interface.

What this means is that clients requesting service from your Linux guest cannot get seamless and transparent connectivity via either interface if they use the "interface" IP address to communicate with the guest.

More information: In the Linux IP stack, IP addresses are *not* bound to any particular interface. Rather, the kernel keeps a list of addresses it will respond to. Addresses are associated with interfaces simply as an indication of the network that is directly reachable by a particular interface.

RFC1122, *Requirements for Internet Hosts -- Communication Layers*, contains a discussion about multi-homed hosts, and categorizes their behavior as either “Weak End System” or “Strong End System”. Linux’s default configuration (which is heavily configurable) displays a combination of these behaviors.

Refer to RFC1122 for a complete discussion of end system behaviors. You can find this RFC at:

<http://www.faqs.org/rfcs/rfc1122.html>

Eliminate common paths

There is not much point in configuring two separate interfaces on a Linux guest if there is excessive duplication along the network path. Obviously there will be some commonality that cannot be avoided (the network card in the client PC, for example), but where possible there should be as much duplication through the environment as technical and budgetary constraints will allow.

Figure 7-2 on page 126 shows a sample configuration which has no duplication inside the z/VM environment.

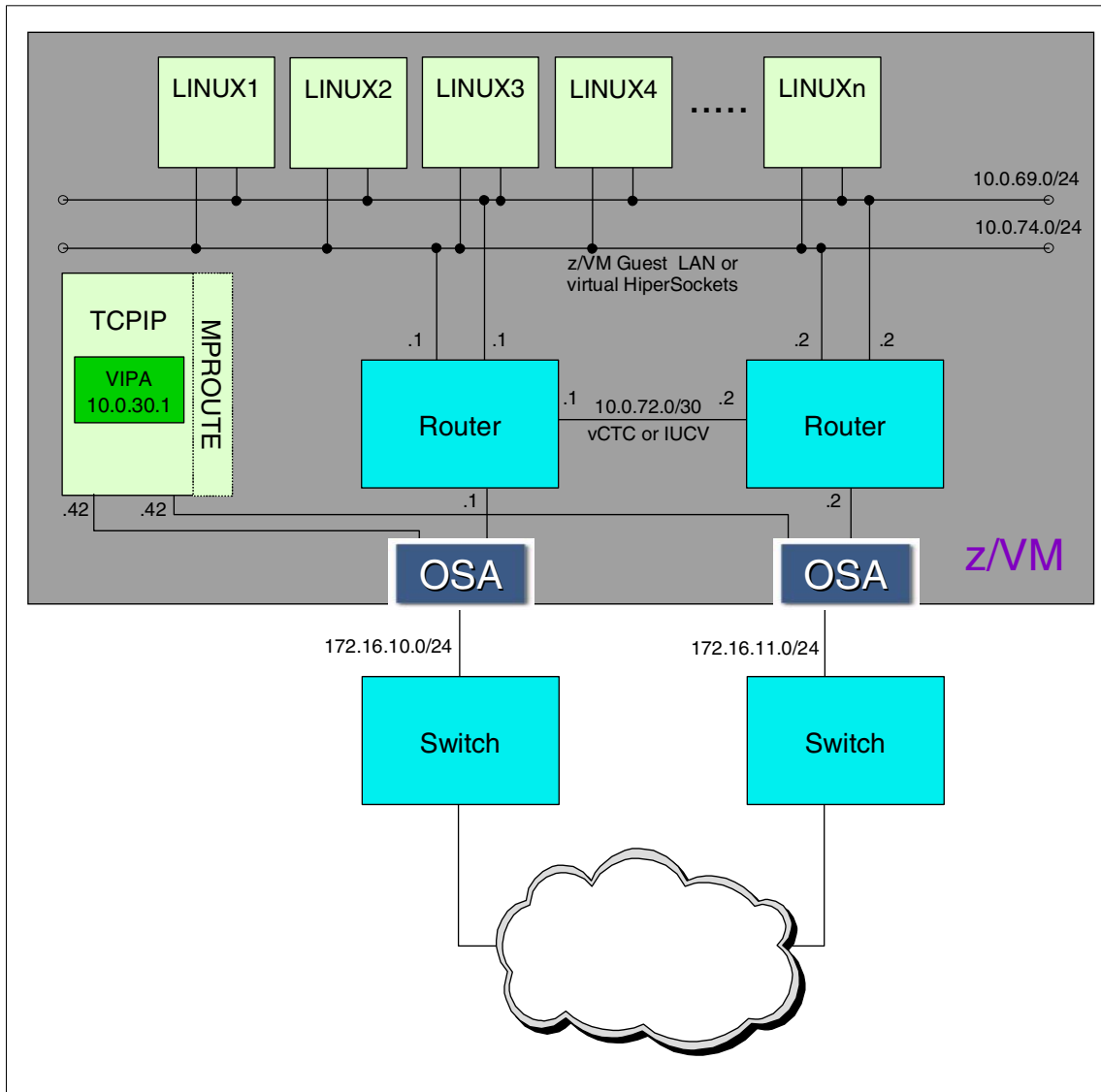


Figure 7-2 Redundant network design 1

Note: The IP addresses and ranges shown in Figure 7-2 and Figure 7-3 on page 131 are for diagrammatic purposes only. The addresses you would use in a real implementation would be determined by your installation.

Here are some points to note regarding this design:

- ▶ We have not specified whether to use z/VM TCP/IP or Linux guests as the routers. Refer to the discussion on this in Section 6.1.5, “Linux router or z/VM TCP/IP router” on page 96.
- ▶ We are sharing the OSA ports with z/VM TCP/IP. For each of the OSA ports, one of the router images is configured as primary router.
- ▶ A link has been defined for direct communication between the two routers. We discuss this in “Connectivity between the routers” on page 127.
- ▶ The z/VM TCP/IP stack is configured with a VIPA address (for reasons discussed in “z/VM TCP/IP availability” on page 122). In the routing network, you could use static routing to reach the VIPA address via either of the OSAs. Alternatively, you could set up the MPROUTE service machine to advertise the VIPA address using dynamic routing.

Note: The contents of the network “cloud” are generally beyond our control as “penguin keepers.” Sometimes we do not even get an opportunity to specify different network switches to connect our OSAs.

We suggest that you work with your network staff and customers to ensure that the level of redundancy and backup in the environment is technically feasible and meets customer needs.

Connectivity between the routers

The link between the routers is required to provide additional backup connectivity. The method shown in Figure 7-2 on page 126 is one way to do this; the other way would be to cross-connect the two routers and the two OSAs. We discuss this method (as well as an undesirable side-effect) in “Cross-connected OSAs” on page 130.

We could use the Linux guest networks as an alternate communication path between the routers. By defining these interfaces to the dynamic routing protocol, the router images would learn about other paths to the exterior network. However, by doing this we are introducing unwanted broadcast/multicast traffic onto the guest networks. This is not a preferred option.

Our objective is to make the Linux guests as highly available as we can, with minimum disruption. In doing this, not only should we minimize the amount of configuration change on the Linux guests, but we should also reduce the processing work they need to do in failure modes. By giving them every possible chance to use their primary default router (the one that we configure highest in their route table), we minimize the potential disruption at the guest. By linking the two routers, a Linux guest’s primary router will be available through more failure modes than otherwise.

The type of link you use for this is up to you. We would expect that it would be used only in a failure scenario, so extremely high performance is not critical. Since it is point-to-point, a CTC or IUCV link is ideal.

Failure modes

Let us examine the failure modes of this environment.

Failures we cannot recover from

There are some failure modes this configuration cannot recover automatically. Eliminating events that would invoke disaster recovery planning rather than failure management (such as failure of the z/VM system or the processor supporting it, or site power failure), we see the following failure events.

- ▶ Network backbone failure

This is an outright failure of the network at large that causes complete loss of connectivity between the data center and the client population. If there is no way for the client traffic to reach the switches that our OSAs attach to, then no configuration inside our installation will help.

- ▶ Failure of the Linux guest

Obviously, if the guest is broken or down, then network connectivity will not solve the problem.

- ▶ Failure of one OSA and the opposing router image

This effectively removes all connectivity paths for the remaining OSA to reach the exterior network. Manual reconfiguration of the remaining good OSA to the remaining active router would resume connectivity, but it may just be easier to fix the broken router image (which is another reason to keep these images as simple as possible, so that they are easy to repair).

Failure of network switch or router

This will disrupt communication to one of our OSAs. There is not much we can do directly in this scenario, but we rely on the rest of the router network delivering the client traffic to the active OSA.

For outbound traffic, we need to prevent our router guests from sending traffic into a “dead network”; using dynamic routing in this case will help, since the dynamic routing protocol will delete paths via the inactive OSA. If we are not using dynamic routing, we can stop the route being used by simply inactivating the connection to the OSA which is attached to the failed switch.

Failure of one OSA

For inbound traffic, the router network must not direct incoming client traffic to the failed OSA. This can be accomplished automatically using dynamic routing, or manually by reconfiguring the routing tables of network routers.

Outbound traffic will be directed through the active OSA by the router guests.

Failure of connectivity between the OSA and the switch

This failure may be physical (a cable inadvertently unplugged or repatched), or due to the configuration (a switch port reconfigured out of a required VLAN). These problems can be notoriously hard to find.

In a dynamic routing scenario, this will behave according to a combination of the two cases above. Even though the interface is still active, dynamic routing adjacency between the router guest and the network's first-line routers will be lost. Both sets of routers will remove the routing entries through that interface.

Without dynamic routing, the configuration will be reliant upon the dead gateway detection of the respective routing equipment. Manual reconfiguration may be required to force the routers to remove routes through the inactive interface.

Important: It may be tempting to take a “shortcut” in this situation, and inactivate the interface to the suspected failing network. Such an approach, however, may have unintended consequences.

From the router side especially, there may be other equipment attached to the Ethernet interface that is working correctly (because the failure is specifically on the connection to the OSA, for example). Therefore, it is very important that you do not inactivate devices indiscriminately, since further problems may be caused that exacerbate the failure scenario.

Failure of one router guest

When dynamic routing is used, the situation is the same as the failure of an OSA. Loss of adjacency would cause the routers to delete entries from their routing tables.

For outgoing traffic, the scenario will depend on how you configure your Linux guests. If you spread the routing load across the routers (by alternately configuring the router images as the first default router), then approximately half of your guests will see no impact. Guests that use the failed router image as their primary path would rely on dead gateway detection to switch their default route to the active router guest.

Failure of one of the guest networks

If the network is destroyed outright (by an operator actually deleting the simulated LAN, for instance), the NIC will disconnect. Linux will bring the network interface down, forcing traffic to pass through the other network interface. The same will happen at the router end, causing inbound traffic to be directed to the other router image and up through the active guest network.

Failure of a NIC at the router image

For inbound traffic, the same as failure of the guest network. For outbound traffic, same as failure of the router guest.

Failure of a NIC at a Linux guest

For outbound traffic, the Linux guest will use the remaining active network interface.

For inbound traffic, the situation becomes more complex. If the traffic is intended for the IP address of the NIC, the traffic will be lost. If a virtual IP address is used, however, dead gateway detection in the router guest can help (because the router sees the Linux guests' interfaces as gateways to the final destination). The router will move on to the next route configured for the virtual address, which will be the IP address of the Linux guest's other NIC.

Cross-connected OSAs

As an alternative to using a point-to-point connection between the two router images, a configuration such as the one shown in Figure 7-3 on page 131 can be used.

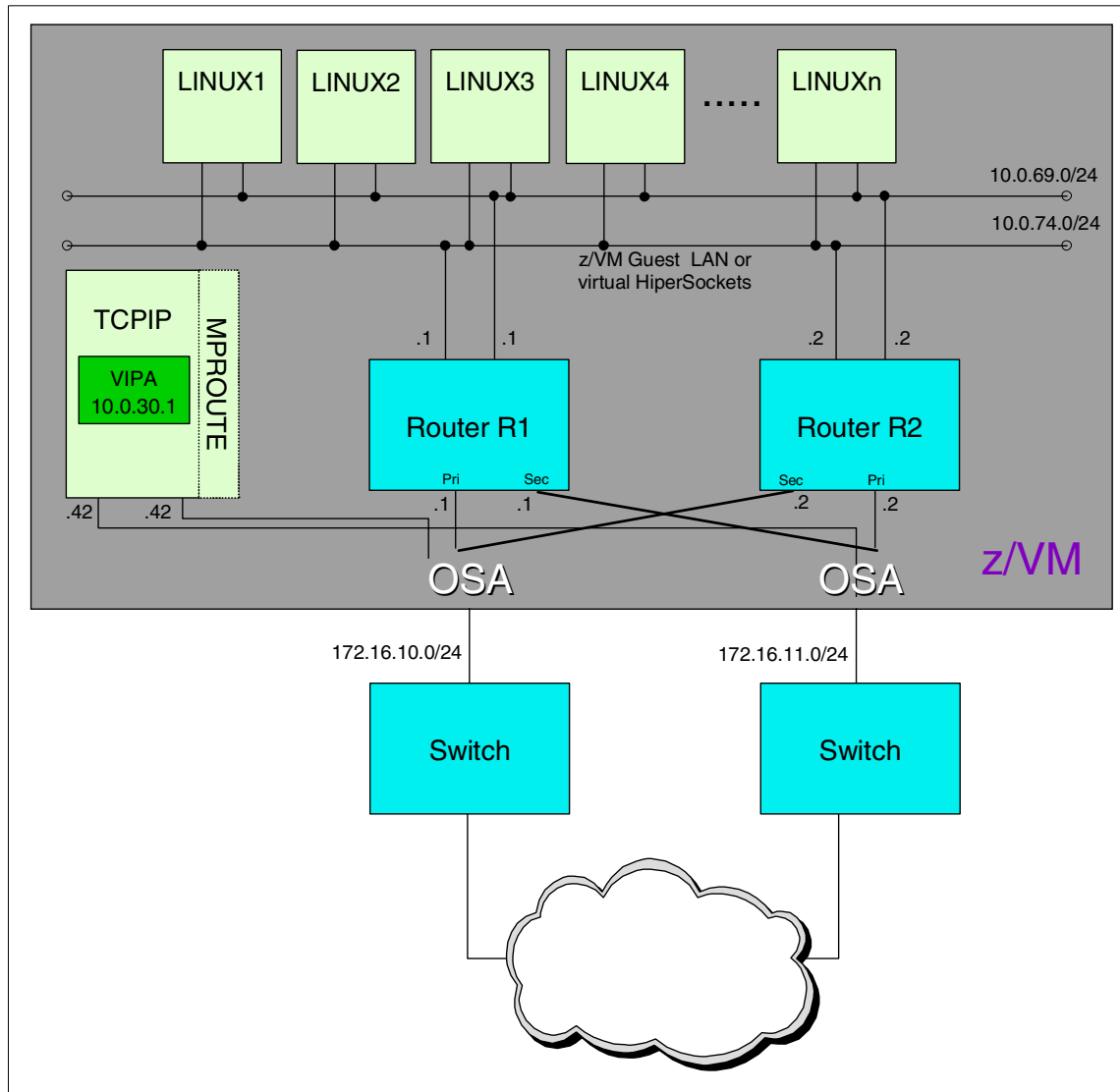


Figure 7-3 Redundant network design 2

Figure 7-3 shows a design similar to the one in Figure 7-2 on page 126, except that both router images connect to both OSAs. Each router activates its OSA devices as either primary or secondary router, so that each OSA has both a primary and secondary router (shown using “Pri” and “Sec” in the diagram). Using the OSA direct routing function, this provides direct connection between the two routers, as well as two paths to the exterior network from each router image. In most cases, this provides better coverage than Figure 7-2.

Unfortunately, this configuration creates a failure mode that would be very difficult to detect and troubleshoot, particularly when dynamic routing is used to the exterior network. The problem arises from an interaction between the dynamic routing processing and the OSA interface.

Picture the network shown in Figure 7-3, using dynamic routing. There are four paths available to get to the router guests, and thereby on to the Linux guest networks. The way that the route tables in the routing network get added is non-deterministic, based simply on the timing of when each router receives and responds to the routing protocol packets. Because of this, we cannot predict which of these four routes will get to the top of the routing table. The problem occurs if the route that gets to the top of the list is a route through a secondary router OSA.

Suppose that traffic was inbound to a Linux guest, and the route chosen via the network was via the address 172.16.11.2 (the left-hand interface on router image R2). Because this is a secondary OSA port, the OSA will actually send the traffic to router R1 because the destination address is unknown and R1 is the primary router. If both of R1's guest network interfaces are up, all will be fine, but if one of R1's interfaces is down (specifically, the interface to our intended network), then there will be a problem. You might think that sending the traffic to R2 via the OSA would be a solution, but again, if the wrong route happens to be at the top of the table (in this case, via 172.16.11.1), then a routing loop is generated.

One way to avoid the effects of this issue is to increase the cost of the secondary links in your dynamic routing configuration. This will make the paths through the secondary definitions less likely to be taken than the primary. However, this approach may have undesirable effects in other parts of the network, forcing traffic onto less preferred paths for the wrong reasons.

Another option is to not define the secondary connections to your dynamic routing protocol. This is actually self-defeating, because if the connections are not defined to dynamic routing, they will not get advertised to the network and will never be taken advantage of by the network. If they cannot be used by the network, they may as well not be defined—and if you don't define them, you don't have the problem!

Because of the caveats that this method introduces, we recommend that you do not cross-connect routers to OSAs in this manner, unless you have a full and complete understanding of the possible implications it may cause.

Multiple default routes

Possibly the biggest issue with a configuration like this is that if both possible network paths are active at the same time, the routing table in the Linux guest will choose one route in preference to the other for all traffic. If the networks beyond

the z/VM complex are asymmetrical, then a suboptimal return path may be chosen for traffic from the Linux guest to the client. Worse still would be a situation where part of the network is reachable via one path but not the other.

One solution to this would be to run a dynamic routing daemon such as **routed**, **mrt** or **zebra** on the Linux guests, to allow them to fully participate in the routing network. This provides the best and most dynamic approach for network connectivity, but is undesirable in a large scale Linux installation due to the amount of CPU and network resource consumed by routing updates.

We consider that it is useful to have your router guests participate in dynamic routing with the network at large, but that it is not desirable to use dynamic routing protocols within the Linux server complex (dynamic routing in large scale Linux installations is discussed more in Section 6.6, “Dynamic routing” on page 116).

Another possibility is to use **rdisc**, the implementation of the Router Discovery Protocol (RDP). There are two advantages to RDP over full dynamic routing protocols:

- ▶ RDP is more directed than full dynamic routing protocols. Generally in OSPF or RIP domains, all routers are peers and share information with all other routers. With RDP, the routers providing routing service are designated differently to the hosts requesting routing configuration. This reduces the amount of routing information being distributed.
- ▶ The default broadcast interval is longer. Routers running RDP only advertise routing information every 10 minutes, by default. This reduces the overall load generated by routing updates (but it does create CPU load spikes).

There are considerations with using RDP, however:

- ▶ RDP is fairly easy to spoof, creating the possibility of a DoS attack. You can block RDP message traffic at your firewall, thus preventing an attack from outside your system, but you may still be vulnerable to attacks from other guests within your system.
- ▶ The default advertisement interval may be too long for your needs, creating extended downtime compared to using a full routing protocol.

Attention: If you want to experiment with RDP, you will obviously want to have routers in your environment advertising RDP updates and answering RDP solicitations. The **in.rdisc** man page as installed on our SuSE system stated that the command **in.rdisc -r** would start the **rdisc** program in “router mode”, but the **rdisc** program as shipped with SuSE does not support this switch. The **gated** program implements RDP, but it is not available as part of the SuSE distribution.

7.2.2 Virtual Router Redundancy Protocol (VRRP)

The Virtual Router Redundancy Protocol (VRRP) is an open protocol that provides a way for router services to be provided to servers or clients.

How VRRP works

VRRP routers provide a “single router image” to clients and other routers. A daemon on the router image listens for the multicast messages of other VRRP routers. If it does not receive any, it assumes that there are none active and takes on the role of “master router”. It adds the virtual router IP address to the interface that attaches to the network the router is providing routing service for, and advertises its details to the network for other VRRP daemons.

If a problem occurs with that router guest, another router running the VRRP daemon will see that the master has not sent its multicast message and take over the master role. There will be an interval when the clients will not be able to reach a default router (while the other routers detect the loss of the master and reconfigure), but this is configurable in the VRRP daemon based on how often the master will send its multicast.

Experiences with VRRP

We obtained the source code for VRRP, built it, and attempted to use it. The configuration we had intended looked like Figure 7-4 on page 135.

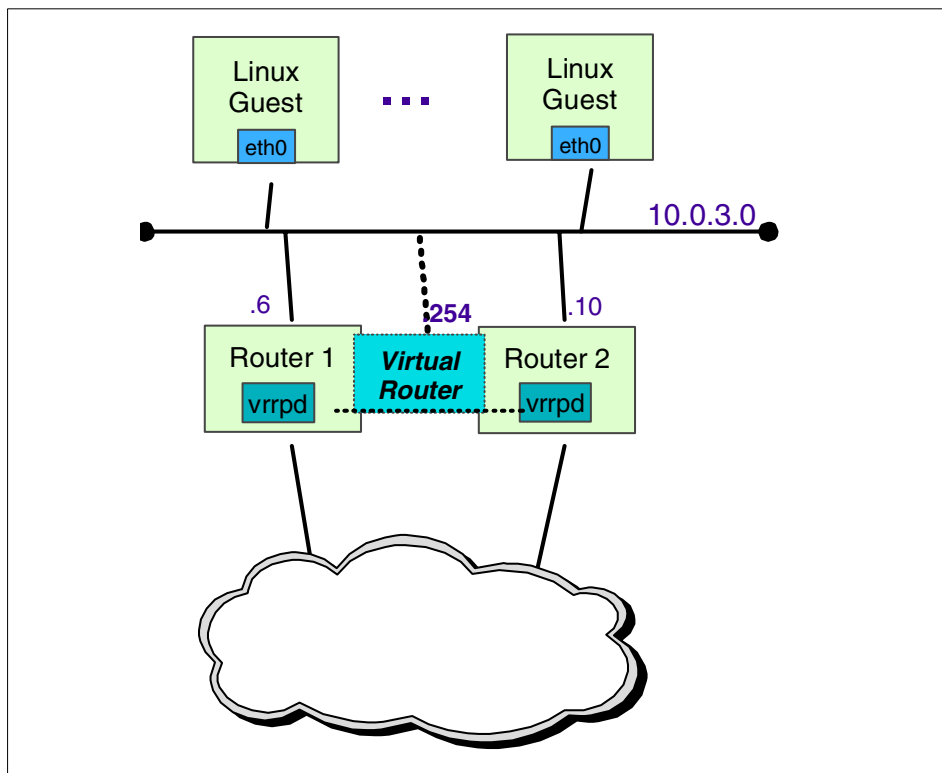


Figure 7-4 Intended VRRP configuration

The two Linux routers would implement the VRRP router 10.0.3.254, which we would then be able to configure as the default router on all the Linux guests. VRRP would ensure that at least one of the router images would always respond to the virtual router IP address.

Unfortunately, we encountered problems with VRRP which we were unable to resolve. When we started the daemon on the first router, it successfully registered the virtual router IP address and started advertising to other VRRP daemons. However, every time a broadcast frame was sent, we got the error messages shown in Example 7-1 in the system log:

Example 7-1 vrrpd: Messages from qeth.o

```
qeth: QETH_IP_VERSION is 0
qeth: skb->protocol=x0=0
qeth: skb:01 00 5e 00 00 12 00 00 00 00 00 00 08 00 45 00
qeth: skb:00 28 00 17 00 00 ff 70 ce 36 0a 00 03 06 e0 00
```

When we started the VRRP daemon on a second router, it appeared as if it was not receiving the multicast advertisements from the master router. Therefore, it assumed there was no other VRRP daemon active, and attempted to become the master. When it tried to register the virtual router IP address, it failed because the first router already was on the network with the address.

Obviously, takeover never occurred either because the second VRRP daemon already believed it was the master.

7.2.3 Virtual IP addresses

The IBM Redbook *Linux on IBM @server zSeries and S/390: ISP/ASP Solutions*, SG24-6299, discusses a method of using multiple attachments in combination with a virtual IP address configured on a dummy interface. This still can form the basis of a high availability connectivity solution. As an alternative to using a dummy interface, you can configure additional addresses against the IP loopback interface (lo). This means you don't need to have another module installed.

Important: The way that the Linux kernel processes incoming IP traffic does not require the virtual IP address to be associated with a dummy or loopback interface. Your virtual address can be just as easily defined against one of your real interfaces. However, to ease possible confusion, we use a different interface to make it clear that the address has nothing to do with any particular physical network device.

The following command will configure the IP address 10.0.100.106 as an alternate IP address bound to the loopback interface:

```
ifconfig lo:0 10.0.100.106 netmask 255.255.255.255 up
```

You can use the `ifconfig` command again to verify the command; see Example 7-2 on page 137.

Example 7-2 *ifconfig* command output

```
lnx6:~ # ifconfig lo:0
lo:0      Link encap:Local Loopback
          inet addr:10.0.100.106  Mask:255.255.255.255
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
```

The **ip** command can be used to achieve a similar result. The syntax for the equivalent **ip** command is shown here:

```
ip addr add 10.0.100.106/32 dev lo
```

When you use the **ip** command to add an address in this way, you will not be able to use the **ifconfig** command to view the address. The way that **ip** adds additional addresses does not appear to be compatible with **ifconfig** (largely due to the BSD heritage of **ifconfig**, compared to the redeveloped Linux IP code introduced with Linux kernel 2.2). You can use the **ip** command to view your address configuration, however.

Example 7-3 *ip addr list* command output

```
lnx6:~ # ip addr list dev lo
1: lo: <LOOPBACK,UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet 10.0.100.106/32 scope global lo
    inet6 ::1/128 scope host
```

Attention: Because of this apparent incompatibility between **ifconfig** and **ip**, we recommend that you make a site-wide decision on which tool and which configuration method you will use across all your Linux guests. This will ensure that your system administrators don't get confused by output they don't expect.

7.2.4 IP connections outbound from Linux guests

If your Linux guest generates outgoing connections to other servers, the dummy interface solution will not provide complete redundancy. This is because by default, when an application does not specify what source address is to be used, the Linux kernel uses the IP address of the most direct interface to the destination as the source address of the connection. If that interface becomes unreachable, the connection will fail. Most applications will deliberately not specify a source address, since doing so relieves the application of the burden of finding out what the address should be.

The ISP/ASP book presents a solution based on Network Address Translation (NAT) using iptables as a way of rewriting the source address of IP packets leaving the Linux guest. There are at least two other ways of performing a similar function.

Understanding Linux outbound address selection

The Linux IP code uses a complex process to determine the source address to be used for an outgoing connection. The final default it will choose (if the application does not specify an address to use) is determined by the routing table, and refers to the interface associated with the route the kernel chooses to send the traffic. The earlier processing steps can be controlled using the `ip` command, which we discuss in the next section.

Another way to control the source address is for the application program to specify one in its call to the `bind()` function. As mentioned earlier, most applications do not do this because in most situations it is best to leave the decision up to the kernel.

Note: Samba is one application that does provide options to specify the bind address.

A utility called `src_vipa` has been written, which allows an application which does not normally specify a source address to be set up to do so. We discuss the `src_vipa` utility in “`src_vipa`” on page 139.

IP rules using the IP utility

The `ip` command allows the routing table to be manipulated to a greater degree than the traditional `ifconfig` and `route` commands. Using `ip`, you can define a rule that will override the source address for outgoing traffic.

You can selectively apply rules to traffic on the interface, allowing you to change which addresses are changed. If you want traffic for different networks to be treated differently, this can be done by creating alternate route tables with the changes you want, and then specifying rules that determine which table will be used for which traffic.

This function is similar to the NAT solution presented in the ISP/ASP book, in that the behavior of the application is not changed.

The table is applied on all IP packets that match the rule, which means that all traffic from a given interface (or all traffic to a specified network, if you are using alternate tables) is changed regardless of application or protocol (for example, TCP or UDP).

Using the ip command

src_vipa

Utz Bacher from IBM Böblingen has written a utility called `src_vipa` that allows you to modify an application's behavior when it does not specify a source address. The function of `src_vipa` is based on the Source VIPA function provided by the Communications Server TCP/IP Feature on z/OS, z/VM and OS/390.

Together with a dummy or loopback IP address as described in 7.2.3, "Virtual IP addresses" on page 136, the `src_vipa` utility can be very useful in providing more deterministic connectivity when multiple interfaces are used.

Note: `src_vipa` can be downloaded from the Developerworks Linux for zSeries and S/390 'Useful Add-Ons' page:

http://oss.software.ibm.com/linux390/useful_add-ons.shtml

The `src_vipa` package provides a wrapper script which is used to invoke a program that you would like to benefit from the source VIPA function. The wrapper script sets up an alternate IP library which is called by the application, and fills in default values, before passing the call on to the system library.

The behavior of `src_vipa` is controlled by a configuration file which, by default, is found at `/etc/src_vipa.conf`. Alternatively, the environment variable `SRC_VIPA_CONFIG_FILE` can override the default location, allowing you to have different configuration files for different situations or for different application programs. The package provides a man page that outlines the configuration options and how to use them.

src_vipa in use

We set up a virtual address on one of our Linux guests, and experimented with `src_vipa`. Our `src_vipa.conf` file is shown in Example 7-4.

Example 7-4 src_vipa.conf

```
# src_vipa.conf
# configuration file for the src_vipa utility
#
9.12.6.0/23 10.0.100.106
.INADDR_ANY 1-623 10.0.100.106
```

This example configuration will ensure both of the following:

- ▶ All connections to the 9.12.6.0/23 network are made with a source address of 10.0.100.106.

- ▶ Any program that binds a socket on ports between 1 and 623 inclusive without specifying a source address will also use a source address of 10.0.100.106.

Refer to the `src_vipa` man page for complete information about the contents of the configuration file.

To demonstrate the first statement in the configuration file, we used the Telnet program to connect to a remote machine. Telnet will not specify an IP address to bind the socket to, so the kernel will choose one according to the route table. Our route table looked as shown in Example 7-5.

Example 7-5 Linux route table for Source VIPA testing

```
lnx6:/etc/init.d # netstat -r
Kernel IP routing table
Destination      Gateway          Genmask         Flags   MSS Window  irtt Iface
10.0.6.0         *               255.255.255.0  U       40  0        0 hsi1
10.0.3.0         *               255.255.255.0  U       40  0        0 eth0
default         10.0.3.1        0.0.0.0        UG      40  0        0 eth0
```

Without `src_vipa`, the information in Example 7-6 is shown about our Telnet session.

Example 7-6 netstat output for a non-VIPA telnet session

```
lnx6:/etc/init.d # netstat -a | grep telnet |grep EST
tcp        0      0 10.0.3.6:1270      9.12.6.83:telnet  ESTABLISHED
```

The kernel chose the address 10.0.3.6 because it is the address of the interface associated with device `eth0`, which is the interface we use to reach the destination (via the default route).

We closed that session, then started the telnet program through the `src_vipa.sh` wrapper script with the following command:

```
/usr/sbin/src_vipa.sh telnet 9.12.6.83
```

When we looked at the telnet session, we saw that the VIPA address was used as the source this time; see Example 7-7.

Example 7-7 netstat output for a VIPA telnet session

```
lnx6:/etc/init.d # netstat -a | grep telnet |grep EST
tcp        0      0 10.0.100.106:1271  9.12.6.83:telnet  ESTABLISHED
```

Tip: It may seem tedious to have to start every IP program through a wrapper script. A much easier way is to copy the contents of the `src_vipa.sh` script to the local profile for those users you wish to get source VIPA function. If you want to give all users source VIPA function, copy to the `/etc/profile.local` file instead.

The two lines in `src_vipa.sh` that you need to copy are as follows:

```
export LD_LIBRARY_PATH=/usr/lib:$LD_LIBRARY_PATH
export LD_PRELOAD=/usr/lib/src_vipa.so
```

The first line adds the path to the `src_vipa` library to the shell's library path, and the second preloads the library so that the system's library is not used.

For a demonstration of the second configuration line in `/etc/src_vipa.conf`, we modified the startup script for the `inetd` daemon, as per the diff output shown in Example 7-8.

Example 7-8 Alteration to `inetd` startup script

```
--- inetd.backup      Wed Jul 31 11:35:11 2002
+++ inetd             Wed Jul 31 13:46:32 2002
@@ -69,7 +69,7 @@

        # startproc should return 0, even if service is
        # already running to match LSB spec.
-       startproc $INETD_BIN
+       startproc /usr/sbin/src_vipa.sh $INETD_BIN

        # Remember status and be verbose
        rc_status -v
```

Prior to restarting `inetd` using this modified script, we verified the addressing of the sockets in use by `inetd`; see Example 7-9.

Example 7-9 `inetd` socket usage, before `src_vipa`

```
lnx6:/etc/init.d # netstat -ap | grep inetd
tcp        0      0 *:login          *:*                LISTEN      30604/inetd
tcp        0      0 *:time           *:*                LISTEN      30604/inetd
tcp        0      0 *:echo           *:*                LISTEN      30604/inetd
tcp        0      0 *:discard        *:*                LISTEN      30604/inetd
tcp        0      0 *:daytime        *:*                LISTEN      30604/inetd
tcp        0      0 *:finger         *:*                LISTEN      30604/inetd
tcp        0      0 *:chargen        *:*                LISTEN      30604/inetd
tcp        0      0 *:ftp            *:*                LISTEN      30604/inetd
tcp        0      0 *:telnet         *:*                LISTEN      30604/inetd
```

udp	0	0 *:talk	*:*	30604/inetd
udp	0	0 *:ntalk	*:*	30604/inetd
udp	0	0 *:echo	*:*	30604/inetd
udp	0	0 *:discard	*:*	30604/inetd
udp	0	0 *:daytime	*:*	30604/inetd
udp	0	0 *:chargen	*:*	30604/inetd
udp	0	0 *:time	*:*	30604/inetd

We then restarted inetd, and reran the `netstat` command; see Example 7-10.

Example 7-10 inetd socket usage, after src_vipa

```
lnx6:/etc/init.d # netstat -ap | grep inetd
```

tcp	0	0 10.0.100.106:login	*:*	LISTEN	30635/inetd
tcp	0	0 10.0.100.106:time	*:*	LISTEN	30635/inetd
tcp	0	0 10.0.100.106:echo	*:*	LISTEN	30635/inetd
tcp	0	0 10.0.100.106:discard	*:*	LISTEN	30635/inetd
tcp	0	0 10.0.100.106:daytime	*:*	LISTEN	30635/inetd
tcp	0	0 10.0.100.106:finger	*:*	LISTEN	30635/inetd
tcp	0	0 10.0.100.106:chargen	*:*	LISTEN	30635/inetd
tcp	0	0 10.0.100.106:ftp	*:*	LISTEN	30635/inetd
tcp	0	0 10.0.100.106:telnet	*:*	LISTEN	30635/inetd
udp	0	0 10.0.100.106:talk	*:*		30635/inetd
udp	0	0 10.0.100.106:ntalk	*:*		30635/inetd
udp	0	0 10.0.100.106:echo	*:*		30635/inetd
udp	0	0 10.0.100.106:discard	*:*		30635/inetd
udp	0	0 10.0.100.106:daytime	*:*		30635/inetd
udp	0	0 10.0.100.106:chargen	*:*		30635/inetd
udp	0	0 10.0.100.106:time	*:*		30635/inetd

As you can see, all of the inetd applications are now bound to the VIPA address.

Attention: Use this option with care. Daemons which are bound to a particular address will accept incoming connections *only* on the address they are bound to. For example, in the above scenario we can no longer use the interface address (or even the loopback address) to connect to any of the inetd services, as inetd is only listening on the VIPA address.

You can use this option to start multiple daemons for the same application on different IP addresses. You might do something like this to provide daemons with different configurations running on the same host—each would be bound to a different VIPA.

The `src_vipa` utility is a useful way to enable the source VIPA functionality at an application level without manipulating system routing tables.

7.3 Redundancy outside the zSeries complex

By introducing additional networking and system intelligence outside a single zSeries system, we can make our Linux guests more available.

7.3.1 Additional z/VM system

We can utilize a second zSeries processor to provide a backup environment for our large scale Linux installation. This gives us options for load sharing, in addition to high availability.

In the following scenario, a parallel image of our z/VM system with its operational Linux guests is set up on a second zSeries processor. We have three choices for how to run this second installation:

- ▶ Warm backup

In this case, z/VM is operational, but no Linux guests are started. In the event of a failure of the primary z/VM system, the Linux disk is connected to the secondary z/VM system (either by reconfiguration of channels, or by restoring disk images from backup media) and the Linux guests are started.¹

- ▶ Hot backup

In this case, z/VM and all Linux guests are operational, but no client load is directed to the backup system. This would require some method of synchronization between the Linux disk on the two different z/VM systems.

- ▶ Parallel operation

In this case, the backup z/VM system is running as per the “hot backup” scenario, but in a load-sharing parallel operation configuration. In addition to the disk synchronization requirements, an external means of sharing client connections between the two systems is now required.

7.4 Linux high availability solutions

A number of high availability solutions exist for Linux. As stated in the introduction to this chapter, some of these solutions can be applied to large scale Linux installations, but others do not (for various reasons). Here, we discuss some of the more popular high availability Linux solutions.

¹ If the method of disk synchronization does not involve a z/VM system on the backup machine, there would be little advantage in even having z/VM running in this scenario; therefore, it might also be described as “cold backup”.

7.4.1 To cluster or not to cluster

Clustering for high-availability is becoming extremely popular, so it may be surprising to some that the situation changes somewhat in the case of Linux under z/VM.

Clustering across z/VM systems

In 7.3.1, “Additional z/VM system” on page 143, we introduce the idea of using a second z/VM installation to provide redundancy of your large scale Linux installation. In the parallel operation scenario, you could use clustering technology to bring together Linux guests in two separate z/VM systems so that they share load and provide backup for each other. Some of the techniques discussed in the following sections could be used for this purpose.

We agree that clustering in this scenario is a useful thing to do in almost all cases.

Clustering within a z/VM system

The benefit of clustering within a single z/VM system, however, is arguable. The current crop of clustering technologies all employ a single “load balancer” machine which distributes connections to back-end “worker” servers.

In a discrete server environment, these are individual machines with no dependencies on each other. In a z/VM environment, however, the load balancer machine, as well as all the back-end servers, share the same processing resource. Does it make sense to subdivide the physical machine into smaller machines, increasing the processing overhead as well as the overall complexity of the installation? Why not have a single, large guest do the work?

We believe there are reasons you might want to use clustering in this case. Some of these reasons are:

- ▶ Comparative cost of “virtual clusters” and discrete server clusters

In a discrete server scenario, a clustering configuration will involve purchasing extra hardware and increase environmental costs (floor space, electricity, cooling, etc.). The added cost raises the point at which customers can cost-justify a cluster configuration.

In a Linux on z/VM environment, a cluster configuration adds nothing to the hardware or environmental cost, so the cost-justification threshold can be much lower.

- ▶ Consistency across Linux guest configurations

If your installation uses standard processes and configurations for guests, it may not be convenient to disrupt those processes if, for example, you have to change one or two guests to give them more storage.

Using clustering might allow you to spread the application load across two or more guests using a standard configuration.

- ▶ More efficient workload scheduling

A single large Linux guest handling multiple concurrent tasks will become processor-bound as the multiple tasks fight with each other for CPU timeslice. You can avoid this by adding virtual CPUs to the guest, allowing Linux to schedule more tasks concurrently, but that might add overhead due to both Linux and z/VM scheduling the multiple CPU workload within the one guest. Spreading the workload across multiple guests in a cluster can even out this effect by letting z/VM schedule individual guests as separate units of work.

- ▶ I/O processing advantages.

At present, the I/O model for Linux under z/VM prevents a guest from having more than one outstanding I/O to a single device. This is mainly important for disk I/O, particularly as disk devices tend toward larger sizes (such as 3390 “Model 27”). The larger a guest is, and the more work it performs, the more likely it is to have multiple processes attempting I/O to the same device. Dividing this processing into smaller clustered guests may provide a way to avoid I/O contention and improve overall performance.

With the network performance available with Guest LAN and HiperSockets, network latency would not be a significant issue in a cluster configuration under z/VM. You can also quite easily define a separated Guest LAN for heartbeat and keepalive processing, if your particular clustering technology requires it (you could even use the TTY mode of the CTC driver for this, allowing you to use a discrete server serial-link heartbeat solution with almost no change).

On this topic, we cannot give a clear recommendation. We leave it up to you to determine what configuration best suits your application load and service requirements.

7.4.2 Linux Virtual Server

In the IBM Redbook *Linux on IBM @server zSeries and S/390: Distributions*, SG24-6264, there is a discussion of a sample implementation of the Linux Virtual Server (LVS) concept. LVS ties a group of server machines together using various network and scheduling techniques, so that they present a single server “image” to clients requesting service. A single load balancing and scheduling machine initially receives all the traffic, which it then sends to back-end servers.

Three network techniques are used in an LVS installation:

- ▶ Network Address Translation (LVS-NAT)
Packets arriving at the LVS load balancer are address-translated to the IP address of one of the back-end servers handling the application. When the response is generated, the load balancer reverses the address translation so that the load balancer appears to have handled the request.
- ▶ IP Tunneling (LVS-TUN)
Packets arriving at the LVS load balancer are “tunnelled” to a back-end server, which processes the request. The back-end server responds to the client directly.
- ▶ Direct Routing (LVS-DR)
Packets arriving at the LVS load balancer are redirected to one of the back-end boxes. The virtual IP address of the cluster is configured on the interfaces of all of the machines in the cluster so that the kernel will correctly receive packets intended for the cluster.

LVS-TUN and LVS-DR have the advantage that the response traffic is sent directly from the back-end server to the client. LVS-NAT requires this outbound traffic to pass through the load balancer. LVS-NAT works around a tricky ARP configuration issue present in the other two options, however.

A complete analysis of LVS is beyond the scope of this book. We recommend that you read the relevant section of the Distributions redbook to learn more about how LVS works on Linux under z/VM.

Note: Another work on high availability on Linux on zSeries under z/VM that we recommend is the Redpaper *Linux on IBM zSeries and S/390: High Availability for z/VM and Linux*, REDP0220, available from

<http://www.redbooks.ibm.com/abstracts/redp0220.html>

Creating and managing a penguin colony

In this part we introduce concepts to aid in the management a Linux server farm. Topics include:

- ▶ Sharing Linux filesystems running under z/VM
- ▶ Building Linux guests as clones operating on shared Linx filesystems
- ▶ Managing a Linux server farm using LDAP.



Shared Linux filesystems

In this chapter we discuss a method of sharing Linux filesystems created on VM minidisks among multiple z/VM Linux guests.

We introduce the theory and provide a sample implementation. You can find the scripts discussed in this chapter in the additional materials available with this redbook.

8.1 Device filesystem mounts

Device filesystem mounts incorporate a filesystem on a block device into the Linux global filesystem namespace.

For example, we consider a filesystem on block device `/dev/dasdc1`. The filesystem contains a `/lib` directory containing a file named `foo.bar`. Upon issuing the following command, the file `foo.bar` in the `/lib` directory on the `/dev/dasdc1` device may be referenced using the global path name `/usr/local/lib/foo.bar`:

```
mount /dev/dasdc1 /usr/local
```

Files and directories which existed in the original `/usr/local` subtree can no longer be accessed by their original path names. In practice, there are no files or directories in the original `/usr/local` directory subtree (as it is created explicitly as a “stub” directory solely for the purpose of being a mount point).

Note: Processes which have open files or have a current working directory in `/usr/local` subtree continue to be able to manipulate those files and directories as usual. The Linux kernel reference counts these open files to ensure access is predictable and safe.

8.2 Bind mount directories

A *bind mount* expands the functionality of the device filesystem mount. Using bind mounts, it is possible to graft a directory subtree from one part of the global filesystem namespace to another. Bind mounts differ from device mounts in that the source is the global filesystem namespace itself - not a block device.

As an example, consider a directory `/guestvol/there` containing a file named `foo.bar`. An additional directory `/mnt/here` exists in the global namespace. Issue the following command:

```
mount --bind /guestvol/there /mnt/here
```

Now the `foo.abc` file can be referenced by two path names:

<code>/guestvol/there/foo.abc</code>	The original path name
<code>/mnt/here/foo.abc</code>	The bind mount path name

This is illustrated in Figure 8-1 on page 151.

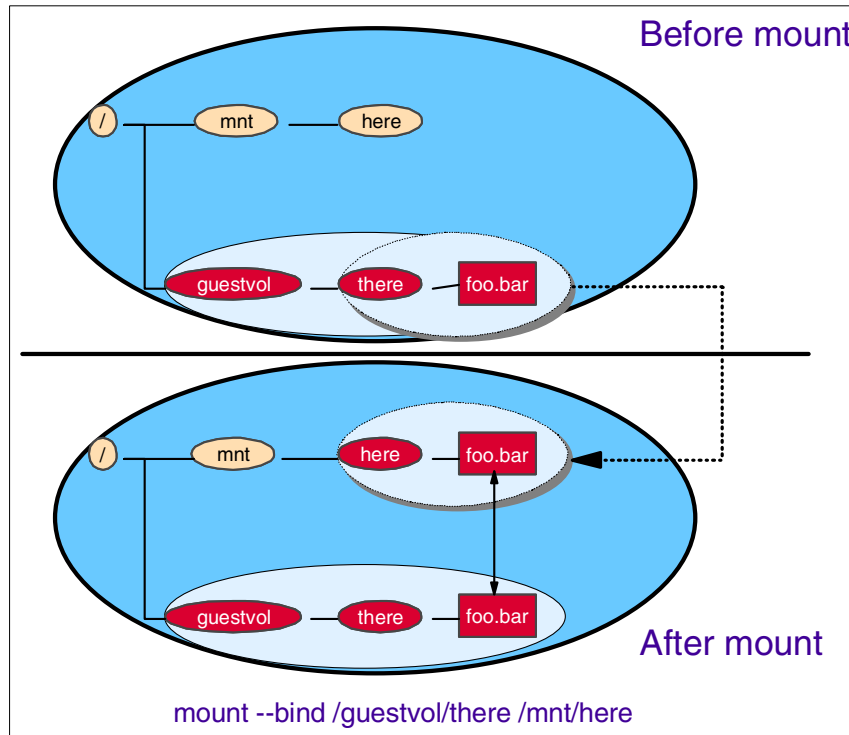


Figure 8-1 The behavior of bind mount

Both names refer to the same underlying file. The Linux kernel maintains coherence and consistency whichever name is used. We refer to this as a “bind mount” in this redbook.

The bind mount feature is available in the Linux 2.4 kernel. It requires no special configuration options. Bind mounts were inspired by similar functionality in the Plan9 operating system.

Note: Plan9 is an operating system developed at Bell Labs by a group that includes Ken Thompson (one of the original inventors of UNIX) and Rob Pike (who also has a long UNIX heritage). For more details, see:

<http://plan9.bell-labs.com/plan9dist/>

Yes, Plan9 did take its name from the film *Plan 9 From Outer Space*.

8.3 Using bind mounts

Bind mounts offer some interesting possibilities for Linux guests running under z/VM. Disk partitions seen by Linux guests are minidisk allocations defined to the guest virtual machine. Because minidisks may be shared by many virtual machines, it is possible to use bind mounts to define a common, read-only root filesystem shared by many guests in a z/VM Linux cluster.

In Figure 8-2, we show the effect on the global filesystem namespace of mounting a read-write filesystem (residing on a read-write block device) on a read-only root filesystem.

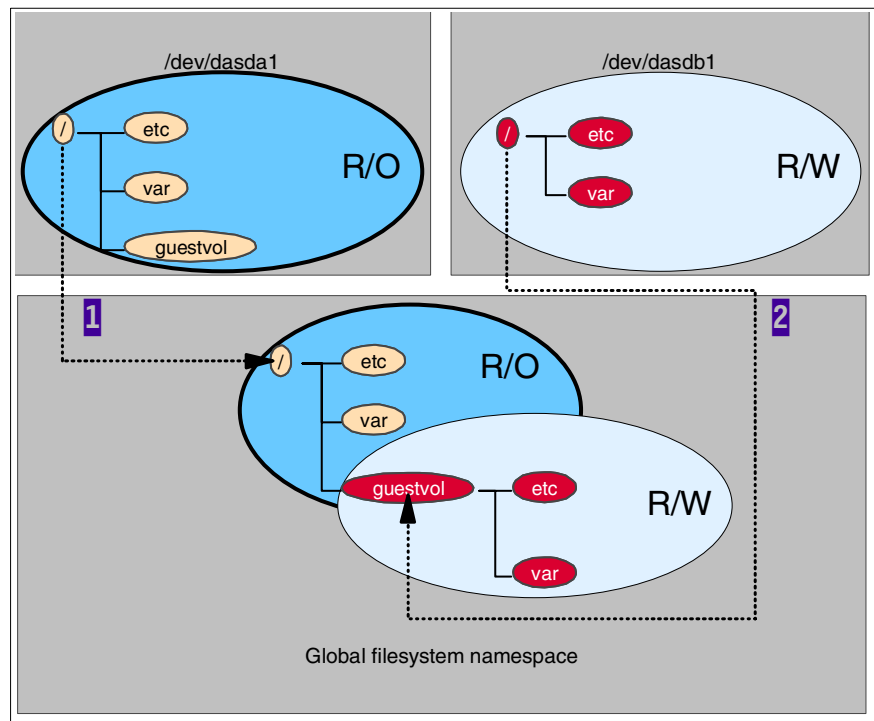


Figure 8-2 Mounting disk partitions on a global filesystem namespace

Note: For simplicity, we illustrate only a portion of the filesystems which may exist on the `/dev/dasda1` and `/dev/dasdb1` disk partitions.

The steps shown in Figure 8-2 on page 152 are described here:

1. The root filesystem is initially mounted from the read-only `/dev/dasda1` device:

```
mount /dev/dasda1 /
```

2. The `/dev/dasdb1` device is mounted on the `/guestvol` stub directory:

```
mount /dev/dasdb1 /guestvol
```

Note: These commands are normally executed as part of the Linux boot sequence; device and mount points are specified in the `/etc/fstab` file.

As shown in Figure 8-2 on page 152, the `/guestvol` filesystem is read-write accessible, while the root filesystem is read-only accessible.

8.3.1 Mounting writable directories on a read-only filesystem

Continuing the example, in Figure 8-3 we:

- ▶ Mount the read-write directories `/guestvol/etc` over the read-only `/etc`:

```
mount --bind /guestvol/etc /etc
```

- ▶ Mount the read-write `/guestvol/var` over the read-only `/var`:

```
mount --bind /guestvol/var /var
```

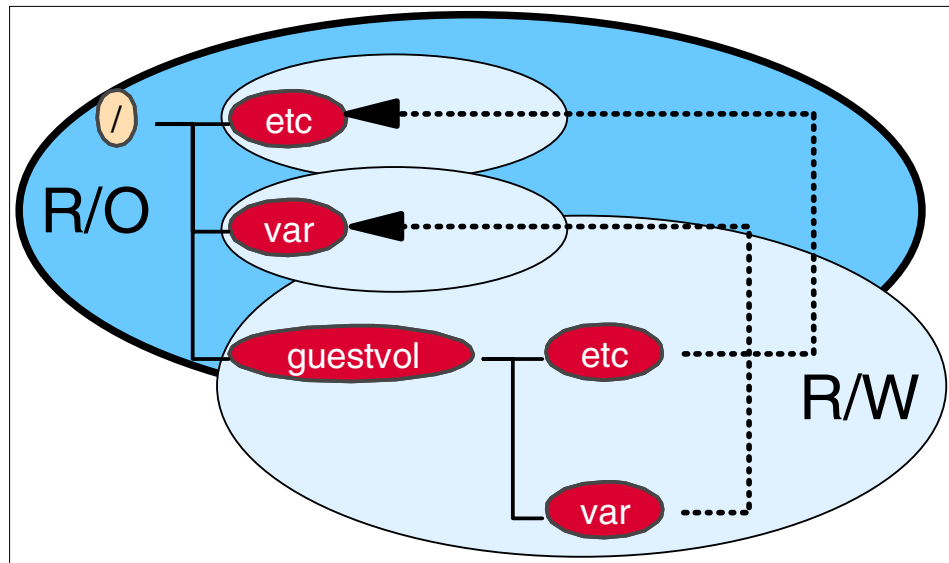


Figure 8-3 Mounting read-write directories on a read-only filesystem

The path name `/etc` now refers to `/guestvol/etc`. Similarly, now `/var` refers to `/guestvol/var`. Files and directories under the `/etc` and `/var` subtrees map to their corresponding locations under the `/guestvol` subtree.

Important: Any files or directories that existed in the `/var` or `/etc` subtree prior to the bind mounts are now hidden from the global filesystem namespace.

8.3.2 Preserving access to the original read-only directories

As noted, files and directories in the directory trees overlaid by bind mounts will be hidden from the global namespace. It can be useful to preserve access to those subtrees.

Note: Although directory trees overlaid by bind mounts are hidden from the global namespace, processes holding references to open files or directories can still access entities.

As illustrated in Figure 8-4, preserve the `/etc` and `/var` read-only directories:

```
mount --bind /etc /basevol/etc
mount --bind /var /basevol/var
```

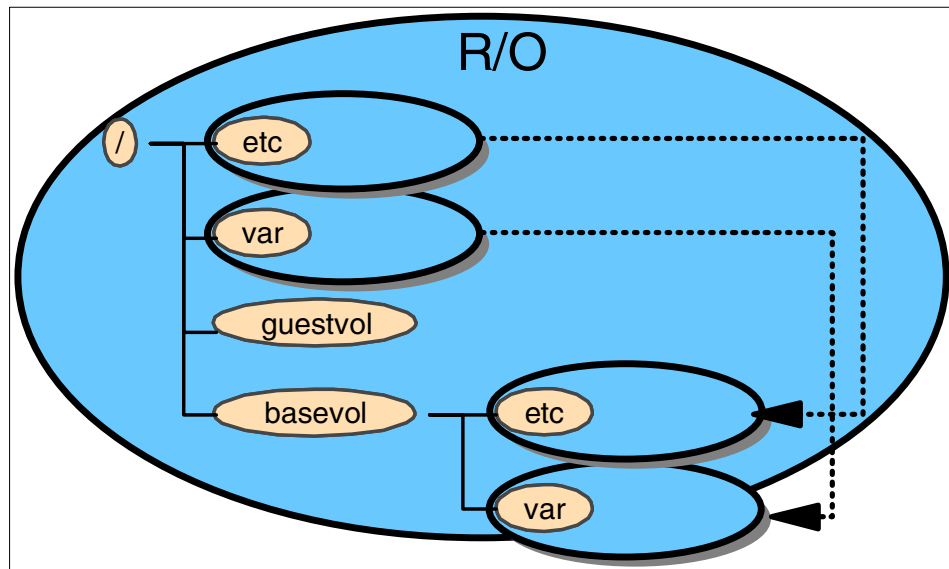


Figure 8-4 Preserving read-only directories using bind mounts

Note: For simplicity, we do not show the read-write device mounts on the `/guestvol` directory.

The `/basevol/etc` and `/basevol/var` directories serve as stub directories for bind mounts. At this point, we can bind mount the read-write `/guestvol/etc` and `/guestvol/var` directories onto the `/etc` and `/var` directories as discussed in 8.3.1, “Mounting writable directories on a read-only filesystem” on page 153. Access to the original read-only copies can be obtained using the `/basevol/etc` and `/basevol/var` path names.

8.4 The basevol filesystem

A base volume or *basevol* filesystem is a bootable root filesystem residing on a VM read-only minidisk shared by many Linux guests. The basevol filesystem contains a set common packages and services needed by all Linux guests in a penguin colony.

A basevol contains a majority of the packages required to operate any single Linux guest in the colony. Directories found on the basevol include:

- ▶ `/sbin`
- ▶ `/bin`
- ▶ `/lib`
- ▶ `/usr`
- ▶ `/var/lib/rpm`

8.5 The guestvol filesystem

A guest volume or *guestvol* filesystem is a filesystem residing on a VM read-write minidisk dedicated to a single Linux guest running in a penguin colony.

The guestvol contains the packages and configuration files needed to personalize a single Linux guest in a colony. A guestvol may require as little as 15 MB of DASD space. Directories found on the guestvol include:

- ▶ `/etc`
- ▶ `/var`
- ▶ `/home`
- ▶ `/opt`
- ▶ `/dev`
- ▶ `/tmp`
- ▶ `/boot`

- ▶ /root
- ▶ /usr/local

8.6 A basevol/guestvol Linux guest

A basevol/guestvol Linux guest is a member of a penguin colony that uses both basevol and guestvol filesystems. Directories on the read-write guestvol are bind-mounted over their respective location on the basevol.

8.7 The File Hierarchy Standard

The File System Hierarchy (FHS) defines a standard for managing a directory hierarchy. In particular, it defines which portions of directory structure should be read-only and which should be read-write. The current version (2.2) can be found at:

<http://www.pathname.com/fhs/>

Discussion of the FHS is beyond the scope of this book. However, it serves to assist in determining how to partition the hierarchy into basevol and guestvol images.

8.8 RPM package management

By default, RPM manages installed packages using database files located in the `/var/lib/rpm` directory. However, the `/var` guestvol filesystem is bind-mounted over `/var`. Therefore, the `/var/lib/rpm` directory refers to the guestvol filesystem.

To prevent a basevol/guestvol Linux guest from unintentionally installing packages on the guestvol, we want the `var/lib/rpm` directory to refer to the basevol filesystem.

Using bind mounts, we do the following:

1. Preserve the basevol `/var` directory in the `/basevol/var` directory:
`mount --bind /var /basevol/var`
2. Bind-mount the `/guestvol/var` directory over `/var`:
`mount --bind /guestvol/var /var`
3. Bind-mount the `/basevol/var/lib/rpm` directory over `/var/lib/rpm`:
`mount --bind /basevol/var/lib/rpm /var/lib/rpm`

Figure 8-5 shows the read-only basevol with a device-mounted guestvol (mounted on the /guestvol directory). Both the filesystems contain a /var/lib/rpm/foo.bar file.

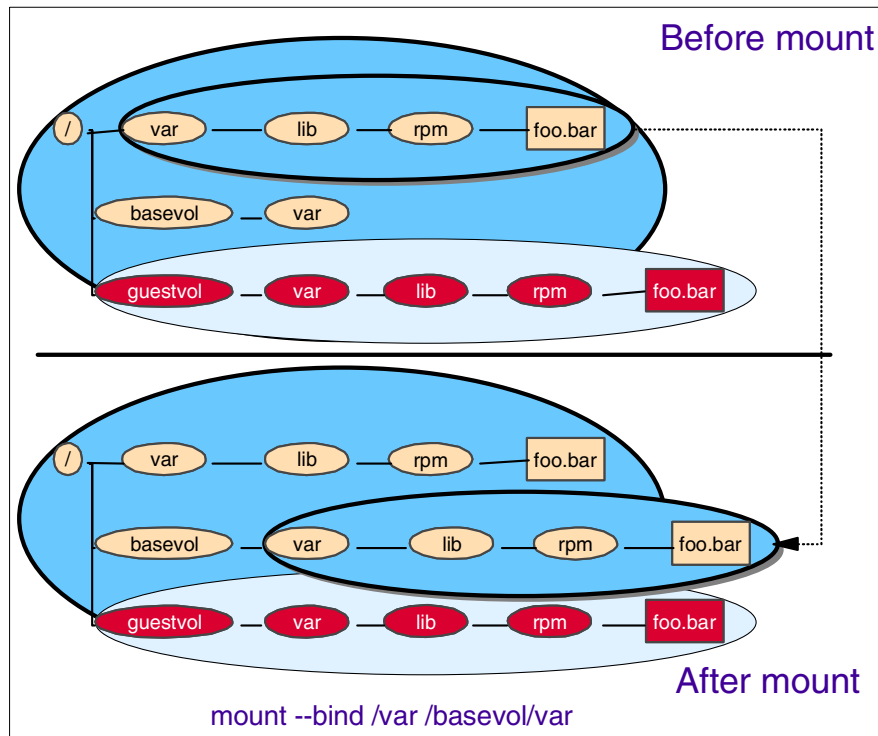


Figure 8-5 The effect of `mount --bind /var /basevol/var`

The read-only basevol /var subtree is preserved in /basevol/var.

Continuing in Figure 8-6 on page 158, we bind-mount the /guestvol/var directory on /var.

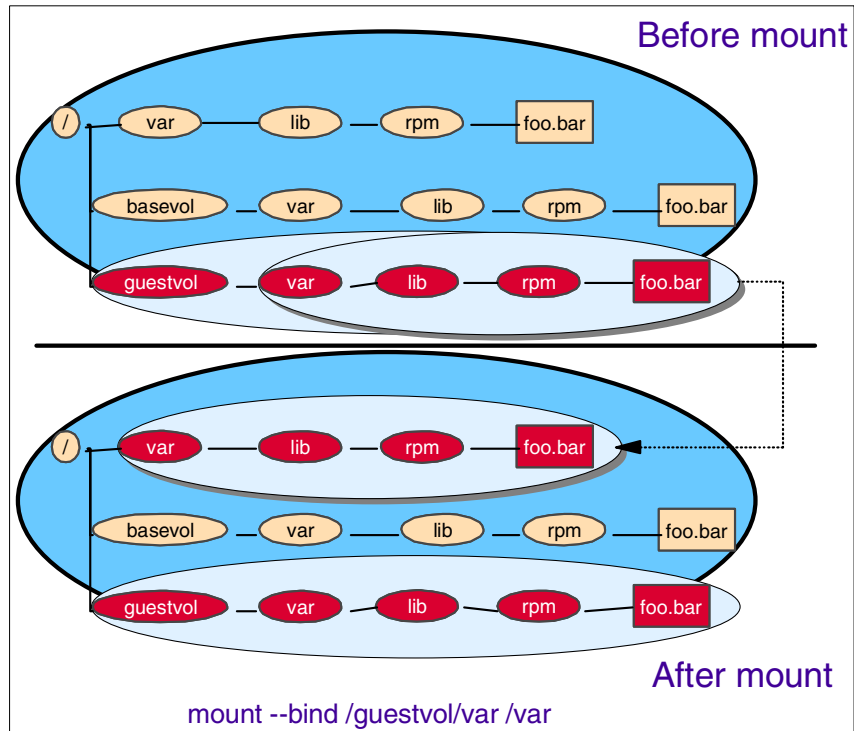


Figure 8-6 The effect of `mount --bind /guestvol/var /var`

Finally, as shown in Figure 8-7 on page 159, the `/basevol/var/lib/rpm` directory is mounted on `/var/lib/rpm`.

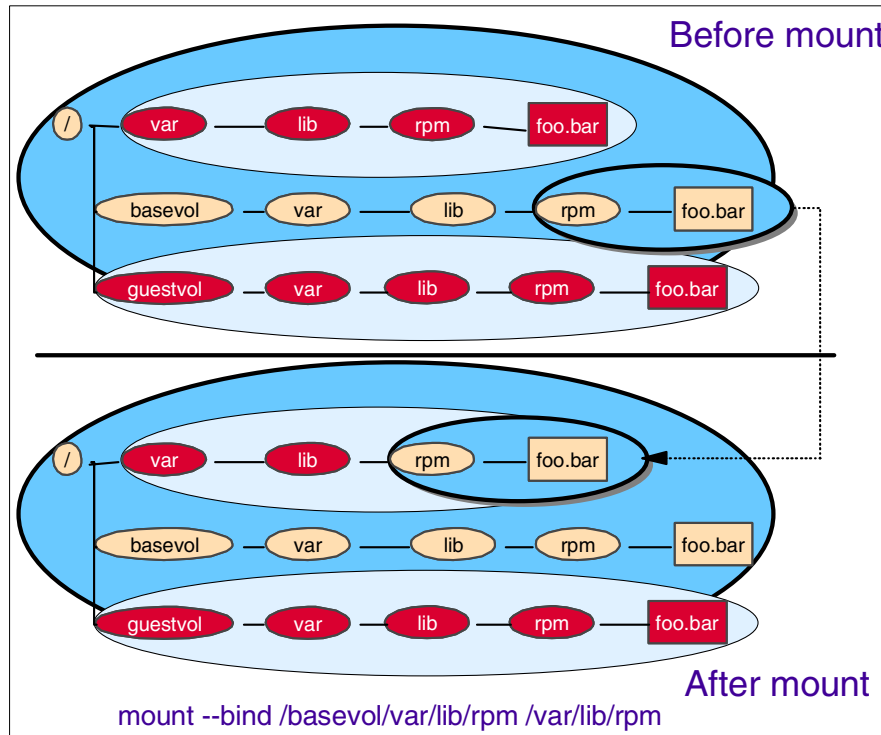


Figure 8-7 The effect of `mount --bind /basevol/var/lib/rpm /var/lib/rpm`

The `/var` directory is now writable (as required by the FHS). The `/var/lib/rpm` directory, however, remains read-only and refers to RPMs installed on the `basevol`.

There may be cases where we wish to install RPM packages on the `guestvol` filesystem image. For instance, two different clones might mount a `basevol` device containing common applications and services, while one of the clones installs additional packages on its `guestvol` device. We discuss how can be accomplished in 9.6, “Guestvol package management” on page 180.

8.9 Booting a basevol/guestvol Linux guest

When booting a Linux guest from `basevol/guestvol` filesystems, we use bind mounts in the Linux boot initialization. In addition, we ensure the Linux guest can boot from the `basevol`, even in the event the read-write `guestvol` is unavailable.

We provide a `sysinit` add-on script (`rc.guestvol`) to perform the necessary bind mounts. We also modify the existing `rc.sysinit` script.

8.9.1 The rc.guestvol script

In a basevol/guestvol Linux guest, the `/etc/inittab` file is modified to specify the `/etc/rc.d/rc.guestvol` command to be run on system startup. This script is responsible for:

- ▶ Determining if the Linux guest uses guestvol mounts.
- ▶ If so, mounting the guestvol device on the `/guestvol` directory (as shown in Figure 8-2 on page 152).
- ▶ Bind mounting the basevol and guestvol directories as discussed in 8.3, “Using bind mounts” on page 152.

A listing of the `rc.guestvol` script can be found in Appendix C.1.1, “The `/etc/rc.d/rc.guestvol` script” on page 232.

8.9.2 Determining if the Linux guest uses a guestvol mount

It may not be possible to mount a guestvol on the base when booting a basevol/guestvol Linux guest. For instance, the guestvol may not be available. Additionally, the Linux guest may wish to boot the basevol as a normal read-write root filesystem (for instance, to create a guestvol filesystem). The basevol startup attempts to deal with a missing guestvol.

The guestvol is assumed to be defined as virtual device number 777. The `/etc/rc.d/rc.guestvol` scripts first examines this device. If no device is found, a DASD device is added:

```
echo "add range=777" > /proc/dasd/devices
```

Note: This utilizes the dynamic DASD device feature of Linux on zSeries. No device would be found (even though a valid guestvol might be defined to the Linux guest virtual machine) if the `dasd=ranges` kernel option was omitted from the IPL record of the basevol device.

Initialization then proceeds:

- ▶ If the device is a DASD device, and the DASD device appears to be a guestvol (it contains a `/guestvol/etc/inittab` file), the guestvol initialization continues.
- ▶ If the device is found to be a printer device, the script assumes the basevol is to be mounted as a normal read-write root filesystem. Guestvol initialization terminates and the normal `/etc/rc.d/sysinit` script is invoked.
- ▶ In all other cases, a maintenance shell is entered.

8.9.3 The maintenance shell

Once in the maintenance shell, the user is given the opportunity to define a `guestvol`. If a `guestvol` device is detected upon exiting the maintenance shell, `guestvol` initialization will proceed as normal.

8.9.4 Example `basevol/guestvol` Linux guest startup

Example 8-1 illustrates the `basevol/guestvol` initialization using the Linux console messages.

Example 8-1 Booting a `basevol/guestvol` Linux guest

```
INIT: version 2.78 booting
Searching for guestvol device 777 1
dasd:/proc/dasd/devices: 'add range=777'
dasd(eckd):0777 on sch 18: 3390/0A(CU:3990/04) Cyl:20 Head:15 Sec:224
dasd(eckd):0777 on sch 18: 3390/0A(CU:3990/04): Configuration data read
debug: reserved 2 areas of 1 pages for debugging dasdd
dasd:waiting for responses...
dasd(eckd):/dev/dasdd(94:12),0777@0x12:(4kB blks): 14400kB at 48kB/trk classic
disk layout
  dasdd:CMS1/ LNX2GV(MDSK): dasdd dasdd1
Guestvol device 777 appears to be at device file /dev/dasdd1 2
Guestvol seems to have been successfully mounted...continuing boot 3
Finished binding guestvol directories, init will now restart... 4
                               Welcome to Red Hat Linux
Mounting proc filesystem: [ OK ]
Configuring kernel parameters: [ OK ]
```

Notes highlighted in Example 8-1 refer to the following points:

1. The `/etc/rc.d/rc.guestvol` script (residing on the `basevol`) checks for the existence of a virtual DASD 777.
In this case, none was found and the script dynamically adds the device.
2. Device number 777 is determined to be a DASD device.
It is mounted on the `/guestvol` directory.
3. A `/guestvol/etc/inittab` file is found.
 - a. Access to read-only root directories is preserved by bind-mounting those directories over respective directories in `/basevol` (as outlined in 8.3.1, “Mounting writable directories on a read-only filesystem” on page 153).
 - b. Directories in the read-write `/guestvol` directory are bind-mounted over their respective locations on the root filesystem.
4. On completion, a restart signal is sent to `init`.

The new `/etc/inittab` (the `/guestvol//etc/inittab` file) is executed.

Configuration of the `basevol/guestvol` Linux guest continues as outlined in 8.10, “Startup configuration” on page 162.

8.9.5 Example `basevol/guestvol` Linux guest maintenance shell

To illustrate the maintenance shell, we simulate a missing `guestvol` by redefining virtual device 777 to be 1777 before booting the Linux guest:

```
REDEFINE 777 AS 1777
```

The boot messages are shown in Example 8-2.

Example 8-2 Booting a `basevol/guestvol` Linux guest with a missing `guestvol`

```
REDEFINE 777 AS 1777
DASD 1777 DEFINED
IPL 202
...
INIT: version 2.78 booting
Searching for guestvol device 777
dasd:/proc/dasd/devices: 'add range=777'
Error: failed to find /guestvol/etc/inittab
About to start maintenance shell. Please either mount guestvol
disk on /guestvol and then type "exit" or else just type
"exit 123" to boot the underlying basevol system in
read-write mode with no guestvol.
bash-2.05#
```

Important: The `/etc/rc.d/rc.guestvol` script examines the return code from the maintenance shell.

- ▶ If zero, the script verifies that the `guestvol` device (number 777) contains an `/etc/inittab` file. If not, it returns to the maintenance shell.
- ▶ If non-zero (for instance, by typing: `exit 123`), the `rc.guestvol` script read-write mounts the `basevol` device.

If the device is read-only linked in the virtual machine, the Linux guest cannot damage the `basevol`. However, Linux startup will fail.

8.10 Startup configuration

At this point in the initialization, `basevol/guestvol` Linux guest has its read-write `guestvol` directories mounted over the read-over `basevol`. Guest-specific

configuration (such as networking configuration) is performed by executing the directives specified in the `guestvol /etc/inittab` file.

Although guest-specific configuration data could be stored on the `guestvol` filesystem, this approach will quickly become too cumbersome to manage, for the following reasons:

- ▶ Guestvols cannot simply be created by copying a master `guestvol` image. This would lead to networking conflicts between `basevol/guestvol` Linux guests.
- ▶ It is not feasible to customize `guestvols` for each `basevol/guestvol` Linux guest. Managing configuration files on the `guestvol` devices is simply too time-consuming and error-prone.

Instead, we implement a two-phase Linux guest configuration design:

- ▶ In the first phase, the `basevol/guestvol` Linux guest bootstraps itself sufficiently to acquire a network temporary configuration. We refer to this phase as *early boot-time configuration*, and its implementation is discussed in 8.11, “Network configuration” on page 164.
- ▶ In the second phase, the `basevol/guestvol` Linux guests acquire configuration information from a centrally managed LDAP database. We discuss this in Chapter 10, “Centralized management using LDAP” on page 191.

8.10.1 The `rc.sysinit-guestvol` script

The standard `/etc/rc.d/rc.sysinit` script presents problems to a `basevol/guestvol` Linux guest:

- ▶ The filesystem on the root device is remounted in read-write mode. The `basevol` device should remain read-only mounted; we comment out this line.
- ▶ The contents of the `/etc/mtab` file are cleared. The version created by the `rc.guestvol` script should be preserved; we comment out this line.
- ▶ The `rc.sysinit` script runs itself through the `initlog` process. We change the script name.

We provide a replacement, the `/etc/rc.d/rc.sysinit-guestvol` script shown in Appendix C.1.2, “The `/etc/rc.d/rc.sysinit-guestvol` script” on page 234.

8.11 Network configuration

We devise a centralized configuration server relying on the services provided by z/VM to enable basevol/guestvol Linux guests to obtain network connectivity. This configuration server is referred to as the *confserv*.

Note: The confserv provides services similar to a DHCP server; clients with no network obtain network configuration from a central management system.

VM Guest LANs do not provide services required for a DHCP server.

We implement the confserv guest using the VM Programmable Operator (PROP). Details on PROP can be found in *z/VM V4R3.0 CMS Planning and Administration*, SC24-6042.

The basevol/guestvol Linux guest requests network configuration from the confserv using only native CP communication methods (thus avoiding the need for any TCP/IP network at boot time).

8.11.1 The z/VM configuration server

PROP runs continuously in its own CMS guest; we use a VM virtual machine named CONFSEV. Virtual machines communicate to the PROP service machine using the CP SMSG command. Upon receiving a message, the PROP virtual machine does the following:

- ▶ It searches its configuration file for a line with a matching message pattern.
- ▶ It executes the action specified on that line.

In Example 8-3, we show a line in the PROP configuration file for confserv, the CONFSEV virtual machine.

Example 8-3 PROP configuration line for CONFSEV

/GETMYCONF /	1	10	4	GETCONF	TAG
--------------	---	----	---	---------	-----

Note: The complete PROP configuration file is shown in Appendix C.6, “The PROP RTABLE configuration file” on page 272.

In the example, if confserv receives a type 4 message (a message sent using the CP SMSG command) containing the text string GETMYCONF, the CONFSEV virtual machine executes GETCONF EXEC.

Parameters passed to GETCONF EXEC include:

- ▶ The name of the guest from which the message originated
- ▶ The final field in the PROP configuration file (in this case, the string "TAG")
- ▶ The contents of the message itself

Note: Although additional parameters are passed as well, we do not use them.

A listing of the GETCONF EXEC script can be found in Appendix C.5, "The GETCONF EXEC script" on page 271.

For example, if virtual machine CLONE1 issues the following command, then the GETCONF EXEC script is invoked in the CONSERV virtual machine:

```
SMSG CONSERV GETMYCONF DEFAULT
```

Note the following:

- ▶ The guest name (CLONE1) and the requested role name (DEFAULT) are passed as parameters.
- ▶ These parameters are used to search for the CLONE1 network parameters specific to role DEFAULT in a CONSERV configuration file (the GUEST CONF file).

In Example 8-4, we show the corresponding line in the GUEST CONF file.

Example 8-4 A configuration line from GUEST CONF

```
CLONE1  DEFAULT HOSTNAME=c1one1;ETH0=10.0.3.101;GATEWAY=10.0.3.1
```

Note: Searches against the GUEST CONF file are case-insensitive.

8.11.2 Generating a CONFSEV response

Because no IUCV API is available for Linux user programs, the confserv uses an alternative method to send a response to the basevol/guestvol Linux guest:

- ▶ The Linux guest is required to have a virtual printer defined as device number 001E.
- ▶ The GETCONF EXEC script uses the CP SEND IP command to send a response. The message is tagged to virtual device 001E.

In the preceding example, GETCONF EXEC composes the line:

```
CP SEND CP CLONE1 TAG 001E HOSTNAME=c1one1;ETH0=10.0.3.101;GATEWAY=10.0.3.1
```

Linux guest CLONE1 retrieves the network configuration information from the tag on its virtual printer as described in 8.11.4, “The vmgetconf script” on page 166.

8.11.3 Security considerations

The CONFSEV virtual machine requires sufficient privileges to be permitted to issue CP commands on behalf of a Linux guest. There are two ways to accomplish this:

- ▶ Assign privilege class C to the confserv virtual machine in its VM user directory entry.

This allows confserv virtual machine to send CP commands *any* virtual machine.

Although PROP is designed for use by high privilege class users, this practice may not conform to your site’s security policy. We do not recommend this approach.

- ▶ Assign the confserv virtual machine as the *secondary user* (SECUSER) for each basevol/guestvol Linux guest.

This can be done by adding the confserv user as the final field in the CONSOLE line of the VM user directory entry for the basevol/guestvol Linux guest. Alternatively, issue the CP command:

```
SET SECUSER confserv-user
```

Using this method, the confserv virtual machine can only respond to virtual machines to which it is designated as the secondary user.

Although this approach can address security concerns, it can have disadvantages: the confserv virtual machine can only respond to a disconnected guest.

For guests started using XAUTOLOG, this is not a problem. However, logging on to the console of a basevol/guestvol Linux guest and attempting to IPL Linux will cause the confserv configuration request to fail.

Tip: Running PROP as a configuration server in this manner is light on memory use. Although CONFSEV’s VM user directory entry gave it a 32 MB virtual machine, the guest’s working set and resident set size both remain at just over 100 pages (400 KB).

8.11.4 The vmgetconf script

The vmgetconf script (found in /etc/init.d directory of the guestvol) generates the request and receives the response from confserv. It uses a configuration file

(/etc/sysconfig/vmconfigserver) to determine the name of confserv virtual machine and to provide the role requested by the basevol/guestvol Linux guest.

A sample /etc/sysconfig/vmconfigserver is shown in Example 8-5.

Example 8-5 The /etc/sysconfig/vmconfigserver file

```
CONFSERV="CONFSERV"  
ROLE="default"
```

Note: The value of the ROLE parameter is case-insensitive.

Sending a request to the confserv virtual machine

To send a confserv request, vmgetconf first defines a virtual printer at device number 001E, then issues the request. The sequence is illustrated here:

```
hcp detach 001e  
hcp define prt 001e  
hcp msg CONFSERV GETMYCONF default
```

Note: We use the cpint package, as described in 1.9.2, “Communicating with CP from a Linux telnet session” on page 15 to communicate with CP.

Receiving the confserv response

After sending the request, vmgetconf polls the virtual printer:

```
hcp tag query 001e
```

- ▶ If the confserv response has not yet been received, the query results in a message that begins:

```
PRT 001E TAG NOT SET
```

- ▶ Once the confserv response is received, the query results in a message of the form:

```
PRT 001E TAG:  
HOSTNAME=c1one1;ETH0=10.0.3.101;GATEWAY=10.0.3.1
```

Note: The vmgetconf script polls the printer once per second for 10 seconds. If a confserv response has not been received in that time, processing continues. However, no network configuration will be possible.

Generating a network configuration file

Once a confserv response is received, vmgetconf generates a network configuration file (/etc/sysconfig/vmconfig) from that response. Example 8-6 on page 168 shows a sample of the generated file.

Example 8-6 The /etc/sysconfig/vmconfig file generated by vmgetconf

```
HOSTNAME=c1one1  
ETH0=10.0.3.101  
GATEWAY=10.0.3.1
```

8.11.5 The itsonet script

The network configuration information available in the `/etc/sysconfig/vmconfig` file is translated to the form expected in a normal Linux boot sequence.

Note: Network configuration varies by Linux distributions. The `itsonet` script described here is designed to operate on a Red Hat 7.1 s390x distribution.

For Red Hat 7.1, the `/etc/rc.d/rc3.d/S10networking` script (a symbolic link to the `/etc/rc.d/init.d/networking` script) configures the network at runlevel 3. The `networking` script configures the network based on information found in:

- ▶ `/etc/sysconfig/network`
- ▶ `/etc/sysconfig/network-scripts/ifcfg-*`

From the `/etc/sysconfig/vmconfig` file, we generate these files before the `/etc/rc.d/init.d/networking` script is executed.

Additionally, we reset the networking hostname (it is set in another init script executed before networking is configured).

The `/etc/init.d/itsonet` script shown in C.1.4, “The `/etc/init.d/itsonet` script” on page 252 performs these actions.

The symbolic link `S09itsonet` in the `guestvol /etc/rc.d/rc3.d` directory `itsonet` is executed at run level 3 before networking.

In Example 8-7 on page 169, we show an example of the network configuration files generated by the `itsonet` script.

Example 8-7 Network configuration files generated by itsonet

```
/etc/sysconfig/network:
NETWORKING=yes
HOSTNAME=clone1
GATEWAY=10.0.3.1

/etc/sysconfig/network-scripts/ifcfg-eth0:
DEVICE=eth0
BOOTPROTO=static
IPADDR=10.0.3.101
NETMASK=255.255.255.0
ONBOOT=yes
```

Note: In addition to generating these files, itsonet will also issue the following command to set the hostname correctly:

```
hostname clone1
```

8.11.6 Example of boot time configuration

Example 8-8 shows the console messages for a basevol/guestvol Linux guest using the vmgetconf configuration.

Example 8-8 Fetching boot time configuration with vmgetconf

```
Entering non-interactive startup
Starting vmgetconf: [ OK ]
Generating network configuration files from vmconfig information
Hostname is clone1
Resetting hostname to clone1 [ OK ]
Setting gateway IP address to 10.0.3.1
Setting eth0 IP address to 10.0.3.101
ETH1 not set: not configuring eth1
Setting network parameters: [ OK ]
Bringing up interface lo: [ OK ]
Bringing up interface eth0: [ OK ]
```

8.12 Shutdown processing

For a basevol/guestvol Linux guest, we modify the system shutdown and reboot scripts (performed by the `/etc/rc.d/init.d/halt` script):

At shutdown, all filesystems must be unmounted. The halt script executes from the `/etc` directory, and init has open devices in the `/dev` directory.

However, the basevol/guestvol Linux guest has bind-mounted both these directories from the guestvol onto the basevol root, which prevents these filesystems from being unmounted.

This problem is solved by splitting the function of `/etc/rc.d/init.d/halt` script into two phases:

- ▶ Phase one executes the `/etc/init.d/guestvol-start-halt` script.
- ▶ Phase two executes the `/sbin/guestvol-final-halt` script.

8.12.1 The guestvol-start-halt script

The `guestvol-start-halt` script (shown in Appendix C.1.5, “The `/etc/init.d/guestvol-start-halt` script” on page 254) performs the same function as the normal `/etc/rc.d/init.d/halt` script up to the point where filesystems are unmounted.

The script then performs an `exec` on the `/sbin/guestvol-final-halt` script. This frees the `/etc` filesystem for `umount`.

8.12.2 The guestvol-final-halt script

The `guestvol-final-halt` script (shown in Appendix C.1.6, “The `/sbin/guestvol-final-halt` script” on page 257) performs the bottom half `/etc/rc.d/init.d/halt` script, unmounting all filesystems.

However, `umount` must be performed using the `-n` option. This prevents updating the read-only basevol `/etc/mstab` file.

At this point, all filesystems except `/guestvol` are unmounted.

The `/guestvol` filesystem cannot be unmounted because the `init` process does the following:

- ▶ It uses the `/dev` directory (it is bind-mounted from the `/guestvol/dev` directory over the `/dev` directory).
- ▶ It keeps `/dev/console` and a FIFO (`/dev/initctl`) open.

To deal with this, `guestvol-final-halt` simply remounts the `/guestvol` filesystem in read-only mode. This effectively frees the `/guestvol` filesystem.

8.12.3 Example of a basevol/guestvol Linux guest shutdown

Example 8-9 on page 171 shows the console messages generated when shutting down a basevol/guestvol Linux guest.

Example 8-9 Shutting down a basevol/guestvol Linux guest

```
bash-2.05# shutdown -h now
Broadcast message from root (console) Wed Aug  7 18:07:50 2002...
The system is going down for system halt NOW !!
INIT: Switching to runlevel: 0
bash-2.05# INIT: Sending processes the TERM signal
Stopping atd: [ OK ]
Stopping sshd:[ OK ]
Shutting down sendmail: [ OK ]
Stopping xinetd: [ OK ]
Stopping crond: [ OK ]
Stopping automount:[ OK ]
Saving random seed: [ OK ]
Shutting down NFS file locking services:
Shutting down NFS statd: [ OK ]
Stopping portmapper: [ OK ]
Shutting down kernel logger: [ OK ]
Shutting down system logger: [ OK ]
Shutting down interface eth0: [ OK ]
Shutting down interface eth1: [ OK ]
Starting killall: [ OK ]
Sending all processes the TERM signal...
Sending all processes the KILL signal... md: recovery thread got woken up ...
md: recovery thread finished ...
md: mdrecoveryd(9) flushing signals.

Turning off quotas:
Switching to final halt script for guestvol environment
Starting final part of halt procedure for guestvol environment
Remounting guestvol readonly on block device /dev/dasdd1
Halting system...
About to eval halt -i -d -p
md: stopping all md devices.
CONNECT= 00:00:59 VIRTCPU= 000:11.02 TOTCPU= 000:11.35
LOGOFF AT 18:10:22 EDT WEDNESDAY 08/07/02
```

1
2
3
4
5

Notes highlighted in Example 8-9 refer to the following points:

1. The `/etc/init.d/guestvol-start-halt` script issues an `exec` of the `/sbin/guestvol-final-halt` script.
2. The `/sbin/guestvol-final-halt` script begins execution.
3. The `/guestvol` is remounted read-only to make it clean for the next boot.
4. The `halt` command is executed to cause the kernel to halt and power down.
5. The kernel powers down by issuing the `CP LOGOFF` command that was configured via its boot time kernel parameter `vmpoff=LOGOFF`.

8.13 Advantages of a basevol/guestvol Linux guest

While the concept of mounting personalized read-write filesystems over a read-only root filesystem is not new, we believe running Linux under z/VM using bind mounts offers some unique advantages.

- ▶ The technique relies on z/VM disk resources; no network-mounted drives are required.
- ▶ No symbolic links are required. Path names retain their canonical names; therefore, applications which might expect canonical names do not complain.
- ▶ Linux guests may be easily added to the cluster using standard z/VM user management directives.
- ▶ Because the basevol filesystem is bootable, even if problems exist in the guestvol definition, it is possible to boot a Linux guest directly from the basevol filesystem.



Building a basevol/guestvol penguin colony

In this chapter, we discuss how to create a basevol/guestvol Linux guest. We then consider how to use that guest to clone a basevol/guestvol penguin colony.

9.1 Overview of the process

Using the concepts discussed in Chapter 8, “Shared Linux filesystems” on page 149, we developed a procedure to create the basevol and guestvol filesystem images. These images will be used in creating a penguin colony.

The procedure involves the following steps:

1. Creating a virtual machine to be used as a Linux guest.
We refer to this as the *development Linux guest* and name it LDV01.
2. Installing Linux on the development guest.
3. Customizing the installation on the development guest.
4. Partitioning the installation into two new filesystems: a basevol filesystem and a guestvol filesystem.

Each of these filesystems will reside on its own DASD device.

5. Creating golden copies of the newly created basevol and guestvol images.
These images will be copied to DASD devices owned by the BASEVOL virtual machine.
6. Developing an automated process to create Linux clones from the golden basevol and guestvol images.

Note: The network topology of the of the penguin colony is represented in Figure 10-1 on page 193.

9.2 The BASEVOL virtual machine

We used a virtual machine named BASEVOL to own golden copies of the basevol and guestvol images. Linux clones will mount the golden basevol images as their read-only root. Each clone will have a dedicated read-write guestvol created from the golden copy owned by BASEVOL.

Example 9-1 on page 175 shows the BASEVOL PROFILE definition.

Example 9-1 The BASEVOL PROFILE definition

```
USER BASEVOL NOLOG
  MDISK 1200 3390 1 3338 LX6711 MR ALL XXXXXXXX XXXXXXXX
  MDISK 1201 3390 1 80 LX6611 MR ALL XXXXXXXX XXXXXXXX
  MDISK 1202 3390 81 80 LX6611 MR ALL XXXXXXXX XXXXXXXX
  MDISK 1203 3390 161 80 LX6611 MR ALL XXXXXXXX XXXXXXXX
  MDISK 1204 3390 241 80 LX6611 MR ALL XXXXXXXX XXXXXXXX
  MDISK 1205 3390 321 80 LX6611 MR ALL XXXXXXXX XXXXXXXX
  MDISK 1206 3390 401 80 LX6611 MR ALL XXXXXXXX XXXXXXXX
  MDISK 1207 3390 481 80 LX6611 MR ALL XXXXXXXX XXXXXXXX
  MDISK 1208 3390 561 80 LX6611 MR ALL XXXXXXXX XXXXXXXX
  MDISK 1209 3390 641 80 LX6611 MR ALL XXXXXXXX XXXXXXXX
```

The full volume 1200 minidisk will act as the golden copy of the basevol filesystem image. We used the 80-cylinder minidisks allocated on volume LX6611 as golden guestvol copies. This allowed us to create several guestvol flavors for a single basevol.

Linux clones will read-only mount the BASEVOL 1200 minidisk as their read-only basevol device. Each clone will have its own read-write guestvol device created from a golden copy of a guestvol minidisk.

9.3 The LDV01 virtual machine

We used a virtual machine named LDV01 to act as the development Linux guest. We show the LDV01 VM directory entry in Example 9-2.

Example 9-2 The LDV01 PROFILE definition

```
USER LDV01 LBYONLY 128M 1G G
  INCLUDE IBMDFLT
  IPL CMS
  LOGONBY MBEATTIE
  MACHINE XA
  XAUTOLOG CONFSERV
  SPECIAL 0700 QDIO 3 SYSTEM PRIVQDIO
  SPOOL 000C 2540 READER *
  SPOOL 000D 2540 PUNCH A
  SPOOL 000E 1403 A
  SPOOL 0777 PRT
  LINK BASEVOL 1200 1200 MR
  LINK BASEVOL 1201 1201 MR
  MDISK 0191 3390 1339 20 LNXU1R
  MDISK 0202 3390 1 3338 LX6710
```

1
2
3
4
5
6

The notes indicated in Example 9-2 on page 175 refer to the following points:

1. The LDV01 user is authorized to XAUTOLOG the CONFSEV virtual machine.
See 8.11, “Network configuration” on page 164 for details on the CONFSEV virtual machine.
2. The LDV01 virtual machine is connected to the PRIVQDIO VM Guest LAN.
3. Device 0777 is defined as a printer to indicate this virtual machine will not use a guestvol bind mount.
See 8.9.2, “Determining if the Linux guest uses a guestvol mount” on page 160 for details.
4. Create a read-write link to the BASEVOL 1200 minidisk.
This disk serves as a golden copy of the basevol filesystem.
5. Create a read-write link to the BASEVOL 1201 minidisk.
This disk serves as a golden copy of the guestvol filesystem.
6. The 202 minidisk will serve as the root filesystem for the LDV01 Linux guest.

9.4 Install Linux on the development image

After creating the BASEVOL and LDV01 virtual machines, we installed Linux on the LDV01 virtual machine. We installed a Red Hat 7.1 s390x distribution. Because the OCO qeth modules are not distributed on the Red Hat installation CD-ROMs, we could either:

- ▶ Install the distribution, and then install the OCO qeth modules available from the IBM Developerworks Web site
- ▶ Use the second initial ramdisk method outlined in on the first CD-ROM of the Red Hat installation distribution

In our case, we created a second initial ramdisk installation and followed the procedure outlined in Appendix B, “Installing Red Hat 7.1 with OCO modules” on page 221.

9.4.1 Choosing the packages to install

You should give some careful consideration as to which software packages will be initially installed on the LDV01 Linux guest. These packages will constitute the contents of the basevol filesystem image. Therefore, choose the minimum set of packages common to all Linux guests intended to operate on this basevol filesystem.

Packages that differentiate Linux guests may be installed on separate guestvol filesystem images. Remember, however, that the basevol filesystem image is intended to be bootable by a Linux guest without any defined guestvol image mount.

Note: In this redbook, we focus on the *methods* used to create basevol/guestvol Linux clones, and not on the criteria used to decide which packages should be installed on the basevol image.

You should give consideration to what software should be installed on a basevol image, and what should be installed on a guestvol image.

9.5 Create the basevol and guestvol filesystem images

Once installed, we used the LDV01 Linux to create basevol and guestvol filesystem images.

9.5.1 Prepare the LDV01 Linux guest

We installed packages on the LDV01 Linux guest which become part of the basevol filesystem image. The packages we installed included:

▶ **basevol+guestvol-1.0.0-1.noarch.rpm**

This package provides startup and shutdown scripts needed to support a basevol/guestvol Linux guest as discussed in Chapter 9, “Building a basevol/guestvol penguin colony” on page 173. In addition, it adds the required /basevol and /guestvol mount points.

▶ **itsobasevol-1.0.0-1.s390x.rpm**

This package includes the configuration startup scripts for managing a penguin farm from a centralized LDAP server. We discuss this in 10.4, “Network configuration and initialization” on page 197.

▶ **pam_ldap-150-1.s390x.rpm**

▶ **nss_ldap-198-1.s390x.rpm**

We examine these modules in 10.5, “UNIX authentication using LDAP” on page 201.

▶ **cpint-1.1.1-1.s390x.rpm**

This package provides a communication facility to VM through the `hcp` command.

Note: You can obtain these packages as part of the additional material distributed with this redbook. See Appendix D, “Additional material” on page 279 for details.

We installed these packages on the LDV01 Linux guest:

```
rpm -Uvh basevol+guestvol-1.0.0-1.noarch.rpm \  
    itsobasevol-1.0.0-1.s390x.rpm \  
    pam_ldap-150-1.s390x.rpm \  
    nss_ldap-198-1.s390x.rpm \  
    cpint-1.1.1-1.s390x.rpm
```

To complete preparation of the development basevol filesystem, we executed:

```
basevol-devel enable
```

This enables the basevol/guestvol startup sequence.

Note: The `/usr/sbin/basevol-devel` script is provided as part of the `basevol+guestvol-10.0-1.noarch.rpm` package. A listing can be found in Appendix C.1.7, “The `/usr/sbin/basevol-devel` script” on page 261.

9.5.2 Create the golden basevol filesystem image

We then created the golden basevol filesystem image. We logged off the LDV01 Linux guest and DDR-copied the LDV01 203 minidisk to the BASEVOL 1200 minidisk. For details on using the DDR command, see 1.11, “Using DDR to copy a minidisk” on page 19.

Attention: We used the same DASD device type and size for both the BASEVOL 1200 and LDV01 202 minidisks to ensure DDR will preserve the underlying ext2 filesystem.

9.5.3 Prepare guestvol filesystem image

Next, we prepared a golden guestvol filesystem.

The LDV01 Linux guest accesses the golden basevol filesystem image on its 1200 minidisk; the golden guestvol filesystem will be created on its 1201 minidisk (see Example 9-2 on page 175).

1. We logged on to the LDV01 Linux guest.
2. We attached the 1200 and 1201 minidisks to the Linux guest:

```
dasd add 1200
dasd add 1201
```

Note: We used the `/usr/sbin/dasd` command supplied in the `itsobasevol-1.0.0-1.s390x.rpm` package to dynamically add DASD devices. See Appendix C.2.2, “The `/usr/sbin/dasd` script” on page 268 for details.

The 1200 minidisk will now be accessed as `/dev/dasdb`; the 1201 minidisk is accessed as `/dev/dasdc`.

3. We formatted, partitioned, and created an ext2 filesystem on the 1201 minidisk:

```
dasdfmt -n 1201 -b 4096
fdasd /dev/dasdc
mke2fs -b 4096 -i 2048 /dev/dasdc1
```

4. We mounted the 1200 minidisk on the `/mnt/newbasevol` directory:

```
mount /dev/dasdb1 /mnt/newbasevol
```

5. We mounted the 1201 minidisk on the `/mnt/newguestvol` directory:

```
mount /dev/dasdb1 /mnt/newguestvol
```

6. We executed the command:

```
mkguestvol /mnt/newbasevol /mnt/newguestvol
```

Note: The `/usr/sbin/mkguestvol` command is supplied in the `basevol+guestvol-1.0.0-1.noarch.rpm` package. Its purpose is to copy the writable portion of a basevol filesystem to a guestvol filesystem. For details, see Appendix C.1.8, “The `/usr/sbin/mkguestvol` script” on page 261.

7. We unmounted the basevol and guestvol filesystems:

```
umount /mnt/newbasevol
umount /mnt/newguestvol
```

9.5.4 Booting the basevol/guestvol Linux guest

Now we could boot the LDV01 using the newly created basevol and guestvol filesystem images:

1. We shut down the LDV01 Linux guest:

```
shutdown -h now
```

2. We detached the existing 202 minidisk and read-only link to the BASEVOL 1200 minidisk as the new 202 device:

```
DET 202  
LINK BASEVOL 1200 202 RR
```

3. We detached the existing 777 virtual printer and read-write link to the BASEVOL 1201 minidisk as the new 777 device:

```
DET 777  
LINK BASEVOL 1201 777 MR
```

4. We IPLed the basevol/guestvol Linux guest:

```
IPL 202
```

Note: The LDV01 Linux guest should be defined in the GUEST CONF configuration file of the CONSERV virtual machine as described in 8.11.1, “The z/VM configuration server” on page 164, to ensure that the startup network configuration is correct.

9.6 Guestvol package management

As mentioned in 8.8, “RPM package management” on page 156, the default RPM database exists on the basevol filesystem. This means RPM packages cannot be directly added to the guestvol filesystem (the RPM database is read-only, and an attempt to install a package will fail).

As an example of how a package can be installed on a guestvol filesystem image, we consider the DDS server component of RMF PM for Linux (used in 3.3, “The Distributed Data Server” on page 39).

We created an RPM package from the downloaded tar file. The SPEC used to create the package can be found in the SPECS directory of additional material available with this redbook (see Appendix D, “Additional material” on page 279).

Tip: For information on how to build RPM packages, consult *Maximum RPM - Taking the Red Hat Package Manager to the Limit* at:

<http://www.rpm.org/max-rpm/>

To install the package, we followed these steps:

1. We created a directory in a guestvol bind mounted directory to store the RPM database files. We selected the `/var/lib/guestrpm` directory:

```
mkdir /var/lib/guestrpm
```

2. We initialized empty RPM database files in that directory:

```
rpm --initdb /var/lib/guestrpm
```

3. We installed the package on the guestvol filesystem image:

```
rpm -ivh --dbpath= rmpms-2.3.33-1.s390x.rpm
```

Attention: When creating the package, you may need to add additional *Provides* lines in the SPEC file definition to resolve package dependencies.

For instance, the package may have a dependency on the Bash shell. Although the shell exists in the basevol filesystem, RPM has no way to know it (it will not access the `/var/lib/rpm` database files).

As a workaround, we added this line:

```
Provides: /bin/sh
```

9.7 Cloning a basevol/guestvol Linux guest

We now considered cloning basevol/guestvol Linux guests, which involves these basic steps:

1. Define a clone virtual machine prototype.
2. Create a Linux clone using the clone prototype.
3. Copy the golden guestvol image to the clone's 777 minidisk.
4. Add the clone to the CONSERV configuration file.
5. XAUTOLOG the new clone.

9.7.1 The LNXCLONE prototype

We used DirMaint prototypes to create Linux clone virtual machines. As discussed in 2.7.4, “Adding a userid using a prototype file” on page 31, we first created the prototype shown in Example 9-3.

Example 9-3 The LNXCLONE PROTODIR definition

```
USER LNXCLONE LBYONLY 64M 512M G      1
  IPL 202                               2
  MACHINE XA
  XAUTOLOG CONFSERV                     3
  LOGONBY MBEATTIE                       4
```

```
CONSOLE 0009 3215 T CONFSERV      5
SPECIAL 0700 QDIO 3 SYSTEM PRIVQDIO 6
SPECIAL 0704 QDIO 3 SYSTEM PUBLQDIO
SPOOL 000C 2540 READER *
SPOOL 000D 2540 PUNCH A
SPOOL 000E 1403 A
LINK MAINT 0190 0190 RR
LINK MAINT 019D 019D RR
LINK MAINT 019E 019E RR
LINK MAINT 0401 0401 RR
LINK MAINT 0405 0405 RR
LINK BASEVOL 0202 0202 RR      7
MDISK 0777 XXXX AUTOG 80 ANY    8
```

The notes highlighted in Example 9-3 refer to the following points:

1. The password field is set to the special value LBYONLY to specify that LOGON to the virtual requires the BY option (see 1.2, “Logging on to z/VM” on page 4).

Note: Although the clone virtual machine normally is started using XAUTOLOG, this allows for logon with providing a visible password.

2. Clones operate as Linux guests IPLed from their 202 virtual device (a read-only basevol filesystem image).
3. Clones are permitted to XAUTOLOG the CONFSERV user - the first stage network configuration server discussed in 8.11.1, “The z/VM configuration server” on page 164.
4. The users permitted to LOGON to a clone virtual machine.

Note: Logging on to a clone virtual machine using LOGON BY is provided as a debugging feature.

5. The CONFSERV virtual machine is assigned as a secondary user for the basevol/guestvol Linux guest as discussed in 8.11.3, “Security considerations” on page 166.
6. The SPECIAL lines create two simulated NICs and connect those NICs to their respective VM Guest LAN.
7. The read-only basevol filesystem image is linked as the 202 minidisk.
8. A read-write 777 minidisk is allocated when the clone virtual machine is created. This minidisk will be the read-write guestvol filesystem image.

To enable the LNXCLONE prototype, we first added the prototype definition to DirMaint:

```
DIRMAINT FILE LNXCLONE PROTODIR A
```

9.7.2 Create the Linux clone virtual machine

To create a new clone virtual machine, we used:

```
DIRMAINT ADD LCL101 LIKE LNXCLONE PW LBYONLY
```

This creates the LCL101 virtual machine based on the LNXCLONE prototype.

9.7.3 Create the Linux clone guestvol

Once the clone virtual machine was created, we created the clone-specific guestvol filesystem image from a golden copy. As discussed in 9.3, “The LDV01 virtual machine” on page 175, we used the BASEVOL 1201 minidisk as the golden copy of the guestvol filesystem image. This image is copied to the 777 minidisk of each clone virtual machine.

To create the guestvol image for clone LCL101 from the BASEVOL 1201 golden copy, we used the command:

```
GVCOPY BASEVOL 1201 LCL101 777
```

Note: The GVCOPY EXEC is listed in C.3, “The GVCOPY EXEC” on page 268.

9.7.4 Define the Linux clone in the GUEST CONF configuration file

Before starting the new Linux, we added a line to the GUEST CONF configuration file of the CONFSERV virtual machine as outlined in 8.11.1, “The z/VM configuration server” on page 164.

9.7.5 XAUTOLOG the Linux clone

To start the new clone, we used XAUTOLOG the clone virtual machine:

```
XAUTOLOG LCL101
```

9.8 Remote startup and shutdown of Linux clones

In this section, we consider a mechanism to remotely start and stop the Linux clones in the penguin colony.

9.8.1 The `ext_int` kernel module

To intercept external interrupts, we used a kernel module called `ext_int`. The source is shown in Example 9-4.

Example 9-4 The `ext_int` kernel module

```
/*
 * Kernel module to hook S/390 and zSeries external interrupts
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License
 * as published by the Free Software Foundation; either version
 * 2 of the License, or (at your option) any later version.
 *
 * Copyright (C) 2001 IBM UK Ltd, IBM Corporation
 * Author: Malcolm Beattie <beattiem@uk.ibm.com>
 */

#include <linux/module.h>
#include <linux/sched.h>
#include <asm/s390_ext.h>

static unsigned short code = 0;
static int pid = 0;
static int sig = 0;

MODULE_PARM(code, "h");
MODULE_PARM_DESC(code, "external interrupt code number to hook");
MODULE_PARM(pid, "i");
MODULE_PARM_DESC(pid, "pid to be signalled when external interrupt arrives");
MODULE_PARM(sig, "i");
MODULE_PARM_DESC(sig, "signal to send when external interrupt arrives");

MODULE_AUTHOR("Malcolm Beattie <beattiem@uk.ibm.com>");
MODULE_DESCRIPTION("Hook S/390 external interrupts to signal a process");

static void send_signal(struct pt_regs *regs, __u16 ourcode)
{
    printk(KERN_DEBUG "ext_int: EXT %hX--sending sig %d to pid %d\n",
           code, sig, pid);
    kill_proc(pid, sig, 1);
}

int init_module(void)
{
    int err;

    if (code == 0 || pid == 0 || sig == 0) {
```

```

        printk(KERN_ERR "ext_int: code, pid & sig must be non-zero\n");
        return -EINVAL;
    }
    err = register_external_interrupt(code, send_signal);
    if (err)
        return err;
    printk(KERN_INFO "ext_int: code=0x%hx, pid=%d, sig=%d\n",
           code, pid, sig);
    return 0;
}

void cleanup_module(void)
{
    unregister_external_interrupt(code, send_signal);
}

```

Using the `ext_int` module, you can send a signal to a specified process upon receiving an external interrupt of a specific type.

Note: External interrupt codes are 16-bit numbers in the hexadecimal range 0000 through FFFF.

When loaded, the `ext_int` module requires three parameters:

code	The hexadecimal external interrupt number to listen for
pid	The process to notify upon receiving the external interrupt code
sig	The UNIX signal to send to pid

For instance, to send UNIX signal 10 to process with pid number 1234 upon receiving external interrupt 0x1243, use:

```
insmod -o ext1234 ext_int code=0x1234 pid=789 sig=10
```

Note: The `-o ext1234` option causes the `ext_int` module to be loaded under name `ext1234`. This allows you to register multiple instances of the `ext_int` module - each handling a different external interrupt.

To unload the module, use:

```
rmmmod ext1234
```

9.8.2 Handling a shutdown external interrupt

The `ext_int` module in conjunction with an external interrupt can be used to remotely trigger shutdown of a Linux guest.

In the `/etc/inittab` file, we added the line:

```
ca::ctrlalddel:/sbin/shutdown -h now
```

This line defines the action the `init` process is to take upon receiving UNIX signal `SIGINT` (signal number 2). We registered the `ext_int` kernel module to send the `init` process a `SIGINT` upon receipt of a `0x0d1e` external interrupt using:

```
insmod -o ext0d1e ext_int code=0x0d1e pid=1 sig=2
```

Note: There does not appear to be a documented range of available external interrupt numbers. However, it seems that interrupt codes `0x8000` and higher are ignored by some Linux kernel and/or z/VM versions.

Attention: We chose to use code `0x0d1e`. However, some codes are actively used by the S/390 and zSeries architecture. Our choice seemed appropriate, but this choice may be reserved for future use.

9.8.3 The management interface

To remotely start or stop a Linux guest, we used the `PROP` facility as outlined in 8.11, “Network configuration” on page 164, together with a Web frontend. We show the management interface HTML in Example 9-5.

Example 9-5 The management interface HTML

```
<html>
<head>
  <title>VMLINUX Management Interface</title>
</head>
<body>
  <h1 align="center">VMLINUX Management Interface</h1>
  <h2>Act on one guest</h2>
  <form action="/cgi-bin/guestact">
    Guest: <input name="guest">
      <input type="submit" name="start" value="Start">
      <input type="submit" name="shutdown" value="Shutdown">
  </form>
</body>
</html>
```

The interface supports startup and shutdown of a Linux guest specified by name.

In Example 9-6 on page 187, we show the `guestact` CGI script which processes management requests.

Example 9-6 The guestact CGI script

```
#!/usr/bin/perl
use strict;

sub hcp {
    system("/usr/sbin/hcp @_ > /dev/null 2>&1");
}

sub start_guest {
    my $guest = shift;
    hcp("msg confserv xautolog $guest");
    print "IPL has been initiated for guest $guest\n";
}

sub shutdown_guest {
    my $guest = shift;
    hcp("msg confserv shutdown $guest");
    print "Shutdown has been initiated for guest $guest\n";
}

my $query = $ENV{QUERY_STRING};
my %q = split(/[&=]/, $query);

print "Content-Type: text/html\n\n";

if ($q{start}) {
    start_guest($q{guest});
} elsif ($q{shutdown}) {
    shutdown_guest($q{guest});
} else {
    print "<h2>Unknown action</h2>\n";
}
}
```

The guestact script utilizes the MSG command to forward a request to the CONFSEVR virtual machine for processing.

9.8.4 PROP actions to manage Linux clones

To enable PROP to handle startup and shutdown requests generated by the guestact script, we first needed to add an appropriate handler to the PROP RTABLE configuration file. In Example 9-7, we show the relevant lines.

Example 9-7 The PROP configuration lines for remote startup and shutdown

/XAUTOLOG /	1	9	4	LNX5	GUESTACT XAUTOLOG
/SHUTDOWN /	1	9	4	LNX5	GUESTACT EXTOD1E

Note: The entire contents of the PROP RTABLE configuration file can be found in Appendix C.6, “The PROP RTABLE configuration file” on page 272.

As shown in the example, PROP will execute the GUESTACT REXX script upon receiving an XAUTOLOG or SHUTDOWN request from the LNX5 virtual machine (the Linux guest running the Web server).

9.8.5 The GUESTACT EXEC script

The GUESTACT EXEC script (shown in Appendix C.4, “The GUESTACT EXEC script” on page 269) issues the CP commands required to perform remote startup and shutdown of Linux guests.

In the case of a startup request, the script executes the XAUTOLOG command on the intended virtual machine. For shutdown, the EXTERNAL CP command will sent to the intended Linux guest’s console, passing the external interrupt code to generate (0D1E in our case). This will generate the external interrupt to be handed by the ext_int module.

9.8.6 Security considerations

The same considerations as discussed in 8.11.3, “Security considerations” on page 166 apply to the use of PROP here, as well. We note some additional security checks implemented in the management interface:

- ▶ The XAUTOLOG and SHUTDOWN actions specified in the PROP RTABLE configuration file are restricted to requests originating from the LNX5 virtual machine (the Web server machine).
- ▶ The GUESACT EXEC script performs a check to ensure the target machine is listed in the GUEST CONF file (to ensure only Linux clones can be remotely managed).

Web server security

In order to execute the **hcp** command, the cpint package must be installed on the Web server machine (in this case, LNX5). Because **hcp** (a user-mode command) requires transition to supervisor mode in order to execute the DIAGNOSE 08 API, cpint provides a kernel module to act on its behalf.

The character device `/dev/cpint8` owner and permissions control which users and groups may use the **hcp** command. By default, this device is owned by root with 0600 permissions. This implies *only* root may execute **hcp**.

In developing this redbook as a proof of concept, we simply granted write authority to the Web server userid (apache) which is an inherently insecure choice. A more acceptable option would be to implement a privileged wrapper command around **hcp** for use by the Web server.

Important: You should consider the security implications and evaluate how these conform to your location's security policy before implementing this type of management interface.



Centralized management using LDAP

In this chapter, we discuss options for centralized management of the penguin colony introduced in Chapter 9, “Building a basevol/guestvol penguin colony” on page 173.

10.1 Using LDAP for centralized management

Centralized management and system administration becomes a crucial consideration in the operation of a penguin colony. Ideally, we would like to store as much server-specific information as possible in a central repository, and have services and applications access that repository. The Lightweight Directory Access Protocol (LDAP) is designed to address this issue.

See the IBM Redbook *Understanding LDAP*, SG24-4986 for details on LDAP. A good source for on-line information on LDAP can be found at the LDAPzone Web site:

<http://www.ldapzone.com/>

In this section, we consider using LDAP for:

- ▶ Network configuration and initialization
- ▶ UNIX user and group authentication
- ▶ An LDAP backend for Domain Name System (DNS)

10.1.1 The OpenLDAP directory server

We use the OpenLDAP directory server, which is an open source LDAP implementation available with many Linux distribution. The OpenLDAP home page is found at:

<http://www.openldap.org/>

10.1.2 The penguin colony network topology

Figure 10-1 on page 193 illustrates the network topology of our penguin colony.

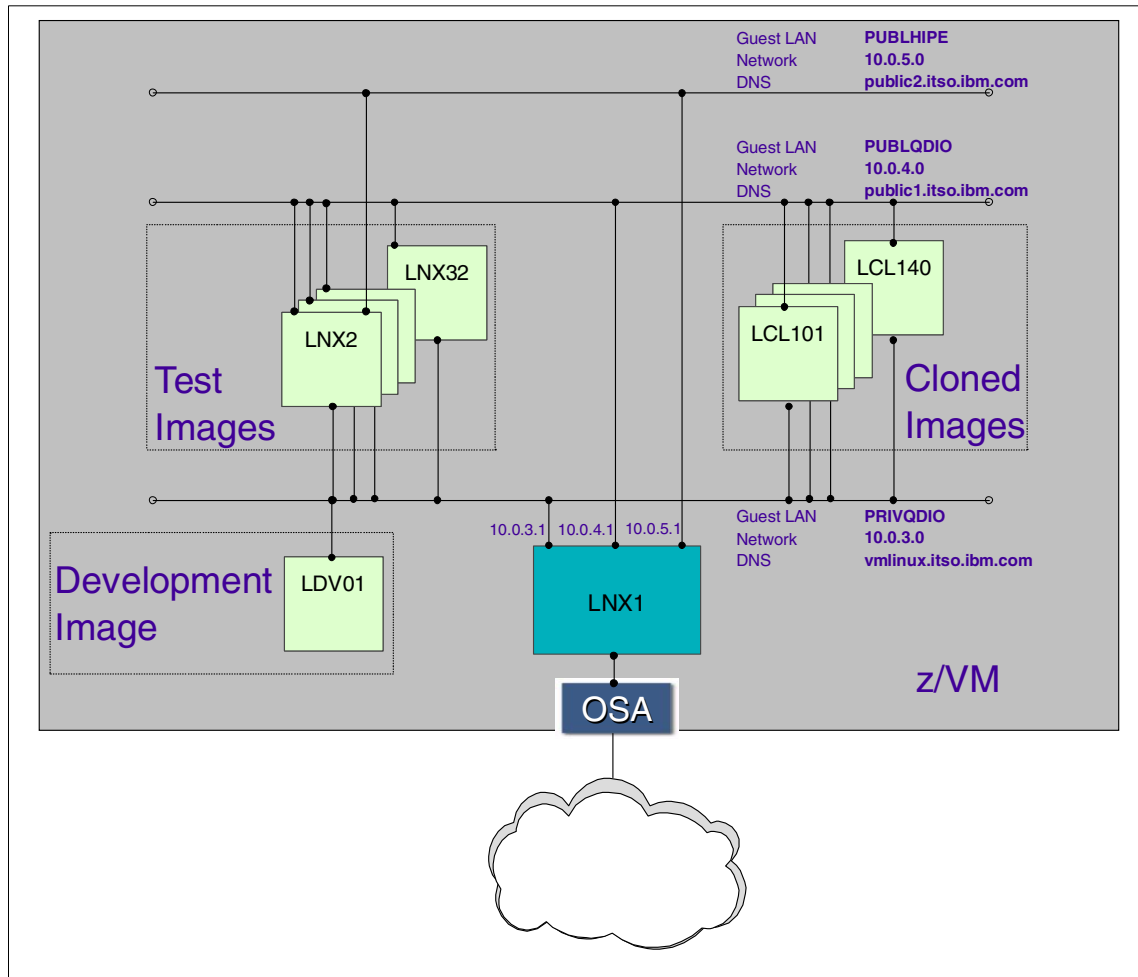


Figure 10-1 The network topology of the penguin colony

Details about key points in the figure are described here.

► The LN_x1 virtual machine

This Linux guest serves as the default gateway to the external network. All network traffic to and from the penguin farm is routed through this guest and forwarded to the appropriate destination using an OSA interface shared with z/VM.

The LN_x1 guest runs services such as:

- **ipchains** for IP forwarding
- **s1apd** for LDAP services

- **bind** for DNS services
- ▶ The test images LNX2 through LNX32

These guests can either run as stand-alone Linux guests or as basevol/guestvol Linux guests. We can use these virtual machines to test the basevol and guestvol images created using the procedure outlined in Chapter 9, “Building a basevol/guestvol penguin colony” on page 173.
- ▶ The cloned images LCL101 through LCL140

These virtual machines are basevol/guestvol Linux guests created from the LNXCLONE prototype outlined in 9.7.1, “The LNXCLONE prototype” on page 181.
- ▶ The PRIVQDIO VM Guest LAN

This QDIO-type Guest LAN is the management network over which Linux clones acquire their initial network connectivity as outlined in 8.10, “Startup configuration” on page 162.
- ▶ The PUBLQDIO VM Guest LAN

This QDIO-type Guest LAN is a public network over which most network traffic in the penguin colony travels. Once a basevol/guestvol Linux guest acquires connectivity to the PRIVQDIO network, it will LDAP to configure its interface to this network, as described in 10.4, “Network configuration and initialization” on page 197.
- ▶ The PUBLHIPE VM Guest LAN

This HiperSocket-type Guest LAN is a public network available to test Linux guests.

10.2 Configuring the LDAP server

As illustrated in Example 10-1, we first configure the LDAP server which runs on the LNX1 guest (shown in Figure 10-1 on page 193).

Example 10-1 The OpenLDAP slapd.conf configuration file

```

include      /etc/openldap/schema/core.schema           1
include      /etc/openldap/schema/cosine.schema
include      /etc/openldap/schema/inetorgperson.schema
include      /etc/openldap/schema/nis.schema           2
include      /etc/openldap/schema/dnszone.schema       3
include      /etc/openldap/schema/redbook.schema       4

pidfile      /var/run/slapd.pid
argsfile     /var/run/slapd.args

```

```
database      ldbm
directory     /var/lib/ldap
suffix        "o=IBM-ITSO,c=US"
rootdn        "cn=Manager,o=IBM-ITSO,c=US"
rootpw        secret
```

5
6

The notes highlighted in Example 10-1 refer to the following points:

1. We include the default LDAP schemas:
 - core schema
The core LDAP schema based on RFC2251 through RFC2256.
 - cosine.schema
An X.500 directory schema based on RFC1274.
 - inetorgperson.schema
A schema to represent a person object class suitable for Internet directory services based on RFC2798.
2. The nis schema maps Network Information Service entities to LDAP based on RFC2307. We use this schema in 10.5, “UNIX authentication using LDAP” on page 201.
3. The dnszone schema maps DNS resource records to LDAP. We discuss DNS and LDAP in 10.6, “Using LDAP with Domain Name System” on page 207.
4. The redbook schema file maps the penguin colony network topology to LDAP. In 10.4, “Network configuration and initialization” on page 197, we discuss using LDAP to manage network configuration.
5. The distinguished name (DN) suffix is defined to be o=IBM-ITSO,c=US.
6. The root distinguished name is defined as cn=Manager,O=IBM-ITSO,C=US.

Note: A complete list and detailed description of all parameters can be found in the *LDAP Implementation HOWTO* available at:

<http://tldp.org/HOWTO/LDAP-HOWTO/>

10.3 LDAP tools

In the following section, we introduce LDAP tools that can assist you in maintaining an LDAP directory.

10.3.1 An LDAP browser

To visually examine and edit the contents of an LDAP directory, use an LDAP browser. A useful Java-based browser, available for both Windows and UNIX/Linux platforms, is the LDAP Browser/Editor at:

<http://www.iit.edu/~gawojar/ldap/>

10.3.2 LDAP Data Interchange Format

You can define LDAP directory entities in an ASCII file format referred to as LDAP Data Interchange Format (LDIF). These entities can then be added to the LDAP directory using the **ldapadd** command supplied with OpenLDAP.

As an example, to add entities defined in the LDIF file `entities.ldif` using the DN `cn=Manager,O=IBM-ITSO,C=US`, issue:

```
ldapadd -D "cn=Manager,O=IBM-ITSO,C=US" -W -x -f entities.ldif
```

where:

- D** Specifies the DN at which to bind the data
- W** The prompt for a password - use the `rootpw` defined in the `slapd.conf` file
- x** Use simple authentication
- f** Specifies the file containing LDIF definitions

For complete details on the **ldapadd** command, consult the man page.

10.3.3 LDAP migration tools

Padl Software provides tools to assist you in migrating existing nameservices to LDAP. You can obtain the migration package from:

<http://www.padl.com/OSS/MigrationTools.html>

Use these tools to create the base structure `o=IBM-ITSO,c=US`, as follows:

1. Download and extract the migration tools package:

```
tar -zxf MigrationTools.tgz
```

2. Customize the `MigrationTools/migrate_common.ph` file for our schema, as shown in Example 10-2 on page 197.

Note: When prompted, use the `rootpw` password specified in the `slapd.conf` file/.

Example 10-2 The customized MigrationTools/migrate_common.ph file

```
...
# Default DNS domain
$DEFAULT_MAIL_DOMAIN = "itso.ibm.com";

# Default base
$DEFAULT_BASE = "o=IBM-ITS0,c=US";
...
$DEFAULT_MAIL_HOST = "itso.ibm.com";

# turn this on to support more general object classes
# such as person.
$EXTENDED_SCHEMA = 1;
...
```

3. Generate an LDIF file:

```
./migrate_base > base.ldif
```

4. Import the LDIF file to the LDAP server:

```
ldapadd -D "cn=Manager,o=IBM-ITS0,c=US" -W -x -f base.ldif
```

10.4 Network configuration and initialization

In 8.11, “Network configuration” on page 164, we examine a procedure used by basevol/guestvol Linux guests to acquire an initial network configuration. The initial network corresponds to the VMGuestManagementNetwork class described in 10.4.1 “The redbook LDAP schema”.

Pictorially, this corresponds to the PRIVQDIO VM Guest LAN depicted in Figure 10-1 on page 193.

10.4.1 The redbook LDAP schema

To represent the network topology we store in LDAP, we create a custom LDAP schema using the redbook.schema file. A complete listing of schema can be found in Appendix C.7, “The redbook.schema file” on page 273.

10.4.2 Redbook LDAP object classes

Object classes defined in the redbook schema include:

VMGuest	A Linux guest virtual machine
VMGuestManagementNetwork	The network over which a VMGuest configures itself
VMGuestPublicNetwork	A public network available to a VMGuest

10.4.3 Redbook LDAP attributes

Attributes defined in the redbook schema include:

VMNodeName	The z/VM machine to which a VMGuest belongs.
VMGuestName	The name of the VMGuest.
VMGuestRole	The role the VMGuest is to assume. We define a single role: DEFAULT.
VMGuestNetwork	The name of a VMGuestPublicNetwork to which a VMGuest is connected.
VMGuestMgmtNetwork	The name of a VMGuestManagementNetwork to which a VMGuest is connected.
VMGuestGatewayIP	The gateway IP address of a VMGuestPublicNetwork or VMGuestManagementNetwork.
VMGuestNet	The network address of a VMGuestPublicNetwork or VMGuestManagementNetwork.
VMGuestNetMask	The network mask of a VMGuestPublicNetwork or VMGuestManagementNetwork.
VMGuestLDAPIP	The IP address of the LDAP server for a VMGuestManagementNetwork.
VMGuestNetType	The network type of a VMGuestPublicNetwork or VMGuestManagementNetwork. We use “eth” for a QDIO VM Guest LAN, and “hsi” for HiperSocket VM Guest LAN.
VMGuestCHANDEV	The virtual device addresses (in the style of an /etc/chandev.conf file) for a VMGuestPublicNetwork or VMGuestManagementNetwork.

In Figure 10-2, we illustrate the redbook schema.

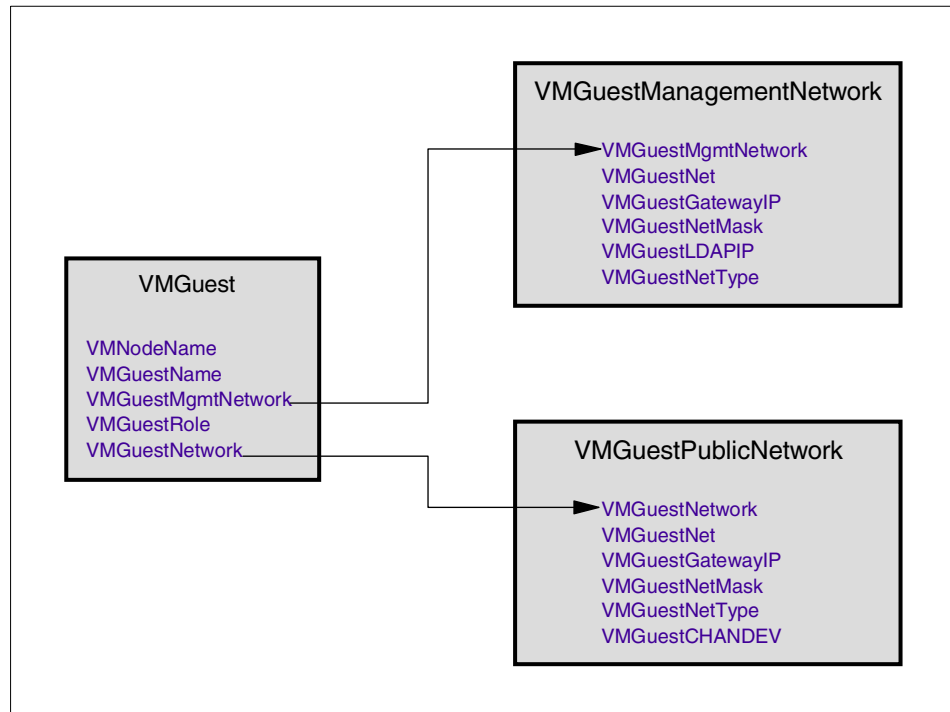


Figure 10-2 The redbook schema

10.4.4 The itsoldap script

After completion of the initialization of a basevol/guestvol Linux guest outlined in 8.10, “Startup configuration” on page 162, the Linux guest has the network interface to the PRIVQDIO network configured but not yet started. At this point, startup initialization will execute the `/etc/init.d/itsoldap` script (shown in Appendix C.2.1, “The `/etc/init.d/itsoldap` script” on page 263) to complete network configuration.

The main function of the `itsoldap` script is to:

1. Save backup copies of the Red Hat network configuration scripts and the `/etc/chandev.conf` file.
2. Start the network interface to the PRIVQDIO network.
3. Connect to the LDAP server running on the LNX1 machine (over the PRIVQDIO network) and acquire the network configuration parameters for this specific Linux guest. These parameters are expressed using the redbook schema shown in 10.4.1, “The redbook LDAP schema” on page 197.

4. For each VMGuestPublicNetwork, based on the LDAP attributes for the network:
 - a. Create a Red Hat network configuration file (a `/etc/sysconfig/network-scripts/ifcfg-*` file for the interface).
 - b. Add an entry to the `/etc/chandev.conf` file.
5. Create a clone-specific `ldap.conf` configuration for LDAP clients as discussed in 10.4.5, “Configuring LDAP clients” on page 200.
6. Shut down the network interface to the PRIVQDIO network.

After completion of `/etc/init.d/itsoldap`, `init` runs the `/etc/init.d/network` script to complete network initialization.

10.4.5 Configuring LDAP clients

System-wide default values for LDAP clients are set in the `/etc/openldap/ldap.conf` file. We specify the LDAP server IP address and the LDAP directory base DN.

As an additional security measure, we would like to restrict access to any server based on a user’s membership in the group `cn=machine-name,ou=HostAccess,o=IBM-ITSO,c=US`.

For example, in Example 10-3, we specify this line to specify only users defined to the PAM_GROUPDN group may login:

```
PAM_GROUPDN cn=lnx16.vmlinux.itso.ibm.com,ou=HostAccess,o=IBM-ITSO,c=US
```

Example 10-3 The `ldap.conf` file for Linux guest `lnx16`

```
ldap_version 3

HOST 10.0.3.1
PORT 389

BASE o=IBM-ITSO,c=US

PAM_GROUPDN cn=lnx16.vmlinux.itso.ibm.com,ou=HostAccess,o=IBM-ITSO,c=US
PAM_CHECK_HOST_ATTR no

PAM_PASSWORD exop

#SSL START_TLS
#TLS_CHECKPEER yes
TLS_CACERTFILE /etc/openldap/ca.cert
```

To create a `ldap.conf` file specific to each Linux clone, we use the template illustrated in Example 10-4.

Example 10-4 The `ldap.template.conf` file used to generate `ldap.conf`

```
ldap_version 3

HOST @LDAPHOST@
PORT 389

BASE @LDAPBASEDN@

PAM_GROUPDN @LDAPPAMGROUPDNE@
PAM_CHECK_HOST_ATTR no

PAM_PASSWORD exop

#SSL START_TLS
#TLS_CHECKPEER yes
TLS_CACERTFILE /etc/openldap/ca.cert
```

10.5 UNIX authentication using LDAP

Using Name Service Switch (NSS) and Pluggable Authentication Modules (PAM), we can configure the penguin colony to authenticate UNIX users and groups from a central LDAP directory.

A large list of information sources on UNIX authentication and LDAP can be found at the PADL software documentation Web site:

<http://www.padl.com/Contents/Documentation.html>

For information on PAM authentication, see the *The Linux-PAM System Administrators' Guide* by Andrew G. Morgan available at:

<http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/pam.html>

10.5.1 The `nss-ldap` and `pam-ldap` modules

To enable LDAP authentication, we use the `nss-ldap` and `pam-ldap` modules. The latest versions are available under the GNU Lesser General Public License from the PADL Software Web site:

<http://www.padl.com/Contents/OpenSourceSoftware.html>

10.5.2 Configuring PAM for LDAP authentication and authorization

To configure PAM to use LDAP for authentication and authorization, we modify the appropriate PAM configuration file found in the `/etc/pam.d` directory.

For a Red Hat Linux system, we modify the `/etc/pam.d/system-auth` file as shown in Example 10-5.

Note: Red Hat uses the PAM module `pam_stack.so` in many of the service-specific PAM configuration files (such as `login`, `passwd`, and `sshd`).

The `pam_stack.so` module refers back to the `system-auth` configuration file, so changes made here immediately affect those services as well.

Example 10-5 LDAP modifications to `/etc/pam.d/system-auth` for Red Hat

```
##PAM-1.0
auth      required      /lib/security/pam_nologin.so
auth      sufficient    /lib/security/pam_pwdb.so shadow nodelay md5
auth      required      /lib/security/pam_ldap.so use_first_pass

account   sufficient    /lib/security/pam_localuser.so
account   required      /lib/security/pam_ldap.so

password  required      /lib/security/pam_cracklib.so
password  sufficient    /lib/security/pam_ldap.so
password  required      /lib/security/pam_pwdb.so shadow use_authtok md5

session   required      /lib/security/pam_limits.so
session   required      /lib/security/pam_mkhomedir.so skel=/etc/skel/ umask=0022
session   required      /lib/security/pam_pwdb.so
```

Note: Sequence is important in the PAM configuration file. The following account lines specify the `pam_localuser.so` module *first*, in order to ensure that a local user can logon in the event the LDAP server is unreachable:

```
account   sufficient    /lib/security/pam_localuser.so
account   required      /lib/security/pam_ldap.so
```

We use `pam_localuser.so` as opposed to `pam_unix.so` to avoid using NSS. This is discussed in 10.5.1, “The `nss-ldap` and `pam-ldap` modules” on page 201.

Although we use a Red Hat distribution as the basis of our penguin colony, a SuSE Linux system would require modification to the `/etc/pam.d/login` file, as shown in Example 10-6 on page 203.

Example 10-6 LDAP modifications to /etc/pam.d/login for SuSE

```
##PAM-1.0
auth    required      pam_nologin.so
auth    sufficient    pam_unix_auth.so
auth    required      pam_ldap.so          try_first_pass

account sufficient    pam_localuser.so
account required      pam_ldap.so

password required     pam_pwcheck.so      nullok
password required     pam_ldap.so        use_first_pass use_authok
password required     pam_unix.so        nullok use_first_pass use_authok

session required      pam_unix_session.so
session required      pam_limits.so
session required      pam_env.so
session optional      pam_mail.so
```

Important: Editing PAM configuration files can be extremely problematic! If done incorrectly, the result may be that you cannot logon to your system. For this reason, we recommend the following measures:

- ▶ Always keep an extra root terminal session active. Save a backup copy of any files you edit.
- ▶ Verify the changes in a new session (login as root and as an LDAP-managed user).
- ▶ Restore the backup configuration files from the root terminal session if you experience problems.

10.5.3 Configuring NSS for LDAP user and group mapping

To configure NSS to use LDAP for UNIX user and group mapping, we modify the /etc/nsswitch.conf file as shown in

Example 10-7 Changes to the /etc/nsswitch file for LDAP user and group mapping

```
...
passwd: files ldap
group:  files ldap
...
```

This specifies user and group information should be read from both local files (/etc/passwd and /etc/group), and from LDAP.

10.5.4 Migrating users and groups to LDAP

Using the LDAP migration tools introduced in 10.3.3, “LDAP migration tools” on page 196, you can import existing `/etc/passwd` and `/etc/group` files into the LDAP directory:

1. Create an LDIF representation of `/etc/passwd`:

```
migrate_passwd.pl /etc/passwd > users.ldif
```

2. Create an LDIF representation of `/etc/group`:

```
migrate_group.pl /etc/group > groups.ldif
```

3. Add the LDIF for users to the LDAP directory:

```
ldapadd -D "cn=Manager,o=IBM-ITS0,c=US" -W -x -f users.ldif
```

4. Add the LDIF for groups to the LDAP directory:

```
ldapadd -D "cn=Manager,o=IBM-ITS0,c=US" -W -x -f groups.ldif
```

Attention: The `users.ldif` and `groups.ldif` files contain readable user and group information. Although passwords are encrypted, treat these as you would password, shadow password, and group files.

10.5.5 Adding users and groups to LDAP

You can add users and groups to the LDAP directory by importing the definitions using the LDIF definition. In Example 10-8, we show the LDIF definition for POSIX user “mwei”.

Example 10-8 The LDIF definition for the “mwei” user

```
dn: uid=mwei,ou=People,o=IBM-ITS0,c=US
uid: mwei
objectClass: account
objectClass: posixAccount
objectClass: top
objectClass: shadowAccount
shadowMax: 10000
gecos: Michael Weisbach
cn: Michael Weisbach
homeDirectory: /home/mwei
gidNumber: 10000
uidNumber: 10000
loginShell: /bin/bash
shadowLastChange: 11901
userPassword: {CRYPT}e1NTSEF9VWdyV
```

In Example 10-9 on page 205, we show the LDIF definition for group “mwei”.

Example 10-9 The LDIF definition for the “mwei” group

```
dn: cn=mwei, ou=Group, o=IBM-ITSO,c=US
objectClass: posixGroup
objectClass: groupOfUniqueNames
objectClass: top
uniqueMember: uid=mwei,ou=People,o=IBM-ITSO,c=US
cn: mwei
gidNumber: 10000
```

10.5.6 Changing passwords stored in LDAP

To permit users to change their own passwords in LDAP, the appropriate Access Control Lists (ACLs) must be added to the LDAP `slapd.conf` configuration file. In Example 10-10, we show the pertinent lines to append to `slapd.conf`.

Example 10-10 The ACLs to add to `slapd.conf` for user password modification

```
access to attr=userpassword
        by self write
        by anonymous auth

access to *
        by self write
        by * read
```

Note: The `rootdn` user (specified as `cn=Manager,o=IBM-ITSO,c=US` in the `slapd.conf` file in Example 10-1 on page 194) has full write access to all data in the LDAP directory.

Once the password ACLs are enabled, LDAP passwords may be changed by using either the `passwd` command or the `ldapmodify` command.

Using the `pam_ldap.so` module

From the command line, LDAP passwords may be changed using `passwd`. This is enabled in the `/etc/pam.d/system-auth` file by the lines:

```
password    required    /lib/security/pam_cracklib.so
password    sufficient  /lib/security/pam_ldap.so
password    required    /lib/security/pam_pwdb.so shadow use_authok md5
```

Example 10-11 on page 206 illustrates the effect of the `pam_ldap.so` module on the `passwd` command.

Example 10-11 Changing password using the pam_ldap.so module

```
bash-2.05$ passwd
Enter login(LDAP) password:
New password:
Password too short
New password:
Re-enter new password:
LDAP password information changed for mwei
passwd: all authentication tokens updated successfully
bash-2.05$
```

Note: Password checking (as seen in the line Password too short) is enabled by the pam_cracklib.so module.

Using the ldapmodify command

LDAP passwords may be changed directly using the `ldapmodify` command:

1. Create an LDIF with the password modification. We show a sample in Example 10-12.
2. Issue the command:

```
ldapmodify -D 'cn=Manager,o=IBM-ITS0,c=US' -x -W -f passwd.ldif
```

Note: Users may modify their own passwords by simply specifying themselves as the DN user. For example, under user “mwei”, issue:

```
ldapmodify -D 'cn=mwei,o=IBM-ITS0,c=US' -x -W -f passwd.ldif
```

Supply the old password when prompted.

Example 10-12 LDIF definition to change password for user “mwei”

```
dn: uid=mwei,ou=People,o=IBM-ITS0,c=US
changetype: modify
replace: userpassword
userpassword: {crypt}68i4YmwUMGhs
```

To automate generation of the LDIF file with an encrypted password, use the Perl script shown in Example 10-13.

Example 10-13 Perl script to generate LDIF for password update

```
#!/usr/bin/perl
$basedn="ou=People,o=IBM-ITS0,c=US";
print "username: ";
$username= <STDIN>;chop ($username);
print "password: ";system ("stty -echo");
```



```
$cleartext= <STDIN>;system ("stty echo");chop ($cleartext);

/* generate "good" password hashes using random salt */
srand (time());
$salt=substr(rand 100,0,2);
$crypttext=crypt($cleartext,$salt);
print "\n\n";

print "dn: uid=$username,$basedn\n";
print "changetype: modify\n";
print "replace: userpassword\n";
print "userpassword: {crypt}$crypttext\n\n";
```

10.6 Using LDAP with Domain Name System

Using the open source Internet Software Consortium (ISC) bind implement of DNS, you can enable an LDAP backend for storing DNS resource records.

10.6.1 The SDB LDAP backend for the ISC bind server

The standard ISC bind packages include LDAP support. We need to build the bind server with LDAP support:

1. Obtain the ISC bind server source code.

The ISC bind server is available from the ISC products Web page:

<http://www.isc.org/products/BIND/>

Choose the current release (version 9.2.1, as of this writing) and download the source package: bind-9.2.1.tar.gz

2. Obtain the SDB LDAP backend source code.

The SDB LDAP backend is available from:

<http://www.venaas.no/ldap/bind-sdb/>

The latest release as of this writing is 0.9. Download the bind-sdb-ldap-0.9.tar.gz package. In addition, download the DNSzone schema: dnszone-schema.txt

Note: Once downloaded in text mode, rename the dnszone-schema.txt to dnszone.txt in the /etc/openldap/schema directory.

3. Uncompress the source packages.

In a directory suitable for compiling source code, issue:

```
tar -zxf bind-9.2.1.tar.gz
tar -zxf bind-sdb-ldap-0.9.tar.gz
```

4. Merge the SDB LDAP modifications into the bind source tree.

Copy the SDB LDAP modifications to the ISC bind directory:

```
cp bind-sdb-ldap-0.9/ldap.h bind-9.2.1/bin/named/include
cp bind-sdb-ldap-0.9/ldap.c bind-9.2.1/bin/named
```

5. Modify the bind Makefile to use the SDB LDAP backend.

Change lines in the bind-9.2.1/bin/named/Makefile.in file that begin with the string DBDRIVER:

```
DBDRIVER_OBJJS = ldapdb.@@
DBDRIVER_SRCS = ldapdb.c
DBDRIVER_INCLUDES = -I/usr/include
DBDRIVER_LIBS = -L/usr/lib64 -lldap -lber -lresolv
```

Attention: Ensure that the ldap.h file exists in the DBDRIVER_INCLUDES directory, and that the ldap.a file exists in the DBDRIVER_LIBS directory.

6. Modify the bind main.c file to use the SDB LDAP backend.

Change lines that contain the string “xxdb” in the bind-9.2.1/named/main.c file:

```
/*
 * Include header files for database drivers here.
 */
/* #include "xxdb.h" */
#include "ldapdb.h"
```

and:

```
/*
 * Add calls to register sdb drivers here.
 */
/* xxdb_init(); */
ldapdb_init();
```

7. Configure, build, and install the bind DNS server.

In the bind-9.2.1 directory, issue:

```
./configure --prefix=/usr/local/bind-9.2.1
make
make install
```

Note: This installs the bind to directory /usr/local/bind-9.2.1.

10.6.2 DNS resource records in LDAP

To mimic DNS zones in our LDAP database, we create DN records of the form:

```
zoneName=somezone,ou=DNS,o=IBM-ITSO,c=US
```

We use this scheme for both forward and reverse DNS mapping. For the topology represented in Figure 10-1 on page 193, we create DN records:

zoneName=vmlinux.itso.ibm.com,ou=DNS,o=IBM-ITSO,c=US

The forward mapping for the PRIVQDIO VM Guest LAN zone.

zoneName=3.0.10.IN-ADDR.ARPA,ou=DNS,o=IBM-ITSO,c=US

The reverse mapping for the PRIVQDIO VM Guest LAN zone.

zoneName=public1.itso.ibm.com,ou=DNS,o=IBM-ITSO,c=US

The forward mapping for the PUBLQDIO VM Guest LAN zone.

zoneName=4.0.10.IN-ADDR.ARPA,ou=DNS,o=IBM-ITSO,c=US

The reverse mapping for the PUBLQDIO VM Guest LAN zone.

zoneName=public2.itso.ibm.com,ou=DNS,o=IBM-ITSO,c=US

The forward mapping for the PUBLHIPE VM Guest LAN zone.

zoneName=5.0.10.IN-ADDR.ARPA,ou=DNS,o=IBM-ITSO,c=US

The reverse mapping for the PUBLHIPE VM Guest LAN zone.

Start Of Authority (SOA) resource records

We create a Start Of Authority (SOA) record for each DN zone. In Example 10-14, we show the LDIF definition of the SOA record for the public1.itso.ibm.com zone.

Example 10-14 LDIF definition of the SOA record for the public1.itso.ibm.com zone

```
dn: zoneName=public1.itso.ibm.com,ou=DNS,o=IBM-ITSO,c=US
objectClass: top
objectClass: dnsZone
relativeDomainName: @
sOARecord: 1nx1.vmlinux.itso.ibm.com. hostmaster.vmlinux.itso.ibm.com. 2002021
400 10800 1800 604800 86400
dNSTTL: 86400
zoneName: public1.itso.ibm.com
nSRRecord: 1nx1.vmlinux.itso.ibm.com.
mXRRecord: 10 mail.public1.itso.ibm.com.
mXRRecord: 20 mail.public2.itso.ibm.com.
```

The SOA record for the reverse mapping for public1.itso.ibm.com (the 10.0.4.0 network) is shown in Example 10-15.

Example 10-15 SOA record for the 4.0.10.IN-ADDR.ARPA zone (LDIF format)

```
dn: zoneName=4.0.10.IN-ADDR.ARPA,ou=DNS,o=IBM-ITSO,c=US
objectClass: top
objectClass: dNSZone
relativeDomainName: @
sOARecord: lnx1.vmlinux.itso.ibm.com. hostmaster.vmlinux.itso.ibm.com. 2002021
400 10800 1800 604800 86400
dNSTTL: 86400
zoneName: 4.0.10.IN-ADDR.ARPA
nSRecord: lnx1.vmlinux.itso.ibm.com.
```

Address and Pointer resource records

We add Address (A) and Pointer (PTR) records to define the hosts in each zone. In Example 10-16, we show the LDIF representation of the A record for the lnx1.public1.itso.ibm.com host.

Example 10-16 A record for host lnx1.public1.itso.ibm.com (LDIF format)

```
dn: cn=lnx1,zoneName=public1.itso.ibm.com,ou=DNS,o=IBM-ITSO,c=US
objectClass: top
objectClass: dNSZone
relativeDomainName: lnx1
aRecord: 10.0.4.1
dNSTTL: 86400
zoneName: public1.itso.ibm.com
```

In Example 10-17, we show the corresponding PTR record in LDIF format.

Example 10-17 PTR record for host lnx1.public1.itso.ibm.com (LDIF format)

```
dn: cn=1,zoneName=4.0.10.IN-ADDR.ARPA,ou=DNS,o=IBM-ITSO,c=US
objectClass: top
objectClass: dNSZone
relativeDomainName: 1
dNSTTL: 86400
zoneName: 4.0.10.IN-ADDR.ARPA
pTRRecord: lnx1.public1.itso.ibm.com.
```

Canonical Name resource records

To define alias 1dap within the public1.itso.ibm.com zone, we use a Canonical Name (CNAME) record, as shown in Example 10-18 on page 211.

Example 10-18 CNAME record for alias ldap.public1.itso.ibm.com (LDIF format)

```
dn: cn=lnx1-ldap,zoneName=vmlinux.itso.ibm.com,ou=DNS,o=IBM-ITSO,c=US
objectClass: top
objectClass: dNSZone
relativeDomainName: ldap
cNAMERecord: lnx1.vmlinux.itso.ibm.com.
dnSTTL: 86400
zoneName: vmlinux.itso.ibm.com
```

10.6.3 Configure the DNS server to use the LDAP backend

To configure bind (the DNS server) to use the populated LDAP, we add the modifications to named.conf file highlighted in Example 10-19.

Example 10-19 Modified named.conf file for LDAP lookup

```
...
zone "public1.itso.ibm.com" {
    type master;
    database "ldap ldap://127.0.0.1/zoneName=public1.itso.ibm.com,ou=DNS,
                                                    o=IBM-ITSO,c=US 172800";
};

...

zone "4.0.10.IN-ADDR.ARPA" {
    type master;
    database "ldap ldap://127.0.0.1/zoneName=4.0.10.IN-ADDR.ARPA,
                                                    ou=DNS,o=IBM-ITSO,c=US 172800";
};

...
```

10.6.4 Adding indexes to speed lookups

To speed DNS lookups against the LDAP database, we create equality indexes on the objectClass, relativeDomainName, and zoneName attributes.

In Example 10-20 on page 212, we show the required additions to the slapd.conf file.

Example 10-20 Additions to the slapd.conf file for DNS indexing

index	objectClass	eq
index	relativeDomainName	eq
index	zoneName	eq

10.7 A remote Web management interface to LDAP

Using the PHP scripting language, we can quickly devise a remote management tool for LDAP basevol/guestvol Linux guests.

Attention: As these examples provide for no encryption or authentication, they are inherently insecure. We provide these simply as samples.

10.7.1 Interface to reset passwords

In Appendix C.8, “The sample-ldap.php script” on page 276, we show a sample script to reset the password of LDAP users under the DN `ou=People,o=IBM-ITSO,c=US`. The interface is shown in Figure 10-3.



Figure 10-3 Interface to reset LDAP passwords

10.7.2 Interface to IPL and shutdown Linux guests

In Appendix C.9, “The ipl-shutdown.php script” on page 277, we show a sample script to reset the password of LDAP users under the DN `ou=People,o=IBM-ITSO,c=US`. The interface is shown in Figure 10-3.

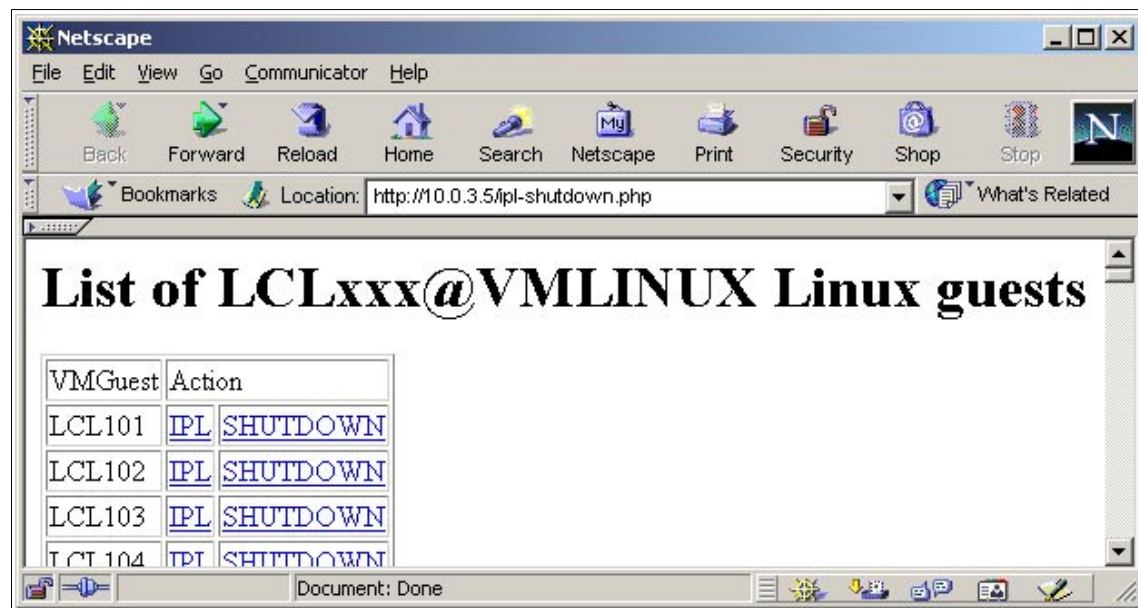


Figure 10-4 Interface to shutdown and IPL Linux guests



A

The Unit Record device driver and utility

In this appendix we describe the Unit Record (UR) device driver and utility, which copies data to and from a Linux guest and a z/VM virtual reader, punch, or printer.

A.1 The UR device driver and utility

The UR device driver provides a Linux character device interface to an attached unit record device for a Linux guest. The UR utility provides a user interface to the UR device driver.

Using the UR driver and utility, it is possible to exchange files between a Linux guest and a z/VM virtual machine (initiated within the Linux guest). The UR utility provides an interface for copying files between UR devices (typically the reader, punch, and printer defined by the virtual machine). It can handle any file block size, and record length, and will perform EBCDIC-to-ASCII conversion as required.

The UR device driver and utility can be downloaded from the Internet as described in Appendix D, “Additional material” on page 279.

A.2 The UR device driver

The UR device driver is distributed in source code format - it is packaged in the `ur-0.4.tar.gz` file. Perform the following steps to install the UR device driver.

A.2.1 Build the UR device driver

Extract the source code from the package file:

```
tar -zxf ur-0.4.tar.gz
```

Change to the UR device driver source directory and build the driver:

```
cd ur-0.4
make
```

Note: The README file explains how to modify the Makefile in case your kernel include files are not accessible using the `/usr/include` directory.

A.2.2 Install the UR device driver

The device driver is built as a dynamically loaded kernel module, the `ur.o` module:

1. Copy the driver to the loaded kernel modules misc directory:

```
cp ur.o /lib/modules/2.4.9-37/misc
```

2. Three helper utilities should be copied to a directory in the PATH:

```
cp makeurdev addvdev removevdev /usr/local/sbin
```

3. Install the device driver:

```
insmod ur
```

A.2.3 Create the UR character devices

After installing the UR driver, you need to create the character devices. Issue the command:

```
makeurdev
```

Attention: The device driver uses major number 240 - defined by LANANA for use by local or experimental drivers. This may clash with another experimental on your system. If so, redefine the UR_MAJOR macro in the ur.c file and re-compile (be sure to re-run `makeurdev` if you need redefine UR_MAJOR).

This creates (using the `mknod` command) the following character devices:

- ▶ `/dev/urctl` (minor number 0)
- ▶ `/dev/ur1` through `/dev/ur8` (minor number 1 through 8)

Then execute the command:

```
addvmur
```

A.2.4 The `addvmur` command

The UR device is typically used to provide access to the VM reader, punch, and printer. The `addvmur` command is used to associate these to a specific UR character device and provide a readable symbolic link to the respective device.

In Table A-1, we show the device node, symbolic links, device minor numbers, VM virtual device number, and record lengths of the character devices created by `addvmur`.

Table A-1 The RDR DR, PUN and PRT devices registered by `addvmur`

Symbolic link	Device node	Minor	Device type	Devno	Reclen
<code>/dev/vmreader</code>	<code>/dev/ur1</code>	1	reader	000C	80
<code>/dev/vmpunch</code>	<code>/dev/ur2</code>	2	punch	000D	80
<code>/dev/vmprinter</code>	<code>/dev/ur3</code>	3	printer	000E	132

Note: After executing `addvmur`, there will be five remaining UR devices (`/dev/ur4` through `/dev/ur8`). If you need more than this, you should execute the `mknod` command, supplying the appropriate major and minor numbers.

A.3 The UR utility

The UR utility provides an interface for dynamically adding and removing unit record devices. It can copy data between UR devices and a Linux guest, or between two unit record character devices. Block size, record length, and EBCDIC to ASCII conversions are handled by the utility.

A.3.1 Install the UR utility

The UR utility is distributed in RPM format, the `ur-utils-0.2-1.s390x.rpm` package. The source code is distributed in the `ur-utils-0.2-1.src.rpm` package.

The binaries can be installed on a RedHat 7.1 s390x Linux guest using:

```
rpm -ivh ur-utils-0.2-1.s390x.rpm
```

The `ur` command will be installed in the `/usr/bin` directory.

A.3.2 The `ur` command syntax

The `ur` command syntax is:

```
ur copy [ -tbf ] [ infile | - ] [ outfile | - ]
ur info devfile
ur list
ur add minor devno blkosz reclen flags [ devname [ perm ] ]
ur remove minor
```

A.3.3 The `copy` subcommand

Use the `copy` subcommand to copy data to and from unit record devices.

The `infile` parameter indicates the data source. If omitted, or specified as “-”, data is read from standard input (stdin).

The `outfile` parameter indicates the data destination. If omitted, or specified as “-”, data is written to standard output (stdout).

A.3.3.1 Flags

The optional flags accepted by `ur copy` are:

-t Indicates a text data transfer. Data records are translated to and from ASCII or EBCDIC as appropriate.

When writing to a unit record device from a Linux input:

1. Input lines are translated from ASCII to EBCDIC.
2. The newline character is removed.
3. The output is padded with the NUL character up to the device record length.
4. If the input line exceeds the output device record length, the copy operation terminates with a non-zero return code. This behavior may be overridden using the `-f` option.

Copy operations which specify unit record devices as both input and output are treated as binary copies.

-b Indicates a binary data transfer - this is the default. No ASCII or EBCDIC translation will apply. Data transfers to and from a unit record device will use the block size associated with the device.

If both input and output are both unit record devices:

- ▶ Data is copied one record at a time.
- ▶ If the output record length exceeds input record length, records will be padded with the NUL character.
- ▶ If the input record length exceeds output record length, the copy operation terminates with a non-zero return code. This behavior may be overridden using the `-f` option.

-f Indicates record truncation are permitted. In situations where the output unit record device record length is exceeded, this option allows the copy to proceed normally.

A.3.4 The `info` subcommand

Use the `info` subcommand to list information about unit record device *devfile*. Reported information includes:

- ▶ The device minor number (in decimal)
- ▶ The subchannel (in hexadecimal)
- ▶ The virtual device number (in hexadecimal)
- ▶ The block size (in decimal)
- ▶ The record length (in decimal)
- ▶ Driver-specific flags (“r” indicates a readable device, “w” a writable device)

A.3.5 The list subcommand

Use the **list** subcommand to list all registered unit record devices. Reported information is preceded by the header:

```
MINOR SUBCH DEVNO BLKSIZE RECLEN FLAGS
```

where each field corresponds to the values reported by the **info** subcommand.

A.3.6 The add subcommand

Use the **add** subcommand to register a new unit record device. Parameters to the subcommand are defined as:

minor	The minor number associated with the device.
devno	The virtual device number for the unit record device. This is the hexadecimal number defined in the device virtual machine.
blksize	The block size associated with the device.
reclen	The record length associated with the device.
devname	An optional device name for the device. If provided, a device node will be created (using mknod) using the provided path name. If the device node exists, it will first be removed using unlink .
perms	The optional permissions to assign to the newly created device node. This defaults to 0700 (argument is assumed to be octal). Note: Permissions are modified by the current umask setting.

A.3.7 The remove subcommand

Use the **remove** subcommand to remove a unit record device. The supplied parameter indicates the minor number of the device to remove.



B

Installing Red Hat 7.1 with OCO modules

In this appendix we describe how to install Red Hat 7.1 on a Linux guest with OCO qeth modules support.

B.1 The Red Hat for zSeries distribution

Red Hat does not provide the OCO qeth modules on the installation CD-ROMs. In order to use Red Hat on zSeries with a VM Guest LAN, you can:

- ▶ Install the Linux guest using a point-to-to network (CTC or IUCV)
- ▶ Boot the Linux guest and install the OCO qeth modules
- ▶ Configure the interfaces using the qeth modules

Alternatively, the Red Hat installer allows you to provide a second initial ramdisk containing the IBM OCO qeth drivers at installation. The installer then uses the second initial ramdisk to install the OCO modules. This method simplifies the process - the qeth drivers and their interfaces can be installed and configured at system installation time.

We describe an installation method using a second initial ramdisk.

B.2 Obtain the latest OCO drivers

To prepare for installation, you need to first obtain the OCO drivers. You can obtain the latest OCO drivers from the IBM Developerworks site:

```
http://www.ibm.com/developerworks/opensource/linux390/special\_oco\_rh\_2.4.shtml
```

1. Download the `redhat-oco-2.4.9-37-s390x-1.tar.gz` file.
2. Extract the package to the root directory:

```
cd /  
tar -zxf /root/redhat-oco-2.4.9-37-s390x-1.tar.gz
```

The package contains files:

- `redhat-oco/LICENSE`

The license agreement for using the OCO modules.

- `redhat-oco/README`

Instructions to read and agree to the license before using the OCO modules.

- `redhat-oco/redhat-oco-2.4.9-37-s390x-1.tgz`

The OCO drivers packaged as gzipped tar files.

Important: You must read and agree to the LICENSE before continuing.

3. After reading and agreeing to the LICENSE, extract the OCO drivers:


```
cd redhat-oco
tar -zxf redhat-oco-2.4.9-37-s390x-1.tgz
```

Note: This extracts the drivers to directories:

- ▶ lib/modules/2.4.9-37
- ▶ lib/modules/2.4.9-37BOOT
- ▶ lib/modules/2.4.9-37BOOTtape
- ▶ lib/modules/2.4.9-37tape

B.3 Create a second initial ramdisk for OCO qeth drivers

Now create a second initial ramdisk containing the OCO qeth modules, following the instructions in the README file of the first Red Hat installation CD-ROM.

1. Create an initrd filesystem. To simplify the process, use the `mkinitrd` command:

```
mkinitrd /boot/initrd-OCO.img 2.4.9-37
```

2. Uncompress and mount the newly created filesystem:

```
zcat /boot/initrd-OCO.img > myimage
mkdir -p /mnt/myimage
mount -o loop myimage /mnt/myimage
```

3. Remove the existing contents of the initrd filesystem, and copy the OCO drivers to the initrd filesystem:

```
rm -Rf /mnt/myimage/*
cp -a /redhat-oco/lib /mnt/myimage
```

4. Unmount the modified initial ramdisk, compress it, and save to disk:

```
umount /mnt/myimage
gzip < myimage > /boot/initrd-OCO.img
```

The `/boot/initrd-OCO.img` file is now ready for use as a second initial disk.

B.4 Copy the installation images to the guest reader

Now copy the installation images and the second initial ramdisk to the reader of the virtual machine to be installed. We used the unit record utility outlined in Appendix A, “The Unit Record device driver and utility” on page 215.

First you need to obtain copies of the installation images from the first installation CD-ROM.

B.4.1 Mount the installation CD-ROM image

The CD-ROM images were previously copied to an ext2 filesystem in the LNX1 virtual machine. The filesystem was created on the 203 DASD device which maps to the /dev/dasdc device.

1. Mount the /dev/dasdc device on the /mnt/iso directory:

```
mount /dev/dasdc1 /mnt/iso
```

2. Mount CD-ROM image on the /rh71/cd1 directory:

```
mkdir -p /rh71/cd1
mount -o loop /mnt/iso/rh-7.1-en-s390x.cd1.iso /rh71/cd1
```

B.4.2 Copy the installation images to the VM reader

Now copy the installation images to the VM reader for the virtual machine to be installed. In Example B-1, we show the command sequence.

Example: B-1 Using ur to copy installation images to a VM reader

```
# cd /rh71/cd1
# hcp spool pun to ldv01
# ur copy kernel.img /dev/vmpunch
# hcp close pun
PUN FILE 0061 SENT TO LDV01    RDR AS  0001 RECS 041K CPY  001 A NOHOLD NOKEEP
# ur copy redhat.prm /dev/vmpunch
# hcp close pun
PUN FILE 0062 SENT TO LDV01    RDR AS  0002 RECS 0001 CPY  001 A NOHOLD NOKEEP
# ur copy initrd.img /dev/vmpunch
# hcp close pun
PUN FILE 0063 SENT TO LDV01    RDR AS  0003 RECS 060K CPY  001 A NOHOLD NOKEEP
# ur copy /boot/initrd-0C0.img /dev/vmpunch
# hcp close pun
PUN FILE 0064 SENT TO LDV01    RDR AS  0004 RECS 5394 CPY  001 A NOHOLD NOKEEP
```

B.5 Install the Linux guest

You are now ready to install the Linux guest. In the instructions that follow, we assume the virtual machine definition for the Linux guest to install include:

- ▶ A 201 DASD device to be used as a Linux swap device
- ▶ A 202 DASD device to be used for the Linux root filesystem
- ▶ A QDIO-type simulated NIC at virtual devices 0700, 0701, and 0702

B.5.1 Beginning the installation

First logon the guest virtual machine (we used the LDV01 user). At this point, the reader contains the Red Hat installation images, along with the second initial ramdisk for the OCO modules.

Install from the CD-ROM images on the LNX1 203 minidisk.

1. Define the minidisk at virtual device 203:

```
LINK LNX1 203 203
```

2. To ensure the installation images will not be deleted from the reader:

```
CH RDR ALL KEEP NOHOLD
```

3. To begin installation, IPL from the virtual reader:

```
IPL 00C
```

B.5.2 First stage configuration

At this point configuration proceeds from the 3270 console. The installer will prompt for network configuration and DASD device assignments. Later, installation will be completed from a telnet session.

Network configuration

On IPL, the installer loaded the second initial ramdisk containing the OCO modules. Provide configuration parameters for network configuration:

1. We choose the **eth0** network interface.

Note: If we had defined a HiperSocket-type simulated NIC (instead of a QDIO-type), we would select the **hsi0** interface.

2. When prompted for the first chandev configuration parameters, we supplied:

```
qeth0,0x0700,0x0701,0x0702,0,0,0
```

Important: Be sure to supply **qeth0** as the first - *not* **eth0**.

3. When prompted for the second chandev configuration parameters, we supplied:

```
add_parms,0x10,0x0700,0x0702,portname:NIC0700
```

Important: A portname *must* be specified. For VM Guest LAN, the actual name used has no special significance. However, the chosen name is used for reporting purposes. We recommend using the same name for all interfaces connected to a guest LAN.

4. The installer then prompts for specific IP configuration parameters, such as:
 - IP address
 - network mask
 - network address
 - default gateway
 - DNS server address

We provided the values specific to our installation.

DASD device configuration

To avoid auto-probing for DASD devices, the installer next prompts for DASD virtual device numbers. We entered the range:

```
202-203
```

These devices map to the Linux names:

202	The /dev/dasda device used for the Linux root filesystem.
203	The /dev/dasdb device containing the installation CD-ROM images.

B.5.3 Second stage configuration

At this point, the Red Hat installer configures the network and DASD devices. When prompted, we started a telnet to the Linux guest using the IP address provided in the first stage configuration. We logged in as root and continued the installation using the command:

```
loader
```

Note: We logged in as root - no password is required.

Choosing the installation medium

Select the **Hard Drive** option when prompted for the installation media, and select the /dev/dasdb1 device.

Format the DASD

Next, format the DASD devices. Begin formatting the /dev/dasda device by choosing the device and selecting **Format DASD**.

Attention: Do not attempt format the /dev/dasdb device - it contains an ext2 filesystem with the CD-ROM images.

Partition the DASD

Next, create a Linux partition on the minidisks. Select the device and choose **Edit Partitions** to start **fdasd**. In Example B-2, we show the command dialog.

Example: B-2 The fdasd command dialog

Command action

```
m  print this menu
p  print the partition table
n  add a new partition
d  delete a partition
v  change volume serial
t  change partition type
r  re-create VTOC
s  show mapping (partition number - data set name)
q  quit without saving changes
w  write table to disk and exit
```

Command (m for help):

Attention: Do not attempt to partition the /dev/dasdb device.

Following are explanations of the actions highlighted in Example B-2.

Add a new partition

To add a new partition, enter: n at the command prompt. Accept all the default parameters in the subsequent dialog to create a Linux ext2 partition.

Note: The dialog will prompt for the starting track number and partition size. We used a single partition starting at track 2 (track 1 contains the VTOC record for OS/390 and z/OS compatibility) and spanning the entire minidisk.

Write table to disk and exit

To write the partition table to disk and exit, enter: w on the command line.

Partition the /dev/dasda1 device

Next, partition the /dev/dasda device for use as a ext2 filesystem.

After selecting the device and choosing **Edit Partitions**:

1. Add a new partition.
2. Write table to disk and exit.

Important: We chose an ext2 filesystem for the /dev/dasda - not ext3. Even when mounting a read-only ext3 filesystem, the kernel attempts to replay the filesystem journal (and therefore, write to the filesystem). When using a basevol/guestvol Linux guest, the read-only basevol cannot be corrupted by an unclean shutdown.

Converting an ext2 filesystem to ext3

You can convert an ext2 filesystem to ext3 using the command:

```
tune2fs -j devname
```

where *devname* is the DASD block device name (/dev/dasda1, for instance).

Converting an ext3 filesystem to ext2

You can convert an ext3 filesystem to ext2 using the command

```
tune2fs -O ^has_journal devname
```

where *devname* is the DASD block device name (/dev/dasda1, for instance).

Note: The -O is the uppercase letter O, the ^ is the caret symbol. This denotes option “turn the **has_journal** option off”.

Important: This should only be issued against a clean filesystem. To verify the filesystem after the conversion, run the command:

```
e2fsck -f devname
```

Kernel boot parameters

The Red Hat installer will prompt for any additional boot parameters to add to the /etc/zipl.conf file. We append the following option to the **Kernel Parameters** entry field:

```
vmppoff=LOGOFF
```

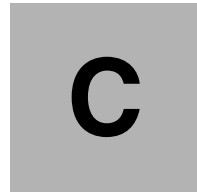
Note: Kernel boot parameters are whitespace delimited - do not use a comma to append to the existing parameter list.

The vmppoff option allows you select a CP command to run when the **halt -p** command is issued. We run the **LOGOFF** command in order to enable XAUTOLOG for the guest virtual machine.

Note: The `vmhalt` option is documented in *Linux for zSeries and S/390 Device Drivers and Installation Commands*, LNUX-1303. However, the `vmppoff` option is not.

When the `vmhalt` option is specified, its value indicates the CP command to run when `halt` is executed; the `vmppoff` command will not be executed. Similarly, when executing the `halt -p` command, the `vmhalt` command will not be executed. To summarize these parameters:

halt -p	Executes the command specified by vmppoff
halt	Executes the command specified by vmhalt



Scripts and configuration files

In this appendix, we provide listings of the scripts and configuration files developed for and used in this redbook.

C.1 The basevol+guestvol-1.0.0-1.noarch.rpm package

We list some of the scripts found in the basevol+guestvol-1.0.0-1.noarch.rpm package.

C.1.1 The /etc/rc.d/rc.guestvol script

In Example C-1, we show the /etc/rc.d/rc.guestvol script discussed in 8.9.1, “The rc.guestvol script” on page 160.

Example: C-1 The /etc/rc.d/rc.guestvol script

```
#!/bin/sh
# rc.guestvol - Find and mount a guestvol on Linux for S/390 or zSeries
# Copyright (C) 2002 IBM UK Ltd
# Author: Malcolm Beattie <beattiem@uk.ibm.com>
#
# This program is free software; you can redistribute it and/or
# modify it under the terms of the GNU General Public License
# as published by the Free Software Foundation; either version
# 2 of the License, or (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# 15 Jul 2002 Initial test version
# 18 Jul 2002 Remove unneeded debug telemetry
#

mount -n -t proc /proc /proc

# First see if we're a development guest which has a virtual
# printer (a 1403 from CP DEF PRT 777) at 777 to "block" the
# guestvol slot. If so, go straight to the ordinary non-guestvol
# boot process. If 777 is merely absent, we give them the benefit
# of the doubt and drop to a maintenance shell so that guestvol
# guests with broken (or unusual) device configs can recover.

if grep -q '^0777.....1403' /proc/subchannels; then
    umount /proc
    exec /etc/rc.d/rc.sysinit
fi

#
# Find and mount the guestvol. Ensure 777 is in the range of active
# DASD devices, look for an active devno 777 in /proc/dasd/devices,
```

```

# find its device name and append a "1" to use partition 1 of that disk.
#
grep -q '^0777' /proc/dasd/devices || echo "add range=777" > /proc/dasd/devices
dev=`awk '/^0777/ && $9 == "active"' { print $7 }' /proc/dasd/devices`

if [ "x$dev" != "x" ]; then
    guestdev="/dev/${dev}1"
    echo "Guestvol device 777 appears to be at device file $guestdev" # debug
    mount -n "$guestdev" /guestvol
fi

rc=0
while [ ! -f /guestvol/etc/inittab -o $rc -ne 0 ]; do
    echo "Error: failed to find /guestvol/etc/inittab" 2>&1
    echo "About to start maintenance shell. Please either mount guestvol"
    echo "disk on /guestvol and then type \"exit\" or else just type"
    echo "\"exit 123\" to boot the underlying basevol system in "
    echo "read-write mode with no guestvol."
    bash -i
    rc=$?
    if [ $rc -eq 123 ]; then
        echo "Maintenance shell exited with code 123."
        echo "Leaving guestvol script and returning to ordinary boot"
        echo "procedure of basevol in read-write mode."
        exec /etc/rc.d/rc.sysinit
    fi
    if [ $rc -ne 0 ]; then
        echo "Shell exited with unexpected return code $rc." 2>&1
    fi
done

echo "Guestvol seems to have been successfully mounted...continuing boot"

#
# Preserve access to original basevol directories via /basevol
#
for dir in etc var boot root dev; do
    mount --bind /$dir /basevol/$dir
done

#
# Bind the necessary directories into the main filesystem namespace
#
mount --bind /guestvol/etc /etc

# Initialise mtab here (on the new guestvol /etc) instead of in rc.sysinit
# We need to find guestvol in /proc/mounts because we may not know the
# underlying device name if someone mounted it from the maintenance shell.
grep /guestvol /proc/mounts > /etc/mtab

```

```

# Use "mount -f" to create more mtab entries while only faking the mounts
for dir in etc var boot root dev; do
    mount -f --bind /$dir /basevol/$dir
done
mount -f --bind /guestvol/etc /etc

umount -n /proc
# Note that /proc has now been unmounted again

# Now bind all the rest from guestvol. We can leave out the "-n"
# from the mount command this time since the entries will be written
# to the "correct" mtab on the new, writable /etc.
# If devfs is in use then dev can be removed from the list below
# If tmpfs is in use then tmp can be removed from the list below
for dir in var root dev opt tmp home usr/local boot mnt; do
    mount --bind /guestvol/$dir /$dir
done

# Put the original (readonly) basevol /var/lib/rpm in place over
# the current empty /var/lib/rpm. RPMs installed by the guest go
# in --dbpath /var/lib/guestrpm.
mount --bind /basevol/var/lib/rpm /var/lib/rpm

echo "Finished binding guestvol directories, init will now restart..."

#
# Kick init to reread the new /etc/inittab
#
kill -1 1

```

C.1.2 The /etc/rc.d/rc.sysinit-guestvol script

In Example C-2, we show the /etc/rc.d/rc.sysinit-guestvol script discussed in 8.10.1, “The rc.sysinit-guestvol script” on page 163.

Example: C-2 The /etc/rc.d/rc.sysinit-guestvol script

```

#!/bin/bash
#
# /etc/rc.sysinit - run once at boot time
#
# Taken in part from Miquel van Smoorenburg's bcheckrc.
#
# This is a modified rc.sysinit for the guestvol environment.
# Modifications by Malcolm Beattie.
# Don't remount the root filesystem read-write (we leave it readonly)
# Don't clear out /etc/mtab (rc.guestvol has already done some magic on it)
#

```

```

# Rerun ourselves through initlog
if [ -z "$IN_INITLOG" ]; then
    [ -f /sbin/initlog ] && exec /sbin/initlog $INITLOG_ARGS -r
    /etc/rc.d/rc.sysinit-guestvol
fi

# If we're using devfs, start devfsd now - we need the old device names
[ -e /dev/.devfsd -a -x /sbin/devfsd ] && /sbin/devfsd /dev

LD_LIBRARY_PATH=/lib
for i in `cat /etc/ld.so.conf`; do LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:$i ; done
export LD_LIBRARY_PATH
# Set the path
PATH=/bin:/sbin:/usr/bin:/usr/sbin
export PATH

HOSTNAME=`/bin/hostname`

# Read in config data.
if [ -f /etc/sysconfig/network ]; then
    . /etc/sysconfig/network
else
    NETWORKING=no
fi

if [ -z "$HOSTNAME" -o "$HOSTNAME" = "(none)" ]; then
    HOSTNAME=localhost
fi

# Source functions
. /etc/init.d/functions

# Print a banner. ;)
echo -en "\t\t\tWelcome to "
[ "$BOOTUP" != "serial" ] && echo -en "\033[1;31m"
echo -en "Red Hat"
[ "$BOOTUP" != "serial" ] && echo -en "\033[0;39m"
echo $" Linux"
if [ "$PROMPT" != "no" ]; then
    echo -en "\t\t\tPress 'I' to enter interactive startup."
    echo
    sleep 1
fi

# Mount /proc (done here so volume labels can work with fsck)

action $"Mounting proc filesystem: " mount -n -t proc /proc /proc

```

```

# Fix up kernel versioning on binary-only modules
if [ -x /sbin/oco-setkver ]; then
    kver=`</proc/sys/kernel/osrelease`
    kernelver=`echo $kver|awk -F '-' '{ print $1 }'`
    if [ "$HOSTTYPE" = "s390x" ]; then
        kernelver="${kernelver}x"
    fi
    if [ "${kver:0:3}" = "2.4" ]; then
        modpath="/lib/modules/$kver/kernel/net"
    else
        modpath="/lib/modules/$kver/net"
    fi
    for i in /lib/modules/ibm/*; do
        [ -e $i ] || break
        /sbin/oco-setkver $kver $i $modpath/`basename $i`
    done
fi

# Fix console loglevel
/bin/dmesg -n $LOGLEVEL

# Unmount the initrd, if necessary
if grep -q /initrd /proc/mounts ; then
    action "$Unmounting initrd: " umount /initrd
    /sbin/blockdev --flushbufs /dev/ram0 >/dev/null 2>&1
fi

# Configure kernel parameters

action "$Configuring kernel parameters: " sysctl -e -p /etc/sysctl.conf

# Set the system clock.
ARC=0
SRM=0
UTC=0

if [ -f /etc/sysconfig/clock ]; then
    . /etc/sysconfig/clock

    # convert old style clock config to new values
    if [ "${CLOCKMODE}" = "GMT" ]; then
        UTC=true
    elif [ "${CLOCKMODE}" = "ARC" ]; then
        ARC=true
    fi
fi

CLOCKDEF=""
CLOCKFLAGS="--hctosys"

```

```

case "$UTC" in
    yes|true)
        CLOCKFLAGS="$CLOCKFLAGS --utc";
        CLOCKDEF="$CLOCKDEF (utc)";
        ;;
    no|false)
        CLOCKFLAGS="$CLOCKFLAGS --localtime";
        CLOCKDEF="$CLOCKDEF (localtime)";
        ;;
esac

case "$ARC" in
    yes|true)
        CLOCKFLAGS="$CLOCKFLAGS --arc";
        CLOCKDEF="$CLOCKDEF (arc)";
        ;;
esac

case "$SRM" in
    yes|true)
        CLOCKFLAGS="$CLOCKFLAGS --srm";
        CLOCKDEF="$CLOCKDEF (srm)";
        ;;
esac

if [ -x /bin/hwclock ] ; then
    /sbin/hwclock $CLOCKFLAGS
    action $"Setting clock $CLOCKDEF: `date`" date
else
    # System date on S390 is always set correctly
    action $"System date: `date` " date
fi

if [ "`/sbin/consoletype`" = "vt" ]; then
    # Load keymap
    if [ -x /bin/loadkeys ]; then
        KEYTABLE=
        KEYMAP=
        if [ -f /etc/sysconfig/console/default.kmap ]; then
            KEYMAP=/etc/sysconfig/console/default.kmap
        else
            if [ -f /etc/sysconfig/keyboard ]; then
                . /etc/sysconfig/keyboard
            fi
            if [ -n "$KEYTABLE" -a -d "/usr/lib/kbd/keymaps" -o -d "/lib/kbd/keymaps" ];
then
                KEYMAP=$KEYTABLE
            fi
        fi
    fi

```

```

if [ -n "$KEYMAP" ]; then
# Since this takes in/output from stdin/out, we can't use initlog
if [ -n "$KEYTABLE" ]; then
    echo -n "$Loading default keymap ($KEYTABLE): "
else
    echo -n "$Loading default keymap: "
fi
loadkeys $KEYMAP < /dev/tty0 > /dev/tty0 2>/dev/null && \
    success "$Loading default keymap" || failure "$Loading default keymap"
echo
fi
fi

# Load system font
if [ -x /sbin/setsysfont ]; then
[ -f /etc/sysconfig/i18n ] && . /etc/sysconfig/i18n
if [ -f /etc/sysconfig/console/$SYSFONT.psf.gz -o \
    -f /usr/lib/kbd/consolefonts/$SYSFONT.psf.gz -o \
    -f /etc/sysconfig/console/$SYSFONT.gz -o \
    -f /usr/lib/kbd/consolefonts/$SYSFONT.gz -o \
    -f /lib/kbd/consolefonts/$SYSFONT.gz -o \
    -f /lib/kbd/consolefonts/$SYSFONT.psf.gz ]; then
    action "$Setting default font ($SYSFONT): " /sbin/setsysfont
fi
fi
fi

# Start up swapping.
action "$Activating swap partitions: " swapon -a -e

# Set the hostname.
action "$Setting hostname ${HOSTNAME}: " hostname ${HOSTNAME}

# Initialize USB controller and HID devices
usb=0
if ! grep -iq "nouse" /proc/cmdline 2>/dev/null && ! grep -q "usb"
/proc/devices 2>/dev/null ; then
    aliases=~ /sbin/modprobe -c | awk '/^alias usb-controller/ { print $3 }'
    if [ -n "$aliases" -a "$aliases" != "off" ] ; then
        modprobe usbcore
        action "$Mounting USB filesystem: " mount -t usbdevfs usbdevfs
/proc/bus/usb
        for alias in $aliases ; do
[ "$alias" != "off" ] && action "$Initializing USB controller ($alias): "
modprobe $alias
        done
[ $? -eq 0 -a -n "$aliases" ] && usb=1
fi
fi

```



```

if ! grep -iq "nousb" /proc/cmdline 2>/dev/null && grep -q "usb" /proc/devices
2>/dev/null ; then
    usb=1
fi

needusbstorage=
if [ $usb = "1" ]; then
    sleep 5
    mouseoutput=`cat /proc/bus/usb/devices 2>/dev/null|grep -E
"^I.*Cls=03.*Prot=02"~`
    kbdoutput=`cat /proc/bus/usb/devices 2>/dev/null|grep -E
"^I.*Cls=03.*Prot=01"~`
    needusbstorage=`cat /proc/bus/usb/devices 2>/dev/null|grep -e "^I.*Cls=08"~`
    if [ -n "$kbdoutput" ] || [ -n "$mouseoutput" ]; then
        action $"Initializing USB HID interface: " modprobe hid 2> /dev/null
        fi
    if [ -n "$kbdoutput" ]; then
        action $"Initializing USB keyboard: " modprobe keybdev
        fi
    if [ -n "$mouseoutput" ]; then
        action $"Initializing USB mouse: " modprobe mousedev
        fi
fi

if [ -f /fastboot ] || grep -iq "fastboot" /proc/cmdline 2>/dev/null ; then
    fastboot=yes
else
    fastboot=
fi

if [ -f /fsckoptions ]; then
    fsckoptions=`cat /fsckoptions`
else
    fsckoptions=
fi

if [ -f /forcefsck ]; then
    fsckoptions="-f $fsckoptions"
fi

if [ "$BOOTUP" != "serial" ]; then
    fsckoptions="-C $fsckoptions"
else
    fsckoptions="-V $fsckoptions"
fi

_RUN_QUOTACHECK=0

```

```

ROOTFSTYPE=`grep " / " /proc/mounts | awk '{ print $3 }'`
if [ -z "$fastboot" -a "$ROOTFSTYPE" != "nfs" ]; then

    STRING="Checking root filesystem"
    echo $STRING
    initlog -c "fsck -T -a $fsckoptions /"
    rc=$?

    if [ "$rc" = "0" ]; then
        success "$STRING"
        echo
    elif [ "$rc" = "1" ]; then
        passed "$STRING"
        echo
    fi

    # A return of 2 or higher means there were serious problems.
    if [ $rc -gt 1 ]; then
        failure "$STRING"
        echo
        echo "$*** An error occurred during the file system check."
        echo "$*** Dropping you to a shell; the system will reboot"
        echo "$*** when you leave the shell."

        PS1=$(Repair filesystem) \# # "; export PS1
        sulogin

        echo "Unmounting file systems"
        umount -a
        mount -n -o remount,ro /
        echo "Automatic reboot in progress."
        reboot -f
    elif [ "$rc" = "1" ]; then
        _RUN_QUOTACHECK=1
    fi
fi

grep -E '[:space:]]+/[:space:]]+' /etc/fstab | \
    awk '{ print $4 }' | \
    grep -q quota
_ROOT_HAS_QUOTA=$?
if [ X"$_RUN_QUOTACHECK" = X1 -a \
    "$_ROOT_HAS_QUOTA" -a \
    -x /sbin/quotacheck ]; then
    if [ -x /sbin/convertquota ]; then
        if [ -f /quota.user ]; then
            action "Converting old user quota files: " \
                /sbin/convertquota -u / && rm -f /quota.user

```

```

        fi
        if [ -f /quota.group ]; then
            action $"Converting old group quota files: " \
                /sbin/convertquota -g / && rm -f /quota.group
        fi
    fi
    action $"Checking root filesystem quotas: " /sbin/quotacheck -nug /
fi

# check for arguments passed from kernel

if grep -iq nopnp /proc/cmdline >/dev/null 2>&1 ; then
    PNP=
else
    PNP=yes
fi

# set up pnp
if [ -x /sbin/isapnp -a -f /etc/isapnp.conf -a ! -f /proc/isapnp ]; then
    if [ -n "$PNP" ]; then
        action $"Setting up ISA PNP devices: " /sbin/isapnp /etc/isapnp.conf
    else
        action $"Skipping ISA PNP configuration at users request: " /bin/true
    fi
fi

# Remount the root filesystem read-write.
# We comment this out for guestvol where the root filesystem is readonly
#state=`awk '/(^\/dev\/root| \/ )/ { print $4 }' /proc/mounts`
#[ "$state" != "rw" ] && \
# action $"Remounting root filesystem in read-write mode: " mount -n -o
remount,rw /

# LVM initialization
if [ -e /proc/lvm -a -x /sbin/vgchange -a -f /etc/lvmtab ]; then
    action $"Setting up LVM:" /sbin/vgscan && /sbin/vgchange -a y
fi

# Clear mtab
# Commented out for guestvol environment--mtab already set up by rc.guestvol
# >/etc/mtab

# Remove stale backups
rm -f /etc/mtab~ /etc/mtab~~

# Enter root, /proc and (potentially) /proc/bus/usb and devfs into mtab.
mount -f /
mount -f /proc
[ -f /proc/bus/usb/devices ] && mount -f -t usbdevfs usbdevfs /proc/bus/usb

```

```

[ -e /dev/.devfsd ] && mount -f -t devfs devfs /dev

# Turn on harddisk optimization
# There is only one file /etc/sysconfig/harddisks for all disks
# after installing the hdparm-RPM. If you need different hdparm parameters
# for each of your disks, copy /etc/sysconfig/harddisks to
# /etc/sysconfig/harddiskhda (hdb, hdc...) and modify it.
# each disk which has no special parameters will use the defaults.

disk[0]=s; disk[1]=hda; disk[2]=hdb; disk[3]=hdc;
disk[4]=hdd; disk[5]=hde; disk[6]=hdf; disk[7]=hdg; disk[8]=hdh;

if [ -x /sbin/hdparm ]; then
  for device in 0 1 2 3 4 5 6 7 8; do
    unset MULTIPLE_IO USE_DMA EIDE_32BIT LOOKAHEAD EXTRA_PARAMS
    if [ -f /etc/sysconfig/harddisk${disk[$device]} ]; then
      . /etc/sysconfig/harddisk${disk[$device]}
      HDFLAGS[$device]=
      if [ -n "$MULTIPLE_IO" ]; then
        HDFLAGS[$device]="-q -m$MULTIPLE_IO"
      fi
      if [ -n "$USE_DMA" ]; then
        HDFLAGS[$device]="${HDFLAGS[$device]} -q -d$USE_DMA"
      fi
      if [ -n "$EIDE_32BIT" ]; then
        HDFLAGS[$device]="${HDFLAGS[$device]} -q -c$EIDE_32BIT"
      fi
      if [ -n "$LOOKAHEAD" ]; then
        HDFLAGS[$device]="${HDFLAGS[$device]} -q -A$LOOKAHEAD"
      fi
      if [ -n "$EXTRA_PARAMS" ]; then
        HDFLAGS[$device]="${HDFLAGS[$device]} $EXTRA_PARAMS"
      fi
    else
      HDFLAGS[$device]="${HDFLAGS[0]}"
    fi
    if [ -e "/proc/ide/${disk[$device]}/media" ]; then
      hdmedia=`cat /proc/ide/${disk[$device]}/media`
      if [ "$hdmedia" = "disk" ]; then
        if [ -n "${HDFLAGS[$device]}" ]; then
          action $"Setting hard drive parameters for
${disk[$device]}: " /sbin/hdparm ${HDFLAGS[$device]} /dev/${disk[$device]}
        fi
      fi
    fi
  done
fi

```

```

# The root filesystem is now read-write, so we can now log via syslog()
directly..
if [ -n "$IN_INITLOG" ]; then
    IN_INITLOG=
fi

if ! grep -iq nomodules /proc/cmdline >/dev/null 2>&1 && [ -f /proc/ksyms ];
then
    USEMODULES=y
else
    USEMODULES=
fi

# Our modutils don't support it anymore, so we might as well remove
# the preferred link.
rm -f /lib/modules/preferred
rm -f /lib/modules/default
if [ -x /sbin/depmod -a -n "$USEMODULES" ]; then
    # If they aren't using a recent sane kernel, make a link for them
    if [ ! -n "`uname -r | grep -- "-"``" ]; then
        ktag="`cat /proc/version`"
        mtag=`grep -l "$ktag" /lib/modules/*.rhkmvtag 2> /dev/null`
        if [ -n "$mtag" ]; then
            mver=`echo $mtag | sed -e 's,/lib/modules/,,' -e 's,/.rhkmvtag,,,' -e
's,[      ].*$,,'`
            fi
            if [ -n "$mver" ]; then
                ln -sf /lib/modules/$mver /lib/modules/default
            fi
            fi
            if [ -L /lib/modules/default ]; then
                INITLOG_ARGS= action "$Finding module dependencies: " depmod -A default
            else
                INITLOG_ARGS= action "$Finding module dependencies: " depmod -A
            fi
        fi

# tweak isapnp settings if needed.
if [ -n "$PNP" -a -f /proc/isapnp -a -x /sbin/sndconfig ]; then
    /sbin/sndconfig --mungepnp >/dev/null 2>&1
fi

# Load sound modules iff they need persistent DMA buffers
if grep -q "options sound dmabuf=1" /etc/modules.conf 2>/dev/null ; then
    RETURN=0
    alias=~ /sbin/modprobe -c | awk '/^alias sound / { print $3 }'`
    if [ -n "$alias" -a "$alias" != "off" ]; then
        action "$Loading sound module ($alias): " modprobe $alias
        RETURN=$?
    fi

```

```

fi
alias=~sbin/modprobe -c | awk '/^alias sound-slot-0 / { print $3 }'~
if [ -n "$alias" -a "$alias" != "off" ] ; then
    action $"Loading sound module ($alias): " modprobe $alias
    RETURN=$?
fi
# Load mixer settings
if grep -q "\(\sparcaudio\|sound\) " /proc/devices 2>/dev/null ; then
    if [ $RETURN -eq 0 -a -f /etc/.aumixrc -a -x /bin/aumix-minimal ] ; then
        action $"Loading mixer settings: " /bin/aumix-minimal -f /etc/.aumixrc
-L
    fi
fi
fi

if [ -f /proc/sys/kernel/modprobe ] ; then
    if [ -n "$USEMODULES" ] ; then
        sysctl -w kernel.modprobe="/sbin/modprobe" >/dev/null 2>&1
        sysctl -w kernel.hotplug="/sbin/hotplug" >/dev/null 2>&1
    else
        # We used to set this to NULL, but that causes 'failed to exec'
messages"
        sysctl -w kernel.modprobe="/bin/true" >/dev/null 2>&1
        sysctl -w kernel.hotplug="/bin/true" >/dev/null 2>&1
    fi
fi

# Load modules (for backward compatibility with VARs)
if [ -f /etc/rc.modules ] ; then
    /etc/rc.modules
fi

# Add raid devices
if [ ! -f /proc/mdstat ] ; then
    modprobe md >/dev/null 2>&1
fi

if [ -f /proc/mdstat -a -f /etc/raidtab ] ; then
    echo -n $"Starting up RAID devices: "

    rc=0

    for i in `grep "^[^*]*raiddev" /etc/raidtab | awk '{print $2}'`~
    do
        RAIDDEV=`basename $i`~
        RAIDSTAT=`grep "^$RAIDDEV : active" /proc/mdstat`~
        if [ -z "$RAIDSTAT" ] ; then
            # Try raidstart first...if that fails then
            # First scan the /etc/fstab for the "noauto"-flag

```

```

# for this device. If found, skip the initialization
# for it to avoid dropping to a shell on errors.
# If not, try raidstart...if that fails then
# fall back to raidadd, raidrun. If that
# also fails, then we drop to a shell
RESULT=1
NOAUTO=`grep "^$i" /etc/fstab | grep -c "noauto"`
if [ $NOAUTO -gt 0 ]; then
    RESULT=0
    RAIDDEV="$RAIDDEV(skipped)"
fi
if [ $RESULT -gt 0 -a -x /sbin/raidstart ]; then
    /sbin/raidstart $i
    RESULT=$?
fi
if [ $RESULT -gt 0 -a -x /sbin/raid0run ]; then
    /sbin/raid0run $i
    RESULT=$?
fi
if [ $RESULT -gt 0 -a -x /sbin/raidadd -a -x /sbin/raidrun ]; then
    /sbin/raidadd $i
    /sbin/raidrun $i
    RESULT=$?
fi
if [ $RESULT -gt 0 ]; then
    rc=1
fi
echo -n "$RAIDDEV "
else
echo -n "$RAIDDEV "
fi
done
echo

# A non-zero return means there were problems.
if [ $rc -gt 0 ]; then
echo
echo
echo $"*** An error occurred during the RAID startup"
echo $"*** Dropping you to a shell; the system will reboot"
echo $"*** when you leave the shell."

PS1=$(RAID Repair) \# # "; export PS1
slogin

echo $"Unmounting file systems"
umount -a
mount -n -o remount,ro /
echo $"Automatic reboot in progress."

```

```

        reboot -f
    fi
fi

_RUN_QUOTACHECK=0
# Check filesystems
if [ -z "$fastboot" ]; then
    STRING="Checking filesystems"
    echo $STRING
    initlog -c "fsck -T -R -A -a $fscsoptions"
    rc=$?
    if [ "$rc" = "0" ]; then
        success "$STRING"
        echo
    elif [ "$rc" = "1" ]; then
        passed "$STRING"
        echo
    fi

    # A return of 2 or higher means there were serious problems.
    if [ $rc -gt 1 ]; then
        failure "$STRING"
        echo
        echo
        echo "$*** An error occurred during the file system check."
        echo "$*** Dropping you to a shell; the system will reboot"
        echo "$*** when you leave the shell."

        PS1=$(Repair filesystem) \# # "; export PS1
        sulogin

        echo $"Unmounting file systems"
        umount -a
        mount -n -o remount,ro /
        echo $"Automatic reboot in progress."
        reboot -f
    elif [ "$rc" = "1" -a -x /sbin/quotacheck ]; then
        _RUN_QUOTACHECK=1
    fi
fi

# Mount all other filesystems (except for NFS and /proc, which is already
# mounted). Contrary to standard usage,
# filesystems are NOT unmounted in single user mode.
action $"Mounting local filesystems: " mount -a -t nonfs,smbfs,ncpfs

# check remaining quotas other than root
if [ X"$_RUN_QUOTACHECK" = X1 -a -x /sbin/quotacheck ]; then
    if [ -x /sbin/convertquota ]; then

```



```

    # try to convert old quotas
    for mountpt in `cat /etc/mstab | awk '$4 ~ /quota/{print $2}'`; do
    if [ -f "$mountpt/quota.user" ]; then
        action $"Converting old user quota files: " \
            /sbin/convertquota -u $mountpt && \
            rm -f $mountpt/quota.user
    fi
    if [ -f "$mountpt/quota.group" ]; then
        action $"Converting old group quota files: " \
            /sbin/convertquota -g $mountpt && \
            rm -f $mountpt/quota.group
    fi
    done
fi
action $"Checking local filesystem quotas: " /sbin/quotacheck -aRnug
fi

if [ -x /sbin/quotaon ]; then
    action $"Enabling local filesystem quotas: " /sbin/quotaon -aug
fi

# Turn on process accounting
if [ -x /sbin/accton ] ; then
    action $"Turning on process accounting" /sbin/accton /var/log/pacct
fi

# Configure machine if necessary.
if [ -f /.unconfigured ]; then
    if [ -x /usr/bin/passwd ]; then
        /usr/bin/passwd root
    fi

    # on S390 console we don't have newt
    if [ "`/bin/arch`" = "s390" -o "`/bin/arch`" = "s390x" ] ; then
        ARCH=".s390"
    else
        ARCH=""
    fi

    if [ -x /usr/sbin/netconfig$ARCH ]; then
        /usr/sbin/netconfig$ARCH
    fi
    if [ -x /usr/sbin/timeconfig$ARCH ]; then
        /usr/sbin/timeconfig$ARCH
    fi
    if [ -x /usr/sbin/authconfig$ARCH ]; then
        /usr/sbin/authconfig$ARCH --nostart
    fi
    if [ -x /usr/sbin/ntsysv$ARCH ]; then

```

```

    /usr/sbin/ntsysv$ARCH --level 35
fi

# Reread in network configuration data.
if [ -f /etc/sysconfig/network ]; then
. /etc/sysconfig/network

# Reset the hostname.
action $"Resetting hostname ${HOSTNAME}: " hostname ${HOSTNAME}
fi

rm -f /.unconfigured
fi

# Clean out /etc.
rm -f /fastboot /fsckoptions /forcefsck /halt /poweroff

# Do we need (w|u)tmpx files? We don't set them up, but the sysadmin might...
_NEED_XFILES=
[ -f /var/run/utmpx -o -f /var/log/wtmpx ] && _NEED_XFILES=1

# Clean up /var
# I'd use find, but /usr may not be mounted.
for afile in /var/lock/* /var/run/*; do
    if [ -d "$afile" ]; then
        [ "`basename $afile`" != "news" -a "`basename $afile`" != "sudo" -a
        "`basename $afile`" != "mon" ] && rm -f $afile/*
    else
        rm -f $afile
    fi
done
rm -f /var/lib/rpm/__db*

# Reset pam_console permissions
[ -x /sbin/pam_console_apply ] && /sbin/pam_console_apply -r

{
# Clean up utmp/wtmp
>/var/run/utmp
touch /var/log/wtmp
chgrp utmp /var/run/utmp /var/log/wtmp
chmod 0664 /var/run/utmp /var/log/wtmp
if [ -n "$_NEED_XFILES" ]; then
>/var/run/utmpx
touch /var/log/wtmpx
chgrp utmp /var/run/utmpx /var/log/wtmpx
chmod 0664 /var/run/utmpx /var/log/wtmpx
fi
}

```

```

# Delete X locks
rm -f /tmp/.X*-lock

# Delete Postgres sockets
rm -f /tmp/.s.PGSQL.*

# Now turn on swap in case we swap to files.
swapon -a
action $"Enabling swap space: " /bin/true

# Initialize the serial ports.
if [ -f /etc/rc.serial ]; then
    . /etc/rc.serial
fi

# If a SCSI tape has been detected, load the st module unconditionally
# since many SCSI tapes don't deal well with st being loaded and unloaded
if [ -f /proc/scsi/scsi ] && grep -q 'Type: Sequential-Access'
/proc/scsi/scsi 2>/dev/null ; then
    if grep -qv ' 9 st' /proc/devices ; then
        if [ -n "$USEMODULES" ] ; then
            # Try to load the module. If it fails, ignore it...
            insmod -p st >/dev/null 2>&1 && modprobe st >/dev/null 2>&1
        fi
    fi
fi

# Load usb storage here, to match most other things
if [ -n "$needusbstorage" ]; then
    modprobe usb-storage >/dev/null 2>&1
fi

# If they asked for ide-scsi, load it
if grep -q "ide-scsi" /proc/cmdline ; then
    modprobe ide-cd >/dev/null 2>&1
    modprobe ide-scsi >/dev/null 2>&1
fi

# Generate a header that defines the boot kernel.
/sbin/mkkerneldoth

# Adjust symlinks as necessary in /boot to keep system services from
# spewing messages about mismatched System maps and so on.
if [ -L /boot/System.map -a -r /boot/System.map-`uname -r` -a \
! /boot/System.map -ef /boot/System.map-`uname -r` ] ; then
    ln -s -f System.map-`uname -r` /boot/System.map
fi
if [ ! -e /boot/System.map -a -r /boot/System.map-`uname -r` ] ; then
    ln -s -f System.map-`uname -r` /boot/System.map

```

```

fi

# Now that we have all of our basic modules loaded and the kernel going,
# let's dump the syslog ring somewhere so we can find it later
dmesg -s 131072 > /var/log/dmesg
# Also keep kernel symbols around in case we need them for debugging
i=5
while [ $i -ge 0 ] ; do
    if [ -f /var/log/ksyms.$i ] ; then
        mv /var/log/ksyms.$i /var/log/ksyms.$(($i+1))
    fi
    i=$((i-1))
done
(/bin/date;
/bin/uname -a;
/bin/cat /proc/cpuinfo;
/bin/cat /proc/modules;
/bin/cat /proc/ksyms) >/var/log/ksyms.0
sleep 1
kill -TERM `sbin/pidof getkey` >/dev/null 2>&1
} &
if [ "$PROMPT" != "no" ]; then
    /sbin/getkey i && touch /var/run/confirm
fi
wait

```

C.1.3 The /etc/init.d/vmgetconf script

In Example C-3, we show the /etc/init.d/vmgetconf script discussed in 8.11.4, “The vmgetconf script” on page 166.

Example: C-3 The /etc/init.d/vmgetconf script

```

#!/bin/sh
#
# vmgetconf - obtain config info from the VM configuration server guest
#
# Copyright (C) 2002 IBM UK Ltd
# Author: Malcolm Beattie <beattiem@uk.ibm.com>
#
# This program is free software; you can redistribute it and/or
# modify it under the terms of the GNU General Public License
# as published by the Free Software Foundation; either version
# 2 of the License, or (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

```

```

# GNU General Public License for more details.
#
# 24 Jul 2002 Initial version
#
# Overview:
# Ensure cpint kernel module is loaded
# Ensure we have a virtual printer at devno 001E (we only use its tag field)
# Read CONFSEVR guest name and our role from /etc/sysconfig/vmconfigserver
# Send configuration server a request via
# CP SMSG CONFSEVR GETMYCONF role
# Config server finds our config_info and does
# CP SEND CP US TAG 001E config_info
# In the meantime, we poll (once a second or so) for the TAG to appear
# (a fancier version of this uses the ext_int kernel module to avoid polling)
# We read the config_info via "CP TAG QUERY 001E".
# We parse it as "ENV1=val1;ENV2=val2;..." and write /etc/sysconfig/vmconfig
# Finally we load ext_int (if possible) to allow remote shutdown trigger
#

# The following file must include a line
# CONFSEVR=name_of_config_server_guest
# and optionally may include a line
# ROLE=our_role_we_want_to_ask_server_for
# The default for ROLE is "default" and the value is case-insensitive

ROLE="default"
. /etc/sysconfig/vmconfigserver

if [ "$CONFSEVR" = "" ]; then
    echo "Error: CONFSEVR not defined in /etc/sysconfig/vmconfigserver" 1>&2
    echo "Continuing without updating $vmconfig_file"
    exit 1
fi

tries=10
interval=1
vmconfig_file=/etc/sysconfig/vmconfig
prtdev=001E

modprobe cpint
hcp detach $prtdev > /dev/null
if ! hcp define prt $prtdev > /dev/null; then
    echo "Error: failed CP DEFINE PRT $prtdev" 1>&2
    echo "Continuing without updating $vmconfig_file"
    exit 1
fi

echo -n "Fetching configuration for role \"$ROLE\" from $CONFSEVR: "

```

```

hcp msg $CONFSERV GETMYCONF $ROLE > /dev/null
tag=""
while [ $tries -gt 0 ]; do
    tag=`hcp tag query $prtdev | awk '
        ($3 == "TAG:") {getline; print; exit}
        ($3 == "TAG" && $4 == "NOT" && $5 == "SET") {exit 1}
        {exit 99}`
    if [ $? -ne 1 ]; then
        break
    fi
    echo -n "."
    sleep $interval
    tries=$((tries - 1))
done
hcp detach $prtdev > /dev/null
if [ "$tag" = "" ]; then
    echo "timeout"
    echo "Error: timeout contacting VM configuration server $CONFSERV" 1>&2
    echo "Continuing without updating config file $vmconfig_file"
    exit 1
else
    echo "done"
fi

echo "$tag" | awk 'BEGIN {RS=";"} {print}' > $vmconfig_file

# Try to load ext_int to intercept external interrupt OD1E and map it
# to a SIGINT for init. That initiates the ctrlaltdel line in /etc/inittab
# which can be configured for a "shutdown -h now".
if insmod -o ext0d1e ext_int code=0x0d1e pid=1 sig=2 > /dev/null 2>&1; then
    echo "ext_int loaded: CP EXT OD1E will trigger init ctrlaltdel"
fi

# Let's ensure CP SET RUN ON while we're here
hcp set run on > /dev/null 2>&1

exit 0

```

C.1.4 The /etc/init.d/itsonet script

In Example C-4, we show the /etc/init.d/itsonet script discussed in 8.11.5, “The itsonet script” on page 168.

Example: C-4 The /etc/initt.d/itsonet script

```

#!/bin/sh
#
# itsonet - init script

```

```

# Use the configuration information in /etc/sysconfig/vmconfig
# (provided dynamically from the VM configuration server CONFSERV
# earlier in the boot process) to create appropriate network config
# files. For RedHat, we need to create /etc/sysconfig/network and
# /etc/sysconfig/network-scripts/ifcfg-eth{0,1}.
# The information in /etc/sysconfig/vmconfig is of the form
#   NETDEV=ifname
#   IPADDR=10.x.y.z
#   NETMASK=255.255.255.0
# along with optional fields
#   GATEWAY=10.d.e.f
# and also there will in practice be a field
#   LDAP=10.p.q.r
# from which a later boot script will find the LDAP configuration server.

. /etc/init.d/functions

# Pick up current settings from /etc/sysconfig/network (in case someone
# has already set the hostname for example and will not be using the
# LDAP host configuration).

. /etc/sysconfig/network

echo "Generating network configuration files from vmconfig information"
# Now get the vmconfig information that has been dynamically fetched for us
. /etc/sysconfig/vmconfig

if [ "$HOSTNAME" = "" ]; then
    HOSTNAME="cloned-but-unset"
fi

cat > /etc/sysconfig/network <<EOT
NETWORKING=yes
HOSTNAME=$HOSTNAME
EOT

if [ "$GATEWAY" != "" ]; then
    echo "GATEWAY=$GATEWAY" >> /etc/sysconfig/network
fi

if [ "$NETDEV" = "" ]; then
    echo "itsonet: NETDEV not set: not configuring network device information"
elif [ "$IPADDR" = "" ]; then
    echo "itsonet: IPADDR not set: not configuring network device information"
elif [ "$NETMASK" = "" ]; then
    echo "itsonet: NETMASK not set: not configuring network device information"
else
    echo "itsonet: net device $NETDEV, IP address $IPADDR, netmask $NETMASK"
cat > /etc/sysconfig/network-scripts/ifcfg-$NETDEV <<EOT

```

```
DEVICE=$NETDEV
BOOTPROTO=static
IPADDR=$IPADDR
NETMASK=$NETMASK
ONBOOT=yes
EOT
fi
```

C.1.5 The `/etc/init.d/guestvol-start-halt` script

In Example C-5, we show the `/etc/init.d/guestvol-start-halt` script discussed 8.12.1, “The `guestvol-start-halt` script” on page 170.

Example: C-5 The `/etc/init.d/guestvol-start-halt` script

```
#!/bin/bash
#
# guestvol-start-halt
# First part of halt procedure for guestvol environment.
# Modifications for guestvol environment by Malcolm Beattie but the
# majority of the script is the first part of the rc.halt script by...
#
# Author:      Miquel van Smoorenburg, <miquels@drinkel.nl.mugnet.org>
#              Modified for RHS Linux by Damien Neil
#

# Set the path.
PATH=/sbin:/bin:/usr/bin:/usr/sbin

export NOLOCALE=1
. /etc/init.d/functions

runcmd() {
    echo -n "$* "
    shift
    if [ "$BOOTUP" = "color" ]; then
        $* && echo_success || echo_failure
    else
        $*
    fi
    echo
}

halt_get_remaining() {
    awk '!/^#|proc|loopfs|autofs|^none|^\|dev\/root| \| \/) / {print $2}'
    /proc/mounts
    awk '{ if ($3 ~ /^proc$/ && $2 !~ /^\/proc/) print $2; }' /proc/mounts
}
```



```

# See how we were called.
case "$0" in
    *halt)
        message="$Halting system..."
        command="halt"
        ;;
    *reboot)
        message="$Please stand by while rebooting the system..."
        command="reboot"
        ;;
    *)
        echo "$0: call me as 'rc.halt' or 'rc.reboot' please!"
        exit 1
        ;;
esac
if [ -n "$1" ]; then
    case "$1" in
        *start)
            ;;
        *)
            echo "$Usage: (halt|reboot) {start}"
            exit 1
            ;;
    esac
fi

# Kill all processes.
[ "${BASH+bash}" = bash ] && enable kill

runCMD "$Sending all processes the TERM signal..." /sbin/killall5 -15
sleep 5
runCMD "$Sending all processes the KILL signal..." /sbin/killall5 -9

# Write to wtmp file before unmounting /var
halt -w

# Save mixer settings, here for lack of a better place.
grep -q "\(\sparcudio\|sound\)" /proc/devices
if [ $? = 0 -a -x /bin/aumix-minimal ]; then
    runCMD "$Saving mixer settings" /bin/aumix-minimal -f /etc/.aumixrc -S
fi

# Sync the system clock.
ARC=0
SRM=0
UTC=0

if [ -f /etc/sysconfig/clock ]; then

```

```

. /etc/sysconfig/clock

# convert old style clock config to new values
if [ "${CLOCKMODE}" = "GMT" ]; then
    UTC=true
elif [ "${CLOCKMODE}" = "ARC" ]; then
    ARC=true
fi
fi

CLOCKDEF=""
CLOCKFLAGS="$CLOCKFLAGS --systohc"

case "$UTC" in
    yes|true)
        CLOCKFLAGS="$CLOCKFLAGS -u";
        CLOCKDEF="$CLOCKDEF (utc)";
        ;;
    no|false)
        CLOCKFLAGS="$CLOCKFLAGS --localtime";
        CLOCKDEF="$CLOCKDEF (localtime)";
        ;;
esac

case "$ARC" in
    yes|true)
        CLOCKFLAGS="$CLOCKFLAGS -A";
        CLOCKDEF="$CLOCKDEF (arc)";
        ;;
esac

case "$SRM" in
    yes|true)
        CLOCKFLAGS="$CLOCKFLAGS -S";
        CLOCKDEF="$CLOCKDEF (srm)";
        ;;
esac

#runcmd "$Syncing hardware clock to system time" /sbin/hwclock $CLOCKFLAGS

# Turn off swap
SWAPS=`awk '! /^Filename/ { print $1 }' /proc/swaps`
[ -n "$SWAPS" ] && runcmd "$Turning off swap: " swapoff $SWAPS

[ -x /sbin/accton ] && runcmd "$Turning off accounting: " /sbin/accton

[ -x /sbin/quotaoff ] && runcmd "$Turning off quotas: " /sbin/quotaoff -aug

# For a guestvol configuration, we need to get /sbin/guestvol-final-halt
# to do the rest. This is because the next step is to unmount filesystems

```

```

# and we are currently running from a script under /etc which is mounted
# from a directory on the guestvol disk.
echo "Switching to final halt script for guestvol environment" # debug
exec /sbin/guestvol-final-halt $0

# Shouldn't get here
echo "Error: failed to execute /sbin/guestvol-final-halt"
echo "Attempting emergency halt"

umount -a -f

mount | awk '/( \| |^\/dev\/root)/ { print $3 }' | while read line; do
    mount -n -o ro,remount $line
done

# Now halt or reboot.
echo "$message"
if [ -f /fastboot ]; then
    echo "On the next boot fsck will be skipped."
elif [ -f /forcefsck ]; then
    echo "On the next boot fsck will be forced."
fi

HALTARGS="-i -d"
if [ -f /poweroff -o ! -f /halt ]; then
    HALTARGS="$HALTARGS -p"
fi

if [ "$command" = halt ] ; then
    if [ -r /etc/ups/upsmon.conf -a -f /etc/killpower -a -f /etc/sysconfig/ups
] ; then
        . /etc/sysconfig/ups
        [ "$SERVER" = "yes" -a "$MODEL" != "NONE" -a -n "$MODEL" -a -n
"$DEVICE" ] && $MODEL -k $DEVICE
        fi
    fi

eval $command $HALTARGS

```

C.1.6 The /sbin/guestvol-final-halt script

In Example C-6, we show the /sbin/guestvol-final-halt script discussed in 8.12.2, “The guestvol-final-halt script” on page 170.

Example: C-6 The /sbin/guest-final-halt script

```

#!/bin/bash
#

```

```

# guestvol-final-halt
# Perform the final part of the halt sequence for a guestvol environment.
# The first part of the halt sequence is /etc/init.d/halt but in a
# guestvol environment, /etc is mounted from a directory on the guestvol
# disk. Now we need to unmount disks, we switch to this script which
# lives in /sbin on the root filesystem.
#
# Modifications for guestvol by Malcolm Beattie but the majority of the
# code is taken from the rc.halt script by...
#
# Author:      Miquel van Smoorenburg, <miquels@drinkel.nl.mugnet.org>
#              Modified for RHS Linux by Damien Neil
#

# Set the path.
PATH=/sbin:/bin:/usr/bin:/usr/sbin

export NOLOCALE=1
. /etc/init.d/functions

runcmd() {
    echo -n "$* "
    shift
    if [ "$BOOTUP" = "color" ]; then
        $* && echo_success || echo_failure
    else
        $*
    fi
    echo
}

halt_get_remaining() {
    awk '!/(^#|proc|loopfs|autofs|^none|^\/dev\/root| \/ )/ {print $2}'
    /proc/mounts
    awk '{ if ($3 ~ /^proc$/ && $2 !~ /^\/proc/) print $2; }' /proc/mounts
}

# See if we want to halt or reboot
# (Note we take this from our first argument, not from $0)
case "$1" in
    *halt)
        message="$Halting system..."
        command="halt"
        ;;
    *reboot)
        message="$Please stand by while rebooting the system..."
        command="reboot"
        ;;
)

```

```

*)
    echo "$0: call me as 'rc.halt' or 'rc.reboot' please!"
    exit 1
    ;;
esac

echo "Starting final part of halt procedure for guestvol environment" # debug

# For a guestvol, unmount "bind" mounts before the others. Use /etc/mtab
# to find them since /proc/mounts only tracks the underlying block device.
# Use -n to avoid updating mtab since /etc becomes the readonly
# basevol instance part way through this procedure.
# Reverse the list so we unmount in the correct order.
mtablist=`awk '!/^(\#|\dev|\guestvol\dev)/ && $4 ~ /,bind/) {print $2}'
/etc/mtab`

revmtablist=""
for dir in $mtablist; do
    revmtablist="$dir $revmtablist"
done

for dir in $revmtablist; do
    umount -n $dir
done

#
# Find /guestvol device for later
#
guestvol_dev=`awk '($2 == "/guestvol") {print $1}' /proc/mounts`
if [ -z "$guestvol_dev" ]; then
    guestvol_dev="none" # if all else fails, try this
fi

# Unmount file systems, killing processes if we have to.
# Unmount loopback stuff first
remaining=`awk '!/^#/ && $1 ~ /^\/dev\/loop/ && $2 != "/" {print $2}'
/proc/mounts`
devremaining=`awk '!/^#/ && $1 ~ /^\/dev\/loop/ && $2 != "/" {print $1}'
/proc/mounts`
[ -n "$remaining" ] && {
    sig=
    retry=3
    while [ -n "$remaining" -a "$retry" -gt 0 ]
    do
        if [ "$retry" -lt 3 ]; then
            runcmd "$Unmounting loopback filesystems (retry):" umount -n
$remaining
        else
            runcmd "$Unmounting loopback filesystems:" umount -n $remaining

```

```

        fi
        for dev in $devremaining ; do
            losetup $dev > /dev/null 2>&1 && \
                runcmd $"Detaching loopback device $dev: "
        losetup -d $device
    done
    remaining=`awk '!/^#/ && $1 ~ /^\/dev\/loop/ && $2 != "/" {print $2}'
/proc/mounts`
    devremaining=`awk '!/^#/ && $1 ~ /^\/dev\/loop/ && $2 != "/"
{print $1}' /proc/mounts`
    [ -z "$remaining" ] && break
    /sbin/fuser -k -m $sig $remaining >/dev/null
    sleep 5
    retry=$((retry -1))
    sig=-9
done
}

umount -n -a -f

# Remount read only anything that's left mounted.
#echo "$Remounting remaining filesystems (if any) readonly"
mount | awk '/( \| ^\/dev\/root)/ { print $3 }' | while read line; do
    mount -n -o ro,remount $line
done

# Now the same for guestvol
echo "Remounting guestvol readonly on block device $guestvol_dev" # debug
mount -n -o remount,ro $guestvol_dev /guestvol

# Now halt or reboot.
echo "$message"
if [ -f /fastboot ]; then
    echo "On the next boot fsck will be skipped."
elif [ -f /forcefsck ]; then
    echo "On the next boot fsck will be forced."
fi

HALTARGS="-i -d"
if [ -f /poweroff -o ! -f /halt ]; then
    HALTARGS="$HALTARGS -p"
fi

if [ "$command" = halt ] ; then
    if [ -r /etc/ups/upsmon.conf -a -f /etc/killpower -a -f /etc/sysconfig/ups
] ; then
        . /etc/sysconfig/ups
        [ "$SERVER" = "yes" -a "$MODEL" != "NONE" -a -n "$MODEL" -a -n
"$DEVICE" ] && $MODEL -k $DEVICE

```

```
        fi
    fi

    echo "About to eval $command $HALTARGS" # debug

    eval $command $HALTARGS
```

C.1.7 The `/usr/sbin/basevol-devel` script

In Example C-7, we show the `/usr/sbin/basevol-devel` script discussed in 9.5.1, “Prepare the LDV01 Linux guest” on page 177.

Example: C-7 The `/usr/sbin/basevol-devel` script

```
#!/bin/sh

if [ "$1" = "enable" ]; then
    awk -F: '{if ($3 == "sysinit") {print "gv::sysinit:/etc/rc.d/rc.guestvol"} \
        else { print }}' < /etc/inittab > /etc/inittab.for-basevol
    rm -f /etc/inittab.pre-basevol
    ln /etc/inittab /etc/inittab.pre-basevol
    mv -f /etc/inittab.for-basevol /etc/inittab
elif [ "$1" = "disable" ]; then
    awk -F: '{if ($3 == "sysinit") {print "si::sysinit:/etc/rc.d/rc.sysinit"} \
        else { print }}' < /etc/inittab > /etc/inittab.not-basevol
    rm -f /etc/inittab.old-basevol
    ln /etc/inittab /etc/inittab.old-basevol
    mv -f /etc/inittab.not-basevol /etc/inittab
else
    echo "Usage: basevol-devel enable|disable" 1>&2
    exit 2
fi
```

C.1.8 The `/usr/sbin/mkguestvol` script

In Example C-8, we show the `/usr/sbin/mkguestvol` script discussed in 9.5.3, “Prepare guestvol filesystem image” on page 178.

Example: C-8 The `/usr/sbin/mkguestvol` script

```
#!/bin/sh
# mkguestvol - copy the "to-be-writable" part of a basevol to a new guestvol
#
runcmd=""

if [ "$1" = "-n" ]; then
    runcmd="echo"
```

```

        shift
    fi

    basevol="$1"
    guestvol="$2"

    if [ -z "$basevol" -o -z "$guestvol" ]; then
        echo "Usage: mkguestvol [-n] /mnt/newbasevol /mnt/newguestvol" 1>&2
        exit 2
    fi

    for dir in etc home opt root usr tmp boot; do
        $runcmd mkdir $guestvol/$dir
    done
    $runcmd chmod 1777 $guestvol/tmp
    $runcmd cp -a $basevol/dev $basevol/etc $basevol/mnt $basevol/var $guestvol
    # "local" subdirectory of source ends up in target dir so below is correct
    $runcmd cp -a $basevol/usr/local $guestvol/usr
    find $guestvol/var/log -type f -name '*. [0-9]' -exec $runcmd rm -f '{}' \;
    find $guestvol/var/log -type f -exec $runcmd cp /dev/null '{}' \;
    find $guestvol/var/run -type f ! \( -name utmp -o -name runlevel.dir \) \
        -exec $runcmd rm -f '{}' \;
    $runcmd awk -F: '{if ($3=="sysinit") \
        {print "si::sysinit:/etc/rc.d/rc.sysinit-guestvol"} \
        else if ($3=="ctrlaltdel") \
        {print "ca::ctrlaltdel:/sbin/shutdown -h now"} \
        else {print}}' \
        < $basevol/etc/inittab > $guestvol/etc/inittab
    $runcmd ln -sf ../init.d/vmgetconf $guestvol/etc/rc.d/rc3.d/S01vmgetconf
    $runcmd ln -sf ../init.d/itsonet $guestvol/etc/rc.d/rc3.d/S04itsonet
    $runcmd ln -sf ../init.d/guestvol-start-halt $guestvol/etc/rc.d/rc0.d/S01halt
    $runcmd ln -sf ../init.d/guestvol-start-halt $guestvol/etc/rc.d/rc6.d/S01reboot
    echo "Don't forget to edit the following files if necessary:"
    echo "    $guestvol/etc/chandev.conf"
    echo "    $guestvol/etc/modules.conf"
    echo "Don't forget to remove the following files if necessary:"
    echo "    unneeded $guestvol/etc/sysconfig/network-scripts/ifcfg-*"

```

C.2 The itsobasevol-1.0.0-1.s390x.rpm package

We list some of the scripts found in the itsobasevol-1.0.0-1.s390x.rpm package.

C.2.1 The /etc/init.d/itsoldap script

The /etc/init.d/itsoldap script shown in Example C-9 we discussed in 10.4.4, “The itsoldap script” on page 199.

Example: C-9 The /etc/init.d/itsoldap script

```
#!/bin/sh
#
# chkconfig: 2345 9 80
# description: VM configuration server
#
# itsoldap.init
# Use the configuration information in /etc/sysconfig/vmconfig
# (provided dynamically from the VM configuration server CONFSEV
# earlier in the boot process) to create a appropriate LDAP client
# config file /etc/openldap/ldap.conf and retrieve other
# informations from the LDAP directory

[ "$1" != "start" ] && exit 0

. /etc/init.d/functions
. /etc/sysconfig/vmconfig

# set VMGUESTNAME and VMNODENAME via "hcp q userid"
eval `hcp q userid | tr "[:upper:]" "[:lower:]" | \
    awk '{ printf "VMGUESTNAME=%s\nVMNODENAME=%s\n",$1,$3 }'`

# set hostname by convention
HOSTNAME=`echo "$VMGUESTNAME.$VMNODENAME.itso.ibm.com" | \
    tr "[:upper:]" "[:lower:]"`

LDAPHOST="10.0.3.1"
LDAPBASEDN="o=IBM-ITSO,c=US"
LDAPBASEDN_VM="ou=VM,$LDAPBASEDN"
LDAPPAMGROUPODN="cn=$HOSTNAME,ou=HostAccess,$LDAPBASEDN"

LDAPTMP="/tmp/itsoldap.$$"

IFNUMCHANDEV="0"
IFNUMETH="0"
IFNUMHSI="0"

#####
###

increment_ifnum ()
{
    case "$IFTYPE" in
```

```

        eth)    IFNUMETH=`expr "$IFNUMETH" + 1`
                ;;
        hsi)    IFNUMHSI=`expr "$IFNUMHSI" + 1`
                ;;
    esac

    IFNUMCHANDEV=`expr "$IFNUMCHANDEV" + 1`
}

get_ifnum ()
{
#     case "$IFTYPE" in
#         eth)    IFNUM="$IFNUMETH"
#                 ;;
#         hsi)    IFNUM="$IFNUMHSI"
#                 ;;
#     esac

    IFNUM="$IFNUMCHANDEV"
}

#####
###

lookup_publicnetwork ()
{
    VMGUESTPUBNET=$1

    ldapsearch -x -H ldap://$LDAPHOST -b "$LDAPBASEDN_VM" \
"(&(objectClass=VMGuestPublicNetwork)(VMGuestNetwork=$VMGUESTPUBNET))" \
| perl -00 -pe 's/\n[ \t]//gs; s/^\(w+): (.*)$/$1="$2"/mg' \
> $LDAPTMP.2

    . $LDAPTMP.2

    if ( cat $LDAPTMP.2 | grep -q "numEntries: 1" )
    then
        PUBNETFOUND="1"
    else
        PUBNETFOUND="0"

        echo "FAILED: Could not retrieve LDAP configuration for
PublicNet \"$VMGUESTPUBNET\"!"
        ERROR="1"
    fi

    rm $LDAPTMP.2
}

```

```

#####
###

config_interface ()
{
    IFTYPE="$VMGuestNetType"
    get_ifnum

    cat > /etc/sysconfig/network-scripts/ifcfg-$IFTYPE$IFNUM <<EOT
#
# autogenerated by /etc/init.d/itsoldap script
#

DEVICE=$IFTYPE$IFNUM
BOOTPROTO=static
IPADDR=$PUBNETIP
NETMASK=$VMGuestNetMask
ONBOOT=yes
EOT

    echo "qeth$IFNUMCHANDEV,$VMGuestNetCHANDEV" | \
        sed -e 's/\^/;/' >> /etc/chandev.conf

    increment_ifnum
}

#####
### INIT
### keep management configuration and backup all others

IFNUM="1"
ERROR="0"

# remove old backup files
rm -f \
    /etc/sysconfig/network-scripts/ifcfg-*.bak \
    /etc/sysconfig/network.bak \
    /etc/chandev.conf.bak

mv /etc/chandev.conf /etc/chandev.conf.bak
cp /etc/chandev.template.conf /etc/chandev.conf

mv /etc/sysconfig/network /etc/sysconfig/network.bak && \
    cat /etc/sysconfig/network.bak | grep -vE "^HOSTNAME=" > \
    /etc/sysconfig/network
echo "HOSTNAME=$HOSTNAME" >> /etc/sysconfig/network

for i in /etc/sysconfig/network-scripts/ifcfg-*

```

```

do
    if [ "$i" != "/etc/sysconfig/network-scripts/ifcfg-$NETDEV" -a \
        "$i" != "/etc/sysconfig/network-scripts/ifcfg-lo" ]
    then
        mv ${i} {.bak}
    fi
done

# bring up management interface now
ifup $NETDEV

IFTYPE=`echo "$NETDEV" | sed -e 's/[0-9]//'^
increment_ifnum

#####
### MAIN

echo "Connecting to LDAP server to lookup PublicNet's parameters..."

VMNODEBASEDN="VMNodeName=$VMNODENAME,ou=VM,o=IBM-ITSO,c=US"
ldapsearch -x -H ldap://$LDAPHOST \
    -b $VMNODEBASEDN "(&(objectClass=VMGuest)(VMGuestName=$VMGUESTNAME))" \
    | perl -00 -pe 's/\n[ \t]//gs; s/^(\\w+): (.*)$/$1="$2"/mg' > $LDAPTMP

if ( cat $LDAPTMP | grep -q "numEntries: 1" )
then
    . $LDAPTMP

    echo "I'm $VMGuestName @ $VMNodeName... searching $VMNODEBASEDN"

    cat $LDAPTMP | grep "^VMGuestNetwork=" | while read PUBNET
    do
        eval `echo $PUBNET`
        PUBNETNAME=`echo $VMGuestNetwork | cut -d ',' -f 1`
        PUBNETIP=`echo $VMGuestNetwork | cut -d ',' -f 2`

        echo "Connecting to PublicNet $PUBNETNAME with IP $PUBNETIP..."
        lookup_publicnetwork $PUBNETNAME

        if [ "$PUBNETFOUND" = "1" ]
        then
            config_interface
        fi
    done
else
    echo "FAILED: Could not retrieve LDAP configuration for guest
    \"$VMGUESTNAME\"!"
    ERROR="1"
fi

```

```

rm $LDAPTMP

#####
### LDAP.CONF

if [ "$ERROR" = "0" ]
then
    cat /etc/openldap/ldap.template.conf | \
        sed -e "s/@LDAPHOST@/$LDAPHOST/g" \
            -e "s/@LDAPBASEDN@/$LDAPBASEDN/g" \
            -e "s/@LDAPPAMGROUPDN@/$LDAPPAMGROUPDN/g" \
    > /etc/openldap/ldap.conf

fi

#####
### DONE

if [ "$ERROR" != "0" ]
then
    echo "Something went wrong. Using backup network configuration!"

    for i in /etc/sysconfig/network-scripts/ifcfg-*.bak
    do
        if [ "$i" != "/etc/sysconfig/network-scripts/ifcfg-$NETDEV.bak"
        ]
        then
            F=`echo $i | sed -e 's/\.bak$//'\`
            mv $i $F
        fi
    done

    if [ -f mv /etc/chandev.conf.bak ]
    then
        mv /etc/chandev.conf.bak /etc/chandev.conf
    fi
else
    echo "LDAP informations retrieved successfully."

    ifdown eth0

    echo "reset_conf" > /proc/chandev
    echo "read_conf" > /proc/chandev
    echo "reprobe" > /proc/chandev

    rmmod qeth && modprobe qeth

fi

```

```
exit 0
```

C.2.2 The /usr/sbin/dasd script

Example C-10 shows the /usr/sbin/dasd script, a systems administration helper script for dynamically adding, listing, enabling and disabling DASD devices. It is merely a wrapper script that hides the underlying manipulation of the /proc/dasd/devices interface to the kernel DASD driver.

Example: C-10 The /usr/sbin/dasd script

```
#!/bin/sh
# dasd - simple utility for dynamic DASD management

if [ "$1" = "add" -a "$2" != "" ]; then
    echo "add range=$2" > /proc/dasd/devices
elif [ "$1" = "on" -a "$2" != "" ]; then
    echo "set device range=$2 on" > /proc/dasd/devices
elif [ "$1" = "off" -a "$2" != "" ]; then
    echo "set device range=$2 off" > /proc/dasd/devices
elif [ "$1" = "list" ]; then
    cat /proc/dasd/devices
else
    echo "Usage: dasd add|on|off vdev_or_range" 1>&2
    echo "      dasd list" 1>&2
    exit 2
fi
```

C.3 The GVCOPY EXEC

In Example C-11, we show the GVCOPY EXEC discussed in 9.7.3, “Create the Linux clone guestvol” on page 183.

Example: C-11 The GVCOPY script

```
/* */
parse upper arg src_user src_vdev dst_user dst_vdev
say "Copying guestvol from" src_user src_vdev "to" dst_user dst_vdev
address command CP "LINK" src_user src_vdev "2777" "RR"
address command CP "LINK" dst_user dst_vdev "3777" "MW"
queue "SYSPRINT CONS"
queue "INPUT 2777 3390"
queue "OUTPUT 3777 3390"
```

```
queue "COPY ALL"
queue "YES"
queue "YES"
queue ""
DDR
address command CP "DETACH 2777"
address command CP "DETACH 3777"
```

C.4 The GUESTACT EXEC script

Example C-12 shows the GUESTACT EXEC script discussed in 9.8.5, “The GUESTACT EXEC script” on page 188.

Example: C-12 The GUESTACT EXEC script

```
/*
 * GUESTACT - perform a variety of actions on a guest in GUEST CONF
 * Intended for use as a PROP action script
 *
 * Copyright 2002, IBM UK Ltd
 * Author: Malcolm Beattie <beattiem@uk.ibm.com>
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License
 * as published by the Free Software Foundation; either version
 * 2 of the License, or (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * Sample code. Do not use in production.
 * 29 Jul 2002 Initial version
 */

conffile="GUEST CONF"

parse upper arg ruser rmode lglopr msgcode puser pnode netid rtable
pull msg
pull action

parse var msg . user restofmsg
okchars = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"
okchars = okchars !! "0123456789_-$"
if verify(user, okchars, "Nomatch") ! length(user)=0 ! length(user)>8
then do
```

```

        say "GUESTACT: Syntax error in requested username:" user
        exit 1
    end

/*
 * Only allow a username which appears in GUEST CONF
 */
found = 0
parse value stream(conffile,'c','open read') with ok fh
if ok == "ERROR:" then do
    say "GUESTACT: Error opening config file" conffile ":" fh
    exit 1
end

do while found == 0 & lines(fh) > 0
    parse value linein(fh) with cuser .
    if translate(cuser) == translate(user) then found = 1
end
ok = stream(fh,'c','close')
/* stream() barfs with RC(-3) used without the "ok =". Go figure. */

if found == 0 then do
    say "GUESTACT: Request for user not in guestlist:" user
    exit 1
end

/*
 * We need to do compare action below with "=" and not "=="
 * since PROP passes us the action string as 8 characters
 * right-padded with spaces.
 */

select
    when action = "XAUTOLOG" then
        address command CP "XAUTOLOG" user
    when substr(action, 1, 3) == "EXT" then do
        extint = substr(action, 4, 4)
        address command CP "SEND CP" user "EXT" extint
    end
    when action = "LOGOFF" then
        address command CP "SEND CP" user "LOGOFF"
    when action = "DISC" then
        address command CP "SEND CP" user "DISC"
    when action = "BEGIN" then
        address command CP "SEND CP" user "BEGIN"
    when action = "RUNON" then
        address command CP "SEND CP" user "SET RUN ON"
    when action = "CPCMD" then
        address command CP "SEND CP" user restofmsg

```



```

when action = "VMCMD" then
    address command CP "SEND" user restofmsg
when action = "CONSTART" then
    address command CP "SEND CP" user "SPOOL CON TO" ruser "START"
when action = "CONCLOSE" then
    address command CP "SEND CP" user "SPOOL CON CLOSE"
when action = "CONOFF" then
    address command CP "SEND CP" user "SPOOL CON OFF"
otherwise do
    say "GUESTACT: Unknown action:" action
    exit 1
end
end
exit 0

```

C.5 The GETCONF EXEC script

Example C-13 shows the GETCONF EXEC script discussed in 8.11.1, “The z/VM configuration server” on page 164.

Example: C-13 The GETCONF EXEC script

```

/*
 * GETCONF EXEC
 * Retrieve guest configuration information.
 * Intended to be called as a PROP action routine, not manually.
 *
 * Copyright 2002, IBM UK Ltd
 * Author: Malcolm Beattie <beattiem@uk.ibm.com>
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License
 * as published by the Free Software Foundation; either version
 * 2 of the License, or (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * Sample code. Do not use in production.
 *
 * 24 July 2002 Initial Version
 */

prtdev="001E"
conffile="GUEST CONF"

```

```

parse upper arg ruser rnode lglopr msgcode puser pnode netid rtable
pull msg
pull method

/* Find role for which guest wants information and sanitise it */
parse var msg . role .
okchars = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"
okchars = okchars !! "0123456789_-"
if verify(role, okchars, "Nomatch") ! length(role)=0 ! length(role)>32
then role = "default"

/*
 * Look in GUEST CONF for the line whose first words are ruser and role
 * where the comparison for both is case *in*sensitive
 */
found = 0
do while found = 0 & lines(conffile) > 0
    parse value linein(conffile) with cuser crole info .
    if translate(cuser) == translate(ruser) ,
        & translate(crole) == translate(role)
        then found = 1
end
if found = 0 then info = "ERROR=No_config_info_found"

/*
 * Send back info as the reply via the appropriate method
 */
select
    when method = 'TAG' then
        address command CP 'SEND CP' ruser 'TAG' prtdev info
    when method = 'MSG' then
        address command CP 'MESSAGE' ruser 'GETCONF:' info
    otherwise
        say 'Method' method 'not supported'
end
exit 0

```

C.6 The PROP RTABLE configuration file

Example C-14 shows the PROP RTABLE configuration file discussed in 8.11.1, “The z/VM configuration server” on page 164.

Example: C-14 The PROP RTABLE configuration file

```

LGLOPR MBEATTIE VMLINUX
TEXTSYM / $ ^

```

```

LOGGING ALL
ROUTE
*
* ALLOW INTERNAL PROP COMMANDS TO AUTHORIZED PEOPLE
*
/SET /                1  4  MBEATTIE          DMSPOR  SET
/GET /                1  4  MBEATTIE          DMSPOR  GET
/CMD /                1  4  MBEATTIE          DMSPOR  TOVM
/QUERY /             1  6  MBEATTIE          DMSPOR  QUERY
/STOP /              1  5  MBEATTIE          DMSPOR  STOP
/LGLOPR /            1  8  MBEATTIE          DMSPOR  LGLOPR
/LOADTBL /           1  8  MBEATTIE          DMSPOL
/*/                  1  1  MBEATTIE
/LOG /               1  4  MBEATTIE
/GETMYCONF /         1 10  4                GETCONF  TAG
/GETMYCONFMSG /     1 13  4                GETCONF  MSG
/XAUTOLOG /          1  9  4 MBEATTIE          GUESTACT XAUTOLOG
/LOGOFF /            1  7  4 MBEATTIE          GUESTACT LOGOFF
/SHUTDOWN /          1  9  4 MBEATTIE          GUESTACT EXTOD1E
/DISC /              1  5  4 MBEATTIE          GUESTACT DISC
/BEGIN /             1  6  4 MBEATTIE          GUESTACT BEGIN
/RUNON /             1  6  4 MBEATTIE          GUESTACT RUNON
/CPCMD /            1  6  4 MBEATTIE          GUESTACT CPCMD
/VMCMD /            1  6  4 MBEATTIE          GUESTACT VMCMD
/STARTCONS /        1 10  4 MBEATTIE          GUESTACT CONSTART
/STOPCONS /         1  9  4 MBEATTIE          GUESTACT CONOFF
/SENDCONS /         1  9  4 MBEATTIE          GUESTACT CONCLOSE
* Web interface
/XAUTOLOG /          1  9  4 LNX5                GUESTACT XAUTOLOG
/SHUTDOWN /          1  9  4 LNX5                GUESTACT EXTOD1E
/STARTCONS /        1 10  4 LNX5                GUESTACT CONSTART
/STOPCONS /         1  9  4 LNX5                GUESTACT CONOFF
/SENDCONS /         1  9  4 LNX5                GUESTACT CONCLOSE

```

C.7 The redbook.schema file

Example C-15 illustrates the LDAP schema definition discussed in 10.4.1, “The redbook LDAP schema” on page 197.

Example: C-15 The redbook.schema

```

#
# SG24-6824-00 Redbook's directory schema items
#
# These are provided for informational purposes only.

```

```

objectIdentifier itsoOID 1.3.6.1.4.1.2.4711
objectIdentifier itsoSNMP itsoOID:1
objectIdentifier itsoLDAP itsoOID:2
objectIdentifier itsoAttributeType itsoLDAP:1
objectIdentifier itsoObjectClass itsoLDAP:2

attributetype ( itsoAttributeType:1 NAME 'VMNodeName'
                DESC 'z/VM Node Name'
                EQUALITY caseIgnoreMatch
                SUBSTR caseIgnoreSubstringsMatch
                SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{8} )

attributetype ( itsoAttributeType:2 NAME 'VMGuestName'
                DESC 'Linux Guest Name under z/VM Node Name'
                EQUALITY caseIgnoreMatch
                SUBSTR caseIgnoreSubstringsMatch
                SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{8} )

attributetype ( itsoAttributeType:3 NAME 'VMGuestRole'
                DESC 'Linux Guest Role'
                EQUALITY caseIgnoreMatch
                SUBSTR caseIgnoreSubstringsMatch
                SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{8} )

attributetype ( itsoAttributeType:4 NAME 'VMGuestMgmtNetwork'
                DESC 'Linux Guest Management Network Name and IP'
                EQUALITY caseIgnoreMatch
                SUBSTR caseIgnoreSubstringsMatch
                SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{128} )

attributetype ( itsoAttributeType:5 NAME 'VMGuestNetwork'
                DESC 'Linux Guest Network Name and IP'
                EQUALITY caseIgnoreMatch
                SUBSTR caseIgnoreSubstringsMatch
                SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{128} )

attributetype ( itsoAttributeType:6 NAME 'VMGuestGatewayIP'
                DESC 'Linux Guest Network - Gateway IP Address'
                EQUALITY caseIgnoreMatch
                SUBSTR caseIgnoreSubstringsMatch
                SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{12} )

attributetype ( itsoAttributeType:7 NAME 'VMGuestNet'
                DESC 'Linux Guest Network - Network'
                EQUALITY caseIgnoreMatch
                SUBSTR caseIgnoreSubstringsMatch
                SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{12} )

```

```

attributetype ( itsoAttributeType:8 NAME 'VMGuestNetMask'
DESC 'Linux Guest Network - Network Mask'
EQUALITY caseIgnoreMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{12} )

attributetype ( itsoAttributeType:9 NAME 'VMGuestLDAPIP'
DESC 'Linux Guest Network - LDAP Server IP Address'
EQUALITY caseIgnoreMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{12} )

attributetype ( itsoAttributeType:10 NAME 'VMGuestNetType'
DESC 'Linux Guest Network - Network Type (eth,hsi)'
EQUALITY caseIgnoreMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{3} )

attributetype ( itsoAttributeType:11 NAME 'VMGuestNetCHANDEV'
DESC 'Linux Guest Network - Network CHANDEV Parameter'
EQUALITY caseIgnoreMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{80} )

objectClass ( itsoObjectClass:1 NAME 'VMGuest'
DESC 'IBM ITSO Redbook SG24-6824 Linux VM Guest Object'
SUP top AUXILIARY
MUST ( VMnodename $ VMguestname $ VMGuestMgmtNetwork )
MAY ( VMGuestRole $ VMGuestNetwork )
)

objectClass ( itsoObjectClass:2 NAME 'VMGuestManagementNetwork'
DESC 'IBM ITSO Redbook SG24-6824 Linux VM Guest Management Network
Object'
SUP top AUXILIARY
MUST ( VMGuestMgmtNetwork $ VMGuestNet $ VMGuestGatewayIP $
VMGuestNetMask $ VMGuestLDAPIP $ VMGuestNetType )
)

objectClass ( itsoObjectClass:3 NAME 'VMGuestPublicNetwork'
DESC 'IBM ITSO Redbook SG24-6824 Linux VM Guest Management Network
Object'
SUP top AUXILIARY
MUST ( VMGuestNetwork $ VMGuestNet $ VMGuestGatewayIP $
VMGuestNetMask $ VMGuestNetType $ VMGuestNetCHANDEV )
)

```

C.8 The sample-ldap.php script

The sample-ldap.php script resets LDAP passwords as discussed in 10.7.1, “Interface to reset passwords” on page 212.

Attention: This script is inherently insecure as there is no encryption or authentication. This is provided only as a sample.

Example: C-16 The sample-ldap.php script

```
<?php
$ldap_host="10.0.3.1";
$ldap_basedn="o=IBM-ITSO,c=US";
$ldap_rootdn="cn=Manager, ". $ldap_basedn;
$ldap_rootdn_password = "secret";
$password="notverysecret";

echo "<H1>LDAP password reset frontend</H1>";

$connect=ldap_connect($ldap_host);
if (!$connect) {
    die ("Could not connect to LDAP server!");
}

/* if 'dn' argument passed, reset users password */
if ($dn) {
    echo "LDAP DN of user is '". $dn .'<BR>";

    /* bind as manager */
    $result=@ldap_bind($connect, $ldap_rootdn, $ldap_rootdn_password);

    if ($result) {
        /* generate a good password hash */
        $random = md5(time() . getmypid());
        $salt = substr($random,0,2);
        $cryptpw = crypt($password,$salt);

        /* update the LDAP attribute "userPassword" */
        $entry[userPassword]="{CRYPT} ". $cryptpw;
        $result = ldap_mod_replace($connect, $dn, $entry);

        if ($result) {
            echo "password is now '". $password ."'";
        }
    }

    /* lookup list of users */
} else {
```

```

/* bind anonymous */
$r=ldap_bind($connect);

$result=ldap_search($connect,"ou=People,". $ldap_basedn,
"objectclass=posixaccount");
$info = ldap_get_entries($connect, $result);

echo "<TABLE BORDER>";
echo "<TR><TD>LDAP DN of user</TD>";
echo "    <TD>Password</TD></TR>";

for ($i=0; $i<$info["count"]; $i++) {
    $dn=$info[$i]["dn"];

    /* create row for each user, pass his DN attribute as 'dn' */
    echo "<TR><TD>". $dn ."</TD>";
    echo "    <TD><A HREF=\" /sample-ldap.php?dn=". urlencode($dn)
."\">reset</A></TD></TR>";
}
echo "</TABLE>";
}

ldap_close($connect);

```

C.9 The ipl-shutdown.php script

The ipl-shutdown.php script shuts down basevol/guestvol Linux guest as discussed in 10.7.2, “Interface to IPL and shutdown Linux guests” on page 213.

Attention: This script is inherently insecure as there is no encryption or authentication. This is provided only as a sample.

Example: C-17 The ipl-shutdown.php script

```

<?php
$ldap_host="10.0.3.1";
$ldap_basedn="ou=VM,o=IBM-ITS0,c=US";

function myhcp($guest, $action) {
    $command="/usr/sbin/hcp msg confserv ". $action ." ". $guest;
    echo $command ."<BR>";

    if (!($p=popen("($command)2>&1","r"))) return 126;
    while (!feof($p)) {
        $l=fgets($p,2342);
        print $l;
    }
}

```

```

    }
    return pclose($p);
}

echo "<H1>List of LCLxxx@VMLINUX Linux guests</H1>";

if ($guest) {
    myhcp ($guest,$action);
} else {
    $connect=ldap_connect($ldap_host);
    if (!$connect) {
        die ("Could not connect to LDAP server!");
    }

    /* bind anonymous */
    $r=ldap_bind($connect);

    $result=ldap_search($connect,$ldap_basedn,
        "(&(objectclass=VMGuest)(VMGuestName=LCL*))");

    $info = ldap_get_entries($connect, $result);

    echo "<TABLE BORDER>";
    echo "<TR><TD>VMGuest</TD>";
    echo "    <TD COLSPAN=2>Action</TD></TR>";

    /* for each VMGuest object found in the LDAP tree */
    for ($i=0; $i<$info["count"]; $i++) {
        $dn=$info[$i]["dn"];
        $vmguest=$info[$i]["vmguestname"][0];

        /* offer IPL and Shutdown option via guestact script */
        echo "<TR><TD>". $vmguest ."</TD>";
        echo "    <TD><A HREF=\"ipl-shutdown.php?guest=". $vmguest
            ."&action=xautolog\">IPL</A></TD>";
        echo "    <TD><A HREF=\"ipl-shutdown.php?guest=". $vmguest
            ."&action=shutdown\">SHUTDOWN</A></TD></TR>";
    }
    echo "</TABLE>";

    ldap_close($connect);
}

```



Additional material

This redbook refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material associated with this redbook is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

<ftp://www.redbooks.ibm.com/redbooks/SG246824>

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the redbook form number, SG24-6824.

Using the Web material

The additional Web material that accompanies this redbook includes the following files:

<i>File name</i>	<i>Description</i>
sg246824.tar.gz	A gzipped tar file containing the sample code.

System requirements for downloading the Web material

The following system configuration is recommended:

Hard disk space: 15 MB

Operating System: Linux for zSeries or S/390

How to use the Web material

Copy and unzip the `sg246824.tar.gz` to your Linux server. The tar file expands to the `sg246824` directory. In this directory, you will find:

▶ The **REXX** directory

This contains the REXX scripts and configuration files discussed in Chapter 8, “Shared Linux filesystems” on page 149. Copy these files to the PROP user virtual machine (CONFSERV, in our configuration).

▶ The **bin** directory

This contains the `ldpasswd.pl` Perl script to generate an LDIF file used to modify a user password as discussed in 10.5.6, “Changing passwords stored in LDAP” on page 205.

▶ The **src** directory

This contains the `ext_int.c` source file for the `ext_int` device driver discussed in 9.8.1, “The `ext_int` kernel module” on page 184.

▶ The **LDAP** directory

This contains the OpenLDAP server configuration and schemas discussed in Chapter 10, “Centralized management using LDAP” on page 191.

▶ The **RPMS** directory

This contains package discussed throughout Part 3, “Creating and managing a penguin colony” on page 147.

▶ The **SPECS** directory

This contains the RPM SPEC files needed to rebuild the packages in the RPMS directory.

Abbreviations and acronyms

BNF	Backus Naur Form	LANANA	Linux Assigned Names and Numbers Authority
CBQ	Class-Based Queue	LDAP	Lightweight Directory Access Protocol
CF	Coupling Facility	LDIF	LDAP Data Interchange Format
CHPID	Channel-path Identifier	LPAR	Logical Partition
CMS	Conversional Monitoring System	LSA	Link State Advertisement
CP	Control Program	LSB	Linux Standard Base
CTC	Channel-To-Channel	LVM	Logical Volume Manager
DASD	Direct Access Storage Device	LVS	Linux Virtual Server
DDS	Distributed Data Server	NIC	Network Interface Card
DMZ	Demilitarized Zone	NSS	Name Service Switch
DN	Distinguished Name	OCO	Object Code Only
DNS	Domain Name System	OSA	Open System Adapter
DoS	Denial of Service	OSPF	Open Shortest Path First
EMIF	Enhanced Multiple Image Facility	PAM	Pluggable Authentication Modules
FHS	Filesystem Hierarchy Standard	PGP	Pretty Good Privacy
FIFO	First In First Out	PHP	PHP: Hypertext Preprocessor
HA	High Availability	PM	Performance Monitor
HSA	HiperSockets Accelerator	PRF	Performance Reporting Facility
IBM	International Business Machines Corporation	QDIO	Queued Direct Input/Output
IGRP	Interior Gateway Routing Protocol	RFC	Request For Comment
IQD	Internal Queued Direct	RIP	Routing Information Protocol
IS-IS	Intermediate System-to-Intermediate System	RMF	RealTime Monitor Function
ITSO	International Technical Support Organization	RPM	RedHat Package Manager
IUCV	Inter-User Communication Vehicle	RSVP	Reservation Protocol
LAN	Local Area Network	RTM	RealTime Monitor
		SATF	Shared-access Transport Facility
		SMB	System Message Block

SNMP	System Network Message Protocol
SSL	Secure Sockets Layer
STI	Self-Timed Interrupt
TBF	Token Bucket Flow
UUID	Universal Unique Identifier
VIPA	Virtual IP Addressing
VMRM	Virtual Machine Resource Manager
VRRP	Virtual Router Redundancy Protocol
WSC	Washington Systems Center
XML	Extensible Markup Language

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 on ordering these publications, see “How to get IBM Redbooks” on page 286.

- ▶ *Linux on IBM @server zSeries and S/390: ISP/ASP Solutions*, SG24-6299
<http://www.ibm.com/redbooks/abstracts/sg246299.html>
- ▶ *Linux on IBM @server zSeries and S/390: Distributions*, SG24-6264
<http://www.ibm.com/redbooks/abstracts/sg246264.html>
- ▶ *Linux for S/390*, SG24-4987
<http://www.ibm.com/redbooks/abstracts/sg244987.html>
- ▶ *zSeries Hipersockets*, SG24-6816
<http://www.ibm.com/redbooks/abstracts/sg246816.html>
- ▶ *Understanding LDAP*, SG24-4986
<http://www.ibm.com/redbooks/abstracts/sg244986.html>
- ▶ *ABCs of System Programming Volume 1*, SG24-5597
<http://www.ibm.com/redbooks/abstracts/sg245597.html>
- ▶ *Linux on IBM zSeries and S/390: High Availability for z/VM and Linux*, REDP0220
<http://www.ibm.com/redbooks/abstracts/redp0220.html>
- ▶ *Linux on IBM zSeries and S/390: Securing Linux for zSeries with a central z/OS LDAP server (RACF)*, REDP0221
<http://www.ibm.com/redbooks/abstracts/redp0221.html>
- ▶ *Linux on IBM zSeries and S/390: Server Consolidation with Linux for zSeries*, REDP0222
<http://www.ibm.com/redbooks/abstracts/redp0222.html>

Other resources

These publications are also relevant as further information sources:

- ▶ *z/VM V4R3.0 CP Command and Utility Reference*, SC24-6008-03
- ▶ *z/VM V4R3.0 Directory Maintenance Facility Function Level 410 Command Reference*, SC24-6025
- ▶ *z/VM V4R3.0 Directory Maintenance Facility Function Level 410 Tailoring and Administration Guide*, SC24-6024
- ▶ *z/VM V4R3.0 CMS Planning and Administration*, SC24-6042
- ▶ *z/VM V4R3.0 Performance*, SC24-5999
- ▶ *z/VM V4R3.0 TCP/IP Level 430 Planning and Customization*, SC24-6019
- ▶ *Linux - Advanced Networking Overview* by Saravanan Radhakrishnan found at:
<http://qos.ittc.ukans.edu/howto/howto.html>
- ▶ *Linux 2.4 Advanced Routing HOWTO* by Netherlabs BV et al. found at:
<http://www.linuxguruz.org/iptables/howto/2.4routing.html>
- ▶ *Designing Large-Scale IP Internetworks* by Cisco Systems found at:
<http://www.cisco.com/univercd/cc/td/doc/cisintwk/idg4/nd2003.htm>
- ▶ *Maximum RPM - Taking the Red Hat Package Manager to the Limit*, by Edward C. Bailey found at:
<http://www.rpm.org/max-rpm/>
- ▶ *OpenLDAP home page* at:
<http://www.openldap.org/>
- ▶ *The Linux-PAM System Administrators' Guide* by Andrew G. Morgan
<http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/pam.html>
- ▶ *LDAP Implementation HOWTO* by Roel van Meer and Giuseppe Lo Biondo, by Roel van Meer and Giuseppe Lo Biondo found at:
<http://tldp.org/HOWTO/LDAP-Implementation-HOWTO/>
- ▶ *LDAP Linux HOWTO* at:
<http://www.tldp.org/HOWTO/LDAP-HOWTO/>
- ▶ *PADL Software Pty Ltd* Open Source Software distribution site at:
<http://www.padl.com/Contents/OpenSourceSoftware.html>
- ▶ *Samba-LDAP PDC HOWTO* at:
<http://samba.idealx.org/>
- ▶ *Proceedings of the 2002 Ottawa Linux Symposium* at:

http://www.linux.org.uk/~ajh/ols2002_proceedings.pdf.gz

- ▶ *Linux for zSeries and S/390 Device Drivers and Installation Commands*, LNUX-1303 at:
<http://www.ibm.com/developerworks/opensource/linux390/docu/linuxdd01.pdf>

Referenced Web sites

These Web sites are also relevant as further information sources:

- ▶ z/VM Version 4 Release 3 Statement of Direction
http://www.ibm.com/servers/eserver/zseries/library/specsheets/gm130075_more2.html#9
- ▶ VM Solutions for System Performance
<http://www.vm.ibm.com/perf/perfprod.html>
- ▶ z/OS RMF PM with Support for Linux Enterprise Server
<http://www.ibm.com/servers/eserver/zseries/zos/rmf/rmfhtmls/pmweb/pmlin.htm>
- ▶ TCP/IP Stack Limitation on OSA-Express
<http://www-1.ibm.com/support/techdocs/atmastr.nsf/PubAllNum/Flash10144>
- ▶ RFC1918 - Address Allocation for Private Internets
<http://www.faqs.org/rfcs/rfc1918.html>
- ▶ RFC1122 - Requirements for Internet Hosts -- Communication Layers
<http://www.faqs.org/rfcs/rfc1122.html>
- ▶ Useful add-ons for Linux on zSeries and S/390 for IBM Developerworks
http://oss.software.ibm.com/linux390/useful_add-ons.shtml
- ▶ The Plan9 from Bell Labs home page
<http://plan9.bell-labs.com/plan9dist/>
- ▶ The Filesystem Hierarchy Standard home page
<http://www.pathname.com/fhs/>
- ▶ The LDAPzone home page
<http://www.ldapzone.com/>
- ▶ The LDAP Browser/Editor home page
<http://www.iit.edu/~gawojar/ldap/>
- ▶ PADL Software LDAP migration tools
<http://www.padl.com/OSS/MigrationTools.html>

- ▶ PADL Software LDAP documentation page
<http://www.padl.com/Contents/Documentation.html>
- ▶ The Internet Software Consortium bind server implementation
<http://www.isc.org/products/BIND/>
- ▶ An SDB LDAP add-on for the the ISC bind server
<http://www.venaas.no/ldap/bind-sdb/>
- ▶ The OCO qeth drivers from IBM Developerworks for RedHat on zSeries and S/390
http://www.ibm.com/developerworks/opensource/linux390/special_oco_rh_2.4.shtml
- ▶ IEEE “Get IEEE 802”(TM) Home Page
<http://standards.ieee.org/getieee802/>
- ▶ IBM Developerworks: Linux for zSeries and S/390
<http://www.ibm.com/developerworks/opensource/linux390/index.shtml>
- ▶ LANANA (Linux Assigned Names and Numbers Authority)
<http://www.lanana.org/>
- ▶ 2002 Ottawa Linux Symposium
<http://www.linuxsymposium.org/2002/>

How to get IBM Redbooks

You can order hardcopy Redbooks, as well as view, download, or search for Redbooks at the following Web site:

ibm.com/redbooks

You can also download additional materials (code samples or diskette/CD-ROM images) from that site.

IBM Redbooks collections

Redbooks are also available on CD-ROMs. Click the CD-ROMs button on the Redbooks Web site for information about all the CD-ROMs offered, as well as updates and formats.

Index

A

addvmur 217

B

basevol 155

 contents 155

basevol+guestvol-1.0.0-1.noarch.rpm 177, 232

basevol/guestvol Linux guest 156

 advantages 172

 configuration 162

 early boot-time configuration 163

 maintenance shell 161

 RPM management 156

 scripts

 /etc/init.d/guestvol-start-halt 170, 254

 /etc/init.d/itsoldap 199, 263

 /etc/init.d/itsonet 168, 252

 /etc/init.d/vmgetconf 166, 250

 /etc/rc.d/rc.guestvol 160, 232

 /etc/rc.d/rc.sysinit 163

 /etc/rc.d/rc.sysinit-guestvol 163, 234

 /etc/rc.d/rc3.d/S09itsonet 168

 /etc/sysconfig/vmconfigsrvr 167

 /sbin/guestvol-final-halt 170, 257

 /usr/sbin/basevol-devel 178, 261

 /usr/sbin/dasd 179, 268

 /usr/sbin/mkguestvol 179, 261

 shutdown 169

 startup 159

bind mount 150

 preserving access to read-only directories 154

 writable directories on a read-only filesystem
 153

C

confserv 164

 CONFSErv virtual machine 164

 PROP configuration file 164

 request 167

 response 165

 security considerations 166

convert ext2 filesystem to ext3 228

convert ext3 filesystem to ext2 228

CP

 command abbreviation 7

 command privilege classes 7

 command structure 6

 command truncation 7

 commands

 ATTach 59

 Begin 13

 COUPLE 66

 DEFine LAN 63

 DEFine NIC 65

 DEFine STORage 19

 DETach LAN 65

 DETach NIC 67

 DISConnect 13

 EXTernal 188

 Help 6

 lpl 14

 Logon 4

 Logon BY 4

 Message 7

 Query DAsd 9

 Query LAN 68

 Query NIC 17

 Query PF 12

 Query Time 6

 Query Virtual CPUS 18

 Query Virtual DAsd 9

 Query Virtual STorage 18

 Set IOPRIORity 21

 Set LAN 68

 Set PF 11

 Set SHARE 21

 SHUTDOWN 188

 SMsg 164–165, 187

 XAUTOLog 188

 issue CP commands from Linux guest 15

 status indicator 11

 CP READ 11

 HOLDING 11

 MORE... 11

 NOT ACCEPTED 11

 RUNNING 11

VM READ 11
cpint package 15
cpint-1.1.1-1.s390x.rpm 177

D

dasdfmt 179
DDR 19
device filesystem mounts 150
DirMaint 23
 access passwords on minidisks 33
 administrator 27
 APAR VM63033 24
 AUTHFOR CONTROL file 27
 command syntax 25
 commands
 ADD 31, 183
 AMDISK 33
 CLASS 34
 DEDICATE 33
 DMDISK 33
 FILE 28, 183
 LOGONBY 35
 MDISK 33
 REVIEW 32
 RLDC 28
 RLDE 30
 SEND 27
 SETOPTN 34
 SPECIAL 34, 71
 STORAGE 34
 TMDISK 34
 EXTENT CONTROL file 28
 prefix keywords 26
 profile directory entry 30
 service machines 24
 4VMDVH10 24
 DATAMOVE 24
 DIRMAINT 24
 DIRMSAT 24
 user directory entry 31
 user prototypes 31
dynamic routing 116
 controlling routing tables 119
 growth of the neighborhood 118
 how it works 116
 use in a penguin colony 117

E

ext_int kernel module 184

F

FCON/ESA 38
 CPU subcommand 49
 customizing for Linux 41
 control file 41
 profile 42
 DEVICE subcommand 51
 Distributed Data Server 39
 download 39
 install 40
 start 40
 viewing monitored data 40
 Linux systems option 42
 STORAGE subcommand 50
 subcommands for Linux guests 43
 LINUX 44
 LXCPU 46
 LXFILESYS 48
 LXMEM 47
 LXNETWR 48
 support for Linux 38
 USER subcommand 52
fdasd 179, 227
FHS 156
firewall features 99

G

GETCONF EXEC 164, 271
GUEST CONF 183
GUEST CONF file 165
guestact CGI script 186
GUESTACT EXEC 188, 269
guestvol 155
 contents 155
 device number 777 160
GVCOPY EXEC 183, 268

H

hcp command 15
HiperSockets 58
 capabilities 58
 configuring 58
 IODF definitions 59
 Linux configuration

- attaching devices 60
 - configuring the interface 61
 - updating system configuration 61
 - updating the dynamic routing daemon 61
 - operating system support 58
 - VM configuration
 - activate interface 60
 - attaching devices 59
 - configure TCP/IP stack 60
 - update MPROUTE 60
- I**
- IEEE 802.1Q VLAN 83
 - broadcast 85
 - configuring 89
 - infrastructure guests 90
 - isolation 85
 - Linux support 85
 - OSA-Express sharing 87
 - theory of operation 83
 - ifconfig command 62, 90
 - ip command 62, 114
 - ipl-shutdown.php 277
 - itsobasevol-1.0.0-1.s390x.rpm 177, 262
- K**
- kernel boot parameters 228
- L**
- LDAP 192
 - authentication using LDAP 201
 - configuring NSS 203
 - configuring PAM 202
 - nss-ldap and pam-ldap 201
 - browser 196
 - configuring clients 200
 - configuring the server 194
 - DNS 207
 - indexes 211
 - LDIF 196
 - migration tools 196
 - OpenLDAP directory server 192
 - schemas
 - core 195
 - cosine 195
 - dnszone 195
 - inetorgperson 195
 - nis 195
 - redbook 195, 197
 - ldapadd 196
 - ldapmodify 206
 - Linux router 98
 - changing 100
 - device support 99
 - routing function 99
 - set up 99
 - LNXCONE prototype 181
- M**
- makeurdev 217
 - mke2fs 179
 - mtr 99
- N**
- NSS 201
 - nss_ldap-198-1.s390x.rpm 177
- O**
- OBEYFILE command 102
 - gotcha 102
 - OSA
 - broadcast traffic issues 83
 - defining as reconfigurable 80
 - defining as shareable 81
 - IP Assist 81
 - LCS mode 80
 - number of Linux guests 78
 - OSA-2 78
 - OSA-Express 78
 - port sharing 78
 - EMIF 80
 - hardware definition 80
 - QDIO mode 80
 - stack-to-stack routing 82
 - OSPF 99
- P**
- PAM 201
 - pam_ldap-150-1.s390x.rpm 177
 - penguin colony xiii
 - PROP 164, 187
 - PROP RTABLE 188, 272

R

- redbook.schema 197, 273
- Redbooks Web site 286
 - Contact us xvi
- RedHat 7.1 with OCO modules 221
 - installing 223
 - second initial ramdisk 223
- RIP 99
- RMF PM for Linux 39
- routed 99
- routing 94
 - considerations with OSAs 97
 - Linux vs z/VM 96
 - device support 97
 - extra routing features 97
 - limited device support in Linux 96
 - Linux reconfigurability 97
 - planning 94
 - address allocation 95
 - connectivity 94
 - isolation 95
 - RFC1918 96
 - traffic shaping 96

S

- sample-ldap.php 276
- SNMP 99

T

- tc command
 - class 114
 - filter 114
 - qdisc 114
- traceroute command 109
- traffic control 110
 - bandwidth choke 114
 - components 111
 - configuring CBQ 112
 - differentiating interactive traffic 115
 - queue disciplines 111
- tune2fs 228

U

- Unit Record 215
 - device driver 216
 - character devices 217
 - installing 216

- utility 218
 - command syntax 218
- ur add 220
- ur copy 218
- ur info 219
- ur list 220
- ur remove 220

V

- vconfig command 89
- Virtual Machine Resource Manager 20

VM

- LBYONLY 182
 - profile directory entry 30
 - prototype directory entry 31
 - SPECIAL statement 70
 - SYSTEM CONFIG file 70
 - SYSTEM DTCPARMS file 59
 - user directory entry 31
 - XAUTOLog 182
- VM Guest LAN 62
 - /etc/chandev.conf file 74
 - /etc/modules.conf file 75
 - automating connections 71
 - changing attributes 68
 - configuration 63
 - attaching simulated NIC 66
 - creating LAN segment 63
 - creating simulated NIC 65
 - establishing lifetime 64
 - establishing owner 64
 - define and couple simulated NIC 70
 - interface naming conventions 74
 - Linux network device drivers 73
 - loading network device driver 73
 - persistent 65
 - restricted 67
 - transient 65
 - viewing attributes 68
 - virtual Hipersockets 62
 - virtual QDIO 63
- vmhalt option 229
- vmpoff option 228

Z

- z/OS router 105
 - HiperSockets Accelerator 105
 - demonstrating 107

- Linux configuration 107
 - set up 106
 - when to use 109
- z/VM router 100
 - device support 100
 - routing function 100
- zebra 99



Redbooks

Large Scale Linux Deployment

(0.5" spine)
0.475" <-> 0.875"
250 <-> 459 pages



Linux on IBM server™ zSeries and S/390: Large Scale Linux Deployment



Redbooks

z/VM concepts and tools for Linux deployment

This IBM Redbook discusses techniques available for building a large farm of Linux servers running under zVM (a “penguin colony”). It has been developed for system administrators and I/T architects who are responsible for developing optimized solutions for large Linux systems installed on IBM zSeries and S/390 machines.

Networking and routing Linux guests running under z/VM

The book is divided into three parts. We first discuss basic VM concepts and commands, and examine tools available on z/VM that can help you create and monitor a server farm.

Building a server farm for Linux on zSeries

We then cover networking features and options, and discuss network high availability issues and solutions. Finally, we demonstrate methods for optimizing virtual Linux server farms (including a novel approach to sharing Linux filesystems among members), and examine central LDAP management. The sample code we discuss is available online.

We also recommend when to use the z/VM IP stack as a router, when to use a virtual Linux IP router, and when to dedicate logical OSA ports to virtual Linux Servers. Linux cloning techniques are also covered.

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

SG24-6824-00

ISBN 0738427373