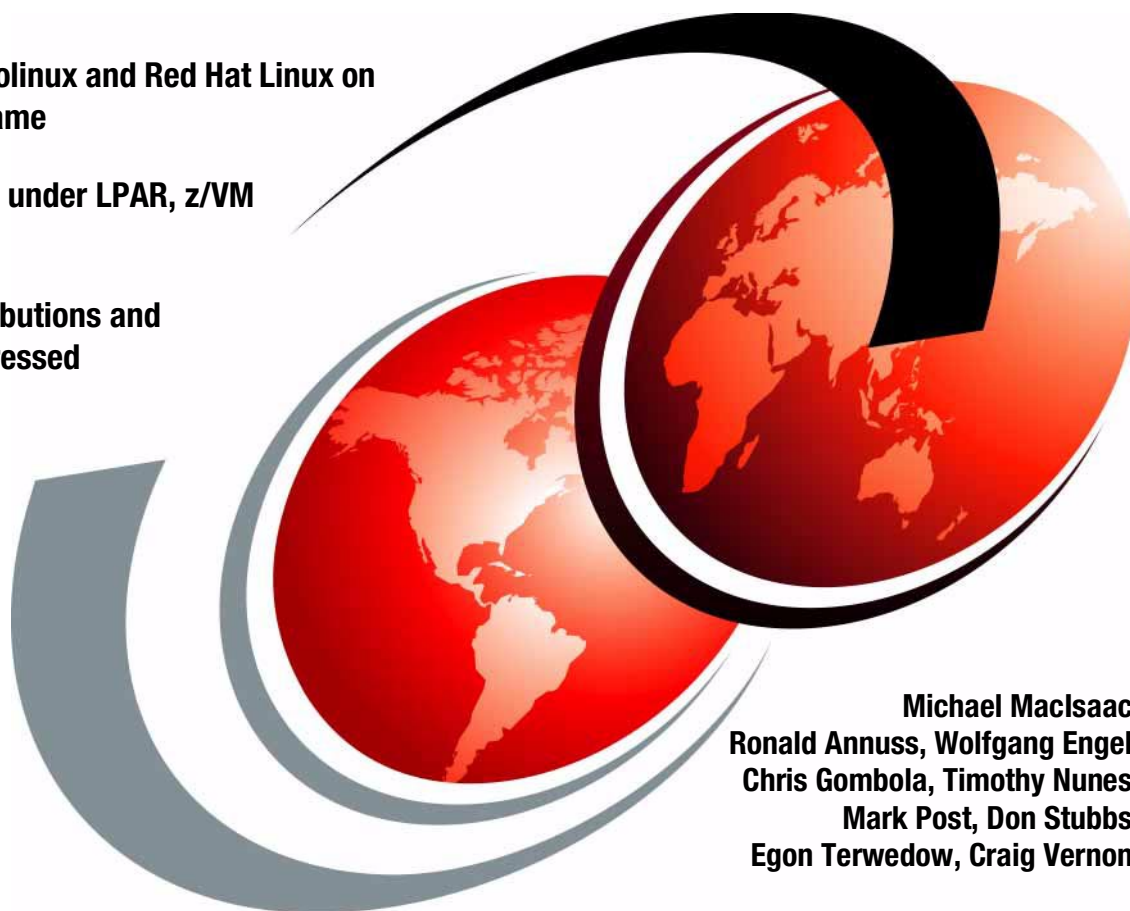


# Linux for IBM server zSeries and S/390: Distributions

SuSE, Turbolinux and Red Hat Linux on  
the mainframe

Installation under LPAR, z/VM  
and VIF

Other distributions and  
topics addressed



Michael Maclsaac  
Ronald Annuss, Wolfgang Engel  
Chris Gombola, Timothy Nunes  
Mark Post, Don Stubbs  
Egon Terwedow, Craig Vernon





International Technical Support Organization

**Linux for IBM @server zSeries and S/390:  
Distributions**

September 2001

**Take Note!** Before using this information and the product it supports, be sure to read the general information in “Special notices” on page 551.

### **First Edition (September 2001)**

This edition applies to many Linux distributions (see 1.2.3, “Linux distributions” on page 9), z/VM V4.1 (ESP) and VIF V1.0.0, SL0500.

Comments may be addressed to:  
IBM Corporation, International Technical Support Organization  
Dept. HYJ Mail Station P099  
2455 South Road  
Poughkeepsie, NY 12601-5400

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 2001. All rights reserved.

Note to U.S Government Users – Documentation related to restricted rights – Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

# Contents

<b>Preface</b> .....	xv
The team that wrote this redbook .....	xv
Special notice .....	xviii
IBM trademarks .....	xix
Comments welcome .....	xix
<b>Part 1. Linux overview</b> .....	1
<b>Chapter 1. Overview</b> .....	3
1.1 A brief history .....	4
1.2 Systems used for this redbook .....	8
1.2.1 Processor .....	8
1.2.2 DASD .....	9
1.2.3 Linux distributions .....	9
<b>Chapter 2. Planning for Linux</b> .....	11
2.1 Planning hardware .....	12
2.1.1 Integrated Facility for Linux (IFL) .....	13
2.1.2 Planning DASD .....	13
2.1.3 Planning for tape devices .....	15
2.2 Planning networking .....	15
2.2.1 Physical networking .....	15
2.2.2 Virtual networking .....	16
2.2.3 Linux OSA cards and drivers .....	16
2.2.4 IBM 2216 .....	22
2.2.5 CTC .....	22
2.2.6 TCP/IP calculations .....	22
2.2.7 Maximum Transmission Unit (MTU) sizes .....	23
2.3 Planning for Linux distribution and software .....	24
2.3.1 Planning file systems .....	24
2.4 Roadmap .....	26
<b>Chapter 3. Preparation</b> .....	27
3.1 Prepare the VM guest for Linux .....	28
3.1.1 The VM user directory .....	28
3.1.2 Processing the VM user directory .....	29
3.1.3 Editing the PROFILE EXEC and XEDIT PROFILE .....	29
3.1.4 Getting the installation files for VM .....	30
3.1.5 Booting the installation kernel from the VM reader .....	31

3.2	Prepare the LPAR for Linux	31
3.3	Swap space size and type	33
3.4	Booting the installation kernel in an LPAR	34
3.4.1	Using the ICKDSF bootstrap	34
3.4.2	Booting from HMC CD-ROM or FTP server	35
3.4.3	Booting from tape	44
<b>Part 2. Strategic Linux distributions</b>		<b>53</b>
<b>Chapter 4. Overview of SuSE Linux</b>		<b>55</b>
4.1	Getting the SuSE distribution	56
4.1.1	Via SuSE sales	56
4.1.2	Via FTP site	56
4.2	Documentation included with the SuSE distribution	57
4.3	Software components of the distribution	58
4.4	How does SuSE Linux differ	60
<b>Chapter 5. Installing SuSE Linux</b>		<b>61</b>
5.1	General installation considerations	62
5.1.1	Boot file processing	62
5.1.2	DASD considerations	62
5.1.3	Installation options and DASD requirements	63
5.2	Installing SuSE under VM	64
5.2.1	SuSE under VM checklist	64
5.2.2	Installing SuSE Linux under VM	65
5.2.3	Transferring installation system files to your VM guest	65
5.2.4	Logging in via telnet and completing the install via YaST	71
5.3	SuSE installation in an LPAR	88
5.4	Installing SuSE under VIF	89
5.5	Assessing the SuSE installation experience	89
<b>Chapter 6. Customizing and using SuSE Linux</b>		<b>91</b>
6.1	Using YaST	92
6.2	Samba installation	95
6.2.1	Install from an RPM	95
6.2.2	Samba configuration	95
6.2.3	Samba password encryption	98
6.2.4	Using SWAT	100
6.2.5	Samba documentation	101
6.3	Apache Web server	102
6.3.1	Apache configuration	102
6.3.2	Password-protecting a directory	103
6.3.3	Enabling CGI scripts	107
6.4	e-mail	107

6.5	Setting up LVM with YaST . . . . .	108
6.6	Graphical interface . . . . .	113
<b>Chapter 7. Overview of Turbolinux . . . . .</b>		<b>115</b>
7.1	History . . . . .	116
7.2	Getting the Turbolinux distribution . . . . .	116
7.2.1	Downloading the CD-ROM images . . . . .	117
7.2.2	Purchasing the distribution . . . . .	117
7.3	Available documentation . . . . .	118
7.4	Notable characteristics of Turbolinux . . . . .	118
7.4.1	Standardized customization and usage . . . . .	119
<b>Chapter 8. Installing Turbolinux . . . . .</b>		<b>121</b>
8.1	Turbolinux installation worksheet . . . . .	122
8.1.1	Planning DASD with Turbolinux . . . . .	123
8.2	Installation of Turbolinux under VM . . . . .	123
8.2.1	VM checklist . . . . .	124
8.2.2	Download boot files to VM . . . . .	124
8.2.3	Initial IPL of Linux . . . . .	125
8.3	Installation of Turbolinux in an LPAR . . . . .	126
8.3.1	Turbolinux under LPAR checklist . . . . .	126
8.3.2	IPL of Turbolinux . . . . .	127

<b>Chapter 10. Overview of Red Hat Linux</b> .....	151
10.1 Obtaining the Red Hat distribution .....	152
10.2 Documentation available .....	155
10.3 How Red Hat Linux differs .....	156
<b>Chapter 11. Installing Red Hat Linux</b> .....	157
11.1 Before you install .....	158
11.2 Preparing for installation .....	158
11.3 Installation of Red Hat under VM .....	162
11.3.1 Red Hat under VM checklist .....	163
11.4 Installation of Red Hat in an LPAR .....	163
11.4.1 Red Hat in an LPAR checklist .....	164
11.4.2 Comments on LPAR installation .....	164
11.5 Installation of Red Hat under VIF .....	164
11.5.1 Red Hat under VIF checklist .....	165
11.5.2 Comments on VIF installation .....	165
11.6 Common installation steps .....	165
11.7 Problems we encountered .....	174
<b>Chapter 12. Customizing and using Red Hat Linux</b> .....	177
12.1 Creating multiple IPL volumes .....	178
12.2 Installing linuxconf .....	178
12.3 Adding a non-root user .....	183
12.4 Managing the network .....	183
12.5 Configuring Apache .....	184
12.6 Managing the FTP Server .....	184
12.7 Managing Samba .....	185
12.7.1 Enabling SWAT .....	186
12.8 Mail server .....	186
12.9 Enabling xdm .....	187
12.10 Installing KDE .....	187
12.11 Installing GNOME .....	188
12.12 Miscellaneous .....	189
12.13 Installing maintenance on Red Hat .....	189
<b>Part 3. Other distributions</b> .....	191
<b>Chapter 13. The Marist File System</b> .....	193
13.1 History and overview of the Marist File System .....	194
13.1.1 Getting the Marist File System .....	194
13.1.2 Documentation .....	195
13.1.3 How the Marist File System differs .....	195
13.2 Installing the Marist File System .....	196
13.3 Installing the Marist File System under VM .....	198



13.3.1	Marist File System under VM checklist . . . . .	199
13.4	Customization . . . . .	206
13.4.1	Installing SSH . . . . .	206
13.4.2	Creating multiple IPL volumes . . . . .	207
13.4.3	Adding a non-root user . . . . .	208
13.4.4	Using Apache . . . . .	208
13.4.5	FTP server . . . . .	209
13.4.6	xdm . . . . .	209
13.4.7	Network . . . . .	209
13.4.8	Misc. . . . .	210
13.5	Problems we encountered . . . . .	210
<b>Chapter 14. Caiman Linux for zSeries . . . . .</b>		<b>213</b>
14.1	Learning more about Caiman . . . . .	214
14.2	What's specific about Caiman Linux . . . . .	214
14.3	System requirements . . . . .	214
14.3.1	Installation worksheet - Caiman . . . . .	215
14.3.2	Caiman disk requirements . . . . .	216
14.4	Installation . . . . .	216
14.4.1	Preparing the first IPL of the install system . . . . .	217
14.4.2	Initial IPL and network setup . . . . .	218
14.4.3	Installation of the base system . . . . .	221
14.4.4	Booting the base system from DASD and adding packages . . . . .	225
14.5	System configuration and administration . . . . .	228
14.5.1	File system layout . . . . .	228
14.5.2	Network setup . . . . .	229
14.5.3	Managing DASD . . . . .	231
14.5.4	Managing users . . . . .	232
14.5.5	Configure mail . . . . .	233
14.5.6	Using Apache . . . . .	234
14.5.7	Using FTP . . . . .	234
14.5.8	Using Samba . . . . .	234
14.5.9	Software package management . . . . .	234
<b>Chapter 15. Think Blue 64 Linux for zSeries . . . . .</b>		<b>239</b>
15.1	What's specific about Think Blue . . . . .	240
15.2	System requirements . . . . .	240
15.2.1	Installation worksheet . . . . .	241
15.2.2	Think Blue disk requirements . . . . .	242
15.3	Installation . . . . .	242
15.3.1	Preparing the first IPL of the install system . . . . .	243
15.3.2	Initial IPL and network setup . . . . .	244
15.3.3	Installation with tbsetup . . . . .	246

15.4 System configuration and administration . . . . .	249
15.4.1 File system layout . . . . .	249
15.4.2 Configure services/daemons . . . . .	250
15.4.3 Linuxconf . . . . .	251
15.4.4 Network setup . . . . .	252
15.4.5 Managing DASD . . . . .	254
15.4.6 Managing users . . . . .	255
15.4.7 Configuring the mail server . . . . .	256
15.4.8 Using Apache . . . . .	259
15.4.9 Using FTP . . . . .	259
15.4.10 Using Samba . . . . .	260
15.4.11 RPM . . . . .	260
15.4.12 Tape support . . . . .	261
15.4.13 Linux 2.4 kernel specific enhancements . . . . .	261
<b>Part 4. Other topics . . . . .</b>	<b>265</b>
<b>Chapter 16. VIF . . . . .</b>	<b>267</b>
16.1 Overview of VIF . . . . .	268
16.2 Planning for VIF . . . . .	269
16.2.1 VIF networking options . . . . .	270
16.2.2 VIF partitions and paging space . . . . .	271
16.3 Installation of VIF with Turbolinux . . . . .	271
16.4 Installation of VIF with Red Hat . . . . .	271
16.5 Installation of VIF with SuSE Linux . . . . .	271
16.5.1 Prepare an FTP server with the VIF install file and Linux . . . . .	272
16.5.2 Format a clean S/390 DASD . . . . .	276
16.5.3 Load VIF tape onto DASD . . . . .	278
16.5.4 IPL the DASD and answer VIF questions . . . . .	279
16.5.5 Begin master Linux installation from HMC . . . . .	281
16.5.6 Telnet to master Linux and finish installation . . . . .	283
16.5.7 Reset master Linux boot partition and reboot . . . . .	286
16.6 Using VIF . . . . .	287
16.6.1 Writing a simple v script . . . . .	288
16.6.2 Applying service to VIF . . . . .	288
16.6.3 Changing the Linux distribution . . . . .	288
16.6.4 Allocating image and paging space . . . . .	289
16.6.5 Installing a template Linux image . . . . .	289
16.6.6 Customizing the template Linux image . . . . .	292
16.6.7 Cloning the template Linux image . . . . .	295
16.6.8 Scripts to clone Linux images . . . . .	296
16.6.9 Shutting down VIF . . . . .	299
<b>Chapter 17. Logical Volume Manager . . . . .</b>	<b>301</b>

17.1	Problem support for LVM	302
17.2	LVM basics	302
17.3	Sample LVM session	303
17.3.1	Using the pvcreate and vgscan commands	303
17.3.2	Using the vgcreate command	304
17.3.3	Using the lvcreate command	304
17.3.4	Using the lvremove command	307
17.3.5	Using the lvextend command	307
17.3.6	resize2fs	307
17.4	Creating a 20 GB logical volume	308
17.5	Striping	310
17.6	Using pvmove to migrate PEs to other PVs	311
<b>Chapter 18. High Availability using clusters</b>		<b>315</b>
18.1	What is a Linux cluster	316
18.2	HA workload balancing Linux cluster for S/390	317
18.3	Linux virtual server architectures	318
18.3.1	Virtual server via network address translation	318
18.3.2	Virtual server via IP tunneling	320
18.3.3	Virtual server via direct routing	321
18.4	Scheduling algorithms	323
18.5	High availability testing	323
18.5.1	The Linux virtual server patch and test	324
18.5.2	Testing LVS under z/VM using tunneling	329
18.5.3	The mon tool and test	333
18.5.4	The IP takeover tool and test	339
18.6	Other tests needed and requirements	343
18.7	Using journaled file systems	343
18.7.1	Using IBM JFS	343
<b>Chapter 19. Debugging and problem diagnosis</b>		<b>347</b>
19.1	Linux diagnostic tools	348
19.1.1	objdump	348
19.1.2	strace	356
19.1.3	ulimit	358
19.1.4	gdb	359
19.1.5	uptime, top commands	362
19.1.6	System.map	363
19.1.7	ksymoops	364
19.1.8	Log files	364
19.1.9	dmesg	366
19.1.10	Standalone dump	368
19.2	VM diagnostic tools	368

19.3	HMC diagnostic facilities	369
19.4	Debugging for developers	369
19.4.1	strace	370
19.4.2	gdb	370
19.5	Support	370
19.5.1	Support Line	371
19.5.2	Getting support	371
19.5.3	Problem description	371
19.5.4	Diagnostic data	372
19.5.5	Data collection	372
19.5.6	Searching	373
<b>Chapter 20. LDAP</b>		<b>375</b>
20.1	What is LDAP	376
20.2	How does LDAP work	378
20.3	Using OpenLDAP	378
20.3.1	Centralized user account management	378
20.3.2	Setting up the OpenLDAP server	380
20.3.3	Migrating data to the LDAP database	383
20.3.4	Setting up the PAM-LDAP module	385
20.3.5	Possible deployment scenarios	386
20.4	Other considerations	392
20.4.1	Security	392
20.4.2	Referrals	393
20.4.3	Replication	393
<b>Chapter 21. System management tools</b>		<b>395</b>
21.1	Common Linux tools	396
21.1.1	The /proc file system	396
21.1.2	The ps command	397
21.1.3	The top command	397
21.1.4	The vmstat command	398
21.1.5	The df command	398
21.1.6	The du command	399
21.1.7	The netstat command	399
21.1.8	Quotas	400
21.2	Administration tools	402
21.2.1	YaST	402
21.2.2	Linuxconf	402
21.2.3	Webmin	403
21.2.4	AdminUX	408
21.3	Remote network management tools	408
21.3.1	snmp tools	409

21.3.2	Scotty/Tkined	411
21.3.3	Big Brother	414
21.3.4	Remstats	423
21.3.5	Other remote network management tools	432
<b>Chapter 22. Overview of security on Linux</b>		<b>435</b>
22.1	Security basics	436
22.1.1	Disable unneeded services	436
22.1.2	Use Secure Shell for remote access	440
22.1.3	Use shadow password utilities	447
22.1.4	Use the Pluggable Authentication Module (PAM)	450
22.1.5	Monitor security news and alerts	456
22.1.6	Use hardening tools	457
22.1.7	Integrate VM and z/VM security	460
<b>Chapter 23. Backup and restore</b>		<b>463</b>
23.1	Linux image backup and restore under VM	464
23.1.1	Backing up a Linux guest using DDR	464
23.2	Tape support under Linux for zSeries and S/390	467
23.2.1	Download kernel source, kernel patch, and lcs module	467
23.2.2	Extract kernel source and patch	468
23.2.3	Apply the patch	469
23.2.4	Recompile kernel	469
23.2.5	Copy the new kernel files	470
23.2.6	Replace the System.map	471
23.2.7	Extract the new LCS driver	471
23.2.8	Enable the new LCS driver	471
23.2.9	Update boot record and reboot to new kernel	471
23.2.10	Recovering from a bad compile	472
23.2.11	Create tape devices	472
23.2.12	Load the tape driver	473
23.2.13	Test the tape driver	473
23.3	Disaster backup and restore	473
23.3.1	Complete file system-based backup and restore	474
23.3.2	Complete file system backup and restore via Linux utilities	474
23.3.3	Complete file system backup and restore via TSM Client	475
23.4	Incremental backup and restore	479
23.4.1	Incremental file system-based backup and restore	479
23.4.2	Incremental file system backup and restore via Linux utilities	480
23.4.3	Incremental file system backup and restore via TSM Client	482
<b>Chapter 24. DB2</b>		<b>485</b>
24.1	DB2 UDB overview	486
24.2	Downloading DB2 UDB for Linux on S/390	486

24.3	Installing DB2 UDB on Linux	487
24.4	Verifying the UDB DB2 installation	488
<b>Chapter 25. WebSphere Application Server</b> . . . . . 491		
25.1	Downloading the evaluation version of WebSphere	492
25.2	Installing WebSphere	492
25.3	Upgrading WebSphere to Fixpack Level 3.5.4	495
25.4	Starting the Web server	496
25.5	Starting the WebSphere Application Server	498
25.6	Test the WebSphere Application Server	498
25.7	Setting up the WebSphere Samples Gallery	500
25.7.1	Configuring the database	501
25.7.2	Configuring Enterprise Java Beans	508
<b>Chapter 26. Conclusion</b> . . . . . 513		
26.1	Observations	514
<b>Appendix A. Commands and other references</b> . . . . . 517		
	Command syntax	517
	VIF commands	518
	Query commands	518
	HYPervisor commands	518
	IMAGE commands	518
	PARTition commands:	519
	Run levels	519
<b>Appendix B. Parameter file values</b> . . . . . 521		
	Condev - 3215 line mode terminal	522
	Syntax	522
	Example	523
	ctc - CTC/ESCON	523
	Syntax	523
	Examples	524
	dasd or dasd_mod - DASD	525
	Syntax	525
	Examples	526
	lcs - LAN channel station (OSA cards)	526
	Syntax	526
	Examples	527
	mdisk - VM/CMS minidisks	528
	Syntax	528
	Example	529
	netiucv - IUCV communications	529
	Syntax	529

Example .....	530
root .....	530
Syntax .....	530
Examples .....	530
xpram .....	531
Syntax .....	531
Examples .....	531
<b>Appendix C. Customizing Linux checklist .....</b>	<b>533</b>
<b>Appendix D. 64-bit Linux .....</b>	<b>535</b>
Building your first kernel .....	535
Building a file system from scratch .....	536
Relevant linux-390 appends .....	537
<b>Appendix E. ICKDSF bootstrap job .....</b>	<b>541</b>
<b>Related publications .....</b>	<b>547</b>
IBM Redbooks .....	547
Other resources .....	547
Referenced Web sites .....	548
How to get IBM Redbooks .....	549
IBM Redbooks collections .....	549
<b>Special notices .....</b>	<b>551</b>
<b>Index .....</b>	<b>553</b>





# Preface

This IBM Redbook describes the Linux distributions available for the mainframe. It will help you to install, customize, and maintain the following distributions:

- ▶ SuSE Linux Enterprise Server for S/390
- ▶ Turbolinux server for zSeries and S/390
- ▶ Red Hat Linux for S/390
- ▶ Marist File System
- ▶ Caiman Linux
- ▶ Think Blue Linux

We address installation and usage of Linux images on a logical partition (LPAR), under z/VM, and under the Virtual Image Facility (VIF). The following topics are also discussed:

- ▶ Managing DASD and file systems
- ▶ Logical Volume Manager (LVM)
- ▶ High Availability
- ▶ Debugging
- ▶ Lightweight Directory Access Protocol (LDAP)
- ▶ Systems management
- ▶ Security
- ▶ Backup and restore
- ▶ DB2
- ▶ WebSphere Application Server

Linux is an open source operating system kernel that was originally developed by Linus Torvalds in 1991. Linux distributions are created from the kernel, the GNU utilities, and additional software developed by thousands of programmers all over the world. One of the platforms now supported by Linux is S/390, which recently was given the new brand name, IBM @server zSeries, when the hardware moved from 32-bit to 64-bit. Such hardware is commonly known as the *mainframe*.

## 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.

**Michael MacIsaac** is a team leader for S/390 Redbooks and workshops at the ITSO Poughkeepsie Center. He writes about and teaches classes on Linux for S/390 and zSeries. Michael has worked at IBM for 14 years, mainly as a UNIX programmer. He has led teams that have written Redbooks on OS/390 UNIX, S/390 file and print serving, and Linux.

**Ronald Annuss** is an IT Specialist working for IBM in Germany. He holds a diploma in Geodesy from the Technical University Berlin. He has six years of experience working with Linux. After joining IBM in 1999, he was trained in S/390 hardware and software. He has worked extensively with Linux for S/390 since it became available in early 2000, and has worked on several Linux for S/390 projects with customers in Germany. He is an Red Hat Certified Engineer (RHCE).

**Wolfgang Engel** is a Technical Supporter working for SuSE in Germany. He has three years of experience in the IT field working with Linux. Before joining SuSE, he worked as a Network Engineer in the ISP area. After joining SuSE, he was trained in Linux on PowerPC and S/390. His areas of expertise include heterogeneous networking and support of Linux on various platforms.

**Chris Gombola** is an Advisory IT Specialist working for IBM in Bethesda, MD. He holds a degree in Business Information Systems from Shippensburg University. He has three years of experience in the Linux and S/390 fields. In addition to his focus on Linux for S/390, he is working on several server consolidation projects with customers.

**Timothy Nunes** is an e-business Technical Consultant working for the IBM Solution Partnership Center in Chicago, IL. He holds a degree in Liberal Studies from CSU Stanislaus. He has 13 years of experience working in Information Technology, six of which have been with IBM; he has been working with Linux specifically for over two years. His areas of expertise include computer networking and systems integration, as well as support for the Linux operating system on a variety of IBM e-server platforms.

**Mark Post** is a Senior Infrastructure Specialist working for EDS in Auburn Hills, Michigan, USA. He holds a Bachelors of Science degree in Computer Science from the University of Missouri - Rolla. He has 27 years of experience in IBM operating systems, including VM, MVS, and OS/390, as well as seven years of experience in with Microsoft operating systems, and three years of experience with Linux. His areas of expertise include operating systems installation and support, systems management, change and problem management, and cross-platform systems integration.

**Don Stubbs** is a Senior IT consultant associated with Albright Consulting. He consults both on information technology and the processes supporting its implementation and effective use. His current interest is in leveraging heterogeneous computing environments (OS/390, UNIX, and Windows) to achieve high performance, cost-effective computing infrastructures. He resides in Greensboro, NC.

**Egon Terwedow** is an OS/390 Technical Specialist working for the Ford Motor Company in Germany. He has more than 13 years of experience in the MVS arena, with expertise in OS/390 systems programming, configuration and performance management. For the Ford datacentres in the US and Europe, he recently set up a new interface to mainframe data using OS/390 Internet Technology and UNIX System Services. He participated in writing the IBM Redbooks *Enterprise Web Serving with the Lotus Domino Go Webserver for OS/390* and *Open Source Software for OS/390*.

**Craig Vernon** is a Senior IT Specialist in IBM Australia. He holds a Bachelor of Applied Science in Computing Sciences from the University of Technology, Sydney, and a Bachelor of Laws from Macquarie University. He has 20 years of experience in the IBM OS/390 and large systems. He has worked at IBM for 12 years. His areas of expertise include OS/390 software problem diagnosis, OS/390 installation, maintenance and customization, and RACF conversion. He has written on FICON implementation and software copyright issues.

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

Terry Barthel, Dave Bennin, Rich Conway  
International Technical Support Organization, Poughkeepsie Center

Bob Haimowitz  
International Technical Support Organization, Raleigh Center

Alan Altmark, Tung-Sing Chong, Romney White  
IBM Endicott

Erich Amrehn, Boas Betzler  
IBM Germany

Sándor Bárány  
IBM Austria

Dennis Wunder  
IBM Poughkeepsie

Marcus Kraft, Bernd Kaindl, Joachim Schroeder, Joerg Arndt  
SuSE Germany

Oliver Paukstadt  
Millenux, Germany

Jae-hwa Park  
LinuxKorea

Julia Karastoianova, Hervey Allen  
Turbolinux, Inc.

Lionel B. Dyck, Systems Software Lead  
Kaiser Permanente Information Technology

Thanks to the *many, many* people whose answers on the linux-390 list server helped make this a better book.

A special thanks to Neale Ferguson of Software AG for additional material for the 64-bit sections.

A special thanks to Jim Sibley of IBM for the OSA notes in the Planning chapter.

A special thanks to Rob van der Heij of IBM Netherlands for the ICKDSF bootstrap section.

A special thanks to Roy Costa of the Poughkeepsie ITSO for carving out VM IDs, minidisks and other resources at a moment's notice.

A very special thanks to Carlos Ordonez who wrote the High Availability chapter and contributed in other areas.



Also a very special thanks to Eva Yan who wrote the LDAP chapter and contributed in other areas.

## Special notice

This publication is intended to help S/390 systems programmers and system administrators to install and manage Linux for zSeries and S/390 systems. The information in this publication is not intended as the specification of any programming interfaces that are provided by Linux for zSeries and S/390. See the PUBLICATIONS section of the IBM Programming Announcement for Linux for zSeries and S/390 for more information about what publications are considered to be product documentation.

## IBM trademarks

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

e (logo)® 	Redbooks
IBM ®	Redbooks Logo 
AFS	Parallel Sysplex
AIX	PowerPC
APPN	RACF
AT	RAMAC
C/VM	RETAIN
CT	RMF
CUA	SAA
Current	S/370
DB2	S/390
DB2 Connect	SecureWay
DB2 Universal Database	Shark
ECKD	SP
Enterprise Storage Server	THINK
ESCON	VM/ESA
FICON	WebSphere
Hummingbird	xSeries
IMS	XT
Multiprise	z/Architecture
MVS	z/OS
Nways	z/VM
OS/2	zSeries
OS/390	400
	Lotus

## Comments welcome

Your comments are important to us!

We want our IBM 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](http://ibm.com/redbooks)
- ▶ Send your comments in an Internet note to:  
[redbook@us.ibm.com](mailto:redbook@us.ibm.com)
- ▶ Mail your comments to the address on page ii.





# Part 1

# Linux overview

In this part of the book, we introduce Linux for IBM @server zSeries and S/390.







# Overview

In this redbook we describe the Linux distributions available for S/390 and the new zSeries hardware. These systems are commonly referred to as the *mainframe*.

However, before we describe the status of Linux on the mainframe in late 2001, let's back up in time to the 1960s and follow a brief history of the development of Linux.

## 1.1 A brief history

In 1964, IBM announced the S/360 architecture. The “360” in the name referred to all points on the compass because it was designed to be universal. These systems have progressed with the brand names S/370, S/390, and now *zSeries*, which has a 64-bit architecture. Improvements in the hardware architecture were made with each generation; in turn, the operating system and application software were able to exploit these improvements to increase reliability, availability and serviceability (RAS), and performance.

In 1969, UNIX, TCP/IP and Linus Torvalds were born. UNIX and TCP/IP grew in the academic and scientific communities with a spirit of openness and cooperation. In the 1980s UNIX became more commercialized and a lot of the spirit of community appeared to be lost. Still, the mainframe, UNIX and TCP/IP have all flourished. They have proven to be reliable, resilient and long-lived, and are all universal.

In 1984, Richard Stallman, a systems programmer at the MIT AI lab, was incensed that he could not get the source code for the software of a new laser printer. Most people would complain, but being a man of principle, Richard quit his job and began writing GNU (an acronym for GNU's not UNIX) software. His goal was to write a completely free operating system. The GNU software collection today represents a large portion of the software necessary to run a Linux distribution.

In addition to the software was another, perhaps more important contribution, the GNU General Public License (GPL). This software license guaranteed everyone the freedom to use, copy, distribute and even sell the software. However, it prevented any restrictions from being added to the license. In addition, it required that new software using other software distributed under the GPL also be distributed under the GPL.

In 1991, Linus Torvalds, a graduate student at the University of Helsinki in Finland, posted the following to an Internet news group:

```
From: torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)
Newsgroups: comp.os.minix
Subject: What would you like to see most in minix?
Summary: small poll for my new operating system
Date: 25 Aug 91 20:57:08 GMT
Organization: University of Helsinki
```

```
Hello everybody out there using minix -
I'm doing a (free) operating system (just a hobby, won't be big and
professional like gnu) for 386(486) AT clones. This has been brewing
since april, and is starting to get ready. I'd like any feedback on
```

things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things).  
I've currently ported bash(1.08) and gcc(1.40), and things seem to work. This implies that I'll get something practical within a few months, and I'd like to know what features most people would want. Any suggestions are welcome, but I won't promise I'll implement them :-)  
Linus (torvalds@kruuna.helsinki.fi)  
PS. Yes - it's free of any minix code, and it has a multi-threaded fs.  
**It is NOT portable (uses 386 task switching etc), and it probably never will support anything other than AT-harddisks, as that's all I have**

The last statement is especially ironic because the Linux kernel has proven to be more portable than any other in history. It has been ported to many tens of computer architectures and platforms. The first port of the Linux kernel from Intel 386 architecture to the Motorola 68000 resulted in *two code trees*.

Linus considered this a serious fork. Problems or improvements to one code tree would probably have to be reflected in the other. When there was interest in a second port to the DEC Alpha processor, Linus rewrote the kernel in 1993 and 1994 to be more portable and to be maintained in a single source tree. On March 14, 1994, the Linux kernel 1.0 was released.

As more and more people on the Internet began to take an interest in the Linux kernel, it became clear that creating a Linux system which was able to boot a computer, use all of its resources in an efficient way, and include useful applications was a common, but also a complicated, task. In order to make it easier for the less computer-savvy, the concept of a *distribution* was born. Linux distributions were usually distributed on CDs or over the Internet as CD images. Some early distributions were available on floppy diskette as well. They contained the Linux kernel at the logical center, GNU essentials such as the compiler, (gcc) the runtime library (glibc) binary utilities (binutils) at the next layer, and the many, many applications around these essentials. The size of distributions continues to grow.

Today there are many distributions. For the zSeries and S/390 platform, there are three important distributions from the following companies:

SuSE	Founded by Roland Dyroff, Hubert Mantel, Thomas Fehr and Burchard Steinbild in September 1992
Turbolinux	Originally founded as Pacific HiTech by Cliff Miller in 1992
Red Hat	Founded by Robert Young and Marc Ewing in 1994

Before IBM worked on helping port the Linux kernel to S/390 hardware, there was another port named Bigfoot, largely done by Linus Vepstas. Though this project has been overshadowed by the size and manpower of the S/390 effort, it still is remarkable that Linux could be ported largely by one person. It is also important to give proper credit for some of the groundwork that was laid by this project. That project is described on the Web at:

<http://linas.org/linux/i370-bigfoot.html>

In 1998, a “skunkworks” project was started at the IBM lab in Boeblingen, Germany. Boas Betzler led a team that started with just a small number of people. Their philosophy was that Linux for S/390 would remain Linux; the structure, development and coding style would remain unchanged, and that S/390 would remain S/390; despite major differences from the PC, the hardware architecture is sufficient for implementing Linux.

This team now is doing much of the work on Linux for S/390. In December 1999, the Linux for S/390 patches were released to open source, and the changes continue to be incorporated into the Linux kernel.

Today the team has grown. At the time of the writing this redbook, the team listed on the developer works site includes Ingo Adlung, Alan Altmark, Utz Bacher, Denis J. Barrow, Reinhard Bündgen, Fritz Elfert, Stefan Hegewald, Michael Holzheu, Horst Hummel, Tuan Ngo-Anh, Carsten Otte, Carl B. Pedersen, Hartmut Penner, Martin Peschke, Jochen Röhrig, Martin Schwidewsky, Holger Smolinski, Gerhard Tonn, Ulrich Weigand and Dieter Wellerdiek.

On February 2, 2000 at LinuxWorld in New York City, Linus Torvalds said in the main keynote address: “Linux has even been ported to S/390. The support will be available in the 2.4 release. This is kind of interesting. When I began Linux, I did not envision it running on S/390. Actually, a year ago I did not even envision it.”

Though Linux has been ported to many different architectures, it can be argued that S/390 is among the important ones today. This assertion can be reinforced by looking at the directories in the Linux source tree arch (for architecture) directory.

```
# ls /usr/src/linux/arch
alpha cris ia64 mips parisc s390 sh sparc64
arm i386 m68k mips64 ppc s390x sparc
```

## Landscape

The mainframe is unsurpassed in reliability because of factors such as error correction and detection, transparent or hot memory and CPU chip sparing, channel I/O and the ability to phone home problems to repair personnel while remaining operational. The Linux kernel and the large amount of software freely available to create a Linux system are perceived to be very reliable due to the

relatively small size of the kernel, and the many thousands of developers able to analyze, improve upon and fix any defects. The combination of this hardware and software, along with the growing acceptance of the open source development model, results in a potent system with plenty of room for growth.

SuSE and Turbolinux have distributions that are generally available for zSeries and S/390, while Red Hat's distribution is still in beta. Other distributions available include Caiman from LinuxKorea Inc., Think Blue from Millenux Inc. and what might be more properly called a Linux for S/390 *file system* from Marist.

With all of these distributions, Linux for zSeries and S/390 is pure Linux. There is no EBCDIC to deal with and most open source software and other UNIX applications simply build (typically, moving software to Linux on the mainframe is just a recompile, not a *port*).

Linux on the mainframe is still relatively young and immature when compared with the more enterprise-ready OS/390 (now z/OS). However, Linux will continue to mature become a more critical player in the mainframe market for the following reasons:

- ▶ It is becoming increasingly accepted by software vendors.
- ▶ It has had a strong initial acceptance by system administrators.
- ▶ There are a large number of freely available, reliable server applications such as Apache, Samba, BIND, Sendmail and INND.
- ▶ It runs on all *@server* hardware and is fully backed by IBM.
- ▶ There is a growing pool of knowledgeable personnel and available support structures from the distributors and from IBM.

Additionally, the combination of z/VM and Linux on a mainframe gives the ability to create a *server farm in a box*. This will allow for server consolidation, which is especially important for large IT shops where the number of servers makes system administration difficult to control. Now that z/VM has a one time charge pricing model, it is more attractively priced than in the past.

We describe many Linux distributions in this book, but specifically do not recommend any. All have their strengths and weaknesses.

## Terminology

In this book we typically use the term *Linux* to refer to a Linux for zSeries and S/390 distribution, and when referring to the *Linux kernel*, we use that term. Sometimes the term *GNU/Linux* is used to describe a distribution. We appreciate all that Richard Stallman and the Free Software Foundation have done; however, the term *GNU/Linux* is somewhat cumbersome and not as commonly used as just *Linux*.

## 1.2 Systems used for this redbook

In this redbook we describe Linux for zSeries and S/390 in a general way. However, it may be helpful to understand the environment in which we worked during the writing stage; the majority of this book was written in May and June of 2001.

### 1.2.1 Processor

We had access to two logical partitions (LPARs) on a z900, machine type 2064, model 1C7. The first LPAR was used for Linux images under VM:

- ▶ 2 shared IFL engines
- ▶ 1280 MB of real memory
- ▶ 256MB of expanded memory
- ▶ OSA-Express Fast Ethernet in both non-QDIO mode (OSE) and QDIO mode (OSD)
- ▶ VM level: z/VM 4.1.0 (early ship program). We installed VM Host using the z/VM Express Installation and Services Procedures. This install includes.
  - Fully functional VM base system (CP, CMS, GCS)
  - Pre-installed products - (TCP/IP, RSCS, RTM, VMPRF, DIRMAINT)

We made modifications to the following z/VM attributes:

- SYSTEM CONFIG file on MAINT 1CF
- TCP/IP for local network features
- RSCS for local network

We enabled and configured DIRMAINT and DATAMOVE for use as our user directory maintenance. We added more SPOOL, PAGE and TDISK space to the system, as follows:

```
VMLPST 2050      1   1000 180000  91363 180000  50% PAGE
                1001  2000   1000     0     0  0% TDISK
                2001  3338 240840  94798 233484  39% SPOOL
```

Most VM user ID directories were similar to the following (though many systems had their storage set to a more reasonable size, such as 64 M):

```
USER VMLINUXA NEWPW4U 128M 512M G
    INCLUDE IBMDFLT
    ACCOUNT ITSOTOT
    IPL CMS PARM AUTOOCR
    MACH ESA 4
    POSIXINFO UID 14
    DEDICATE 2124 2124
    DEDICATE 2125 2125
    DEDICATE 2126 2126
    MDISK 0191 3390 471 50 VMLU1R MR
```

```
MDISK 0201 3390 0001 2600 LNX012 MR
MDISK 0202 3390 2601 2600 LNX012 MR
MDISK 0203 3390 5201 0100 LNX012 MR
MDISK 0204 3390 5301 0200 LNX012 MR
MDISK 0205 3390 5501 0200 LNX012 MR
MDISK 0206 3390 5701 0200 LNX012 MR
*DVHOPT LNK0 LOG1 RCM1 SMSO NPW1 LNGAMENG PWC20010514 CRC}É
```

The second LPAR was used for Linux “native” and VIF. It had the following resources:

- ▶ 2 shared IFL engines
- ▶ 1280 MB of real memory
- ▶ 256 MB of expanded memory
- ▶ OSA-Express Fast Ethernet in non-QDIO mode (OSE)
- ▶ VIF level: V1R1M0, service level 0500 (early ship program)

## 1.2.2 DASD

Most DASD volumes were based on a RAMAC Virtual Array (RVA), 9393-X82, formatted as 3390-9s. Some volumes were based on the Enterprise Storage Server (Shark), 2105-E20, formatted as 3390-3s. Some volumes were RAMAC, 9392-B23, formatted as 3390-3s.

## 1.2.3 Linux distributions

We worked with the following Linux distributions:

- ▶ SuSE - A development version that is planned to be an update to the GA version of SuSE Linux for S/390.
- ▶ Turbolinux - V6.0 (GA January 30, 2001) was initially used, then a beta of V6.5 became available.
- ▶ Red Hat - The public beta dated May, 2001, based on their 7.1 distribution.
- ▶ Caiman - The public beta dated May, 2001 from LinuxKorea.
- ▶ Marist - The public version updated May 1, 2001.
- ▶ Think Blue - V 7.1 from Millenux Inc.







## Planning for Linux

In this chapter we describe what planning should be done before installing Linux for zSeries or S/390. To plan for Linux, you must first think hardware, then networking, then distribution and software.

## 2.1 Planning hardware

In general, we recommend that Linux be run on a G2 to G6 9672, a zSeries 900 (machine type 2064) or later, or a Multiprise 3000. However, if applications are to be run with heavy use of floating point calculations, a G5 9672 or later is recommended. Table 2-1 shows the machines that will run Linux.

Table 2-1 Processor table

Machine	Pertinent attributes	Notes
G1 or earlier (9672-Rn1, 9672-Enn, 9672-Pnn)	31-bit, no CSP or relative instructions, no IEEE FP	“Vintage patch” (see linux-390 mailing list) is required; poor performance due to additional layers
P/390, R/390, Integrated Server	P/390 or P/390E card on a RISC or Intel-based PC server	Relatively poor performance
Multiprise 2000 (2003- nnn)	31 bit, no IEEE Floating Point	Poor performance with applications that heavily use floating point
G2 (9672-Rn2, 9672-Rn3) G3 (9672-Rn4) G4 (9672-Rn5)	31 bit, no IEEE Floating Point	Performance issues with applications that heavily use floating point
G5 (9672-Rn6, 9672-Yn6, 9672-Tn6) Multiprise 3000 (7060- nnn) G6 (9672-Xn7, 9672-Zn7)	31 bit, IEEE Floating Point	
z900 (2064)	31 or 64 bit, IEEE Floating Point	

Linux can run in a number of configurations: on the mainframe natively; in a logical partition (LPAR); under z/VM; under the Virtual Image Facility (VIF); or in an emulated environment. Table 2-2 shows some considerations. Emulated environments are not discussed in this redbook.

Table 2-2 Platforms you can run Linux on

Platform	Reasonable number of Linux images	Notes
Native	1	With the exception of the smaller platforms such as the P/390, this is generally not economically feasible.

Platform	Reasonable number of Linux images	Notes
LPAR	15	Useful if a very small number of Linux images is needed for testing or development
VIF	Tens to hundreds	Management of Linux images becomes difficult when greater than a few tens.
z/VM	Hundreds to thousands	Probably the most cost-effective way to run and manage Linux on the mainframe.
Simulated	1	Flex-ES or other simulators; relatively poor performance, however, the only option when no zSeries or S/390 hardware is available.

### 2.1.1 Integrated Facility for Linux (IFL)

The heart of zSeries and S/390 hardware is the Multi-Chip Module (MCM) which contains up to 20 Processing Units (PU), commonly referred to on other platforms as CPUs or *engines*. All PUs are identical, but can take on different roles in the system. Each PU can be one of:

- ▶ A central processor (CP), sometimes called a *standard engine*
- ▶ A System Assist Processor (SAP)
- ▶ An Internal Coupling Facility (ICF) for Parallel Sysplex
- ▶ An Integrated Facility for Linux (IFL), sometimes called an *IFL engine*

The IFL engine is a new type of PU which allows additional engines to be dedicated to z/VM or Linux workloads. It is lower priced than standard engines and allows traditional S/390 software charges to remain the same.

### 2.1.2 Planning DASD

A disk drive on the mainframe is called a direct access storage device (DASD). As with many other entities, the acronym (usually pronounced “dasdee”) is commonly used as a word. The types of S/390 DASD are referred to by their machine type and model number. A machine type of 3380 is seen occasionally, but most commonly a machine type of 3390 is seen.

For 3390s, there are four possible model numbers: 1, 2, 3 and 9. Models 3 and 9 are seen most commonly and are sometimes called a *pack* or *volume*. A 3390 model 3 (or more simply, a 3390-3) has a capacity of approximately 2.8 GB, but when formatted for Linux, that is reduced to about 2.3 GB. A 3390 model 9 has a capacity of about 8.4 GB, but when formatted, that is reduced to about 7.0 GB.

The data format of S/390 DASD is called Extended Count Key Data (ECKD); it differs from most other disk drives, which have Fixed Block Architecture (FBA). In the past, each of these DASD was a large physical device, but today they are emulated by disk arrays with much larger capacities such as RAMAC, RAMAC Virtual Array (RVA), and the Enterprise Storage Server (ESS) commonly referred to as *Shark*.

Because a 3390-9 is the largest capacity DASD, 7 GB is the largest partition that can be allocated for a Linux system. By today's standards, this is a relatively small capacity. Alternatives are needed to mass multiple DASD together in order to bypass the 7 GB limit. The most commonly used and recommended method is the Logical Volume Manager (see Chapter 17, "Logical Volume Manager" on page 301).

Besides not being able to have very large file systems, the inability to partition DASD can also be a limitation. When Linux is running under z/VM or VIF, this problem is overcome by the ability to create minidisks or partitions, respectively. If Linux is running natively or in an LPAR, the inability to create a partition can be a problem. Thankfully this issue is being addressed by Linux for zSeries developers with the addition of a new `fdasd` command. See 2.4, "Roadmap" on page 26 for more details.

UNIX and Linux use the concept of major and minor numbers associated with devices. Major numbers are assigned to each *type of device* and minor numbers are assigned to *instances of each type of device*. Minor numbers are stored in a single byte, so it is only possible to have 256 devices of a specific type for that major number.

To make matters worse on S/390, four minor numbers are allocated for each DASD: the first for the whole *volume*, and then three for each *partition* on a DASD. To avoid the low maximum of 64 DASD volumes ( $256/4$ ), major numbers can be dynamically allocated at boot time or when the DASD module is loaded. The *well-known* major number for S/390 DASD is 94, and if more than 64 are found, major numbers are dynamically allocated beginning at a larger number, for example, 235.

Every DASD has a unique device address. The Linux kernel auto-senses attached devices at boot time, or when a machine check is presented due to an important hardware event. All DASD type devices come to a specific pool and the kernel identifies all of them. This information is available from the virtual proc file system in the file named `/proc/dasd/devices`. This file shows:

- ▶ The DASD address
- ▶ The allocated major and minor numbers
- ▶ The device file name
- ▶ The status of the DASD

- ▶ Whether the DASD is accessible
- ▶ The block size, number of blocks and total size of the DASD

Following is an example of a Linux system with three DASD. Because of its large size, you can tell that the DASD at address 204 (/dev/dasdd) is a 3390-9:

```
$ cat /proc/dasd/devices
0203(ECKD) at (94:0) is dasda:active at blksize: 4096, 36000 blks, 140 MB
0202(ECKD) at (94:4) is dasdb:active at blksize: 4096, 204840 blks, 800 MB
0201(ECKD) at (94:8) is dasdc:active at blksize: 4096, 360000 blks, 1406 MB
0204(ECKD) at (94:12) is dasdd:active at blksize: 4096, 1802880 blks, 7042 MB
```

You cannot use the S/390 device address in Linux except with the **-n** flag to the **dasdfmt** command. Usually you have to specify your DASD via the corresponding /dev/dasdX file.

Some of the limitations discussed here are being addressed. As the Linux kernel and the S/390 additions to the code mature, Linux distributors will pick this function up. See 2.4, “Roadmap” on page 26 for a short discussion of what may be coming.

### 2.1.3 Planning for tape devices

The tape driver for zSeries and S/390 hardware is not yet built into Linux distributions (the Turbolinux v6.5 beta does have a tape driver, however). In order to use a tape driver at the time of writing of this redbook, we had to rebuild a 2.2.19 Linux kernel; see 23.2, “Tape support under Linux for zSeries and S/390” on page 467.

## 2.2 Planning networking

The types of networking that Linux on zSeries and S/390 can do are either physical or virtual.

### 2.2.1 Physical networking

Physical networking is typically done through one or more flavors of the Open Systems Adapters (OSA), or equivalent hardware; see 2.2.3, “Linux OSA cards and drivers” on page 16. Physical channel-to-channel networking can also be done.

## 2.2.2 Virtual networking

Virtual networking can be performed on some of the platforms under which Linux is run. z/VM can perform virtual networking via virtual channel-to-channel (vCTC) communications or the Inter-User Communication Vehicle (IUCV). VIF can perform virtual networking via IUCV only.

Both virtual and physical channel-to-channel networking use the same driver (ctc.o). IUCV networking uses the IUCV driver (iucv.o). Both of these drivers are distributed with Linux distributions.

## 2.2.3 Linux OSA cards and drivers

The drivers for OSA cards are object code only (OCO). There two sets of Linux IBM OCO drivers to support OSA cards:

- ▶ LAN Channel Station (LCS) Ethernet and Token Ring support normal ESCON protocols. The file name of the driver is lcs.o.
- ▶ Gigabit Ethernet (GBE). QDIO is a new S/390 architecture that allows an OSA-Express Gigabit Ethernet feature to communicate directly with system memory through the use of queues and a never-ending channel program. There are two drivers used to support these cards:
  - Queued Direct I/O (qdio) protocol driver - the file name is qdio.o.
  - Gigabit Ethernet driver the file name is qeth.o.

The latest versions of the OCO lcs, qdio and qeth modules are found at the IBM developers works download site (OCO Download). The modules must be untarred, and then copied and renamed to `/lib/modules/<kernel_version>/net` (where `kernel_version` is a number such as 2.2.16) as `qdio.o` and `qeth.o`. If you do these tasks *before* a kernel build, the `depmod -a -v` command should build the correct `/lib/modules/<kernel_version>/modules.dep` file.

## HCD and IOCDs considerations

Generally, 16 addresses for each OSA can be coded and the last entry at xxFE must be “genned” for OSA/SF as OSAD to read any of the cards or update the OSA2 cards. This number of addresses may vary according to the number of IP addresses you need for an LPAR.

Table 2-3 IOCDs code samples

Card	IOCDs code
OSA2 ENTR or OSA2 FE	CHPID PATH=(00),SHARED, PARTITION=((STLBE1F,STLNXEAF)), <b>TYPE=OSA</b> * CNTLUNIT CUNUMBR=2000,PATH=00,UNIT=OSA IODEVICE ADDRESS=(2000,016),CUNUMBR=(2000),UNIT=OSA IODEVICE ADDRESS=(20FE,001),CUNUMBR=(2000),UNIT=OSAD
OSAX-FE (lcs.o)	CHPID PATH=(FD),SHARED, PARTITION=((STLBE1F,STLNXEAF)), <b>TYPE=OSE</b> * CNTLUNIT CUNUMBR=4000,PATH=(FD),UNIT=OSA IODEVICE ADDRESS=(4000,016),CUNUMBR=(4000),UNIT=OSA IODEVICE ADDRESS=(40FE,001),CUNUMBR=(4000),UNIT=OSAD
OSAX-FE (qdio.o/qeth.o)	CHPID PATH=(FE),SHARED, PARTITION=((STLBE1F,STLNXEAF)), <b>TYPE=OSD</b> * CNTLUNIT CUNUMBR=5000,PATH=(FE),UNIT=OSA IODEVICE ADDRESS=(5000,016),CUNUMBR=(5000),UNIT=OSA IODEVICE ADDRESS=(50FE,001),CUNUMBR=(5000),UNIT=OSAD
OSAX-GB (qdio.o/qeth.o)	CHPID PATH=(FF),SHARED, PARTITION=((STLBE1F,STLNXEAF)), <b>TYPE=OSD</b> * CNTLUNIT CUNUMBR=6000,PATH=(FF),UNIT=OSA IODEVICE ADDRESS=(6000,016),CUNUMBR=(6000),UNIT=OSA IODEVICE ADDRESS=(60FE,001),CUNUMBR=(6000),UNIT=OSAD

## Types of OSA cards

Currently, there are two flavors of OSA cards: OSA2 and OSA-Express. The OSA-Express cards can only be installed in S/390 G5 or above. The zSeries OSA-Express (Hydra 1.5 cards) are functionally the same as two (2) G5+ cards (Hydra 1). These abbreviations are used to describe the cards in the sections that follow:

OSA2-ENTR	OSA2 Ethernet/Token Ring
OSA2-FE	OSA2 Fast Ethernet
OSAX-FE	OSA Fast Ethernet Express
OSAX-GB	OSA Gigabit Express

When an OSA Express card can be used in QDIO mode (TYPE=OSD in the IOCDS), it is much more efficient than when it is in non-QDIO mode (TYPE=OSE in the IOCDS). QDIO mode is shown in Figure 2-1 on the right side; it is clearly a more streamlined communication path. A rule of thumb is that this mode will perform 20% better than non-QDIO mode.

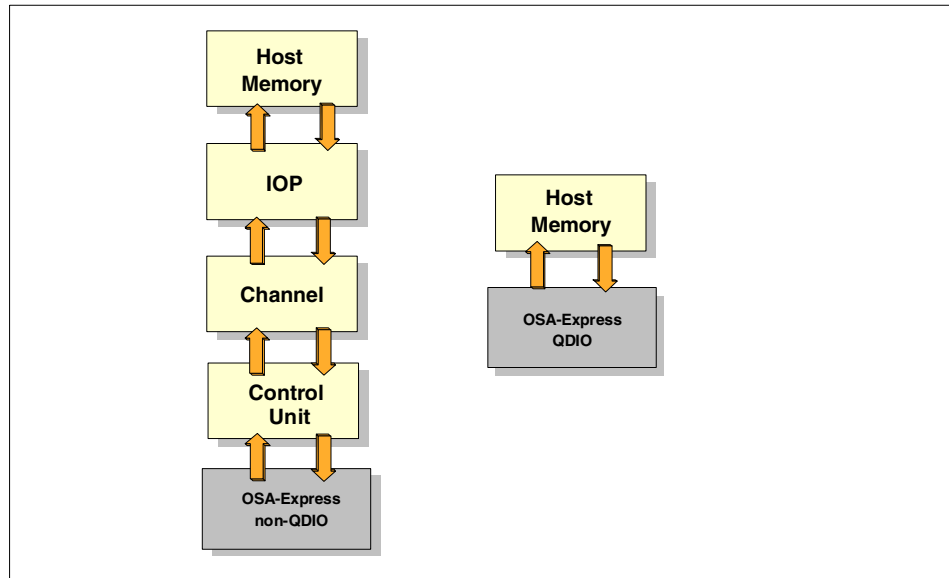


Figure 2-1 OSA Express in non-QDIO mode vs. QDIO mode

Table 2-4 shows information for OSA cards for 9672 machines G6 and earlier. OSA Express cards (OSAX) can only be used in G5 or G6 machines.

Table 2-4 S/390 OSA cards for G6 and earlier

Attribute	OSA2-ENTR	OSA2-FE	OSAX-FE (OSE)	OSAX-FE (OSD)	OSAX-GB (OSD)
Connection	Ethernet or Token Ring	Ethernet	Ethernet	Ethernet	Ethernet
Mb/sec throughput	10 (ethernet) or 16 (T/R)	100	100	100 (200 if full duplex)	1000
Cable	Copper	Copper	Copper	Copper	Fiber optic
HCD/IOCDS device type	OSA	OSA	OSE	OSD	OSD
Ports/CHP	2	1	1	1	1



Attribute	OSA2-ENTR	OSA2-FE	OSAX-FE (OSE)	OSAX-FE (OSD)	OSAX-GB (OSD)
Multiple IP address support	OSA/SF	OSA/SF	Automatic	Automatic	Automatic
Max IP addresses*	16	16	16	240	240
Access IP addresses on other LPARs?	Yes	No	Yes	Yes	Yes
Linux device driver	lcs.o	lcs.o	lcs.o	qdio.o, qeth.o	qdio.o, qeth.o
* Many more IP addresses are possible in non-shared mode with VM TCP/IP or Linux hub					

Table 2-5 shows information for OSA cards for zSeries machines (machine type 2064). All cards use one port per channel path (CHP), but two channel paths per card.

Table 2-5 zSeries OSA cards

Card (mode)	OSAX-FE (OSE)	OSAX-FE (OSD)	OSAX-GB (OSD)
Connection	Ethernet	Ethernet	Ethernet
Mb/sec throughput	100	100 (200 if full duplex)	1000
Cable	Copper	Copper	Fiber optic
HCD/IOCDS CHP type	OSE	OSD	OSD
Ports/CHP	1	1	1
Max IP addresses	32	480	480
Linux device driver	lcs.o	qdio.o, qeth.o	qdio.o, qeth.o

## Options in /etc/modules.conf

### *lcs* parameters

The addresses are in even-odd pairs, but you only have to specify the even addresses. Only one *lcs* parameter should be present. Multiple parameters would be coded on the same line. Linux detects whether the card is T/R or Ethernet and it assigns the interface name sequentially, as appropriate to the card (tr0, tr1, eth0, eth1, etc.).

1 chp, 1 port/chp	options lcs noauto=1 devno_portno_pairs=0x2000,0
1 chp, 2 ports/chp	options lcs noauto=1 devno_portno_pairs=0x2000,0,0x2000,1
2 chp, 1 ports/chp	options lcs noauto=1 devno_portno_pairs=0x2000,0,0x3000,0

### ***qdio/qeth parameters***

The addresses are in quadruplets and start on *even* address boundaries. You code the first three addresses. Only one qeth parameter should be present. If no interface name is present, it assumes eth0. It is probably safer to code the explicit interface name. The qeth module has a dependency on the qdio module in /lib/modules/2.2.16/modules.dep, so both modules are loaded when qeth is loaded.

1 chp	options qeth qeth_options=noauto,0x4000,0x4001,0x4002,eth0
2 chp	options qeth qeth_options=noauto,0x4000,0x4001,0x4002, eth0,0x5000,0x5001,0x5002,eth1

### **Relating interface with OSA card and driver parameters**

The network is generally brought up with a single script, for example, /etc/rc.d/network. This script needs to tie the following four items together:

- ▶ The module name
- ▶ The module parameters
- ▶ The device addresses
- ▶ The TCP/IP parameters

They are tied together with the interface name from the **ifconfig** command (i.e. "eth0" or "tr0").

In the file /etc/modules.conf, the line

```
alias eth0 lcs
```

relates the interface name (eth0) with the module name (lcs.o). Also, the line

```
options lcs
```

relates the device address and card options with the module name.

In the file /etc/rc.config, the line:

```
NETCONFIG="_0 _1"
```

defines the subscripts for TCP/IP devices (in this case there are two: "\_0" and "\_1"). The lines with IPADDR specify the IP address to use for each interface. For example:

```
IPADDR_0="9.112.50.115"  
IPADDR_1="10.32.56.119"
```

The line with NETDEV assigns the interface names. For example:

```
NETDEV_0="tr0"  
NETDEV_1="eth0"
```

The lines with IFCONFIG then specify the parameters for the `ifconfig` command. For example:

```
IFCONFIG_0="9.112.50.115 broadcast 9.112.50.255 netmask 255.255.255.0 up"  
IFCONFIG_1="10.32.56.119 broadcast 10.32.56.255 netmask 255.255.255.0 up"
```

## Native IPL

If you IPL native, your parameter file must have an `ipldelay` of at least 3 minutes (`ipldelay=3m`) for the OSA cards to stabilize. If you IPL as an LPAR, this is not necessary. Remember to use the `sil0` command if you change any of these parameters (see “Command syntax” on page 517 for the syntax).

## Resetting an OSA card

Once an OSA is “hung” in an LPAR, you have to re-IPL that LPAR. However, sometimes the IPL will not fix the hang, so you must reset the OSA card CHP. Resetting an OSA card or configuring the CHPID off/on may cause the Linux LPARS with OSA card genned to hang, whether you have specified it in `/etc/modules.conf` or not! The May 5, 2001 version of the `qdio/qeth` drivers may hang during IPL in LPAR mode with 100% CPU busy. IPL the machine with the `clear` option to avoid this. Unfortunately, this precludes `shutdown -r` or `reboot`. In most shops, resetting the OSA card is handled by field support for the processor. Some shops allow the systems programmer to do this procedure. If you are going to do this, you must first know the CHPID for the OSA card you want to reset. Then OSA cards can be displayed and reset from the HCD, as follows:

1. Click **Defined CPCs**.
2. Highlight the CPC you want.
3. Click **Single Object Operations**.
4. Click **CPC**.
5. Right-click **CPC** to get CHPIDs.
6. Use Advanced Facilities to view, set the parameter on OSA2 cards, and reset the cards, as follows:
  - a. All cards should be full duplex.
  - b. OSA2-ENTR - 10 Mbs.

- c. All others 100 Mbs.
- 7. After you have finished in Advanced Facilities, toggle the CHPIDs off, then on under Configure ON/OFF.
- 8. Log out of Single Object Operations when you're done.

Refer to *OS/390 OSA/SF User's Guide*, SC28-1855, for more information.

## 2.2.4 IBM 2216

The 2216 is another network interface you can use to connect your Linux system. The 2216 Nways Multiaccess Connector functions as a host gateway for SNA/APPN and TCP/IP applications and devices attached to LANs, WANs, and ATM. The 2216 Model 400, which has eight slots for network adapters and one for a system card, supports two different channel adapters for channel attachment: an ESCON channel adapter and a Parallel channel adapter.

Of the eight slots available for network adapters in the 2216, up to four channel adapters for ESCON/parallel or a mixture of ESCON and PARALLEL can be used. The ESCON channel adapter can attach directly to the mainframe ESCON channel or an ESCON director.

## 2.2.5 CTC

The channel-to-channel function simulates an I/O device that can be used by a system control program to communicate with another system control program. It provides the data path and synchronization for data transfer between two channels. When the CTC option is used to connect two channels that are associated with different systems, a loosely coupled multiprocessing system is established. The CTC connection, as viewed by either of the channels it connects, has the appearance of an unshared I/O device.

Channel-to-channel (CTC) support exists for:

- ▶ Parallel channels via 3088 Multisystem Channel Communication Unit (MCCU)
- ▶ ESCON channels
- ▶ FICON channels via ESCON Director with the FICON Bridge Feature

## 2.2.6 TCP/IP calculations

To obtain the *network address* or *network IP address*, the network mask (netmask) is logically ANDed with the Host IP address.

To obtain the *broadcast address* or *broadcast IP address*, the network IP address is logically eXclusively ORed with the netmask and the result is logically NOTted to derive the broadcast IP address.

## 2.2.7 Maximum Transmission Unit (MTU) sizes

The Maximum Transmission Unit (MTU) is the maximum number of bytes that can be transferred across a given physical network. When a datagram to be sent comprises more bytes than the MTU, IP performs fragmentation, breaking the datagram up into smaller pieces. If any bridge or router in the network infrastructure does not perform IP-layer fragmentation of packets, you must select an MTU size of the smallest MTU in use by that router. Selecting an MTU size that is too large may cause client applications to hang. As you can see, there are many factors involved and this is a case where you really do need to talk to your network administrator. For reference, here is a table of MTU sizes for various types of networks, which can be used as guidelines:

Ethernet 802.2/802.3	1492
Ethernet Version 2 IEEE	1500
Token ring	2000 or 4000
FDDI	2000 or 4000
CTC	65527
CLAW	4096

**Attention:** You have to use 1492 for an MTU when using Ethernet and running Linux on a Multiprise 3000. The Ethernet adapter uses IEEE 802.3 networking.

If it appears that networking has been properly set up on your Linux system, but then you see very long transmission delays or network time-outs, you might suspect an MTU size problem. A rudimentary test for determining MTU size problems is to **ping** a machine on a different subnet with a packet size larger than the default of 56 bytes (using the **-c** flag), for example, 4096 bytes:

```
$ ping -c4 9.12.0.88
PING 9.12.0.88 (9.12.0.88): 56 data bytes
...
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 6.784/6.864/6.937 ms
$ ping -c4 -s4096 9.12.0.88
PING 9.12.0.88 (9.12.0.88): 4096 data bytes
...
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 12.591/13.731/14.999 ms
```

Here we see that a packet size of 4096 bytes takes about twice as long to get to another machine, but there are no MTU size problems. If there had been problems, the ping would have timed out.

## 2.3 Planning for Linux distribution and software

While planning for hardware might seem challenging, planning a Linux distribution and any additional software to be added and configured is probably harder. The remainder of this redbook is devoted to covering these topics.

In this section we offer a general discussion on file systems.

### 2.3.1 Planning file systems

The term *file systems* is used to refer to a few different concepts. Let us classify file systems into:

- ▶ Conventional
- ▶ Journaled
- ▶ Virtual
- ▶ Swap

#### Conventional file systems

Conventional file systems are limited. They can only be contained on a single raw device, and the file size is limited by the device size. An example of a conventional file system (and the most commonly used file system in Linux) is ext2.

To check and repair a conventional file system, the command `fsck` must be run against it. With small partitions, this process is reasonable; however, with large partitions, it can become prohibitively slow. Therefore, a more sophisticated file system must be used.

#### Journaled file systems

A journaled file system is one in which changes are recorded or *journalled*. Most modern UNIX operating systems use journaled file systems, and Linux is certainly moving in that direction. However, while the conventional second extended file system (ext2) is clearly the de facto standard on Linux, there is no such standard with journaled file systems. We investigated the Reiser file system (reiserfs) and the IBM Journaled File System (JFS).

### ***The Reiser file system***

The Reiser file system (reiserfs) is one of the most well-known journaling file systems that works on Linux. Unfortunately, during the residency we discovered that the official source tree does not contain support for big-endian architectures such as the S/390, POWERpc, etc. (Big-endian architectures store the four bytes of an address in the same order, lowest to highest, as the network byte order, or that which is sent across the network. With little-endian architectures, such as Intel, the lowest memory address contains the low order byte, and the order must be reversed to be sent over the network.)

Jeff Mahoney has written patches to reiserfs to make it “endian safe.” These patches have not yet been merged into the official source tree. Until that time, we *cannot* recommend the use of reiserfs on Linux. More information, including the endian-safe utilities, can be found at:

<http://www.suse.com/~jeffm/>

### ***The IBM Journaled File System (JFS)***

JFS is the journaling file system from IBM’s AIX operating system. It was made open source sometime last year and IBM started a project to port it to Linux. Beta 1 was released in December 2000, Beta 2 in March 2001, and Beta 3 in April 2001. The first release of the JFS was made available on June 28, 2001. The project team workign on porting to Linux has focused its efforts on the 2.4 kernel, although patches can be found for the 2.2.x version of the Linux kernel. Version 1.0.0 of the JFS can be found at:

<http://oss.software.ibm.com/developerworks/opensource/jfs/>

See 18.7, “Using journaled file systems” on page 343 for details on using this file system.

### **Virtual file systems**

*Virtual file systems* are those which are not actually written to disk. The prime example of a virtual file system is the /proc file system. It contains data structures used by the kernel and is available to the user via this file system.

Another virtual file system, the /dev file system, has been added to the Linux 2.4 kernel.

### **Swap file systems**

Setting the size of a swap file can cause vigorous debates, because there is no one right answer. Less is sometimes more because Linux will use swap space aggressively, and keeping the memory size of Linux images small on Linux for S/390 is important.

One approach is to start with a very small swap size, and if your system begins to swap, analyze the workload, memory size and swap size; and address swap file systems at that time.

This issue is discussed in more detail in the IBM Redbook *Linux for zSeries and S/390: ISP/ASP Solutions*, SG24-6299.

## 2.4 Roadmap

On June 29, 2001, a new code drop was put out on the Linux for S/390 developerworks site. The items of importance are:

- ▶ 2.4 kernel and the /devfs file system
- ▶ Ability to partition DASD with the new **fdasd** command
- ▶ Common disk layout (CDL)

This code drop was not incorporated into any of the Linux distributions at the time of writing of this redbook; however, it is anticipated that this new function will be incorporated into Linux distributions soon.





# Preparation

In this chapter we discuss the preparation of the mainframe system that is common to all distributions. We first consider Linux under VM, then under an LPAR.

## 3.1 Prepare the VM guest for Linux

This section briefly describes the basics of what is required to define a new guest to VM. For further details, and an in-depth treatment of the subject, refer to VM/ESA publications on planning and administration.

### 3.1.1 The VM user directory

The VM user directory will need entries or statements added to compose a definition for each Linux guest. Linux guests are similar to other operating system guests under VM in that they require a name, disk space, storage, and other devices.

The VM user directory resides in the file called USER DIRECT, and requires you to log as MAINT to have the authority required to edit and process this file.

The following example defines a guest called VMLINUX7 with a password of NEWPW4U with 64 MB of storage. CMS will be IPLed automatically and the machine will run in ESA mode. There are 2 dedicated devices, which in fact are OSA cards, and there are 7 minidisks. Notice that 6 are on the same volume, LNX009, and are mapped by cylinder. A beginning cylinder number is followed by the size in cylinders. Care has to be taken not to overlap these minidisks.

The MDISK parameters (in sequence) are: virtual device; device type; beginning cylinder; size in cylinders; volume name; mount attributes. The other statement of interest is the INCLUDE, which eliminates duplication of statements that are going to be common to all guests by referencing a PROFILE definition.

```
USER VMLINUX7 NEWPW4U 64M 64M G
  INCLUDE IBMDFLT
  ACCOUNT ITSOTOT
  IPL CMS PARM AUTOCR
  MACH ESA 4
  DEDICATE 292C 292C
  DEDICATE 292D 292D
  MDISK 0191 3390 321 50 VMLU1R MR
  MDISK 0201 3390 0001 2600 LNX009 MR
  MDISK 0202 3390 2601 2600 LNX009 MR
  MDISK 0203 3390 5201 0100 LNX009 MR
  MDISK 0204 3390 5301 0200 LNX009 MR
  MDISK 0205 3390 5501 0200 LNX009 MR
  MDISK 0206 3390 5701 0200 LNX009 MR
```

The following is an example of a profile definition.

```
PROFILE IBMDFLT
  SPOOL 000C 2540 READER *
```

```
SPool 000D 2540 PUNCH A
SPool 000E 1403 A
CONSOLE 0009 3215 T
LINK MAINT 0190 0190 RR
LINK MAINT 0191 0191 RR
LINK MAINT 019E 019E RR
```

The profile definition in this example shows the devices and disks that, typically, every VM guest will need. The devices 0190, 0191 and 019E are to be accessed as read-only and typically hold the system commands and execs used by all VM guests.

### 3.1.2 Processing the VM user directory

The USER DIRECT file can be easily edited using the XEDIT command. Once any updates are complete, run the following commands to process them:

```
DISKMAP USER
```

This will check for any overlaps in all the minidisk definitions. If any are found, they are placed in a file called USER DISKMAP, along with other information.

Use these commands to find information on any overlaps created.

```
X USER DISKMAP
```

```
/overlap
```

Once these overlaps have been corrected, use the following command to syntax-check the definitions in the user direct file.

```
DIRECTXA USER (EDIT
```

If there are any syntax errors, this step will allow them to be identified and corrected. When all are correct, the following command will implement the definitions:

```
DIRECTXA USER
```

This loads the updated definitions to VM, and on its completion, the new guests may be logged on.

### 3.1.3 Editing the PROFILE EXEC and XEDIT PROFILE

It's a good idea to add the following commands to your Linux/390 guests' profile exec. The first command will allow you to disconnect the VM console from the Linux/390 guest and reconnect to it later, without putting it into a CP READ state.

The next two commands allow you to use the PF12 key to “recall” or “retrieve” previously entered commands. You can then re-issue the commands without having to completely retype them:

```
'CP SET RUN ON'  
'CP SET PF12 RETRIEVE'  
'CP SET RETRIEVE MAX'
```

The kernel boot parameter file is *case sensitive*. If you edit it without putting XEDIT into mixed-case mode, any line you modify in any way will be converted to all uppercase letters. (Many times this goes unnoticed until the IPL fails in some way.) To automatically put XEDIT into mixed-case mode every time it is invoked, create a file on your 191 A disk named XEDIT profile, and put in this line:

```
'SET CASE M I'
```

### 3.1.4 Getting the installation files for VM

Every distribution includes three files that are needed to IPL from the VM virtual card reader. The names of the files will differ, but how you get them to the reader will not. The (generic) files are:

- ▶ A kernel image to be IPLed from the VM reader
- ▶ A parameter file to be used when IPLing from the reader
- ▶ A RAMdisk which contains the installation scripts.

Most distributions also include a REXX EXEC to “punch” the files to your VM reader, then IPL from the reader. You will need to copy the three critical files and, optionally, the installation EXEC to the VM user ID that will become your Linux/390 guest. An easy way to do this is with the VM/CMS FTP client (on the TCPMAINT 592 minidisk). The important files on our Red Hat distribution are:

The installation exec	inst.exec
The kernel parameter file	inst.parm
The installation ramdisk	vminrd22.bin
The installation kernel	vmkrnl22.bin

We show the important FTP commands to get the Red Hat startup files from an FTP server to VM:

```
ftp linux4.itso.ibm.com  
...  
ftp> cd /home/redbooks/linuxSystems/RedHat/boot  
ftp> locsite fix 80  
ftp> get inst.exec  
ftp> get inst.parm  
ftp> binary  
ftp> get vminrd22.bin
```

```
ftp> get vmkrnl22.bin
ftp> quit
```

Because each of the ftp get subcommands includes only one argument, the VM file name and file type are virtually identical to the Linux file names. The transferred files had the following attributes:

```
list (1)
FILENAME FILETYPE FM FORMAT LRECL RECS BLOCKS
INST EXEC A1 F 80 9 1
INST PARM A1 F 80 1 1
VMINRD22 BIN A1 F 80 44837 876
VMKRNL22 BIN A1 F 80 20166 374
```

### 3.1.5 Booting the installation kernel from the VM reader

Most of the REXX execs we saw both punched the files to the VM reader and then IPLed from the reader.

Others simply punched the files to the reader, and left it up to the installer to issue the IPL command:

```
IPL 00C CLEAR
```

The installation kernel should now boot. From here you should continue with the documentation that is specific to your distribution.

## 3.2 Prepare the LPAR for Linux

Traditional S/390 operating systems see three layers of I/O configuration:

1. The physical layer containing the hardware and cabling.
2. The logical layer provided by the S/390 or zSeries “firmware”. Devices, control units, and channel paths are defined and this information is stored in the Hardware System Area (HSA). The HSA is initially built using data in the IOCDS at power-on reset; anytime after this, it can be dynamically modified using Hardware Configuration Definition (HCD). The Channel Subsystem (CSS) uses this path information for routing I/O.
3. The layer provided by the operating systems itself. I/Oand means that a device must be varied online (or attached) before any I/O can be scheduled to a device.

These three layers provide the convenience of being able to define all devices, which physically exist, so that they can all be seen at the “firmware” level—yet their use and access can be restricted at the operating system level.

Linux does not have the concept of online and offline devices as with other S/390 operating systems. Thus, a problem arises because Linux cannot restrict access to only “its” devices unless the DASD parameter is used in the kernel parameter file.

If this parameter is *not* used, then devices belonging to other systems may have I/O directed to them—and whether that happens inadvertently or deliberately—it is both a security and integrity problem. While it is true that Linux can have its parameter file tailored using a DASD statement to restrict the DASD devices it will see, one is not used during boot of the driver system for the current Turbolinux install procedure. So it is entirely possible for an incorrect DASD device to be formatted if a mistake is made during the Turbolinux install. As you can imagine, this could be disastrous if the device contains critical business or system data.

Therefore, you need to know how to prevent this potential problem in production environments. How? The Linux partition can be “shielded” from a full view of the configuration. HCD provides features, through the I/O Configuration Program (IOCP), that will restrict resources in the I/O configuration to particular logical partitions (LPARs). There are two ways to do this, which we detail in the following examples.

You’ll need two non-shared DASD for the Linux LPAR, as Linux requires one for a file system and one for a swap space. The following example shows the IOCDs definitions (in IOCP statement format) for 2 DASD: an OSA and a tape device, and their corresponding CU and CHPIDs.

### **Example - Using PARTITION on the CHPID statement**

This example restricts the Linux LPAR (LIN01) by using the PARTITION parameter on the CHPID statement. The CHPIDs are defined as reconfigurable, which means they can be *configured* between LPARs, but cannot be shared between LPARs. Assuming that these devices defined by the IODEVICE are the only ones that are connected, via CNTLUNIT definitions, these are the only devices seen by the LPAR LIN01.

```
CHPID PATH=(54),TYPE=CNC,PARTITION=(LIN01,REC)
CHPID PATH=(5A),TYPE=CNC,PARTITION=(LIN01,REC),SWITCH=06
CHPID PATH=(7D),TYPE=CNC,PARTITION=(LIN01,REC)
CHPID PATH=(E9),TYPE=CNC,PARTITION=(LIN01,REC)
...
CNTLUNIT CUNUMBER=0008,PATH=(54),UNIT=OSA
CNTLUNIT CUNUMBER=5A00,PATH=(5A),UNITADD=((00,16)), X
LINK=(CE),UNIT=3490
CNTLUNIT CUNUMBER=7D00,PATH=(7D,E9),UNITADD=((00,32)), X
UNIT=3990,CUADD=1
...
IODEVICE ADDRESS=(5400,02),UNITADD=00,CUNUMBER=(0008),UNIT=OSA
```

```

IODEVICE ADDRESS=(0A10,02),UNITADD=00,CUNUMBER=(5A00),STADET=Y,UNIT=3490
IODEVICE ADDRESS=(21C,02),CUNUMBER=7D00, X
UNITADD=1C,STADET=Y,UNIT=3390

```

### **Example - Using PARTITION on the IODEVICE statement**

This example restricts the Linux LPAR (LIN01) by using the PARTITION parameter on the IODEVICE statement. The CHPIDs are defined as shared between the specified LPARs.

```

CHPID PATH=(54),TYPE=CNC,PARTITION=((LIN01,PRODMVS1,PRODMVS2),SHARED)
CHPID PATH=(5A),TYPE=CNC,PARTITION=((LIN01,PRODMVS1,PRODMVS2),SHARED),X
SWITCH=06
CHPID PATH=(7D),TYPE=CNC,PARTITION=((LIN01,PRODMVS1,PRODMVS2),SHARED)
CHPID PATH=(E9),TYPE=CNC,PARTITION=((LIN01,PRODMVS1,PRODMVS2),SHARED)
...
CNTLUNIT CUNUMBER=0008,PATH=(54),UNIT=OSA
CNTLUNIT CUNUMBER=5A00,PATH=(5A),UNITADD=((00,16)), X
LINK=(CE),UNIT=3490
CNTLUNIT CUNUMBER=7D00,PATH=(7D,E9),UNITADD=((00,32)), X
UNIT=3990,CUADD=1
...
IODEVICE ADDRESS=(5400,02),UNITADD=00,CUNUMBER=(0008),UNIT=OSA,X
PARTITION=(LIN01)
IODEVICE ADDRESS=(5402,14),UNITADD=??,CUNUMBER=(0008),UNIT=OSA,X
PARTITION=(PRODMVS1,PRODMVS2)

IODEVICE ADDRESS=(0A10,02),UNITADD=00,CUNUMBER=(5A00),STADET=Y,UNIT=3490
PARTITION=(LIN01)
IODEVICE ADDRESS=(0A12,14),UNITADD=??,CUNUMBER=(5A00),STADET=Y,UNIT=3490
PARTITION=(PRODMVS1,PRODMVS2)
IODEVICE ADDRESS=(200,30),CUNUMBER=7D00, X
UNITADD=00,STADET=Y,UNIT=3390,PARTITION=(PRODMVS1,PRODMVS2)
IODEVICE ADDRESS=(21C,02),CUNUMBER=7D00, X
UNITADD=1C,STADET=Y,UNIT=3390,PARTITION=(LIN01)

```

## **3.3 Swap space size and type**

If Linux is running under VM, it is recommended that perhaps a 100 cylinder minidisk be allocated for swap space. Because there is no partitioning tool for LPAR or native Linux (however, see 2.4, “Roadmap” on page 26), a swap file may be used in preference to a swap disk. If a swap disk is used, the *whole* disk will be dedicated to the swap function—which could waste a lot of DASD space. Further, under VM, this may lead to a performance problem known as “double paging”.

In any event, these problems can be avoided by either having no swap space or initially having none and then defining a swap file, as a part of the file system, which can be tuned to the correct size without wasting any disk space. The correct size will depend on many factors particular to each system. See the IBM Redbook *Linux for zSeries and S/390: ISP/ASP Solutions*, SG24-6299, for more details. It is on the Web at:

<http://www.redbooks.ibm.com/abstracts/sg246299.html>

## 3.4 Booting the installation kernel in an LPAR

There are a number of ways to install Linux into an LPAR:

- ▶ By using the ICKDSF bootstrap (see 3.4.1, “Using the ICKDSF bootstrap” on page 34)
- ▶ From the HMC CD-ROM or FTP server (see 3.4.2, “Booting from HMC CD-ROM or FTP server” on page 35).
- ▶ Via an IPL tape (see 3.4.3, “Booting from tape” on page 44)

We discuss each of these methods in detail in the following sections.

### 3.4.1 Using the ICKDSF bootstrap

The following recipe should allow you to use OS/390 to prepare a Linux IPL volume in an LPAR. Note that the bootstrap only works for ECKD devices (i.e. 3390, or 3380 on a 3990 control unit).

1. Up/download the following files to your TSO user ID:
  - The sample job (see Appendix E, “ICKDSF bootstrap job” on page 541)
  - The kernel (e.g. cd1/suse/images/tapeipl.ikr)
  - The parameter file (e.g. cd1/suse/images/parmfile)
  - The RAMdisk (e.g. cd1/suse/images/initrd)
2. Take one of the volumes you are going to use in your Linux LPAR and vary it online on OS/390. The remainder of this process will erase the current contents of the volume.
3. Adapt the sample job according to your installation needs:
  - Volume label of the Linux volume you varied online.
  - Data set name of the files you downloaded on your TSO user ID.
4. Run the job to assemble the bootstrap, write a new VTOC on the volume, write the bootstrap records, and copy the three Linux data sets to the volume.
5. Vary the volume offline on OS/390 and IPL it in your Linux LPAR. The prompts will show on the HMC.



The remainder of this procedure is as documented for the IPL from tape.

Although you did an IPL from DASD, the device is not usable in Linux as you have it now. We recommend that you run **dasdfmt** against it and use it as your root device for Linux. You can now complete the install in the fashion specific to your distribution.

If you prefer to do your own job, you can also start with the bootstrap source and do it as fits. (Keep in mind that when you use other names for the three data sets, you will have to adjust the definition in the bootstrap source as well.) The data sets must be on the IPL volume, must not be migrated, and must consist of just a single extent. The bootstrap does not care about record length or block size, but the data set must be F or FB. Also, when you edit the parameter file, notice that the kernel only uses the first 896 bytes of it.

### 3.4.2 Booting from HMC CD-ROM or FTP server

For G5 and later machines, perhaps the most convenient method of booting the installation system is either via FTP server or from CD-ROM at the HMC. No knowledge of any of the various mainframe operating systems is required when installing in either of these ways, and dealing with unlabeled tapes is avoided.

The CD-ROM drive on the HMC is used only to stage the kernel image, parameter file, and RAMdisk image. The installation files of your particular Linux distribution must still be “network reachable” from the target LPAR during the installation process. The installation process that follows the successful booting of the installation kernel is the same, regardless of where the kernel was booted from.

The following series of figures graphically depict the booting of the installation system from a CD by using the CD-ROM drive on the HMC or from an FTP server which is network-accessible to the HMC.

For consistency, we will select the object to be acted upon with a single click of the left mouse button on its icon. Then we will initiate the action by double-clicking with the left mouse button on the icon of the action to be taken.

In Figure 3-1, we have selected Defined CPC and are poised to double-click **Single Object Operations**.

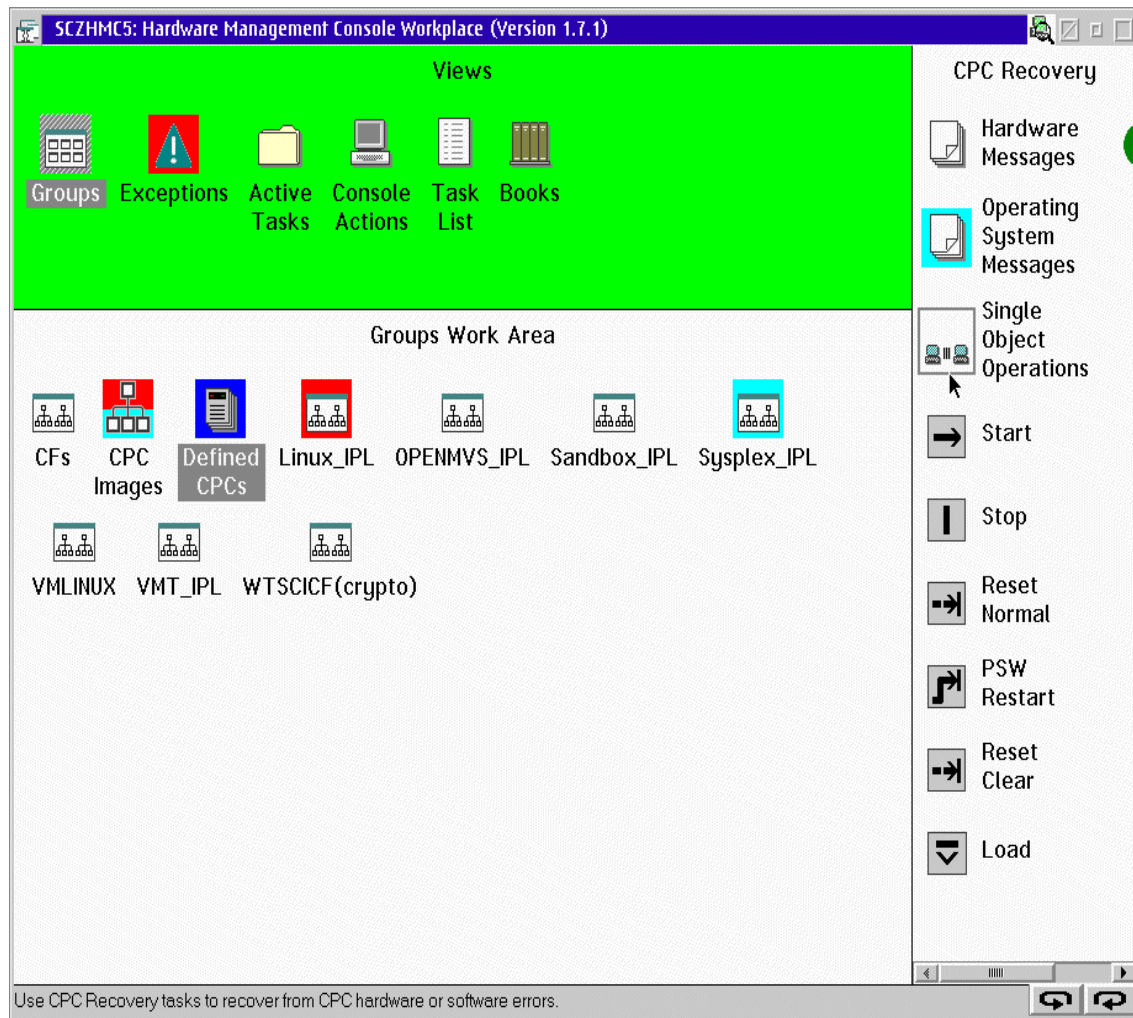


Figure 3-1 Defined CPCs selected

Since there is more than one Central Processor Complex (CPC), we are prompted to select the one that is to be the target of the Single Object Operation. We click **Object Name** to select the target, and then click **OK** to proceed; see Figure 3-2.

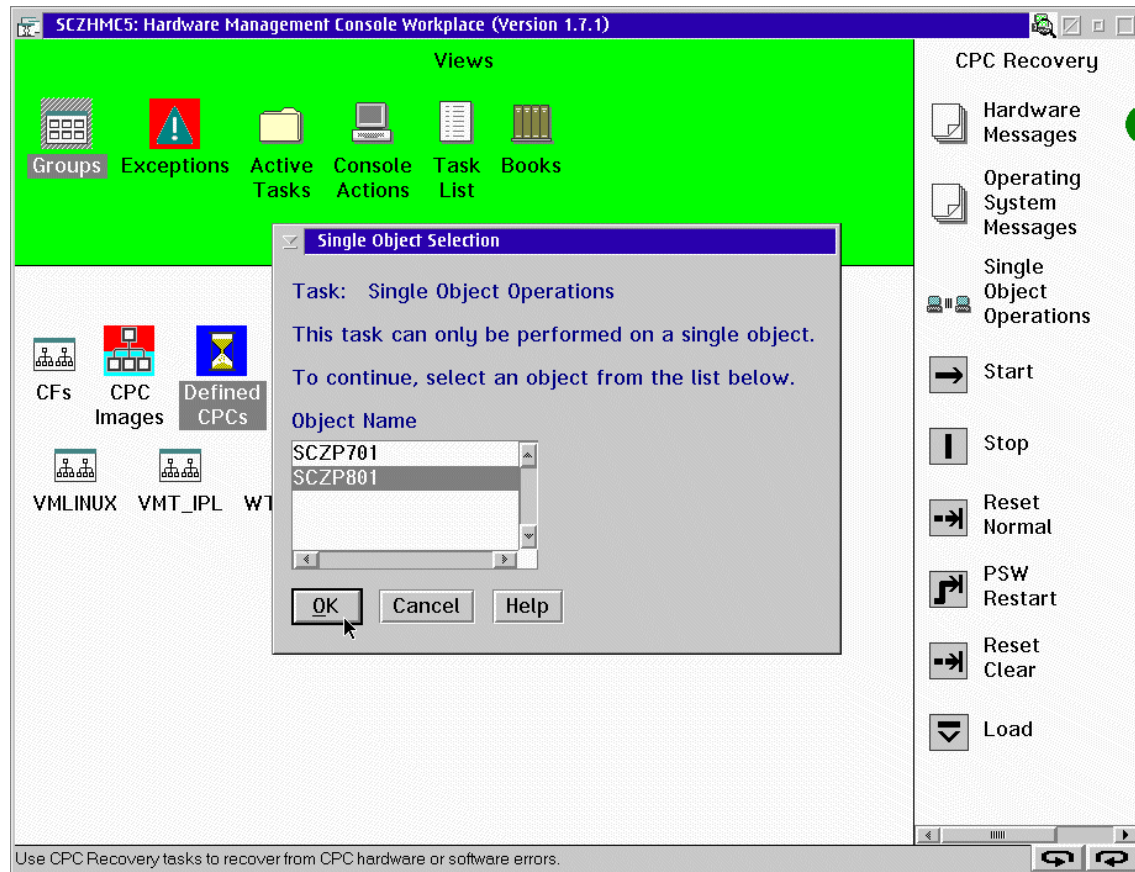


Figure 3-2 Selection of CPC

We now have displayed, in the Images Work Area, all of the LPARs defined for this CPC. The target LPAR will be A12, which in our case is an Integrated Facility for Linux (IFL) LPAR (note the penguin in this icon, denoting the IFL processor).

To initiate the load process, we double-click the icon labeled **Load from CD-ROM or Server**; see Figure 3-3.

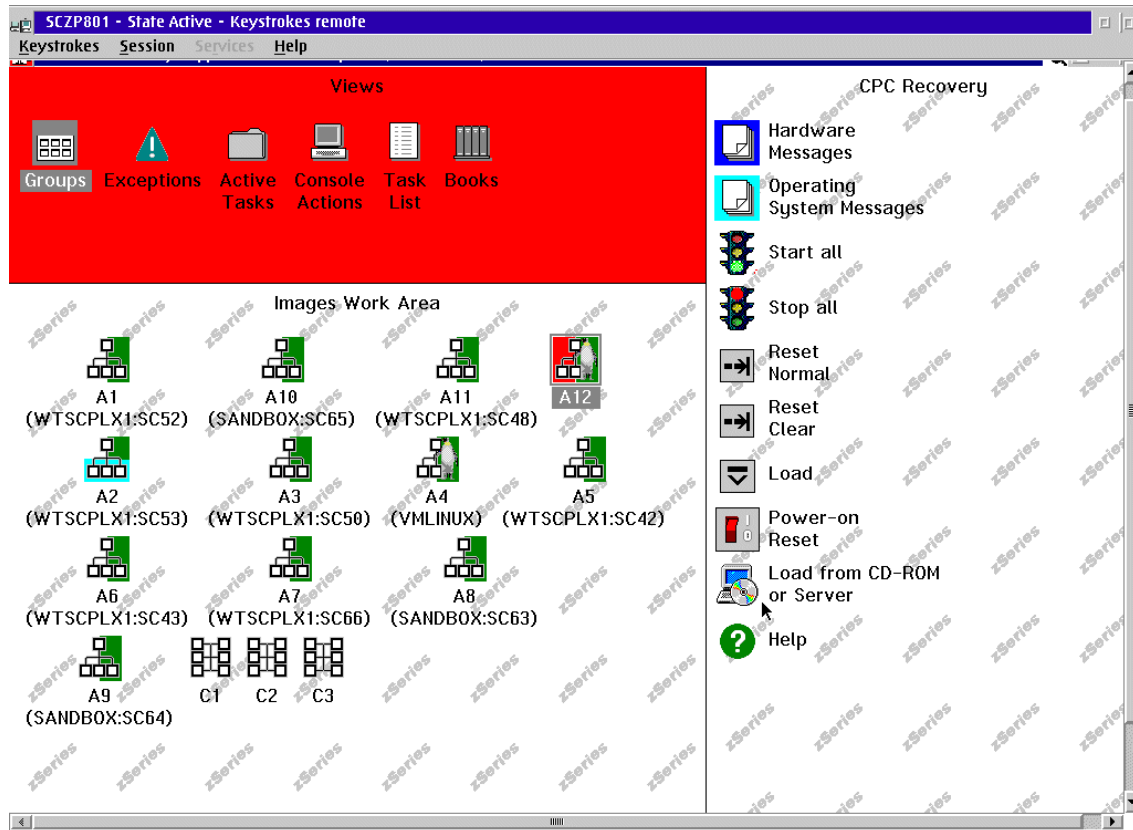


Figure 3-3 LPARs in CPC

Confirmation will be requested to continue with the load. Click **Yes** to continue if the target LPAR is correct; see Figure 3-4.

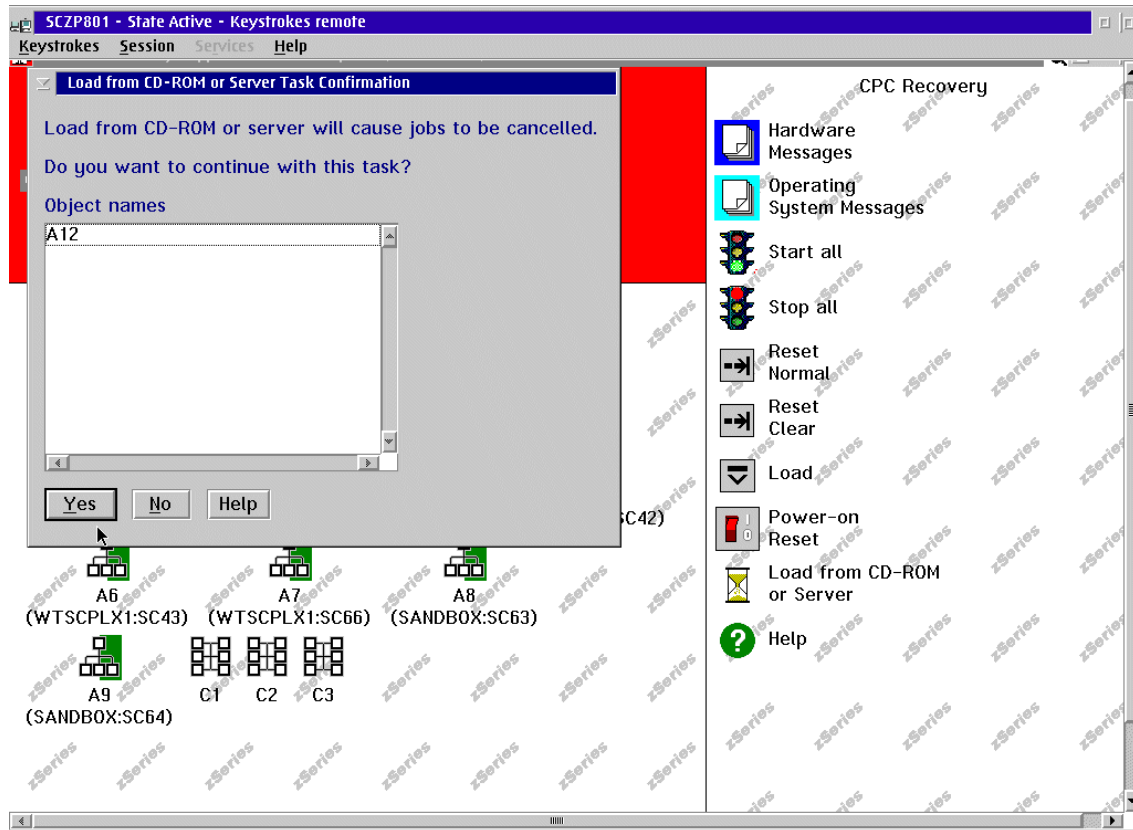


Figure 3-4 Confirming the "Load from CD-ROM or Server"

Once the Load from CD-ROM or Server has been confirmed, we are prompted to select the source for the load. In the first example, we will be loading from CD-ROM, which is the default; see Figure 3-5.

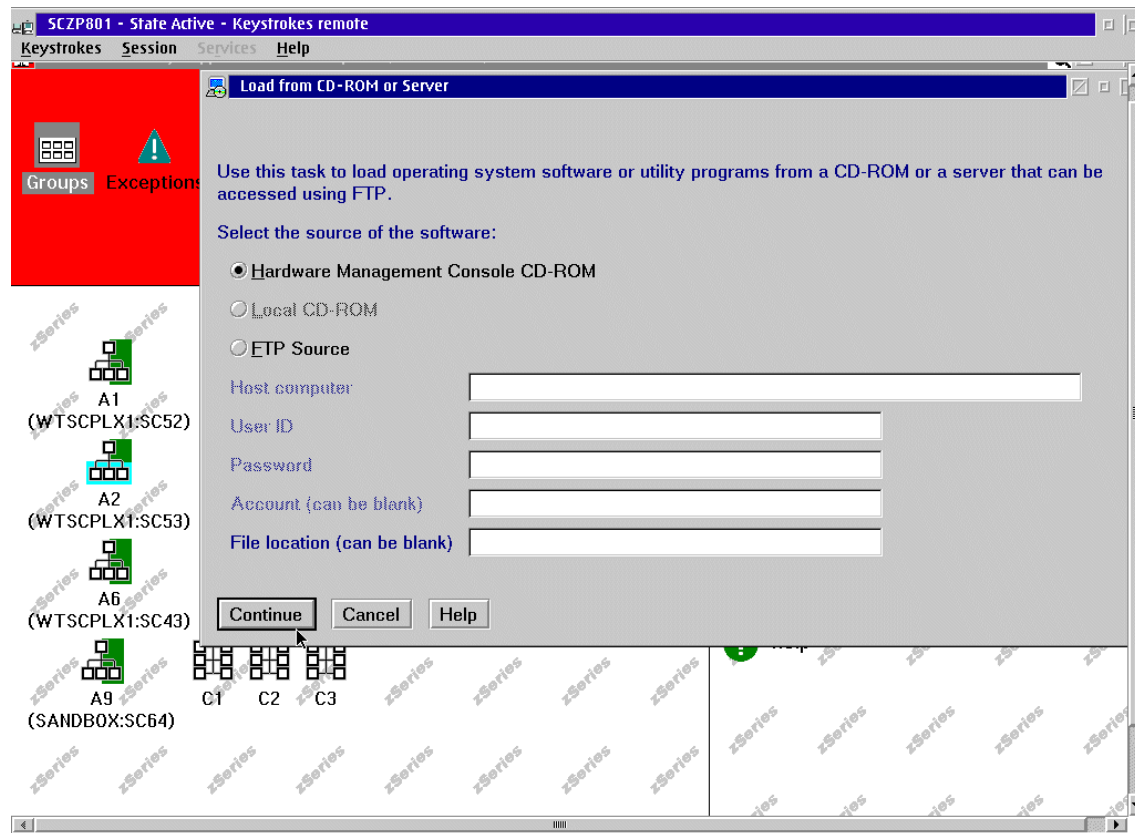


Figure 3-5 Specifying the HMC CD-ROM drive as the load source

In this example, we show specifying an FTP server as the source for our startup files; see Figure 3-6.

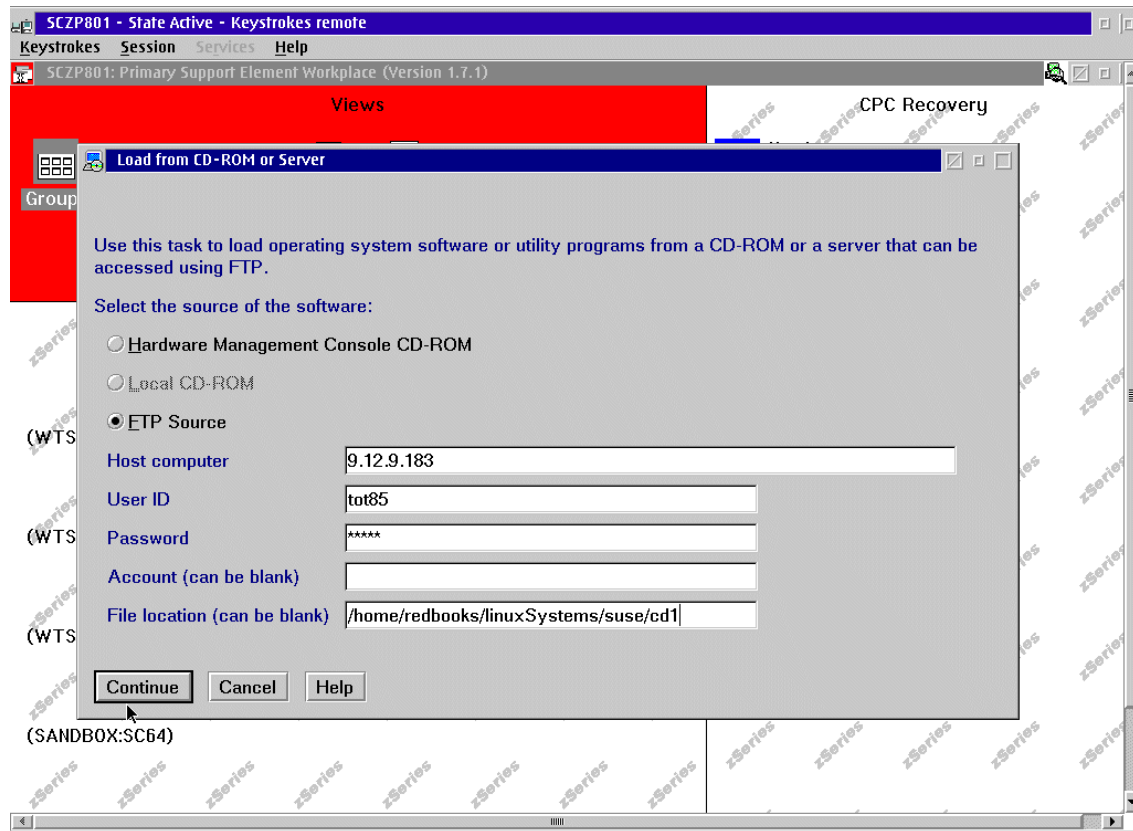


Figure 3-6 Specifying an FTP server as the load source

The root directory of the load source is scanned for files whose names end in .ins. The internal contents of such files have the format as follows:

```
* SuSE Linux for S/390 Installation/Rescue System (default)
suse/images/tapeipl.i kr 0x00000000
suse/images/initrd 0x00800000
suse/images/parmfile 0x00010480
```

The file names and comment (first line) of all such files are then displayed. Since we were not going to be using IUCV communications, we selected the first file; see Figure 3-7 on page 42.

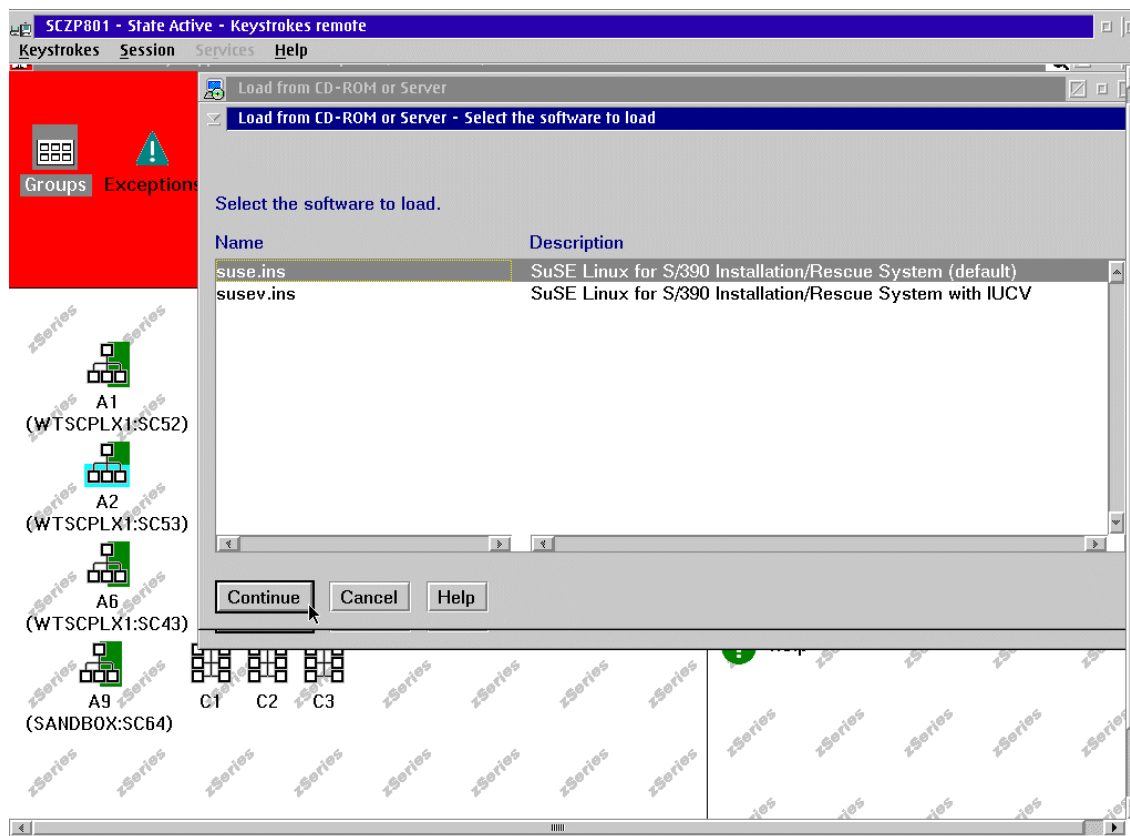


Figure 3-7 Selecting the software



Confirmation of our selection is then requested. We clicked **Yes**.

The selected set of files are then staged from the selected source. We get a confirmation box that says: Retrieving code from source.

A **Reset Clear** is then performed on the target LPAR. We get a confirmation box that says: Performing a clear reset. Following this, the load for the LPAR begins. We get a confirmation box that says: Loading data into system.

Notification of successful completion is the next display that we receive; see Figure 3-8.

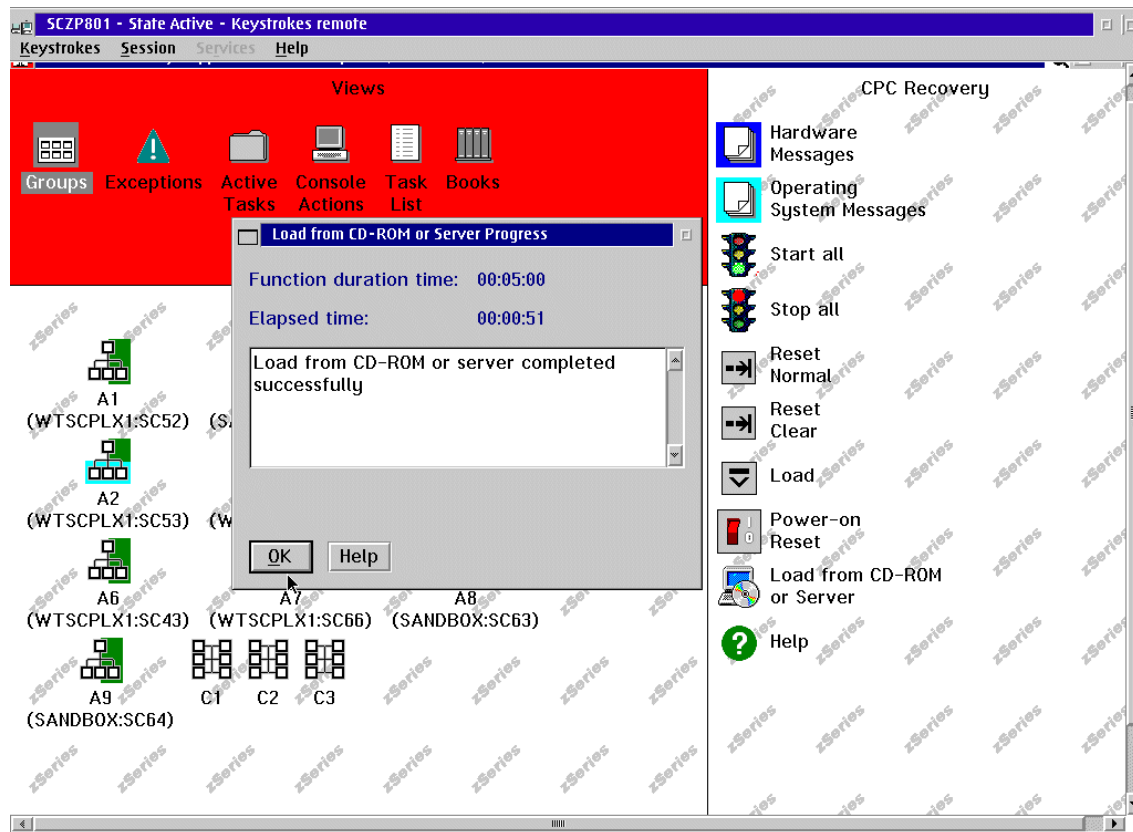


Figure 3-8 Boot complete

The installation kernel has been booted. From here you should continue by using the documentation that is specific to your distribution.

### 3.4.3 Booting from tape

If no FTP server with the installation files is reachable from the HMC, and if no physical CD can be created, then the “boot” of the installation system must be done from tape.

Every distribution includes three files that are needed to create a boot tape. The names of the files will differ, but how you get them to tape will not. The (generic) files are:

- ▶ A kernel image to be IPLed from tape
- ▶ A parameter file to be used when IPLing from tape
- ▶ A RAMdisk which contains the installation scripts

You will need to copy these files to some S/390 operating system environment in order to create the tape. For our installation, we used an OS/390 system. You can also use z/VM (see “Creating an IPL tape on VM” on page 50) or VSE (see “Creating an IPL tape on VSE” on page 50).

Note that you can select any data set name you desire, and that your data set allocations must conform to your installation’s data set naming and allocation standards. In the examples that follow, we transferred the files from a LAN-attached Windows 2000 machine via FTP to OS/390. The files themselves were actually hosted via Samba on a Linux/390 system running under VM. These transfers occurred in the following order:

1. The installation RAMdisk file - /suse/images/initrd
2. The parameter file - /suse/images/parmfile
3. The installation kernel - file /suse/images/tapeipl.ikr

```
ftp wtsc69
...
ftp> quote site lrecl=1024 blksize=8192 recfm=fb track pri=260 sec=50
ftp> bin
ftp> put
Local file f:/linuxSystems/suse/cd1/suse/images/initrd
Remote file linux390.initrd.txt
125 Storing data set MIKEM.LINUX390.INITRD.TXT
ftp: 10132453 bytes sent in 31.74Seconds 319.27Kbytes/sec.
ftp> quote site lrecl=1024 recfm=f blksize=1024 track pri=1
ftp> put
Local file f:/linuxSystems/suse/cd1/suse/images/parmfile
Remote file linux390.parm.line
125 Storing data set MIKEM.LINUX390.PARM.LINE
ftp: 38 bytes sent in 0.03Seconds 1.27Kbytes/sec.
ftp> quote site recfm=fb lrecl=1024 blksize=8192 bl pri=200 sec=100
ftp> put
```

```

Local file f:/linuxSystems/suse/cd1/suse/images/tapeipl.ikr
Remote file linux390.image.txt
125 Storing data set MIKEM.LINUX390.IMAGE.TXT
250 Transfer completed successfully.
ftp: 1501168 bytes sent in 4.96Seconds 302.84Kbytes/sec.
ftp> quit

```

Following are the attributes of the files as they were created on the OS/390 system:

Dataset Name	Tracks	%Used	Device	Dsorg	Recfm	Lrecl	Blksize
MIKEM.LINUX390.IMAGE.TXT	31	100	3390	PS	FB	1024	8192
MIKEM.LINUX390.INITRD.TXT	207	100	3390	PS	FB	1024	8192
MIKEM.LINUX390.PARM.LINE	1	100	3390	PS	F	1024	1024

Before transferring the files to tape, we prepared the tape by creating an empty file which simply wrote tape marks at the beginning of the tape. This destroyed any volume label that might have been on the tape.

In a typical OS/390 production environment, you will have to bypass tape volume protection in order to write to the tape or even be able to specify the “bypass label processing” option BLP which was used on DD statement SYSUT2 in the sample job:

```

//LINUXTP JOB (999,POK),'CREATE NL',NOTIFY=&SYSUID,
// CLASS=A,MSGCLASS=T,MSGLEVEL=(1,1)
//STEP0 EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD DUMMY,DCB=(RECFM=F,BLKSIZE=1024)
//SYSUT2 DD DISP=(NEW,KEEP),LABEL=(1,BLP),DSN=SUSE.BOOT,
// DCB=(RECFM=F,LRECL=1024),VOL=SER=NLTAPE,
// UNIT=B4F
//SYSIN DD DUMMY
//

```

Other volume protection measures in effect by your installed tape management product may require additional DD statement parameters to be specified.

We created the boot tape by running the job as shown:

```

//LINUXTP JOB (999,POK),'CREATE BOOT TAPE',NOTIFY=&SYSUID,
// CLASS=A,MSGCLASS=T,MSGLEVEL=(1,1)
//IMAGE EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD DISP=SHR,DSN=MIKEM.LINUX390.IMAGE.TXT
//SYSUT2 DD DISP=(NEW,PASS),LABEL=(1,NL),DSN=DUMMY,
// DCB=(RECFM=F,LRECL=1024),
// UNIT=B4F

```

```

//SYSIN DD DUMMY
//PARMLINE EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD DISP=SHR,DSN=MIKEM.LINUX390.PARM.LINE
//SYSUT2 DD DISP=(NEW,PASS),LABEL=(2,NL),DSN=DUMMY,
// DCB=(RECFM=F,LRECL=1024),
// VOL=(,RETAIN,,REF=*.IMAGE.SYSUT2),
// UNIT=B4F
//SYSIN DD DUMMY
//INITRD EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD DISP=SHR,DSN=MIKEM.LINUX390.INITRD.TXT
//SYSUT2 DD DISP=(NEW,KEEP),LABEL=(3,NL),DSN=DUMMY,
// DCB=(RECFM=F,LRECL=1024),
// VOL=(,RETAIN,,REF=*.IMAGE.SYSUT2),
// UNIT=B4F
//SYSIN DD DUMMY
//

```

**Note:** There may be environments where a non-labelled tape is simply not possible. If you must write the Linux boot files to a standard-labelled tape, it can still be IPLed by IPLing five times. The first four times will fail, but the label will be skipped over, and the fifth IPL should succeed.

We then transferred the cartridge to a tape drive attached to the LPAR where we were going to install Linux/390. To start the IPL process, we selected our A12 LPAR, and then the Load command; see Figure 3-9.

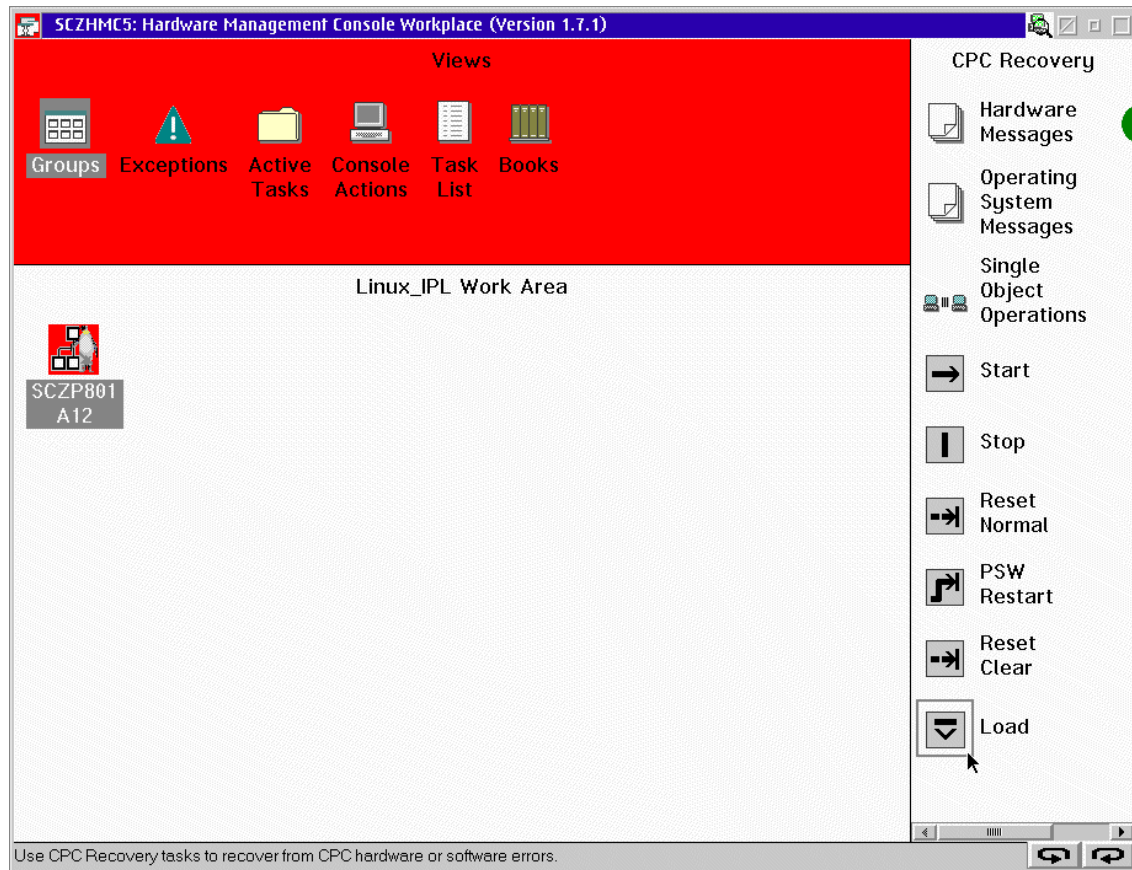


Figure 3-9 Selecting LPAR A12 for Load

When the load panel was displayed, we filled in the unit number of the tape drive with our IPL tape, and clicked **OK**; see Figure 3-10.

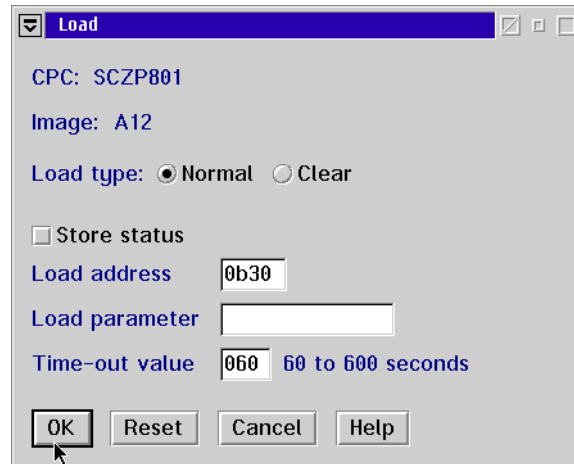


Figure 3-10 Load screen

When asked to confirm the load, we clicked **YES**; see Figure 3-11.

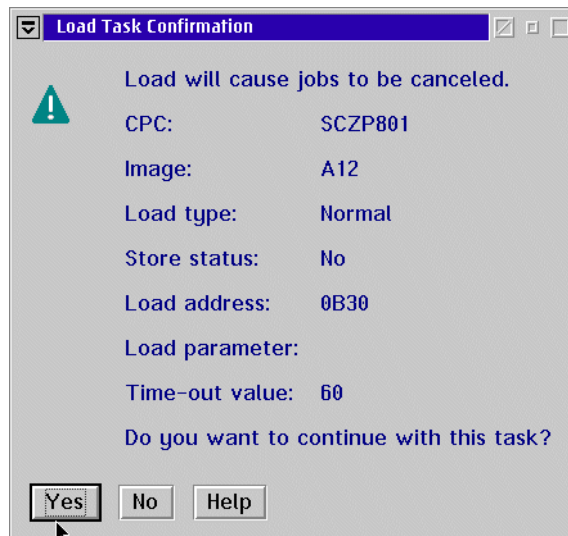


Figure 3-11 Load Confirmation screen

The Load Progress box was then displayed while the tape was read.

The reading of the tape was eventually successful, as indicated as in Figure 3-12. However, as often occurs, we experienced a number of load failures due to other systems accessing the tape controllers, etc, so patience and persistence are definitely virtues in this scenario.

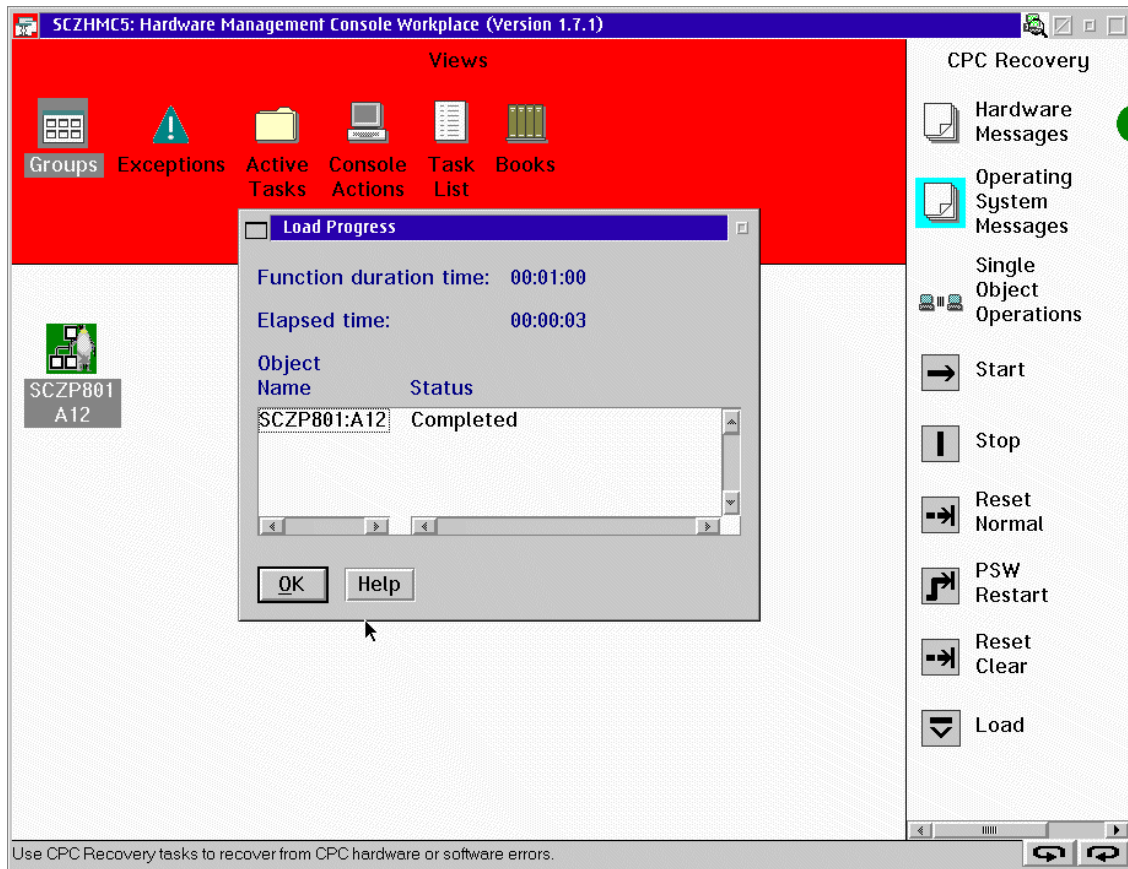


Figure 3-12 Load completed!

At this point, the installation kernel has been booted. If you double-click the Operating System Messages icon, you will (eventually) start to see the usual Linux/390 startup messages. This may take a minute or more, depending on your hardware and system load. From here you should continue by using the documentation that is specific to your distribution.

## Creating an IPL tape on VM

We show the following sequence of FTP commands to download the required boot files for an IPL from tape. Note that files are reblocked to a record length of 1024 bytes, and that the kernel file is specific to the tape IPL method.

```
# ftp 9.12.0.88
...
cd /mnt/cdrom
bin
locsite fix 1024
get /turbo/image-2.2.nn-127-ramdisk-tape turbot.image
get /turbo/initrd turbot.initrd
asc
get /turbo/parmline turbot.parm
quit
```

First a tape device must be attached to the VM guest. The following CP command is an example of attaching the tape device B30, which must be issued from a privileged VM user ID:

```
ATT B30 VMLINUX7 AS 181
```

In this example the user ID VMLINUX7 now has a virtual device 181, which is the real tape drive B30. Insert a blank non-labelled tape and issue a rewind command, **REW 181**, to ensure the tape is rewound. Then the following REXX EXEC will copy the boot files to the tape device 181:

```
/* */
'FILEDEF IN DISK TURBOT IMAGE A'
'FILEDEF OUT TAP1 (RECFM F LRECL 1024 BLOCK 1024 PERM) '
'movefile in out'
'FILEDEF IN DISK TURBOT PARM A'
'movefile in out'
'FILEDEF IN DISK TURBOT INITRD A'
'movefile in out'
```

The tape can now be either unloaded via the **TAPE RUN** command, or can just be rewound for an immediate IPL.

## Creating an IPL tape on VSE

A job similar to the following can be used to preallocate the kernel, parameter file, and RAMdisk:

```
* $$ JOB JNM=ALLOCATE,CLASS=0,DISP=D,NTFY=YES
// JOB ALLOCATE DEFINE FILES
// EXEC IDCAMS,SIZE=AUTO
  DEFINE CL(NAME(LINUX.IMAGE.FILE) -
            CYL(3 3) SHR(1) RECSZ(1024 1024) -
            VOL(DOSRES SYSWK1) -
```



```

        NOREUSE NIXD COMPRESSED -
        FREESPACE(15 7) TO(99366)) -
        DATA(NAME(LINUX.IMAGE.FILE.@D@) -
        CISZ(8192)) -
        CAT(VSESP.USER.CATALOG)
DEFINE CL(NAME(LINUX.PARM.FILE) -
        CYL(2 2) SHR(3) RECSZ(1024 1024) -
        VOL(DOSRES) -
        REUSE NIXD NOCOMPRESSED -
        FREESPACE(15 7) TO(99366)) -
        DATA(NAME(LINUX.PARM.FILE.@D@) -
        CISZ(8192)) -
        CAT(VSESP.USER.CATALOG)
DEFINE CL(NAME(LINUX.INITRD.TXT) -
        CYL(6 6) SHR(1) RECSZ(1024 1024) -
        VOL(DOSRES SYSWK1) -
        NOREUSE NIXD COMPRESSED -
        FREESPACE(15 7) TO(99366)) -
        DATA(NAME(LINUX.INITRD.TXT.@D@) -
        CISZ(8192)) -
        CAT(VSESP.USER.CATALOG)
        IF LASTCC NE 0 THEN CANCEL JOB
/*
// OPTION STDLABEL=ADD
// DLBL IMAGE,'LINUX.IMAGE.FILE',,VSAM,CAT=VSESPUC
// DLBL PARMLIN,'LINUX.PARM.FILE',,VSAM,CAT=VSESPUC
// DLBL INITRD,'LINUX.INITRD.TXT',,VSAM,CAT=VSESPUC
/*
// EXEC IESVCLUP,SIZE=AUTO
A LINUX.IMAGE.FILE          IMAGE VSESPUC
A LINUX.INITRD.TXT          INITRD VSESPUC
/*
/&
* $$ E0J

```

Now a job similar to the following can be used to download the files using FTP:

```

* $$ JOB JNM=FTPJOB,CLASS=A,DISP=D
// JOB GETFILES FTP FILES
// DLBL IMAGE,'LINUX.IMAGE.FILE',,VSAM,CAT=VSESPUC,DISP=(OLD,KEEP)
// DLBL INITRD,'LINUX.INITRD.TXT',,VSAM,CAT=VSESPUC,DISP=(OLD,KEEP)
// DLBL PARMLIN,'LINUX.PARM.FILE',,VSAM,CAT=VSESPUC,DISP=(OLD,KEEP)
// EXEC FTP,PARM='IP=9.12.0.88,PORT=21,ID=00'
<userid>
<password>
<iccf_userid>
<iccf_password> BINARY
cd /mnt/cdrom
get /turbo/image-2.2.nn-127-ramdisk-tape FILE

```

```

get /turbo/parmline TXT
get /turbo/initrd TXT
quit
/*
/&
* $$ E0J

```

Ensure a tape device is available. If VSE is operating as a VM guest, use the **ATTACH** command in the previous job. If not, ensure a tape device is available. Then a job similar to the following can be used to create the IPL tape.

```

* $$ JOB JNM=DITTO,CLASS=A,DISP=D
// JOB DITTO TO CREATE IPL-ABLE LINUX TAPE
// ASSGN SYS006,cuu
// DLBL IMAGE,'LINUX.IMAGE.FILE',,VSAM,CAT=VSESPUC,DISP=(OLD,KEEP)
// DLBL INITRD,'LINUX.INITRD.TXT',,VSAM,CAT=VSESPUC,DISP=(OLD,KEEP)
// DLBL PARMLIN,'LINUX.PARM.FILE',,VSAM,CAT=VSESPUC,DISP=(OLD,KEEP)
// LIBDEF *,SEARCH=PRD1.BASE
// UPSI
// EXEC DITTO
$$DITTO REW OUTPUT=SYS006
$$DITTO WTM OUTPUT=SYS006,NTMKS=5
$$DITTO REW OUTPUT=SYS006
$$DITTO VTP FILEIN=IMAGE,OUTPUT=SYS006
$$DITTO VTP FILEIN=PARMLIN,OUTPUT=SYS006
$$DITTO VTP FILEIN=INITRD,OUTPUT=SYS006
$$DITTO E0J
/*
/&
* $$ E0J

```



## Part 2

# Strategic Linux distributions

In this part of the book, we describe the IBM strategic distributions of Linux for S/390 and zSeries. They are:

- ▶ SuSE Linux AG, which has a generally available Linux distribution
- ▶ Turbolinux Inc., which has a generally available Linux distribution
- ▶ Red Hat Inc., which has a beta Linux distribution





## Overview of SuSE Linux

In this chapter we describe the Linux distribution offered by SuSE Linux AG (SuSE). SuSE's product is called SuSE Linux Enterprise Server for S/390. We will use the term SuSE Linux for brevity.

SuSE Linux was announced on October 31, 2000 and became generally available around November 5, 2000. It was based on SuSE Linux 7.0 for the PC platform. A beta was originally available around July 2000.

The product is primarily intended for S/390 G5 and G6, Multiprise 3000, and zSeries machines, but can be installed and used on S/390 G3 and G4 series, and Multiprise 2000 machines with certain limitations. Service and support is available as is project planning, implementation, performance tuning, and other consulting services.

The version of the product that we installed was from a "release candidate" provided by SuSE that had been created around May 9, 2001. It was Linux version 2.2.16 with a kernel compile date of May 6, 2001 06:15:49 GMT.

## 4.1 Getting the SuSE distribution

You can acquire the SuSE Linux distribution via SuSE sales, or on the Web.

### 4.1.1 Via SuSE sales

USA/Canada:

phone: +1-510-628-3380  
fax: +1-510-628-3381  
E-mail: [s390-sales@suse.com](mailto:s390-sales@suse.com)  
Web: [shop.suse.com](http://shop.suse.com)

Europe:

phone: +49-911-74053-31  
E-mail: [s390-sales@suse.de](mailto:s390-sales@suse.de)

### 4.1.2 Via FTP site

The SuSE Linux distribution is contained on three ISO images and can be downloaded from:

<ftp://ftp.suse.com/pub/suse/s390/suse-us-s390>

**Note:** If you downloaded the images but would like installation assistance and/or support, then you must contact one of the above listed sales locations.

Along with the ISO images, the file MD5SUM is available which can be used to validate the accuracy of any copies made of the ISO images. To perform this validation, download file MD5SUM to the same directory and enter the following command:

```
# md5sum -c MD5SUM
```

Successful validation of the files will produce output similar to the following:

```
suse-us-s390-cd1.iso: OK  
suse-us-s390-cd2.iso: OK  
suse-us-s390-cd3.iso: OK
```

The display will contain the names of the ISO files that you downloaded (which will not necessarily be the ones displayed in the example above).

## Creating the CDs

Typically, CDs will be created from the ISO images. Creating the CDs will provide a backup of the contents of the large, downloaded ISO files, provide a convenient means to copy the contents of the distribution to a network server for use during installation, and provide the capability to boot the installation system from CD.

Given the many CD “burning” programs available on various hardware platforms and operating systems, it is not practical to attempt to describe here how to create a CD from an ISO image. However, use care to ensure that the ISO image is not simply *copied* to the CD but that the file system created on the CD was, in fact, populated with all the files contained within the ISO image.

## Using the ISO images directly

If you have a Linux system available and if it can be accessed from the z900 or S/390 system on which Linux is to be installed, then you can use the contents of ISO images directly.

Simply loopback-mount them as follows:

```
# mkdir /suse
# cd /suse
# mkdir cd1 cd2 cd3
# mount -o loop -r /<your location>/suse-us-s390-cd1.iso /suse/cd1
# mount -o loop -r /<your location>/suse-us-s390-cd2.iso /suse/cd2
# mount -o loop -r /<your location>/suse-us-s390-cd3.iso /suse/cd3
```

**Important:** Mounting a CD image via loopback devices requires ISO 9660 support in the kernel which is built into most, but not all, kernels

## 4.2 Documentation included with the SuSE distribution

The first CD contains the following installation documentation for SuSE Linux:

- ▶ l390gp3.pdf - *Preparing for Installing SuSE LINUX for S/390* @, April 5, 2001  
LINUX-1001-01, 76 pages
- ▶ l390ga3.pdf - *Installing SuSE LINUX for S/390* @, April 5, 2001,  
LINUX-1002-01, 146 pages
- ▶ manual.pdf - *SuSE 7.0 Installation, Networking, Know How*, 271 pages

Additionally SuSE includes the file sg244987.pdf, which is the IBM Redbook entitled *Linux for S/390*, SG24-4987. This publication consists of 533 pages and is one of the most comprehensive publications to date for Linux on S/390, providing an introduction to Linux for management while covering installation, configuration, and customization. It is on the Web at:

<http://www.redbooks.ibm.com/abstracts/sg244987.html>

**Tip:** Read and follow the excellent installation documentation found in the “Preparing” and “Installing” manuals.

The list of software RPMs that are installed on SuSE Linux are listed in an appendix entitled “Linux Software Packages”. Due to the size of this document, it is not included in this redbook, but can be found separately on the Web at:

<ftp://www.redbooks.ibm.com/redbooks/SG246264/RPMappendix.pdf>

## 4.3 Software components of the distribution

SuSE groups various related RPM’s into “series”. In this section we summarize each series with a short descriptive phrase, along with the number of packages contained in the series.

RPM is an acronym for Red Hat Package Manager. When used as a noun, it usually describes a binary package that can be run on a specific platform. SRPM is a Source code RPM. It is a package that contains source code which typically must be built into binaries.

### ***Series on CD #1***

Series	Description	RPM’s
a1	Linux Base System (You need it!)	60
ap1	Programs that don’t need X	88
ap2	Programs that don’t need X	19
d1	Development (C, C++, Lisp, etc.)	75
e1	Emacs	12
fun1	Games and more	1
gra1	all about graphics	22
j1	Japanese packages	23
kde1	Desktop Environment	2
kde2	Desktop Environment	1
kpa1	KDE applications	5
n1	Network-Support (TCP/IP, UUCP, Mail, News)	124
n2	Network-Support (TCP/IP, UUCP, Mail, News)	16
pay1	Commercial Software	8
perl1	Perl modules	27
sec1	Security related software	18
snd1	Sound related stuff	2



sp11	Spell checking utilities and databases	20
sp12	Spell checking utilities and databases	3
tc11	Tcl/Tk/TclX,Tcl-Language and Tk-Toolkit for X	4
tc12	Tcl/Tk/TclX,Tcl-Language and Tk-Toolkit for X	37
tex2	TeX/LaTeX and applications	1
x1	Base X Window System - XFree86	26
x3d1	3D stuff for X11 and console	3
xa[1	X Applications	76
xdev1	Development under X11	5
xdev2	Development under X11	22
xwm1	Window Manager and Desktop	22
xwm2	Window Manager and Desktop	5

29 series and 727 RPMs on CD #1

### **Series on CD #2**

d2	Development (C, C++, Lisp, etc.)	20
doc1	Documentation	6
doc2	Documentation	149
doc3	Documentation	24
fun2	Games and more	15
gnm1	GNOME - GNU Network Object Model Environment	4
gnm2	GNOME - GNU Network Object Model Environment	5
gnm3	GNOME - GNU Network Object Model Environment	19
gnm4	GNOME - GNU Network Object Model Environment	4
gra2	All about graphics	34
han1	Korean packages	8
sgm1	Components for a SGML system	19
xv1	XView (OpenLook, Applications)	1
zq2	Source packages	439

14 series - 308 RPMs and 439 SRPMs

### **Series on CD #3**

tex1	TeX/LaTeX and applications	32
x3	Base X Window System - XFree86	1
zq1	Source packages	250

3 series - 33 RPMs and 250 SRPMs

There are 46 series with a total of 1068 RPMs and 689 SRPMs.

There are 1757 total packages in distribution.

## 4.4 How does SuSE Linux differ

Most distributions provide system administration tools which were developed expressly for it. Such tools provide a means to achieve increased productivity in the installation, customization, and support of a Linux system. This provides one of the major opportunities for a distribution builder to add value.

SuSE has developed YaST (Yet another Setup Tool) to provide such a facility for their various distributions, including S/390. Some of the tasks supported by YaST are as follows:

- ▶ Installation
- ▶ Changes to installation
- ▶ Install/delete packages
- ▶ Add/delete/list users and groups
- ▶ Configure new hardware
- ▶ Maintain security settings
- ▶ Create/maintain file systems
- ▶ Back up the system
- ▶ Maintain the system configuration file

In response to the system administrator's changes, YaST alters the contents of the SuSE configuration file `/etc/rc.config`. Then YaST invokes `SuSEConfig` in order to implement the changes.

Other differences:

- ▶ More documentation is available that has been developed specifically for the SuSE distribution.
- ▶ Support for the Logical Volume Manager is provided in the supplied kernels. Logical volumes can be defined and used during the installation.
- ▶ Currently, the largest selection of software packages for a S/390 distribution is provided. The distribution occupies three full CDs.
- ▶ Professional services are available for installation and support.
- ▶ CISCO CIP is now a supported network interface and can be selected at installation.



# Installing SuSE Linux

In this chapter we discuss the installation of SuSE Linux for S/390 and zSeries under VM and in an LPAR. Installing SuSE under VIF is documented in Chapter 16, “VIF” on page 267.

In performing the installation and customization activities for SuSE Linux, we followed the supplied installation documentation in order to assess its usability and completeness. Using the documentation provided on CD1; *Preparing for Installing SuSE LINUX for S/390 @–April 5, 2001 (l390gp3.pdf)* and *Installing SuSE LINUX for S/390 @–April 5, 2001 (l390ga3.pdf)*. Our installation scenarios were completed without incident.

What minor exceptions to the documentation that were encountered during the installation process are described where found.

Regardless of the method of installation, the contents of CD #1 and CD #2 must be accessible to installation kernel via NFS, FTP, or SMB. The default installation does not use any files on CD #3. CD #3 is only used for TeX/LaTeX and related applications, for the Base X Window System (XFree86), and for many of the source RPMs. We recommend, however, that you make CD #3 accessible if you have the available disk space. Then, regardless of the type of system to be installed—or when adding package groups late—the necessary files will be available.

## 5.1 General installation considerations

Note the following considerations in relation to the installation task.

### 5.1.1 Boot file processing

In order to boot the *starter* system through which the installation will be accomplished, three files are placed on the boot device in the following order:

1. Installation kernel image
2. Parameter file
3. RAMdisk image

For an LPAR or native installation, these three files are either copied to tape or emulated tape, or staged on the HMC from CD #1.

When booting, a hardware-generated read to the boot device will transfer the first block of data into storage, and execution control will pass to the first byte in the data block. This is the initial record in the kernel's self-loading routine.

Once loaded, the kernel does a single read to the boot device to get its boot-time parameters. Then the next file, which is the image of the memory-resident file system (RAMdisk), is read from the boot device. The RAMdisk contains all of the programs, scripts, etc. required for the installation process.

### 5.1.2 DASD considerations

When the SuSE installation routines create the file `/boot/parmfile`, the device number of the swap file system will always be declared first. Therefore, if there is a swap device, it will always be `/dev/dasda1`.

Both the "Installation Worksheet for LPAR or native" on page 5 of the *Preparing for Installing SuSE LINUX for S/390* manual and the "Installation Worksheet for VM" on page 7 of the same manual ask for the "First DASD volume (root file system) and the "Second DASD volume (swap space)". This would lead one to casually assume that the root file system will be created on DASDA and that the swap file will be located on DASDB. Be aware though that, typically, the root file system will be on DASDB if a swap volume is specified during the installation process.

Logical Volume Manager (LVM) support is included in the installation kernel. YaST supports the creation and management of LVM groups and can be used at installation to build such for use by the system to be installed. We chose not to perform our installations to LVM DASD. We did, however, exercise YaST in creating an LVM environment; that experience is described in Chapter 6, “Customizing and using SuSE Linux” on page 91.

### 5.1.3 Installation options and DASD requirements

We typically used SuSE’s default for the type of installation. Installation types, along with their approximate package counts and DASD space requirements, are summarized in Table 5-1:

*Table 5-1 Disk space requirement by installation type*

Installation type	Package count	Space required
SuSE Almost Everything	834	2.72G
SuSE Development System	319	1.18G
SuSE DMZ Base System	96	309.1M
SuSE Minimum System	75	160.4M
SuSE Network Oriented System	337	918.1M
SuSE Default System with Server Applications and Development	244	784.3M

Our actual installation of the SuSE Default System with Server Applications and Development contained 248 packages. Therefore, the values listed should be not taken as absolutes, but rather as values intended for planning purposes.

The distribution of disk occupancy by key directory for the installation type of SuSE Default System with Server Applications and Development is described in Table 5-2:

*Table 5-2 Disk usage by directory*

Directory	Size
/bin	5.7 MB
/boot	1.7 MB
/dev	28 KB
/etc	3.6 MB

Directory	Size
/lib	12 MB
/opt	31 MB
/root	20 KB
/sbin	8.2 MB
/tmp	4 KB
/usr	654 MB
/var	19 MB
Total	736 MB

## 5.2 Installing SuSE under VM

In this section we describe how to install SuSE under VM, provide various checklists, and detail how to FTP files and create REXX execs to help you in the install process.

### 5.2.1 SuSE under VM checklist

For convenience, we start with an overall checklist.

1. Read 3.1, "Prepare the VM guest for Linux" on page 28.
2. Plan your DASD configuration:
  - a. Which minidisks are going to be used
  - b. What file systems are going to be on what minidisks
  - c. How big those minidisks need to be - see 5.1.3, "Installation options and DASD requirements" on page 63
3. Complete the worksheet in Table 5-3 on page 65 (or, alternatively, the "Installation worksheet for VM" located in *Preparing for Installing SuSE LINUX for S/390* @-April 5, 2001 I390gp3.pdf).
4. Make sure the three appropriate boot files and an IPL EXEC are available - see 5.2.3, "Transferring installation system files to your VM guest" on page 65.
5. IPL Linux from reader (or possibly tape).
6. Complete the installation with YaST - see 5.2.4, "Logging in via telnet and completing the install via YaST" on page 71.

## 5.2.2 Installing SuSE Linux under VM

Table 5-3 is a copy of the “Installation worksheet for VM” found in *Preparing for Installing SuSE LINUX for S/390* @–April 5, 2001, which is also provided here for your convenience.

Table 5-3 SuSE installation worksheet

VM Data	Value
VM Guest user ID (if installing under VM)	
VM guest password (if installing under VM)	
Disk with FTP program	
First DASD volume (root file system)	
Second DASD volume (swap space)	
Tape unit (if installing from tape)	
Host name	
IP address	
Net mask	
Broadcast address	
Gateway address	
IP address of the DNS	
DNS search domain	
IP address or host name	
Path to CD-ROM data	
Root password	

## 5.2.3 Transferring installation system files to your VM guest

We used FTP to transfer the image of the installation kernel, the parameter file, and the RAMdisk image to our VM guest system. Following are the important FTP commands:

```
FTP 9.12.9.183
...
cd /suse/cd1
bin
locsite fix 80
```

```

get suse/images/vmldr.ikr suse.image
get suse/images/initrd suse.initrd
ascii
get suse/images/parmfile suse.parm
quit

```

**Note:** SuSE's documentation refers to the "parmfile" as "parmline". The file on CD #1 is named "parmfile".

## Creating LIN and LIPL EXECs

We then created the LIPL REXX EXEC, which will be used to place the installation files on the VM reader, and then to boot the installation system.

Under CMS, we created a REXX EXEC file **LIN EXEC**, which consisted of the following statements.

**Note:** Keep in mind that a REXX EXEC must always start with a C style comment to let VM know that REXX should interpret the file.

```

/* */
'CLOSE RDR'
'PURGE RDR ALL'
'SPOOL PUNCH * RDR'
'PUNCH SUSE IMAGE A (NOH'
'PUNCH SUSE PARM A (NOH'
'PUNCH SUSE INITRD A (NOH'
'CHANGE RDR ALL KEEP NOHOLD'
'IPL OOC CLEAR'

```

We also created the REXX **LIPL EXEC**, which can be used to reboot the kernel. The statements for LIPL are as follows:

```

/* */
'CHANGE RDR ALL KEEP NOHOLD'
'IPL OOC CLEAR'

```

We now had all the files created or staged that we would use to boot the installation system. A list of our files at this point is as follows:

Cmd	Filename	Filetype	Fm	Format	Lrec1	Records	Blocks	Date	Time
	SUSE	PARM	A1	V	36	2	1	6/12/01	17:46:36
	SUSE	INITRD	A1	F	80	126656	2474	6/12/01	17:40:53
	SUSE	IMAGE	A1	F	80	18972	339	6/12/01	17:39:16
	LIN	EXEC	A1	V	28	9	1	6/17/01	17:47:46
	LIPL	EXEC	A1	V	28	3	1	6/17/01	17:54:49
	VMLINUX5	NETLOG	A0	V	103	1	1	5/14/01	17:34:36
	PROFILE	EXEC	A1	V	31	5	1	5/14/01	17:28:34



## Booting the installation system under VM

From our CMS prompt, we invoked the LIN EXEC as follows:

```
lin
0000003 FILES PURGED
RDR FILE 0019 SENT FROM VMLINUX5 PUN WAS 0019 RECS 019K CPY 001 A NOHOLD NOKEEP
RDR FILE 0020 SENT FROM VMLINUX5 PUN WAS 0020 RECS 0002 CPY 001 A NOHOLD NOKEEP
RDR FILE 0021 SENT FROM VMLINUX5 PUN WAS 0021 RECS 127K CPY 001 A NOHOLD NOKEEP
0000003 FILES CHANGED
0000003 FILES CHANGED
```

Since the last command in the REXX EXEC was an IPL of our installation system, momentarily we received the following output from the startup of the kernel:

```
Linux version 2.2.16 (root@ikr_rdr.suse.de) (gcc version 2.95.2 19991024
(release)) #1 SMP Tue May 1 11:44:38 GMT 2001
Command line is: ramdisk_size=32768 root=/dev/ram0 ro
We are running under VM
This machine has an IEEE fpu
Initial ramdisk at: 0x02000000 (10132480 bytes)
Detected device 2928 on subchannel 0000 - PIM = 80, PAM = 80, POM = FF
Detected device 2929 on subchannel 0001 - PIM = 80, PAM = 80, POM = FF
Detected device 0191 on subchannel 0002 - PIM = F0, PAM = F0, POM = FF
.....
Detected device 019E on subchannel 000F - PIM = F0, PAM = F0, POM = FF
Jun 12 21:59:26 suse syslogd 1.3-3: restart.
Checking if LVM is supported in the kernel... Ok.
```

## Network setup

With the startup of the kernel complete, the network setup script started and the following display was received, to which we replied **2**, as shown:

```
=
==- Welcome to SuSE Linux S/390 7.0
=
```

```
First, select the type of your network device:
0) no network
1) OSA Token Ring
2) OSA Ethernet
3) OSA-Express Gigabit Ethernet
4) Channel To Channel
5) Escon
6) IUCV
7) CLAW (Cisco Mainframe Channel Connection)
Enter your choice (1-7): 2
```

The SuSE installation documentation had selection numbers 3 and 6, followed by (*experimental*); it did not have a selection number 7. Because Cisco routers are common, having CLAW support available at installation is a plus.

Next, we were informed that we must read and confirm the license information for the driver provided by IBM:

```
To set up the network, you have to read and confirm the license information
of the network device driver provided by IBM.
Do you want to see the license (Yes/No) ? yes
```

Following our reply, we were presented with the terms of the agreement (you must display the entire agreement and then reply affirmatively to the prompt at the end). Following our affirmative reply, we were prompted to begin specifying our network interface information.

```
Ok, now we can set up the network configuration.
```

```
Please enter the device address of the network device
Ask your system administrator for the correct settings.
If there is only _ONE_ OSA network device attached to your machine,
you may type "auto" for automatic detection; use this option carefully.
Network device address (e.g. FC20): 2928
```

```
Please enter the relative port number on device address 2928
Relative port: 0
```

To the above prompt, we replied with our OSA device number and port number.

The installation routines attempted to activate the lcs module with our supplied values. The output of the activation process was as follows:

```
Unloading LCS module if active...
rmmod: module lcs is not loaded
Trying to start the LCS module now...
insmod lcs noauto=1 devno_portno_pairs=0x2928,0 :
Using /lib/modules/2.2.16/net/lcs.o
Starting lcs
lcs: eth0 configured as follows read subchannel=0 write subchannel=1
read_devno=2928 write_devno=2929
hw_address=00:06:29:6C:CB:CE rel_adapter_no=0
lcs configured to use sw statistics,
ip checksumming of received packets is off.
autodetection is off.
configured to detect
cu_model 0x01,15 rel_adapter(s)
cu_model 0x08,15 rel_adapter(s)
cu_model 0x60,1 rel_adapter(s)
cu_model 0x1F,15 rel_adapter(s)
lsmod now shows all loaded modules:
```

```
lcs                14896  0 (unused)
```

Was the start of the LCS module successful (Yes/No) ? **yes**

Next we were prompted to supply the remaining network configuration parameters:

```
Please enter your full host name (e.g. s390.suse.com):  
vmlinux5.itso.ibm.com  
Please enter your IP address: 9.12.6.76  
Please enter the net mask: 255.255.255.0  
Please enter the broadcast address (9.12.6.255):  
Please enter the gateway address: 9.12.6.75  
Please enter the IP address of the DNS server (leave blank for none):  
9.12.14.7  
Please enter the DNS search domain (e.g. suse.com): itso.ibm.com  
Please enter the MTU (Maximum Transfer Unit,  
leave blank for default: 1492):
```

Note that we took the defaults for the broadcast address and MTU size by simply pressing Enter at the prompt. We were prompted to confirm that our configuration was correct. (If it had not been correct, a reply of *no* would have started the process from the beginning.) The values were correct for our installation, as shown:

```
Configuration for eth0 will be:  
Full host name   : vmlinux5.itso.ibm.com  
IP address       : 9.12.6.76  
Net mask        : 255.255.255.0  
Broadcast address: 9.12.6.255  
Gateway address  : 9.12.6.75  
DNS IP address   : 9.12.14.7  
DNS search domain: itso.ibm.com  
MTU size        : 1492  
Is this correct (Yes/No) ? yes
```

Next we had to set a temporary password for user root.

```
For security reasons you have to set an temporary installation  
system password for the user "root".  
You'll be asked for it only when you telnet into this installation  
system to limit the access to it and it will be cleared as soon  
as you shut down or reset the installation system
```

```
Please enter the temporary installation password: *****
```

We received confirmation of the temporary password change, followed by activation of our network. Note (in the following results) the use of the **ping** command to verify that our supplied network addresses were reachable:

```
Temporary installation password set.
restarting syslogd:
Jun 13 11:24:32 vmlinux5 syslogd 1.3-3: restart.
ifconfig eth0 9.12.6.76 netmask 255.255.255.0 broadcast 9.12.6.255 mtu 1492
/sbin/ifconfig eth0 :
eth0      Link encap:Ethernet  HWaddr 00:06:29:6C:CB:CE
          inet addr:9.12.6.76  Bcast:9.12.6.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1492  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
```

```
Trying to ping my IP address:
PING 9.12.6.76 (9.12.6.76): 56 data bytes
64 bytes from 9.12.6.76: icmp_seq=0 ttl=255 time=62.260 ms
64 bytes from 9.12.6.76: icmp_seq=1 ttl=255 time=0.106 ms
64 bytes from 9.12.6.76: icmp_seq=2 ttl=255 time=0.091 ms
--- 9.12.6.76 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.091/20.819/62.260 ms
Trying to ping the IP address of the Gateway:
PING 9.12.6.75 (9.12.6.75): 56 data bytes
64 bytes from 9.12.6.75: icmp_seq=0 ttl=64 time=2.981 ms
64 bytes from 9.12.6.75: icmp_seq=1 ttl=64 time=2.319 ms
64 bytes from 9.12.6.75: icmp_seq=2 ttl=64 time=0.896 ms
--- 9.12.6.75 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.896/2.065/2.981 ms
Gateway seems to respond to our pings, continuing.
Trying to ping the IP address of the DNS Server:
PING 9.12.14.7 (9.12.14.7): 56 data bytes
64 bytes from 9.12.14.7: icmp_seq=0 ttl=128 time=1.161 ms
64 bytes from 9.12.14.7: icmp_seq=1 ttl=128 time=0.797 ms
64 bytes from 9.12.14.7: icmp_seq=2 ttl=128 time=0.736 ms
--- 9.12.14.7 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.736/0.898/1.161 ms
Network Setup finished, running inetd...
```

Network setup was now finished. To perform the rest of the installation, we left our 3270 session and resumed via telnet session. First, however, we had to logoff from VM while leaving our virtual machine running. Following the last shell prompt above, we replied with the following VM CP commands:

```
#cp set run on
#cp disc
```

## 5.2.4 Logging in via telnet and completing the install via YaST

Since we would be running YaST in this telnet session, we had to ensure that our emulated terminal was capable of supporting a screen of 80 characters by 25 lines, at a minimum.

We started our telnet client and logged in as root, using the temporary password that we had set at the end of network setup. We logged in and received the following screen:

```
>>> >>> >>> >>> >>> >>> SuSE Linux S/390 7.0 <<< <<< <<< <<< <<< <<<
```

1. If you want to check which devices the dasd driver can see, run 'insmod dasd probeonly' (may take long) and check the output of 'cat /proc/dasd/devices'. Remove the dasd driver with 'rmmod dasd' afterwards.
2. Choose the device numbers you want to use for SuSE Linux S/390  
!!! BE CAREFUL WHEN SELECTING DASDs - !!!  
!!! YOU MAY DESTROY DATA ON SHARED DEVICES !!!
3. Enter 'insmod dasd dasd=<list of devices>' Remember to separate devices by commas (<dev\_no>,<dev\_no>), syntax for ranges is <from\_dev\_no>-<to\_dev\_no> like

```
'insmod dasd dasd=FD00-FD0F,FD40,FD42,FD80-FD86'
```

Note: When updating, you have to load the dasd driver with the same DASD order as in the installed system - see documentation for further information.

4. Start installation or update with 'YaST'.

### Declaring then formatting DASD devices

We now defined the DASD devices that we wanted to make available to Linux (our VM ID had 6 minidisks defined of various sizes), so we entered the following at the shell prompt:

```
# insmod dasd dasd=201-206
```

We were notified of the loading of the DASD driver module:

```
Using /lib/modules/2.2.16/block/dasd.o
```

To verify that our devices were in fact available for use, we entered **cat /proc/dasd/devices** and received the following output:

```
0201(ECKD) at (94:0) is dasda:active at blocksize: 4096, 468000 blocks, 1828 MB
```

```
0202(ECKD) at (94:4) is dasdb:active at blocksize: 4096, 468000 blocks, 1828
MB
0203(ECKD) at (94:8) is dasdc:active at blocksize: 4096, 18000 blocks, 70 MB
0204(ECKD) at (94:12) is dasdd:active at blocksize: 4096, 36000 blocks, 140 MB
0205(ECKD) at (94:16) is dasde:active at blocksize: 4096, 36000 blocks, 140 MB
0206(ECKD) at (94:20) is dasdf:active at blocksize: 4096, 36000 blocks, 140 MB
```

Before we could proceed, we had to format the DASD volumes. We formatted three volumes, to be used as follows:

```
201          /dev/dasda - boot volume
202          /dev/dasdb - home directories
203          /dev/dasdc - swap space
```

At the shell prompt, we entered the command to format the first volume:

```
# dasdfmt -f /dev/dasda -b 4096
```

This resulted in the following output:

```
I am going to format the device /dev/dasda in the following way:
Device number of device : 0x201
Major number of device  : 94
Minor number of device  : 0
Labelling device        : yes
Disk label               : LNX1 x80405238
Blocksize                : 4096
```

```
---->> ATTENTION! <<----
```

```
All data in the specified range of that device will be lost.
```

```
Type "yes" to continue, no will leave the disk untouched: yes
```

```
Formatting the device. This may take a while (get yourself a coffee).
```

```
Finished formatting the device.
```

```
Rereading the partition table... done.
```

We repeated the **dasdfmt** command for the other two volumes:

```
dasdfmt -f /dev/dasdb -b 4096
dasdfmt -f /dev/dasdc -b 4096
```

## Software installation and configuration under YaST

Note that, in general, we use the *Tab* key to move to (and thereby highlight) an action to be taken, and we use the *arrow* keys to navigate within a list of items from which we need to make a selection. We entered the command **yast** to start the installation process.

**Important:** If you are telnetted in from a Windows desktop, having function keys recognized is often a problem. If it appears a function key is not recognized, try pressing **Ctrl-F**, and then the number of the function key. For example, to send an F4, press **Ctrl-F** and **4**.

The initial display is a language selection screen. We highlighted the language for our installation, selected **Continue**, and pressed **Enter** to proceed to the next screen.

Next we had to declare the source of the installation files. We were using an FTP server, so we chose **Installation from an FTP site**. An NFS server is also a useful vehicle for supplying access to the installation files.

Then we had to declare the type of installation to be performed. We would be performing a complete installation, so we selected **Install Linux from Scratch**.

We were prompted to specify our swap device. We used device 203 for our swap volume, so we selected `/dev/dasdc (0203)` and then proceeded as shown in Figure 5-1 on page 74.

**Note:** In this sample installation, we specified a swap device. During our other installation of SuSE Linux, which we performed in an LPAR, we chose not to specify swap space in order to illustrate that Linux can be installed and run without it. Swap space can always be added and/or modified later.

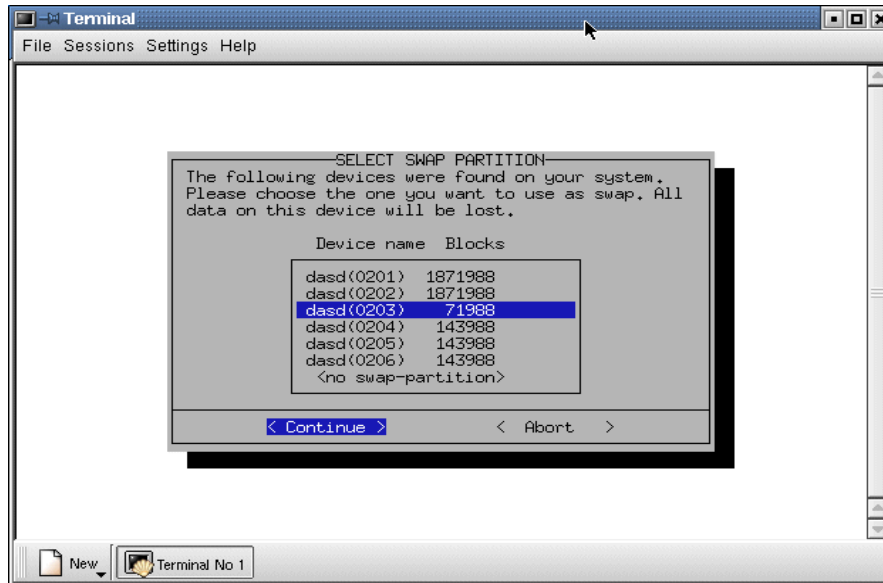


Figure 5-1 Specifying the swap device

The next prompt is somewhat misleading in that mainframe volumes are not partitioned. They currently are used in their entirety. So the real purpose for this prompt is to provide an opportunity to configure DASD for use by LVM. For this installation, we would not be using LVM, so we selected **Do not partition**.

Next, we navigate through a series of screens that associate the available minidisks with Linux file systems and their corresponding mount points. We use the **F4** function key to select the mount point, and the **F6** Function key to select the formatting option; see Figure 5-2 on page 75.



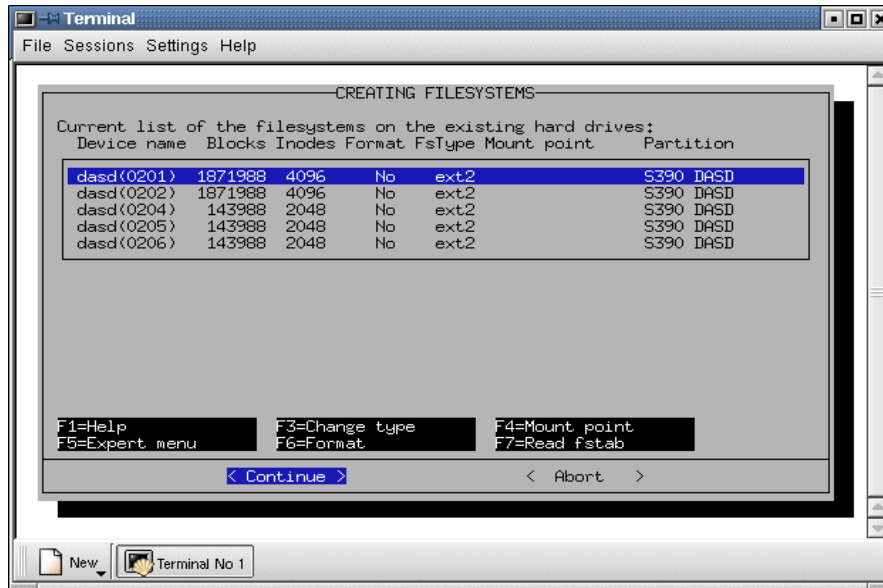


Figure 5-2 Creating the file systems panel before settings are applied

We selected the first DASD device on which to create a file system. We pressed the **F4** key in order to get the CREATING FILESYSTEMS pop-up window, as shown in Figure 5-3 on page 76. On this panel we set each file system's mount point and formatting option.

Our first volume at address 201 will contain the root file system, so we selected the line containing the single character **/**, and then selected **Continue**, as shown in Figure 5-3 on page 76.

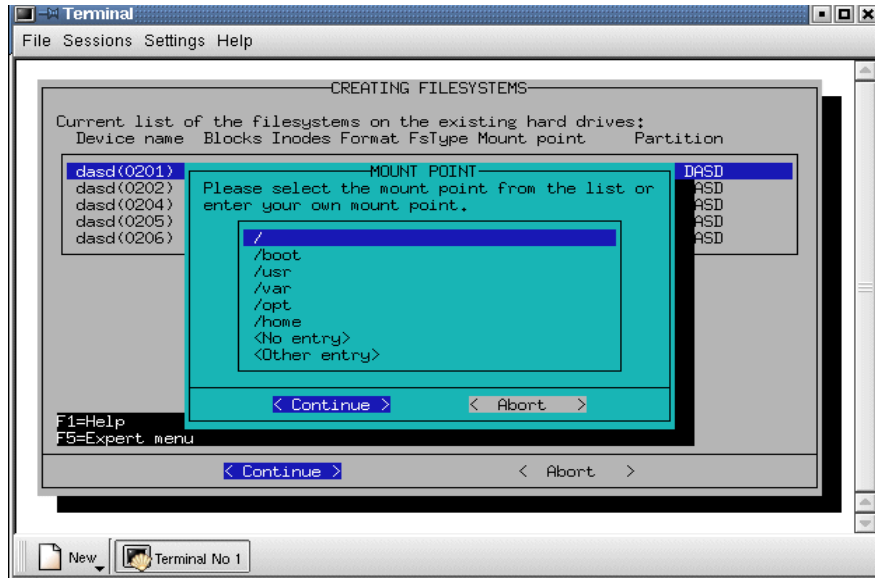


Figure 5-3 Specifying the mount point for a file system

In your case, you'll need to repeat the steps represented by Figure 5-2 and Figure 5-3 for all mount points that will be loaded with files during the installation. In this installation, we also set a mount point of /home for the DASD (minidisk) at address 202.

Depressing **F6**, we received a pop-up for selecting format mode. We selected the **Normal Format** option and then we selected **Continue**.

We then had set our mount points and format option for both volumes as shown in Figure 5-4 on page 77.

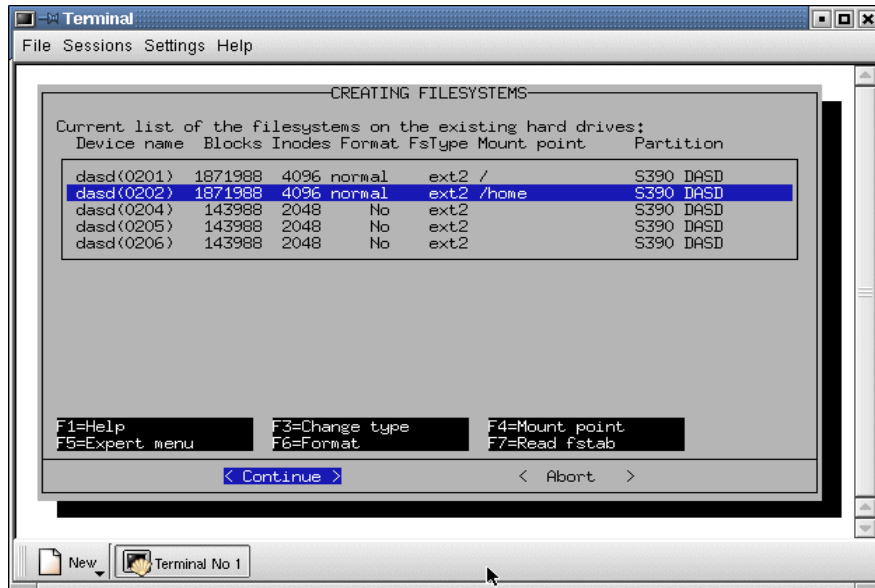


Figure 5-4 Creating the file systems panel after settings are applied

We now selected **Continue** to start the file system creation process. A pop-up window was displayed, requesting confirmation of the volume format request (note that both volumes are listed; one will be formatted, followed by the other). We selected **Yes**. A pop-up message was displayed as each file system was being created.

Having prepared the volumes, we were now ready to start the software installation phase. We used an FTP server as the source for the installation files. Since we would not be logging in anonymously, we supplied our user ID and password information; see Figure 5-5 on page 78.

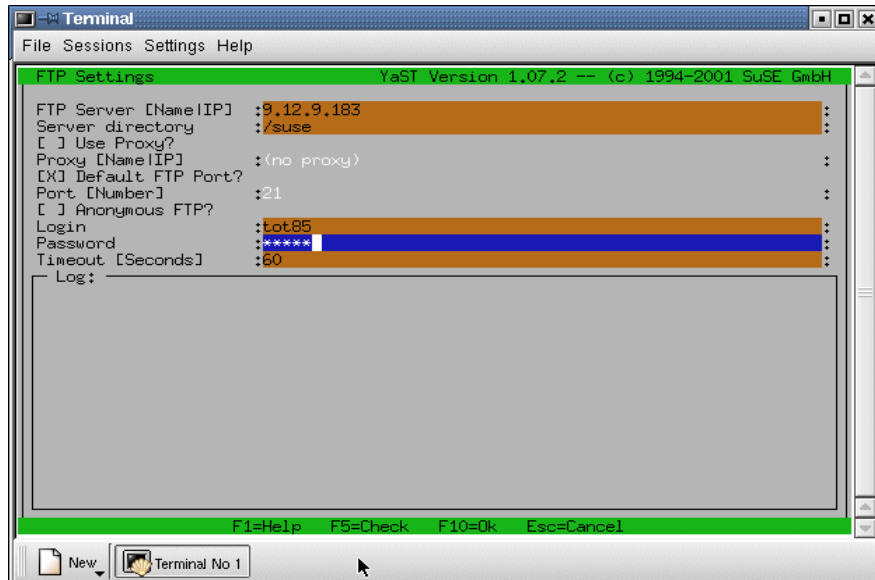


Figure 5-5 Specifying FTP server information

We pressed **F5** to ensure that our installation files were reachable. The installation program now attempts to open an FTP session and, if successful, to locate the installation files in the supplied server directory.

Note that our supplied installation directory information was `/suse`, which is not the top level directory of an installation CD hierarchy. Therefore, the installation routine searched downward in the directory structure to find where the needed files were located (Figure 5-6). Since we received a message line of `Settings Ok.`, we can proceed by pressing **F10**.

**Important:** If you are telnetted in from a Windows desktop, having function keys recognized is often a problem. If that is the case, pressing F10 is especially tricky on this screen. To avoid having a problem, move the cursor to the top of the Log: area with the Tab key and press **Ctrl-F** and **0**.

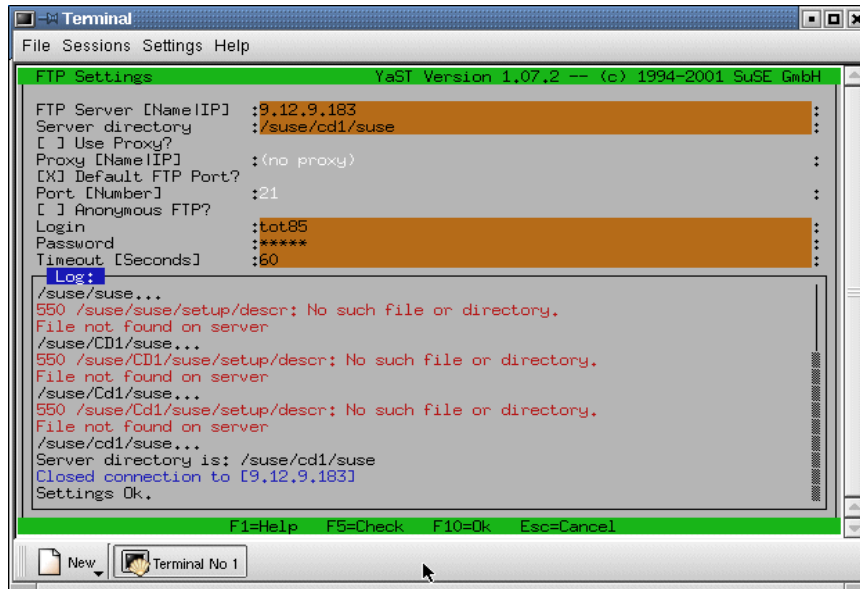


Figure 5-6 Confirmation that the FTP server is reachable

A selection list of pre-selected software groupings is displayed next. We would be installing the **SuSE Default system with Server Applications...**, so we selected our choice by pressing the space bar when the cursor was over that choice. An [X] was then shown in the square brackets. We selected **Add** to include this set of packages to be installed; see Figure 5-7 on page 80.

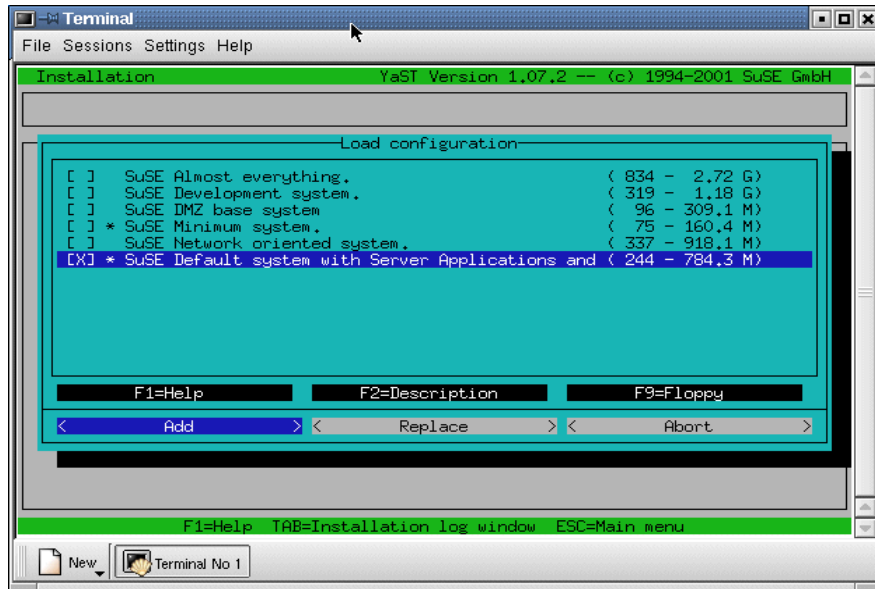


Figure 5-7 Selecting the software configuration

After selecting the software environment to be installed, we selected **Start installation** from the pop-up window, as shown in Figure 5-8.

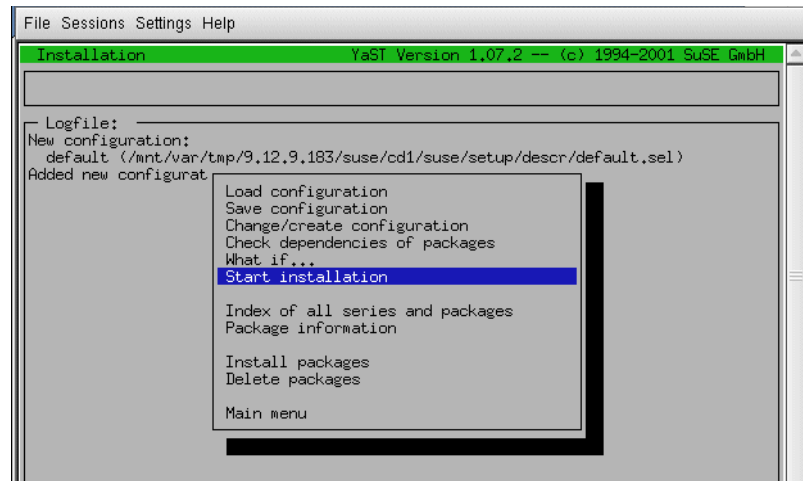
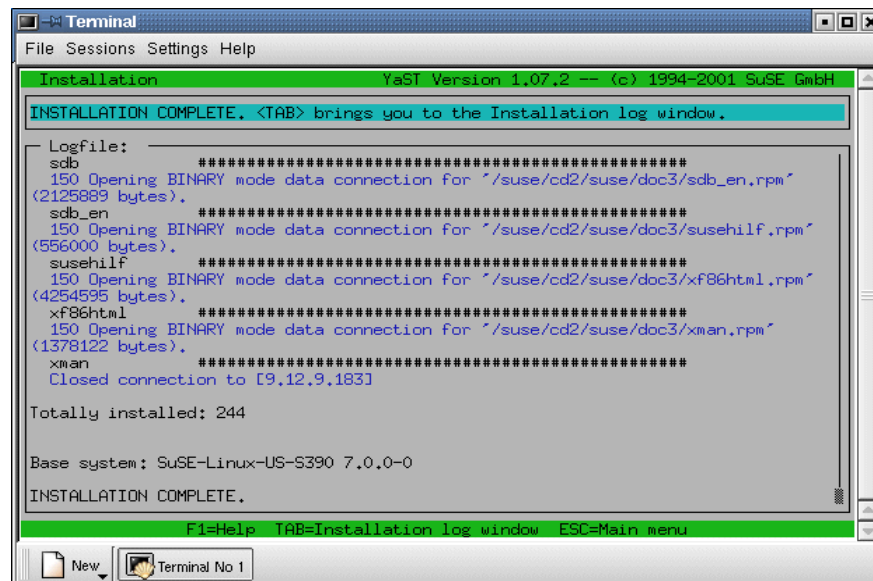


Figure 5-8 Starting the software installation

This process takes a number of minutes. As the packages are being installed, progress information is displayed which includes the name of the current package and the number of packages remaining to be installed.

At the end of the installation of the selected software environment, you will receive a display showing the total numbers of packages installed, the name of the base system that was installed, and finally the message `INSTALLATION COMPLETE.`, as shown in Figure 5-9. If the main pop-up menu is obscuring this status screen, it can be hidden by pressing the **Tab** key.



```
Terminal
File Sessions Settings Help
Installation YaST Version 1.07.2 -- (c) 1994-2001 SuSE GmbH
INSTALLATION COMPLETE. <TAB> brings you to the Installation log window.
Logfile:
sdb *****
150 Opening BINARY mode data connection for "/suse/cd2/suse/doc3/sdb_en.rpm"
(2125889 bytes).
sdb_en *****
150 Opening BINARY mode data connection for "/suse/cd2/suse/doc3/susehlf.rpm"
(556000 bytes).
susehlf *****
150 Opening BINARY mode data connection for "/suse/cd2/suse/doc3/xf86html.rpm"
(4254595 bytes).
xf86html *****
150 Opening BINARY mode data connection for "/suse/cd2/suse/doc3/xman.rpm"
(1378122 bytes).
xman *****
Closed connection to [9.12.9.183]
Totally installed: 244
Base system: SuSE-Linux-US-S390 7.0.0-0
INSTALLATION COMPLETE.
F1=Help TAB=Installation log window ESC=Main menu
New Terminal No 1
```

Figure 5-9 Software installation complete

We recommend that the software installation process be run one more time in order to ensure that no unsatisfied dependencies exist. To do this, we reselected the **Start installation** option from the main menu. We had no unresolved dependencies, so we got another `INSTALLATION COMPLETE.` message.

We then selected the **Main Menu** option in order to continue the installation process; see **Figure 5-10 on page 82.**

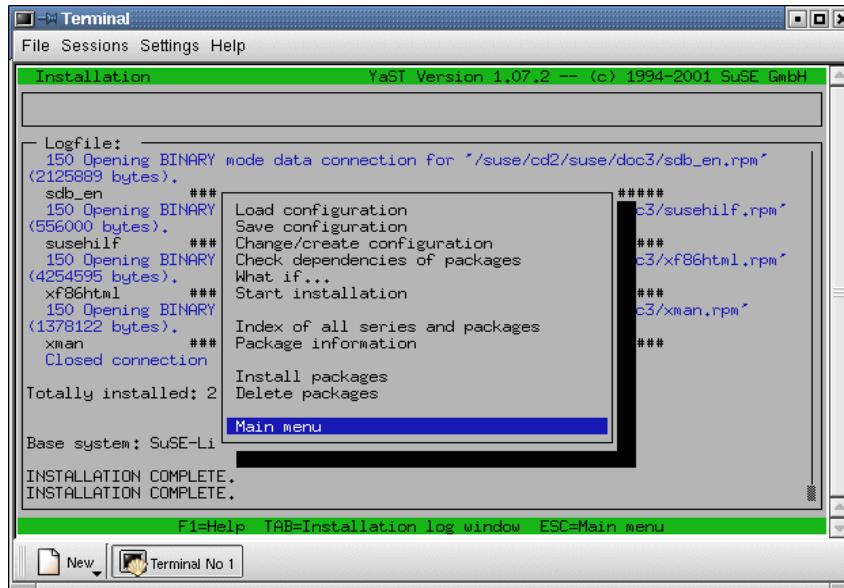


Figure 5-10 “Doubly” complete software installation

Next we had to select the kernel to be installed. We selected the default kernel (our only choice), followed by **Continue**. A pop-up message was displayed with the message Installing the selected kernel.

We now had the software installed on DASD and a kernel to run, but we still needed to customize our system. On the panel entitled TIME ZONE CONFIGURATION, we selected our local time zone and then **Continue**.

On the panel entitled ADJUSTMENT OF HARDWARE CLOCK, we were asked whether our hardware clock is set to Greenwich Mean Time (GMT, also referred to as UTC), or to local time. Our hardware clock is set to GMT, so we selected **GMT**.

Next we had to supply the DNS host name and domain name, as shown in Figure 5-11 on page 83. It is convenient to have this value and the IP address set up before installation, so that you can immediately start accessing the new host via its DNS name.



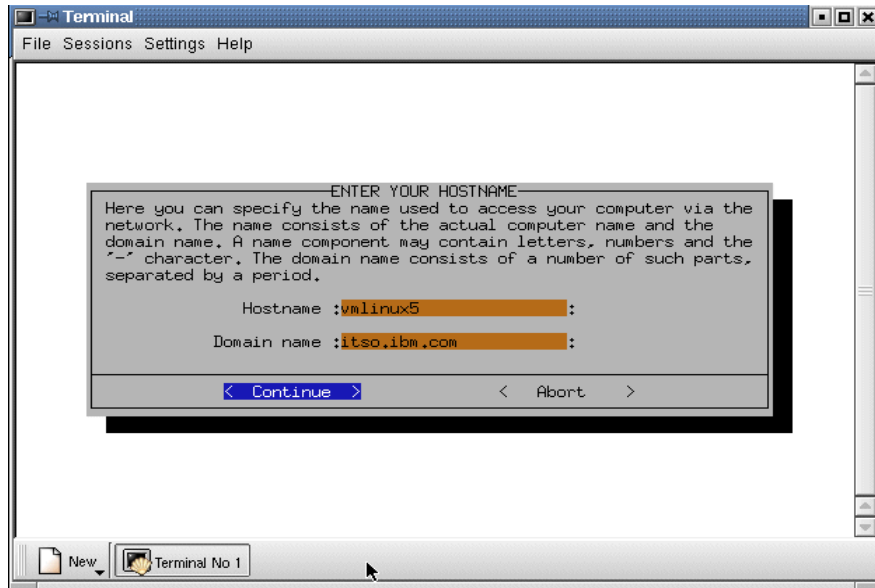


Figure 5-11 Prompt for host name and domain name

The next panel is entitled CONFIRMATION, and < **Real network** > should be chosen.

Because we had a fixed IP address and would not be using DHCP client services, we selected **No** to the question Do you want to use the machine as DHCP client?.

Referring to our installation worksheet (Table 5-3 on page 65), we got the values for our IP address, network mask, IP address of default gateway, and our MTU size that we need to supply in the ENTER THE NETWORK ADDRESSES screen as shown in Figure 5-12 on page 84. After entering the requested information, we selected **Continue**.

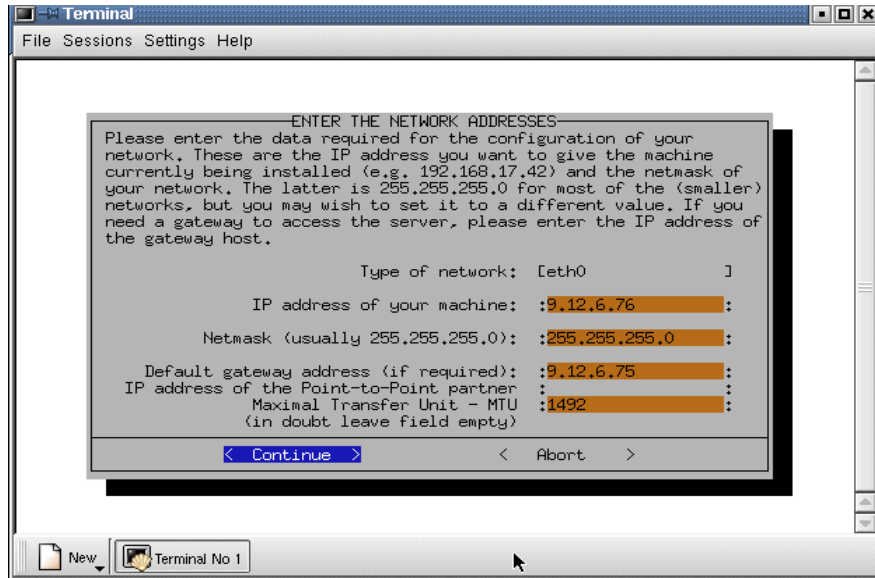


Figure 5-12 Specify the network interface and address

In the panel entitled START INETD?, we had to state whether we want **inetd** to be started automatically (the important services this daemon supplies are telnet and FTP). Since we did, we selected **YES** and then pressed **Enter** to continue.

In the panels entitled START THE PORTMAPPER? and START NFS-SERVER? we were asked whether these NFS services should be started at boot time. Because it is common to use NFS services on a server, we selected **YES** for both of these.

In the panel entitled ADJUST NEWS FROM-ADDRESS, we accepted the default of our fully-qualified DNS name and selected **Continue**.

Because we wanted to access a DNS name server, we replied **YES** to the panel entitled CONFIRMATION and were presented with the panel entitled NAMESERVER CONFIGURATION, as shown in Figure 5-13 on page 85.

From our installation worksheet, we supplied the required values and then selected **Continue**.

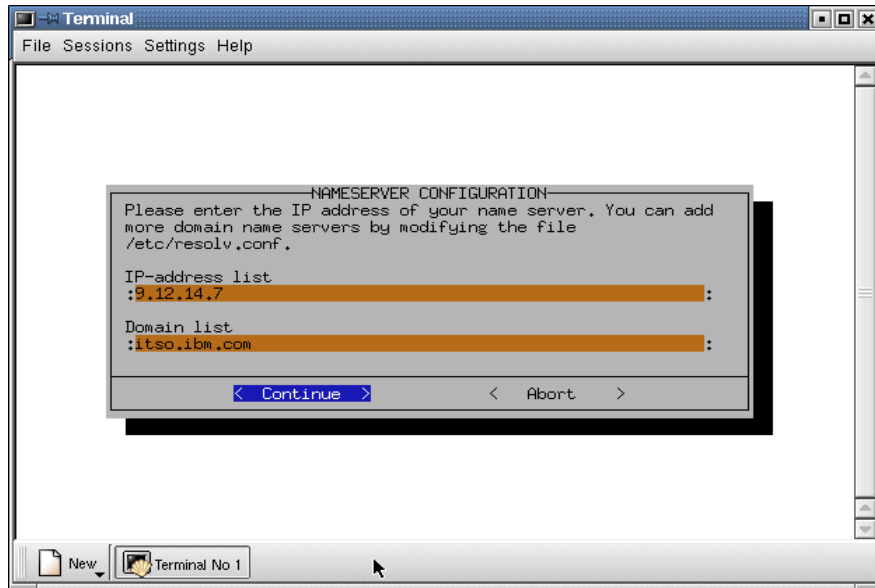


Figure 5-13 Specifying the address of our DNS and our domain name

In the panel entitled SENDMAIL CONFIGURATION, we selected the **Host with permanent network connection (SMTP)**, and then selected **Continue**.

At this point the installation script started the SuSE configuration tool, SuSEconfig, in order to apply the values that we provided; this is shown in Figure 5-14 on page 86.

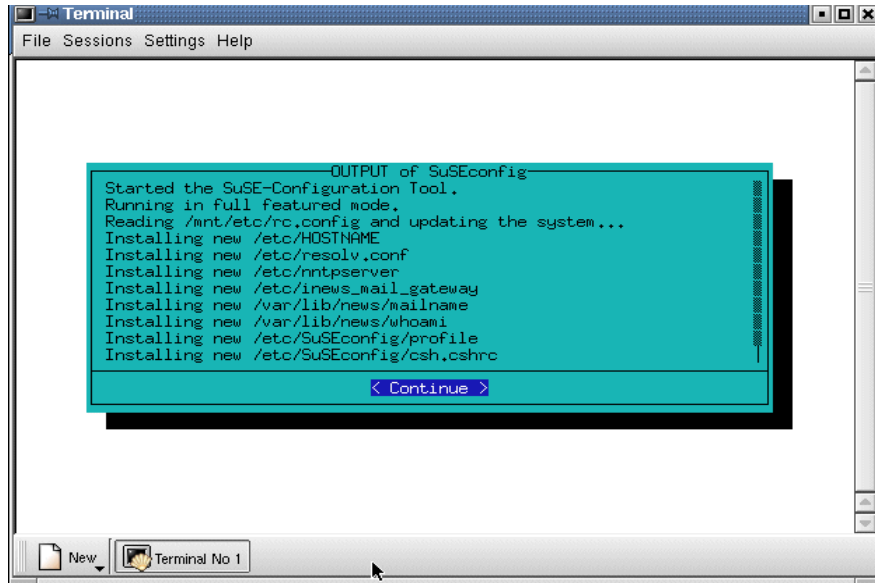


Figure 5-14 Progress messages from the configuration tool

Following the execution of the configuration tool, the message Clean-up is displayed. The final step in the installation process was to unmount the new file systems that we created and populated, though we received a message that /mnt is not mounted:

```
Running YaST...
umount: /mnt: not mounted
```

We then shut down in order to prepare for our initial boot of our newly create SuSE Linux system:

```
# shutdown -h now
```

## Booting the new system

Having completed the installation to disk, we needed to IPL our newly created system. We reconnected to our VM guest user ID. Because the root file system was written to the 201 minidisk, we issued the following command to boot our new system:

```
IPL 201 CLEAR
```

The start of our newly install system was announced by the following initial set of messages:

```
Linux version 2.2.16 (root@Tape.suse.de) (gcc version 2.95.2 19991024
(release))
```

```
#1 SMP Sun May 6 06:15:49 GMT 2001
Command line is: ro dasd=0203,0201,0202 root=/dev/dasdb1 noinitrd
```

```
We are running under VM
This machine has an IEEE fpu
Initial ramdisk at: 0x02000000 (16777216 bytes)
Detected device 2928 on subchannel 0000 - PIM = 80, PAM = 80, POM = FF
.....
```

**Note:** On the line that starts **Command line is:**, you will see that the DASD order has been changed. That happens because the installation process sets the swap device as the first DASD device.

```
201 /dev/dasdb was dasda          boot volume
202 /dev/dasdc was dasdb          home directories
203 /dev/dasda was dasdc          swap space
```

After the system has completed booting, we were asked to reset the root password. After the password had been entered and verified, additional configuration steps were started via the **SuSEconfig** command:

```
Started the SuSE-Configuration Tool.
Running in quick mode.
Reading /etc/rc.config and updating the system...
Installing new /etc/SuSEconfig/profile
Installing new /etc/SuSEconfig/csh.cshrc
Executing /sbin/conf.d/SuSEconfig.pam...
Installing new /etc/pam.d/login
Installing new /etc/pam.d/rlogin
Executing /sbin/conf.d/SuSEconfig.sendmail...
Installing new /etc/sendmail.cf
Rebuilding /etc/aliases.db.
/etc/aliases: 37 aliases, longest 10 bytes, 420 bytes total

setting /bin/mount to root.root 4755.
setting /bin/umount to root.root 4755.
setting /usr/lib/mc/bin/cons.saver to root.root 4755.
setting /usr/sbin/suexec to root.root 4755.
setting /etc/permissions to root.root 644.
setting /etc/permissions.secure to root.root 644.
setting /etc/permissions.easy to root.root 644.
setting /etc/permissions.paranoid to root.root 644.
Finished.
```

-----

Now scripts have to be started. They will be started in one minute. You can find a log file under `/var/log/Config.bootup`. It will also be printed on console 9.

You can now already use your system. If you shut down the system before the scripts are finished, they are executed again at the next system startup.

Press <RETURN> to continue...

Have a lot of fun!

Your SuSE Team

All installation tasks have been performed when you see this message:

**Give root password to login:**

This completes the illustrative installation under VM. See Chapter 6, “Customizing and using SuSE Linux” on page 91, for examples of customizing and using your SuSE Linux system.

## 5.3 SuSE installation in an LPAR

Following is a high-level checklist for installing SuSE Linux in an LPAR:

1. Read 3.2, “Prepare the LPAR for Linux” on page 31.
2. Plan your DASD configuration:
  - a. Which volumes are going to be used
  - b. What file systems are going to be on what volumes
  - c. How big those volumes need to be
3. Complete the worksheet in Table 5-3 on page 65.
4. Prepare media, depending on one of the following IPL methods:
  - a. From a tape - see 3.4.3, “Booting from tape” on page 44.
  - b. From the HMC CD-ROM or an FTP server - see 3.4.2, “Booting from HMC CD-ROM or FTP server” on page 35
  - c. From an ICKDSF bootstrap - see 3.4.1, “Using the ICKDSF bootstrap” on page 34
5. IPL Linux, using the appropriate media depending on IPL method.
6. Continue with the common portion of the install documented in 5.2.4, “Logging in via telnet and completing the install via YaST” on page 71.

## 5.4 Installing SuSE under VIF

Since SuSE was the initial Linux image created during our installation of the S/390 Virtual Image Facility for Linux (VIF), see Chapter 16, “VIF” on page 267 for documentation on how to install SuSE under VIF.

## 5.5 Assessing the SuSE installation experience

We encountered no problems of note in performing any of our various installations of SuSE Linux Enterprise Server for S/390 . SuSE was typically the distribution that we used when creating systems to test various functions and capabilities on S/390. This was due to its completeness and to the maturity of SuSE distribution.







## Customizing and using SuSE Linux

In this chapter we describe many aspects of customizing and using "SuSE Linux Enterprise Server for S/390" . The vehicle for performing most system administration functions is the tool YaST. YaST stands for "Yet another Setup Tool".

## 6.1 Using YaST

YaST not only provides a convenient installation method, but also simplifies many system administration and system management tasks. Starting YaST from the command line or in KDE will allow you to manage the installed packages on your system, update your system, and perform system administration tasks. Enter **yast** to start the program and have its main menu displayed (see Figure 6-1 on page 92).

Note that the minimum terminal size of YaST is 80 characters in width and 25 lines in depth. Invoking YaST from a terminal or window set to display less than 25 lines will result in a warning message that some functions may fail.

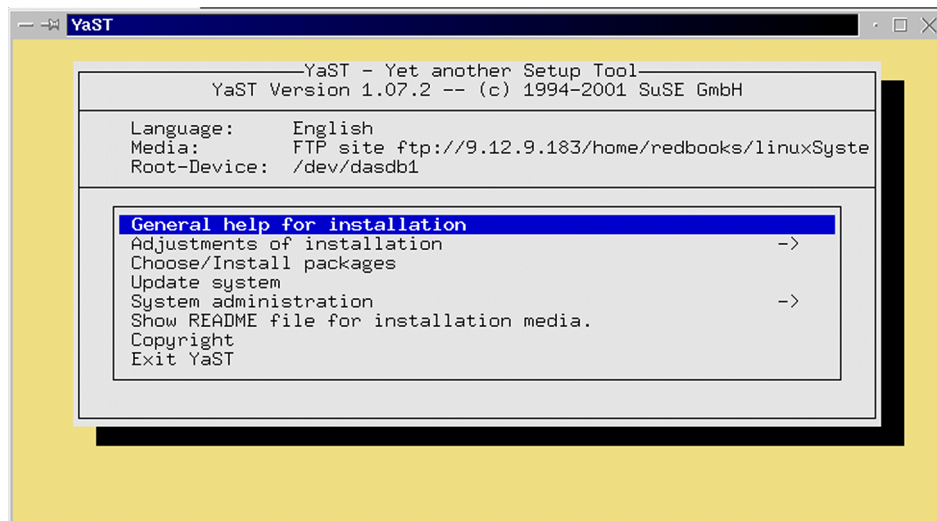


Figure 6-1 YaST main menu

YaST allows you to adjust your installation by specifying the FTP server to use, target volumes, languages, etc. Disk management can also be accomplished with YaST by using the Logical Volume Manager option; see 6.5, “Setting up LVM with YaST” on page 108.

YaST allows you to perform the following system administration tasks with an intuitive menu system:

- ▶ Install new packages from the specified media
- ▶ Retrieve package updates from SuSE's FTP server
- ▶ Add devices
- ▶ Specify which kernel to boot
- ▶ Configure the network

- ▶ Perform user/group administration tasks
- ▶ Create backups
- ▶ Manage system security

In addition to navigating the YaST menus, you can open YaST with the `--mask` argument and then jump (or “fast path”) to YaST functions.

For example, after installing the system, we recommend that you create a non-root user on your system. To jump to the user administration section of YaST, issue:

```
# yast --mask user --autoexit
```

This takes you directly to the panel shown in Figure 6-2 on page 93:

```

┌──────────────────────────USER ADMINISTRATION──────────────────────────┐
│ In this dialog you can get information about existing users, create new  │
│ users, and modify and delete existing users.                            │
│                                                                           │
│ User name                       :newuser      :                        │
│ Numerical user ID                :515         :                        │
│ Group (numeric or by name)       :users       :                        │
│ Home directory                   :/home/newuser :                        │
│ Login shell                      :/bin/bash   :                        │
│ Password                         :*****     :                        │
│ Re-enter password                :*****     :                        │
│ Access to modem permitted        [ ]          :                        │
│                                                                           │
│ Detailed description of the user                                         :
│ :                                                                           │
│ F1=Help                          F3=Selection list                    F4=Create user
│ F5=Delete user                    F6=Password times                   F10=Leave screen
└──────────────────────────────────────────────────────────────────────────┘

```

Figure 6-2 YaST fast path to USER ADMINISTRATION menu

Supply a user name and password, and YaST will fill in the defaults for the rest of the fields.

On multiuser systems, it may be a good idea to restrict the amount of disk space each user can consume by using the quota utility. For an explanation of quota configuration, refer to 21.1.8, “Quotas” on page 400.

To save some keystrokes, create a small script called `yast`, as follows:

```
# /sbin/YaST --mask $1 --autoexit
```

\$1 takes the first argument from the command line (in our case, a mask argument) and substitutes the value into the command.

Make the script executable and copy it to /usr/local/sbin. Rename /sbin/yast to /sbin/yastold. This allows you to simply type: **yast** argument .

Table 6-1 lists the **--mask** options that are applicable to Linux for S/390:

*Table 6-1 YaST mask arguments*

<b>Main menu</b>	<b>--mask argument</b>
General help for installation	help
Adjustments of installation	
Choose/Install packages	install
Show README file for installation media	readme
Copyright	copyright
<b>Adjustments of installation</b>	
Select language	language
Select keymap	keymap
Select installation medium	medium
Set target partitions/file systems	filesys
Configure the Logical Volume Manager	lvm
<b>System administration</b>	
User Administration	user
Group administration	group
Create backups	backup
Security settings	security
Set the time zone	timezone
Change configuration file	rcconfig
<b>Integrate hardware into system</b>	
Mouse configuration	mouse
Configure networking device	netcard
<b>Kernel and boot configuration</b>	

Select boot kernel	kernel
<b>Network configuration</b>	
Network base configuration	network
Change host name	name
Configure network services	services
Configuration nameserver	nameserver
Configure YP client	ypclient
Configure sendmail	sendmail
Administer remote printers	netprinter
Connect to printer via Samba	smbprinter

## 6.2 Samba installation

Samba is an open source implementation of the Server Message Block (SMB) protocols. Samba allows many popular platforms to provide NT-like file and print services to SMB clients, such as Windows 95/98 and Windows NT. This can be very useful in a Linux for S/390 environment, providing these services to clients via a virtual server environment as opposed to a large population of physical servers.

The Samba server is composed of two daemons: `smbd` provides the file and print services to clients, while `nmbd` works with NetBIOS over IP name service requests and participates in the browsing protocols that make up the Windows Network Neighborhood.

### 6.2.1 Install from an RPM

Samba is usually installed by default with SuSE; if you need to add it or reinstall, the Samba RPM can be found on the SuSE installation CD 1. It is named `samba-2.0.8-2.s390.rpm`. It can be installed from the command line via the following command:

```
# rpm -ivh samba-2.0.8-2.s390.rpm
```

### 6.2.2 Samba configuration

Here is a summary of the files used for Samba on SuSE:

Name	Samba
------	-------

```

daemon          /usr/sbin/smbd
                /usr/sbin/nmbd
configuration files /etc/smb.conf
startup script   /etc/rc.d/smb
log files       /var/log/log.smb
                /var/log/log.nmb

```

## Samba configuration methods

Several methods can be used to configure Samba, but the two most common are editing the `/etc/smb.conf` file (as in the following example), and using the Samba Web Administration Tool (SWAT, see 6.2.4, “Using SWAT” on page 100).

### **Configuration example**

Here we show you how to configure a simple Samba configuration to allow a Windows workstation to access a shared directory on Linux for S/390 (file server).

Edit the following parameters in the [global] section of the Samba configuration file, `/etc/smb.conf`:

1. Change the `workgroup = TUX-NET` to the workgroup of your Windows client.  
If you would like your Samba server to take part in an NT domain, you may also need to change the domain.
2. `ctc` is not a default interface that Samba looks for when starting up, so for `nmbd` to start, you need to add `interface = <your-ip>/24`. (However, this line will not be needed if you are using an ethernet device.)
3. Uncomment: `encrypt passwords = yes`.
4. Add a line after `encrypt passwords` with: `password level = 8`.
5. Add another line with: `username level = 8`.

Example 6-1 illustrates these changes:

*Example 6-1 smb.conf changes*

---

```

;
; /etc/smb.conf
;
; Copyright (c) 2001 SuSE GmbH Nuernberg, Germany.
;
[global]
workgroup = TUX-NET      <--- 1
interface = 9.12.9.184/24 <--- 2
guest account = nobody
keep alive = 30
os level = 2
kernel oplocks = false

```

```
security = user

; Uncomment the following, if you want to use an existing
; NT-Server to authenticate users, but don't forget that
; you also have to create them locally!!!
; security = server
; password server = 192.168.1.10

encrypt passwords = yes <--- 3
password level = 8 <--- 4
username level = 8 <--- 5

printing = bsd
printcap name = /etc/printcap
load printers = yes

socket options = TCP_NODELAY

map to guest = Bad User
```

---

Save your work and exit. Test your smb.conf syntax by running the **testparm** command.

## Starting Samba automatically

With a default SuSE system, Samba is installed but not running. In order for Samba to be started automatically, the `/etc/rc.config` file must be updated, either manually or by using YaST.

You can update this file as follows:

1. Open `rc.config` with `vi` and search for SMB. You will find the line:  
`START_SMB="no"`; change this to: `"yes"`.  
  
You can also issue: **yast --mask rcconfig --autoexit** (or **yast rcconfig**, assuming you created the script mentioned in 6.1, "Using YaST" on page 92").
2. Press **F4** and search for SMB. When you see `START_SMB` highlighted, press **F3** and change the field to: `yes`.
3. **F10** exits.

Refer to Figure 6-3 on page 98.

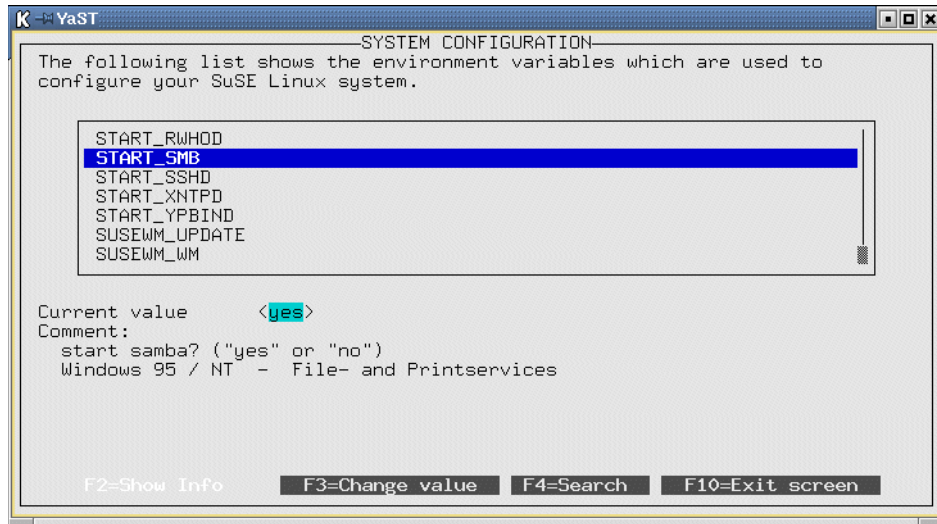


Figure 6-3 Setting Samba to execute at startup

Issue **rcsmb restart** (this is the same as issuing **/etc/rc.d/smb restart**, but it has been placed into **/usr/sbin** as a script); you should see the following messages:

```
Shutting down SMB services:           done
Starting SMB services:                done
```

Issue **rcsmb status** to ensure that **smbd** and **nmbd** are running:

```
Checking for service smb: OK OK
```

### 6.2.3 Samba password encryption

To supply the password to Samba, you need to know whether the password going over the network is encrypted. Most Windows clients (Windows 95 OEM2 and later, and Windows NT, SP3 and later) now encrypt passwords going over the network.

You can either turn off the encryption via a registry setting, or you can instruct Samba to create a password file that maintains encrypted passwords. The latter is recommended as it is more secure and it can be burdensome to add a setting to the Windows registry on all clients. For additional information, see the files **Win95.txt**, **WinNT.txt**, **ENCRYPTION.txt** located in the directory **/usr/share/doc/packages/samba/textdocs/**.



We recommend that you use the `smb.conf` parameters `password level` and `username level` when connecting with a Win95 or earlier client. These parameters test the user ID and password in several combinations of mixed case. This ensures that a user ID or password that has been converted to upper case will be handled appropriately.

To have Samba handle password encryption, do the following:

1. Ensure changes 4, 5, and 6 from Example 6-1 on page 96 have been made.
2. Create the `smbpasswd` file. The script `mksmbpasswd.sh` is provided to generate the `smbpasswd` file from the `/etc/passwd` file.
  - a. Do `chmod +x` on `/usr/lib/samba/scripts/mksmbpasswd.sh`.
  - b. Execute the following command:

```
cat /etc/passwd | /usr/lib/samba/scripts/mksmbpasswd.sh > /etc/smbpasswd
```

That command will pipe the contents of `/etc/passwd` to the `mksmbpasswd` script. The resulting password file will contain all Xs in the password field. It's a good idea to remove all of the accounts you will not use for Samba (ie. `root`, `bin`, `daemon`, and so on).

Once the `smbpasswd` file has been created, you still need to actually set the `smb` password for each user in the `smbpasswd` file. (This can be done either by the individual users, or by the root user.) Issue the command `smbpasswd <userid>` and then enter the password twice, when prompted.

Note: If you try to add a user who is not already a normal user of the system, you will receive an error, so ensure that those who need Samba access are first created with YaST or `useradd`. Also, ensure that a home directory exists for the user ID, and that permissions and ownership are correct.

To access the share from Windows, run `net use drive_letter: \\hostname\homes` from a command prompt. This will give you a drive, `drive_letter`, which is the home directory for the user on your Linux system. The equivalent can be done by mapping a network drive with Windows Explorer.

## Supplying the user ID

Windows NT is easier than 9x because it allows you to supply the user ID. From the Map Network Drive dialog, there is an additional text field entitled Connect As. If the user ID with which you logged onto NT is different from that on Linux, you can specify your Linux user ID in this field. Similarly, the DOS `net use` command allows a `/USER:username` flag. On Windows 9x, you cannot supply a user ID different from the one you logged on with.

Therefore, we recommend that your Windows networking and Linux user IDs be the same. If this is not feasible, there is an alternative: Samba allows the user ID to be “tacked on” to the share name by appending it after a percent (%) character delimiter. For example, accessing the share `\\linux390\myShare%mikem` will allow the share `myShare` to be accessed with the user ID `mikem`.

## Creating a public share

To create a public share, duplicate the `[home]` section with your share information; that is, `[fileServer]`. Ensure that you change the `path =` statement to `/path/name`.

Change `browseable = no` to `yes`.

```
[fileServer]
comment = user files
browseable = yes          <--- yes
read only = no
path = /pub
create mode = 0750
```

## 6.2.4 Using SWAT

There is a second option for configuring Samba which is called the Samba Web administration tool (SWAT). On the default install, SWAT is enabled. To run SWAT, specify the following URL in your browser:

```
http://<your.domain.name>:901/
```

This URL forces communication through port 901 rather than the well-known http port of 80. SWAT is elegantly designed in that it is not a daemon. Rather, when requests come in on port 901, it is forked via `inetd`, replies with a Web page, and terminates. Figure 6-4 on page 101 shows the SWAT welcome page.

If you do not get to this page, ensure that the following lines exist `/etc/inetd.conf` and `/etc/services` and are not commented out via a leading `#`:

```
$ grep ^swat /etc/inetd.conf /etc/services
/etc/inetd.conf:swat    stream tcp  nowait.400  root    /usr/sbin/swat swat
/etc/services:swat    901/tcp    # XXX Samba Web Administration Tool
```

When prompted for a user ID and password, login as a user with root privileges. You can now navigate through the tool and configure your shares.

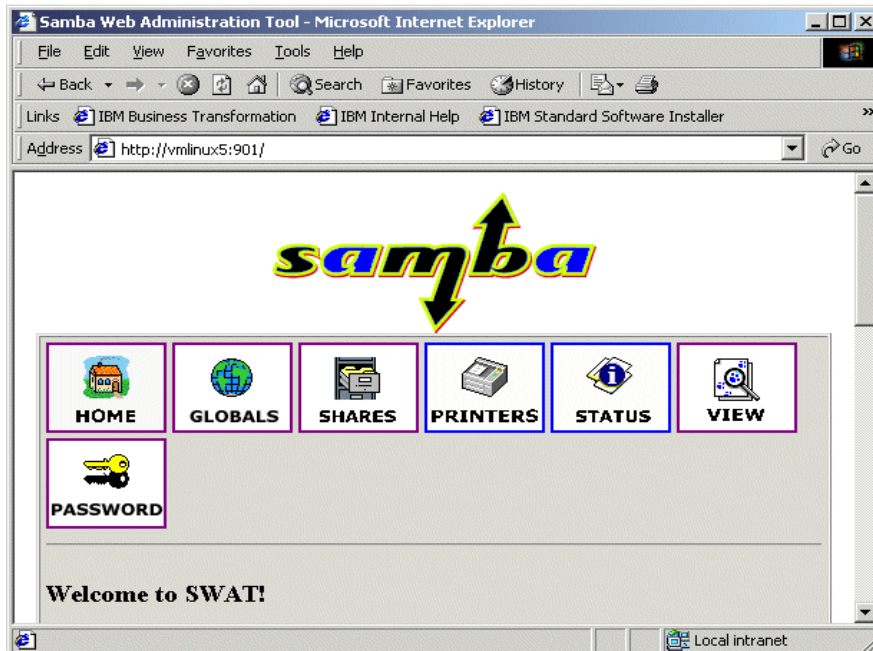


Figure 6-4 SWAT welcome page

## 6.2.5 Samba documentation

Samba is a powerful and flexible application, which you can learn about by accessing Samba documentation via `usr/share/doc/packages/samba`. In addition, a complete softcopy version of the O'Reilly book *Using Samba* is packaged with Samba and can be found in HTML format at the bottom of the SWAT welcome page; see Figure 6-5 on page 102.

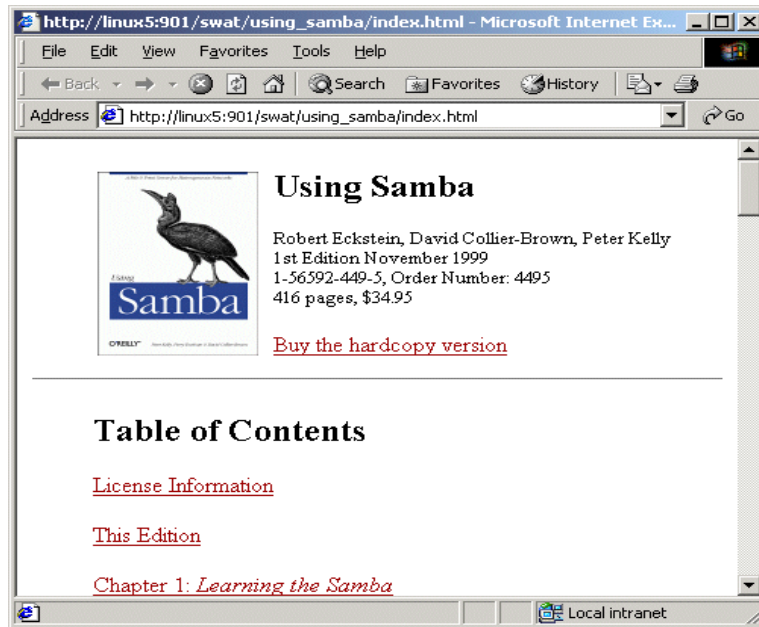


Figure 6-5 Online copy of “Using Samba”

## 6.3 Apache Web server

The SuSE default system includes the Apache Web server. Apache, one of the most popular Web servers on the market today, is based on the NCSA public domain httpd server and has become a major project in the open source community. At the time of writing, it is estimated that 62% of the world’s Web servers are running Apache (netcraft.com).

After the SuSE installation, Apache will be installed and running. If you plan to use Apache, proceed with configuration. However, if you have no need for a Web server, you can turn Apache off by removing the rc script (/etc/rc(1,2, & 3).d/S20apache) and turning the server off (**rcapache stop**). Better yet, you can uninstall Apache by using the **rpm -e** command.

### 6.3.1 Apache configuration

The following is a summary of the files used for Apache on SuSE.

Name	Apache
daemon	/usr/sbin/httpd

```

configuration files /etc/httpd/httpd.conf
                   /etc/httpd/srm.conf
                   /etc/httpd/access.conf
                   /etc/mime.types or /etc/httpd/mime.types
startup script     /etc/rc.d/apache
lock file          /var/lock/subsys/httpd
log files          /var/log/httpd/access_log
                   /var/log/httpd/access_log.1
                   /var/log/httpd/error_log
                   /var/log/httpd/error_log.1

```

By default, all configuration files are included in `/etc/httpd`. You will find three configuration files: `httpd.conf`, `srm.conf`, and `access.conf`. All three files are combined in `httpd.conf` (however, `srm.conf` and `access.conf` are maintained for historical reasons, and do not contain any configuration data).

**Note:** While `srm.conf` and `access.conf` do not contain configuration data, they must be present in order for the server to operate.

Initially, Apache will display a default `index.html` located in `/usr/local/httpd/htdocs`. This is the Web page that will be shown when a client (browser) comes to your site. This default file can be replaced with a custom home page.

Observe the following in `/etc/httpd/httpd.conf`:

- ▶ Note the `ServerRoot` entry - this is the top of the directory tree under which the server's configuration, error, and log files are kept.
- ▶ The first `<Directory>` entry restricts Apache's access to the file system. From here, any access needs to be specified with additional directory and file containers.

### 6.3.2 Password-protecting a directory

In your environment, you may have a requirement to publish some content on your Web site that is restricted to some users. One method of restricting access to particular directories on your server is by using the `.htaccess` file.

To limit access in this way, begin by placing a file named `.htaccess` in the restricted directory, `/usr/local/httpd/secret` in our example. The format of the `.htaccess` file looks like this:

```

AuthUserFile etc/httpd/.htpasswd
AuthName "Restricted files"
AuthGroupFile /dev/null
AuthType Basic

```

```
<Limit GET>
require valid-user
</Limit>
```

The file `.htaccess` points to `.htpasswd`, which stores the user IDs and passwords of all users who have permission to access the restricted documents.

Create the `.passwd` file in `/etc/httpd` with one user named `tux`. The `-c` flag instructs `htpasswd` to create a new password file:

```
# cd /etc/httpd
# htpasswd -c .htpasswd tux
New password:
Re-type new password:
Adding password for user tux
```

One final step must be executed for this to work. Edit `httpd.conf` and replace `AllowOverride = None` with `AllowOverride = All` under the directory container for your document root.

```
<Directory "/usr/local/httpd/htdocs">

#
# This may also be "None", "All", or any combination of "Indexes",
# "Includes", "FollowSymLinks", "ExecCGI", or "MultiViews".
#
# Note that "MultiViews" must be named *explicitly* --- "Options All"
# doesn't give it to you.
#
    Options Indexes -FollowSymLinks -Includes MultiViews
#
# This controls which options the .htaccess files in directories can
# override. Can also be "All", or any combination of "Options", "FileInfo",
# "AuthConfig", and "Limit"
#
AllowOverride All    <----Allows .htaccess to override httpd.conf settings
```

Now access the restricted Web page with a browser; you will be prompted for a user ID and password as shown in Figure 6-6 on page 105:



Figure 6-6 Restricted Web site user ID and password prompt

Figure 6-7 illustrates successful access.



Figure 6-7 Access to restricted files

## Apache documentation

The Apache User's Guide is available in the directory `/usr/share/doc/packages/apache/manual`. Because these are HTML files, to view them you will either need a browser running locally and displaying to your desktop, or you will need to make the directory available with Samba.

To try accessing the Apache documentation via Samba, we added the following to the `/etc/smb.conf` file on the Linux server with an IP address of 9.12.6.53:

```
[apachedoc]
    path = /usr/share/doc/packages/apache/manual
    writeable = No
```

After Samba reread this file, we were able to map the network drive using the share name `\\9.12.6.53\apachedoc`.

We were then able to view the documentation by opening the file `index.html` as shown in Figure 6-8.

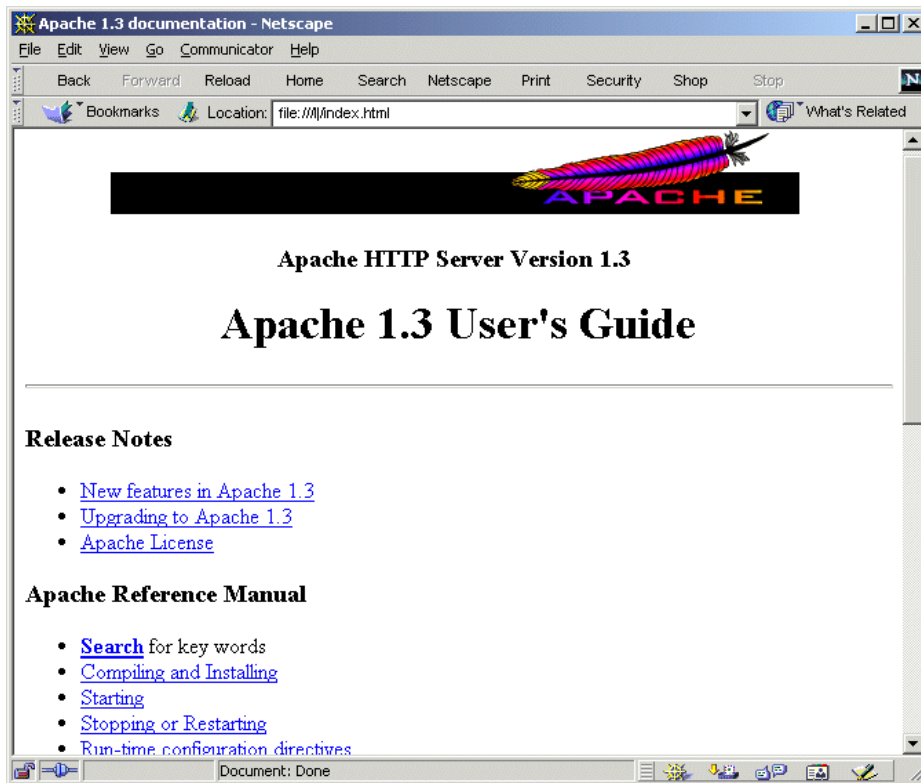


Figure 6-8 Accessing Apache documentation via Samba and a browser



### 6.3.3 Enabling CGI scripts

If you have a requirement to run CGI scripts, you must tell Apache where the scripts are located. Modify the ScriptAlias entry:

```
ScriptAlias /cgi-bin/ "/usr/local/httpd/cgi-bin/"
                URL part                cgi program locations
```

## 6.4 e-mail

SuSE Linux uses sendmail as the mail transfer agent (MTA). It can either be configured with YaST, or you can edit the configuration files directly. The following example shows how to configure sendmail with an external mail host in order to be able to send and receive e-mails between your Linux system and the outside world.

**Note:** DNS must be set up with MX records to have mail for your domain delivered to your Linux system.

Edit /etc/sendmail.cf; lines 71 and 72 (approximately) should contain a comment and a configuration parameter:

```
# "Smart" relay host (may be null)
DS
```

Put in the DNS name of one of your company's mail forwarding systems. If we use smtp.mycompany.com as an example, the file would be changed to this:

```
# "Smart" relay host (may be null)
DSsmtp.mycompany.com
```

Save your change, and exit the editor.

Send the "1" signal to sendmail to tell it to reread its configuration file. Determine the process id (PID) by doing `ps ax | grep sendmail`. Then issue a `kill -1 <pid>` command, where <pid> is the process ID from the previous command.

At this point, sendmail is configured and you should be able to send mail to, and receive mail from, other hosts.

## 6.5 Setting up LVM with YaST

SuSE supports the use of the Logical Volume Manager (LVM). LVM is a way to aggregate multiple DASD devices in order to create larger “logical” volumes. Without LVM, the largest file system is limited to largest real or emulated ECKD volume attached to the system. The largest possible volume (emulated or real) is an IBM 3390 model 9, which has a capacity of approximately 7.2 GB when formatted with 4,096 byte blocks.

In the following section, we demonstrate how to implement LVM via the YaST tool. For a more exhaustive explanation of LVM, refer to Chapter 17., “Logical Volume Manager” on page 301.

LVM has been fully integrated with YaST since SuSE Linux 6.3. The LVM in SuSE Linux is Heinz Mauelshagen’s implementation. The home page is now available on the Web:

<http://www.sistina.com/lvm/>

We recommend that you review this site and become familiar with LVM concepts before proceeding with configuration.

In this example we have six DASD volumes:

- ▶ Volume 203 is used for the SuSE Swap device.
- ▶ Volume 202 is used for the file system /.
- ▶ Volume 201 is used for the file system /usr.
- ▶ Volumes 204 through 206 will be used for LVM. Initially we will create a Volume Group (VG) of two volumes, and then expand it to include a third. The steps to reduce a Volume Group are similar and intuitive with the YaST menus.

We need to add volumes 204 through 206 to an existing system. We can do this by updating `/boot/parmfile` with the new volume numbers, running the `silob` command (`silob -d /dev/dasdb -f image -p parmfile -b bootsectfile`), and then rebooting. We perform a `dasdfmt` on each volume once the volumes are visible to the system.

We launch YaST and navigate to **Configure the Logical Volume Manager**, as shown in Figure 6-9 on page 109.

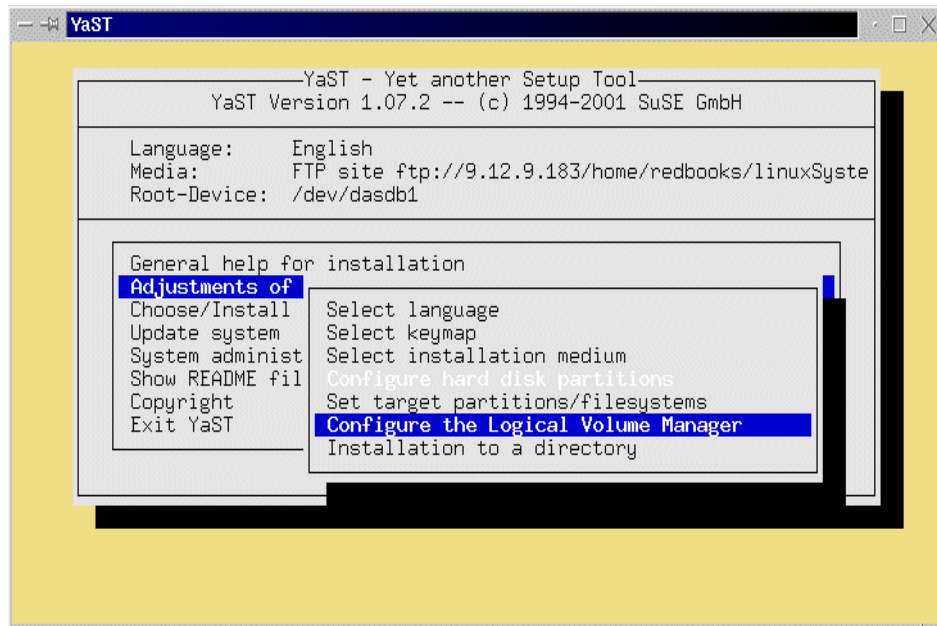


Figure 6-9 Launching Logical Volume Manager configuration

At this point, we select **<Create new Volume Group>** and choose the volumes we want to add. We will use `dasdd1` and `dasde1` initially.

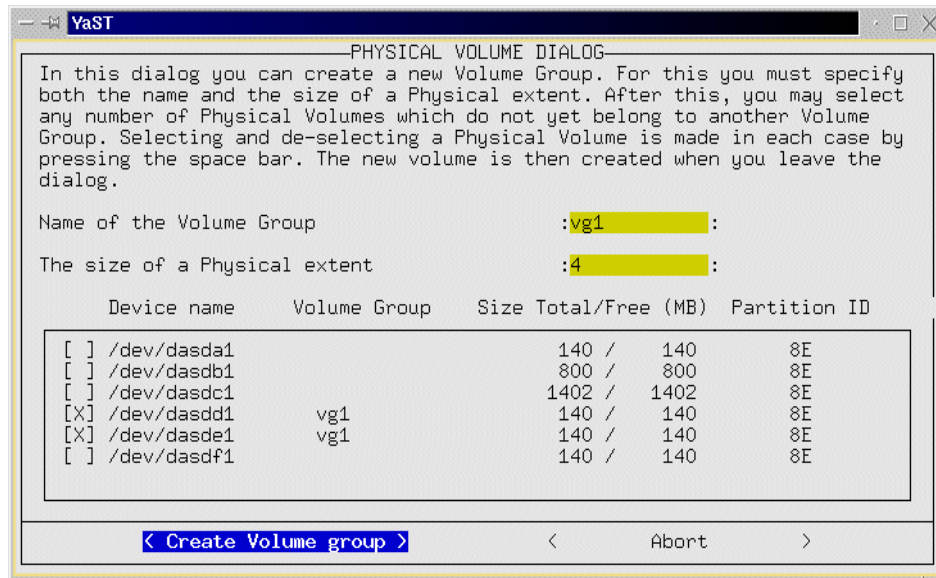


Figure 6-10 Volume selection

The Volume Group **vg1** is created; we press **F3** to create a Logical Volume.

To view the maximum space available for a Logical Volume, we enter 1000T in the size field. This will cause an error—and will also report what the total available space is. In our example, we can create a 280 MB volume.

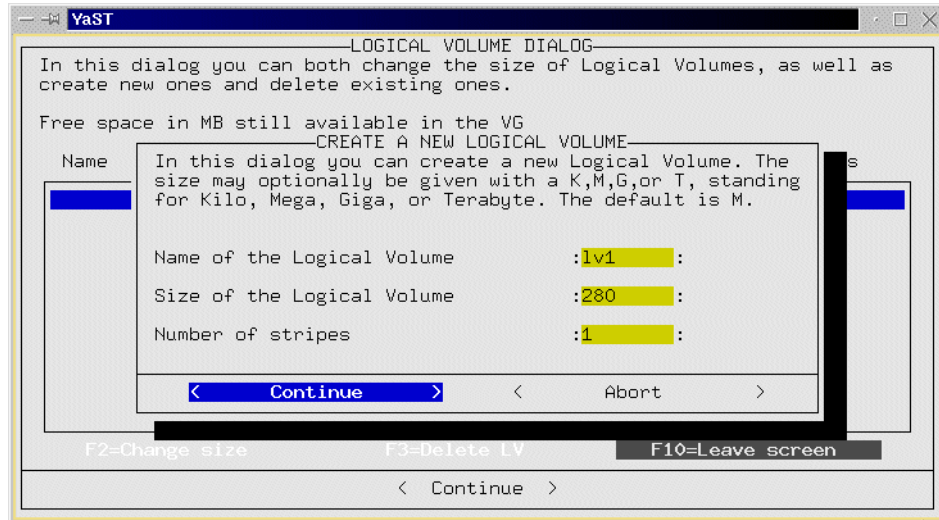


Figure 6-11 Creating a Logical Volume

We create a file system on lv1 with mke2fs and mount the volume:

```
# mke2fs -b 4096 -m2 /dev/vg1/lv1
# mount /dev/vg1/lv1 /mnt/lvm
```

The power of LVM lies in its ability to expand and contract Volume Groups and Logical Volumes to fit your needs. Adding space to a file system is quite simple with YaST.

Launch LVM in YaST and press **F2=change PVs** on the Volume Group menu. Select the volume you'd like to add to your vg and continue. Press **F3=change LVs** and adjust the size of your LV.

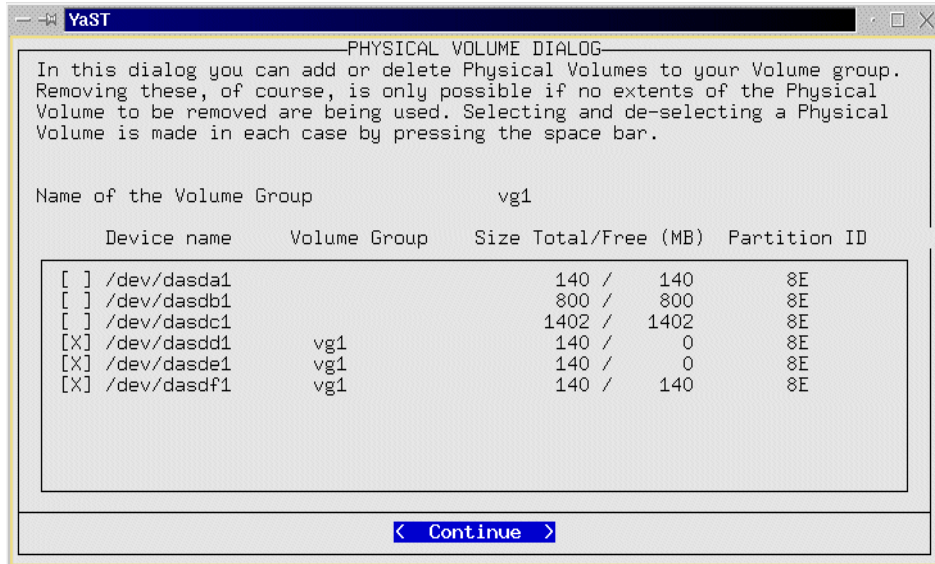


Figure 6-12 Adding a Physical Volume to an existing Volume Group

Expanding the Logical Volume is a two-step process; refer to Figure 6-13.

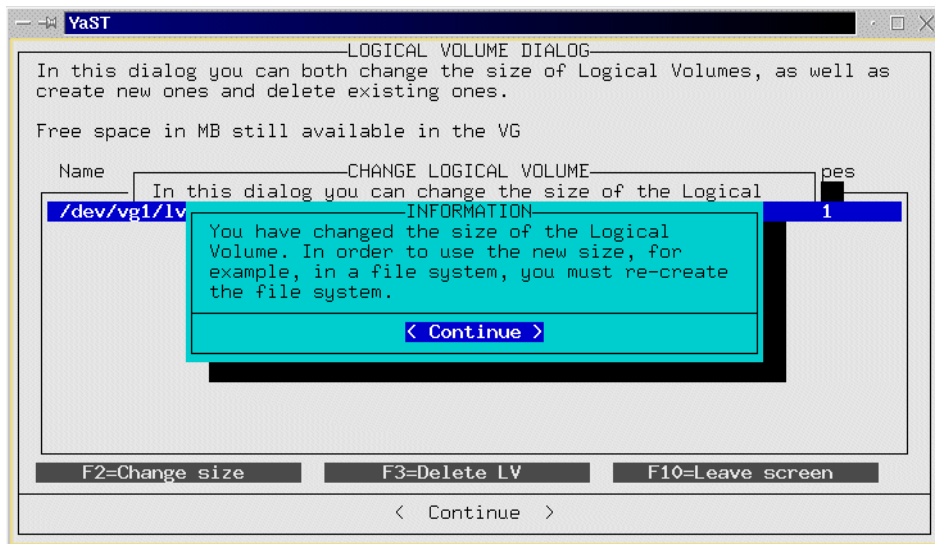


Figure 6-13 Extending the Logical Volume

After adding the new Physical Volumes and extending the LV, you must also extend the file system that resides on that LV.

To extend your FS, unmount the volume and run `resize2fs /dev/vg1/1v1`.

The last step is to add an entry to `/etc/fstab` to automatically mount your Logical Volumes.

## 6.6 Graphical interface

Although S/390 and zSeries hardware does not have graphics capability, a GUI environment can still be established. If accessed through a Linux or other UNIX client which is running an X Window server, or through the use of an X window server for Windows (such as Hummingbird's Exceed, MI/X, or X-Win32), you can run Xclients or a full desktop environment running multiple Xclients. While Linux for S/390 is running the Xclient application, it is sending the display output to your workstation, where your Xserver displays the windowed, GUI output.

Several desktop environments are available for Linux; KDE 1.1.2 (K Desktop Environment) comes standard with SuSE for S/390. Gnome is also available, as are several other window managers (twm, fvwm, olvwm). In this example, we download KDE2 from the SuSE ftp site and demonstrate how to run a few Xclients.

As of the time of writing, KDE 2.1.1 was available and we decided to install it on one of our SuSE test systems. We downloaded the packages from:

```
ftp://ftp.suse.com/pub/suse/s390/KDE2/update\_for\_7.0
```

There are some prerequisite packages (from CD 1) to install before we apply KDE:

```
# rpm -ivh qt-2.3.0-17.rpm (found in ./xdev1/RPM)
# rpm -ivh --nodeps kdelibs-2.1.2-9.rpm (found in ./k2de1/RPM/)
# rpm -ivh kbase-2.1.1-24.rpm (found in /k2de1/RPM/)
```

Normally, installing kdelibs will produce a dependency of libaudiofile.so.0. To work around this, we used the `--nodeps` parameter, which performs the package install without the dependency check.

We now had KDE 2.1.1 installed in `/opt/kde2`. You will need a suitable X Window emulator on another workstation to run KDE.

```
$ export DISPLAY=9.12.2.166:0 (your workstation IP here)
$ startkde &
```

Following are some additional useful KDE packages:

- ▶ kadmin2.1.1 - system administration tools (ftpd configuration, task scheduler (cron), user manager)

- ▶ kdeutils2.1.1 - various utilities (disk monitoring, menu editor, printer administration, etc.)
- ▶ koffice1.1beta - office suite (word processing, spreadsheet, presentation)

Using a Windows 2000 PC running an X Window emulator, we displayed a complete KDE 2 desktop from our mainframe Linux (through which we had opened several windows) as shown in:

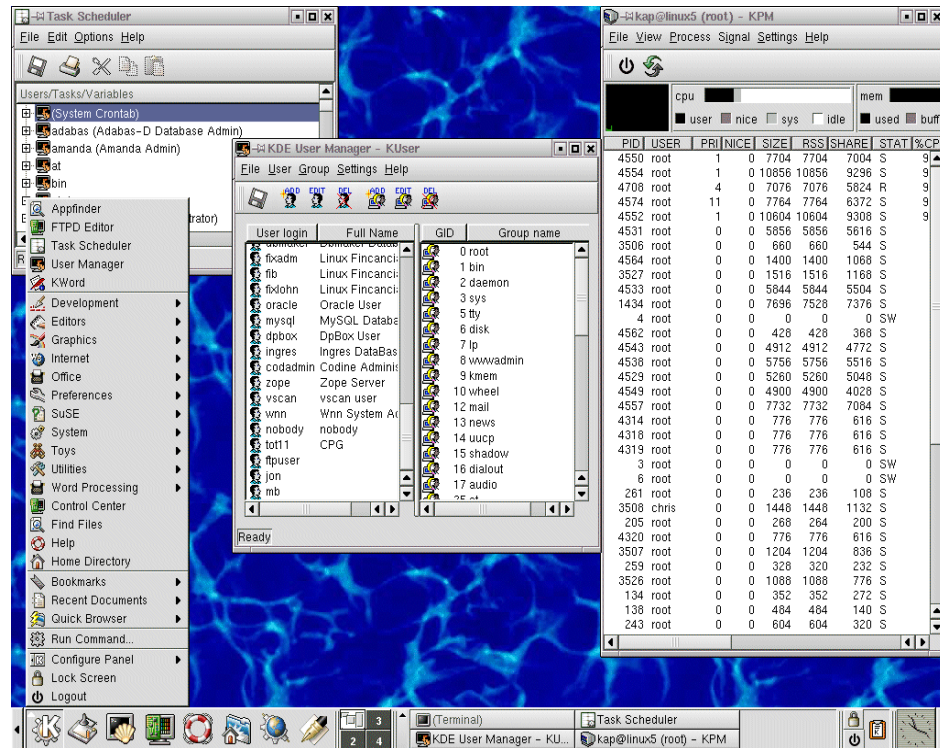


Figure 6-14 Customized KDE 2 desktop running on S/390





## Overview of Turbolinux

This chapter describes the Turbolinux for zSeries and S/390 distribution of Linux by examining the following topics:

- ▶ A history of the Turbolinux for zSeries and S/390 distribution
- ▶ How to acquire the Turbolinux for zSeries and S/390 distribution
- ▶ Available documentation related to the Turbolinux for zSeries and S/390 distribution
- ▶ Notable characteristics of the Turbolinux on zSeries S/390 distribution

## 7.1 History

Turbolinux Inc. was founded in 1992 (as Pacific HiTech Inc.), and has been a Linux player since 1993. This company's history first converged with IBM's history on May 25, 1999, when the two made an agreement to become working partners to promote and deliver Linux solutions. As a result, the IBM DB2 Universal Database could be shipped with Turbolinux.

This initiative also included joint development of marketing, education, training and support. On June 8, 1999, Pacific HiTech changed its name to Turbolinux Inc.

The next chapter in Turbolinux and IBM's history as working partners started with the announcement, on May 17, 2000 that Turbolinux for S/390 would be generally available later the same year. These were the beginnings of Turbolinux on the IBM S/390 and zSeries mainframe hardware platforms.

On October 12, 2000 Turbolinux announced that it would run on S/390 and zSeries. This announcement was part of a more general announcement that Turbolinux would run on all IBM hardware platforms, which includes S/390 and zSeries. This made Linux the first operating system to run on all IBM server hardware.

Turbolinux Server 6 for zSeries and S/390, which is the version of Turbolinux used for installation and customization for this book, has been available since January 30, 2001. An alpha version of the next release (V6.5) was made available for testing on June 5, 2001 and some testing, though not extensive, had been performed at the time of writing.

## 7.2 Getting the Turbolinux distribution

The Turbolinux for zSeries and S/390 distribution can be acquired in one of the following ways:

- Download the ISO CD-ROM images, or the RPM files, from the Turbolinux FTP site
- Purchase the distribution from Turbolinux

## 7.2.1 Downloading the CD-ROM images

**Attention:** Just as this book was being published, the GA version of Turbolinux became available. The following was posted on the linux-390 list server on September 16, 2001:

We now have all our Turbolinux for zSeries product ISO's, updates, and documentation available on our Website. We have a Download Center located at:

<http://www.turbolinux.com/products/s390/download.html>

that includes ISO's for our current Server 6.5 product (kernel 2.2.19), our 6.0 product (kernel 2.2.16), updates to both these products, beta software (kernel 2.4.5 based, new features not fully tested for current release, and upcoming 2.4 kernel betas), as well as some HOWTO's, documents, etc. To download software you need to register first at:

<http://www.turbolinux.com/products/s390/s390reg.php3>

1. Connect to the Turbolinux FTP site at:  
<ftp://ftp.turbolinux.com/pub/product/s390>
2. Change to the version-specific subdirectory (Jan2001, for example) and download the version of Turbolinux you are interested in as an \*.iso file. (If you have an installed Turbolinux for zSeries and S/390 system, you may also download \*.rpm files from another subdirectory under the version directory). The V6.5 beta is on the Web at:  
<ftp://ftp.turbolinux.com/pub/beta/s390/>
3. Use CD creation software such as Adaptec EZ CD-Creator to recreate the original CD-ROMs.

**Note:** Downloading the \*.iso files, while providing the actual installation code, does *not* provide any other benefits that may be available from the distribution vendor, such as (but not exclusive to) support services, upgrade services, integration services, special hardcopy documentation, etc.

## 7.2.2 Purchasing the distribution

Turbolinux can be purchased as follows:

1. Call Turbolinux Sales at: 1-650-228-5076
2. Send e-mail to Turbolinux Sales at: [sales@turbolinux.com](mailto:sales@turbolinux.com)

## 7.3 Available documentation

The Turbolinux for zSeries and S/390 documentation can be found in \*.pdf format on the installation CD-ROMs, and may also be downloaded from the Turbolinux Web site at:

<http://www.turbolinux.com/products/s390/index.html>

The following documents are available in PDF format from this Web site. All were dated December 2000:

Install guide	<i>Preparing for Installing Turbolinux for S/390</i>
Pre-install guide	<i>Installing Turbolinux for S/390</i>
User guide	<i>Turbolinux for zSeries and S/390: User Guide</i>

The same documents were found on the CD:

```
$ ls /mnt/documentation/*.pdf
/mnt/documentation/IBM_Install_Dec0400.pdf
/mnt/documentation/IBM_Prep_Install_Dec0400.pdf
/mnt/documentation/TLS6zSeriesUserGuide.pdf
```

The list of software RPMs that are installed on Turbolinux are listed in an appendix entitled *Linux Software Packages*. Due to the size of this document, it is not included in this redbook, but can be found separately on the Web at:

<ftp://www.redbooks.ibm.com/redbooks/SG246264/RPMappendix.pdf>

## 7.4 Notable characteristics of Turbolinux

The Turbolinux for zSeries and S/390 distribution, while probably notable in a variety of ways, is general distinguishable in two areas:

1. It utilizes a script-based installation process, rather than a menu-based installation.
2. It uses standardized customization methods, such as the editing of configuration files, rather than the use of configuration tools such as YaST or linuxconf.

### 7.4.1 Standardized customization and usage

This characteristic of the Turbolinux for xSeries and S/390 distribution refers to the use of standard Linux commands such as **useradd** and **userdel**, as well as standard Linux tools such as **vi** to edit configuration files like `smb.conf`, the Samba configuration file. We discuss the customization and usage of Turbolinux for S/390 and zSeries further in Chapter 9, “Customizing and using Turbolinux” on page 141.





# Installing Turbolinux

This chapter describes installation of Turbolinux Server Version 6.0 for S/390 and zSeries under the following scenarios:

- ▶ Under VM (see 8.2, “Installation of Turbolinux under VM” on page 123)
- ▶ In an LPAR (see 8.3, “Installation of Turbolinux in an LPAR” on page 126)
- ▶ Under VIF (see 8.4, “Installation of Turbolinux under VIF” on page 129)
- ▶ An introduction to installing the Version 6.5 “beta” (see 8.6, “The Version 6.5 “beta”” on page 136)

Unlike some other Linux distributions, Turbolinux version 6.0 for S/390 and zSeries does not currently provide a full-screen menu-driven interface during the Linux-based installation phase. Instead, an installation script, **install.pl**, is employed that prompts the user for each piece of required information. Once all the questions have been correctly answered, the installation is complete, and the newly installed Linux for S/390 and zSeries system can then be re-IPLed.

## 8.1 Turbolinux installation worksheet

As can be seen from the Installation Worksheet rows 7 through 13 (see Table 8-1 on page 122), a host name, IP addresses for the host, the netmask, the gateway and DNS (Domain Name Server) are required.

For the devices, use the virtual device addresses that are defined for the guest in the user directory (see 3.1.1, “The VM user directory” on page 28).

*Table 8-1 Turbolinux installation worksheet*

1. VM guest user ID (if installing under VM)	
2. VM guest password (if installing under VM)	
3. Device: 1st DASD/vdev(VM) (/ file system)	
4. Device: 2nd DASD/vdev(VM) (swap space)	
5. Device: tape unit *optional (e.g. 0181)	
6. Network device 1st of pair (e.g. 292C)	
7. Network device port 1st of pair (e.g. 0)	
8. Network device type (e.g. OSA-E)	
9. Host name (e.g. vmlinux.itso.ibm.com)	
10. Host IP address (e.g. 9.12.6.73)	
11. Netmask (e.g. 255.255.255.0)	
12. Network IP address (e.g. 9.12.6.0)	
13. Broadcast IP address (e.g. 9.12.6.255)	
14. Gateway IP address (e.g. 9.12.6.75)	
15. DNS IP address (e.g. 9.12.14.7)	
16. MTU size (e.g. 1492)	
17. FTP server IP address or hostname	
18. FTP path (e.g. /mnt/cdrom)	
19. FTP user ID (e.g. TOT82)	



1. VM guest user ID (if installing under VM)	
2. VM guest password (if installing under VM)	
20. FTP password	
21. Linux root user password	

### 8.1.1 Planning DASD with Turbolinux

A *full development* install of Turbolinux onto a 2600 cylinder minidisk uses 52% of the minidisk (or a little over 1 GB). This would equate to about 40% of a full 3390-3 volume. The size of other types of installations is shown in Table 8-2.

Table 8-2 Turbolinux disk requirements

Directory	Installation type	
	default	all
/bin	5.4 MB	6 MB
/boot	1.7 MB	1.7 MB
/dev	104 KB	104 KB
/etc	8.2 MB	12 MB
/lib	17 KB	17 MB
/opt	0	0
/root	32 KB	44 KB
/sbin	5.1 MB	9 MB
/tmp	140 KB	288 KB
/usr	630 MB	2,300 MB
/var	11 MB	49 MB
Total	677 MB	2,400 MB

## 8.2 Installation of Turbolinux under VM

The installation process can be divided into three major steps: preparation; initial IPL of Linux; building the target Linux system. The initial IPL of Linux can be done from the VM reader or from tape.

In the following sections we describe how to prepare for both methods. IPLing from the VM reader is often the most convenient; however, IPLing from tape allows you to reuse the same tape from both VM and when installing in an LPAR.

## 8.2.1 VM checklist

1. Read 3.1, “Prepare the VM guest for Linux” on page 28.
2. Plan your DASD configuration:
  - a. Which minidisks are going to be used.
  - a. What file systems are going to be on what minidisks.
  - a. How big those volumes need to be (see 8.1.1, “Planning DASD with Turbolinux” on page 123).
3. Complete the worksheet in Table 8-1 on page 122.
4. Make sure the three appropriate boot files and an IPL EXEC are available (see 8.2.2, “Download boot files to VM” on page 124).
5. IPL Linux from reader (or possibly tape).
6. Run the install script **install.pl**.
7. IPL the built target system from DASD.

## 8.2.2 Download boot files to VM

Using FTP on VM, locate and download the Linux for S/390 boot files. In our case, we mounted the single Turbolinux CD on an FTP server running Linux. We downloaded the boot files to our 191 (A) disk.

The files must be reblocked to a record length of 80 bytes, and the kernel file must be specific to the reader IPL method. The parameter file is converted to EBCDIC via the **asc** FTP subcommand. Following is a summary of the FTP commands we used:

```
# ftp 9.12.0.88
cd /mnt/cdrom
bin
locsite fix 80
get /turbo/image-2.2.nn-127-randisk-vm turbo.image
get /turbo/initrd turbo.initrd
asc
get /turbo/parmline turbo.parm
quit
```

Once we had the boot files on VM, we created an **LBOOT EXEC** as follows, in order to be able to IPL from the reader:

```
/* */
```

```

'close rdr'
'purge rdr all'
'spool punch * rdr'
'punch TURBO IMAGE A (NOH' /* this the kernel image and must be first*/
'punch TURBO PARM A (NOH' /* this is the parmline file and must be 2nd*/
'punch TURBO INITRD A (NOH' /*this is the RAM disk and must be 3rd*/
'change rdr all keep nohold'
'ipl 00c clear'

```

### 8.2.3 Initial IPL of Linux

We IPLed from the VM reader using only the command **LB00T**. If you have to re-IPL and the same files are still in your reader, the following command will suffice.

```
#CP IPL 00C CLEAR
```

To IPL from tape, ensure the installation tape has been mounted and rewound and issue:

```
#CP IPL 181 CLEAR
```

We tried to start with a relatively small VM machine size of 64 M, but received the following warning:

```

There is only 62640 KB RAM available. This install is likely to have
problems. Some packages may not get installed. In the worst case, you
may end up with a non-functional Linux system.

```

At the moment, we recommend 128 MB of RAM.

We increased the VM machine size to 128 M, but still got the warning. After increasing to 132 M, we did not receive this message.

After a successful boot, the following message will appear on the VM console (it will be preceded by normal Linux boot messages):

```

-----
                Welcome to
                TUBOLINUX
for IBM S/390 and zSeries
    Astrodon Release
    brought to you by the
    FRONTIER TEAM
    October 31 2000
-----
Do you want to configure the eth0 device? (y/n) [y]:

```

This is the beginning of the customization of the bootstrap system with the required local network information, so the bootstrap system can download the files necessary for the target system. When this process is completed, the bash prompt will appear:

```
...  
bash-2.04#
```

From this point on, a telnet session to the bootstrap system is started in order to finish building the target system. Continue with the instructions in 8.5, “Building the target system” on page 130.

## 8.3 Installation of Turbolinux in an LPAR

Whether you install Linux in an LPAR, or native on a S/390 or zSeries machine, the process is the same and the steps are similar. In the following section, we first summarize this process with a high-level checklist.

### 8.3.1 Turbolinux under LPAR checklist

1. Read 3.2, “Prepare the LPAR for Linux” on page 31.
2. Plan your DASD configuration:
  - a. Which volumes are going to be used.
  - a. What file systems are going to be on what volumes.
  - a. How big those volumes need to be.
3. Complete the worksheet in Table 8-1 on page 122.
4. Prepare media, depending on one of the following initial IPL methods:
  - a. From a tape (see 3.4.3, “Booting from tape” on page 44 then 8.3.2, “IPL of Turbolinux” on page 127).
  - a. From the HMC CD-ROM or an FTP server (see 3.4.2, “Booting from HMC CD-ROM or FTP server” on page 35).
  - a. From an ICKDSF bootstrap (see 3.4.1, “Using the ICKDSF bootstrap” on page 34).
5. IPL Linux, using the appropriate media depending on the IPL method.
6. Run the install script **install.pl**.
7. IPL the built target system from DASD.

## 8.3.2 IPL of Turbolinux

The IPL from tape is done from the HMC of the S/390 or zSeries processor, from a tape drive using the Turbolinux boot tape.

From the HMC, select the processor and then select an image (see Figure 8-1 on page 127) which represents the LPAR to be used for Linux. Drag the icon with the right mouse button onto the **Load** icon in the right-hand column.

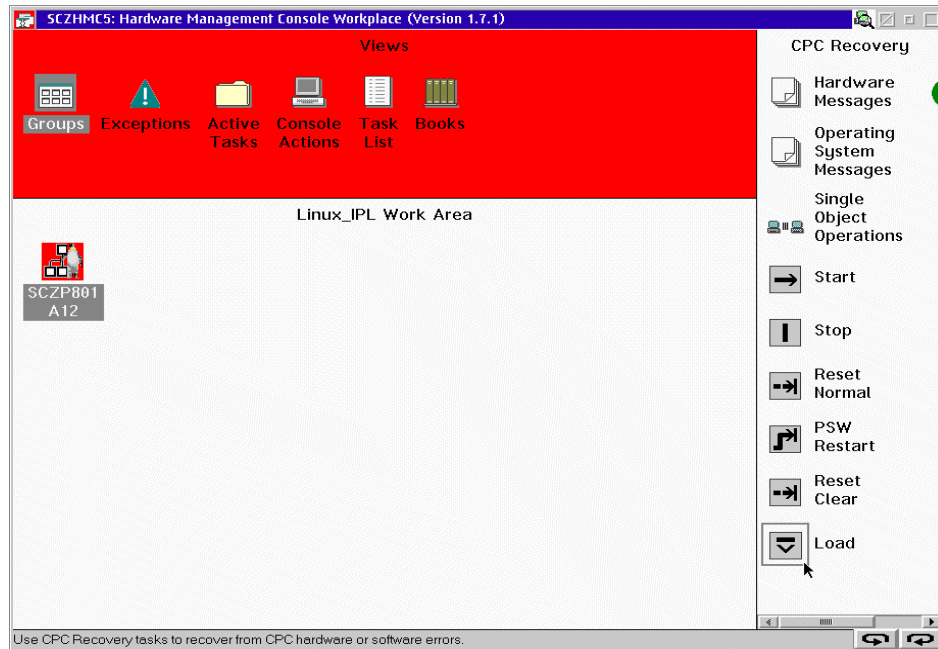


Figure 8-1 Linux LPAR on HMC

This will display the load dialog box as shown in Figure 8-2. Select the **Clear** button, enter the address of the tape drive, and press **OK**.

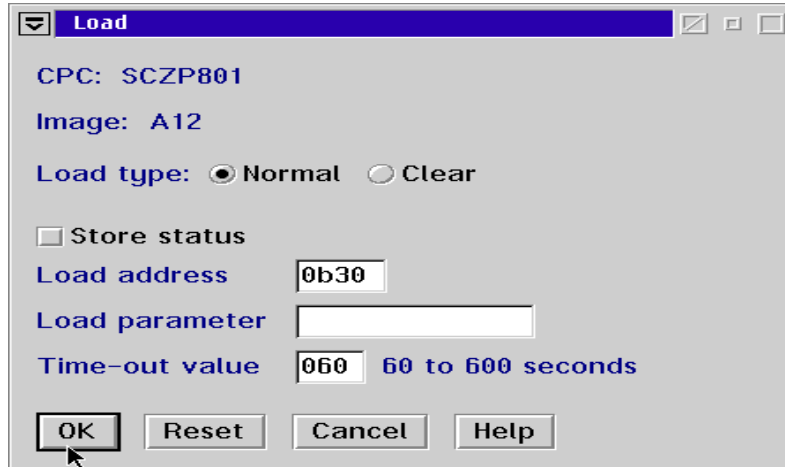


Figure 8-2 Dialog load screen

There will be a confirmation box (to which the reply is yes), and then you will see some progress information boxes.

After a successful boot, the following messages will finally appear on the HMC operator messages interface (preceded by normal Linux boot messages):

```

-----
                Welcome to
                TUBOLINUX
    for IBM S/390 and zSeries
                Astrodon Release
                brought to you by the
                FRONTIER TEAM
                October 31 2000
-----
Do you want to configure the eth0 device? (y/n) [y]:

```

This is the beginning of the customization of the bootstrap system with the required local network information, so the bootstrap system can download the files necessary for the target system. When this process is completed, the bash prompt will appear:

```
bash-2.04#
```

From this point on, a telnet session is started to the bootstrap system to complete building of the target system. Continue with the instructions in 8.5, “Building the target system” on page 130.

## 8.4 Installation of Turbolinux under VIF

Chapter 16, “VIF” on page 267, provides a complete description of VIF installation. However, that chapter describes installing SuSE under VIF, and several unique issues arose during our testing of the Turbolinux under VIF installation process. We document those differences in the following sections.

### 8.4.1 The turbo.ins file

The turbo.ins file included on the *Turbolinux Server 6 for zSeries and S/390* CD will not work with VIF without modification. If you are installing from a CD mounted directly on your FTP server (as we were), you will need to move the turbo.ins file to a read/write directory on the same FTP server, so that the file can be modified. If you have copied the entire CD to a read/write directory on the FTP server and will be installing directly from that directory, you can simply modify the file.

The modification consists of the addition of *fully-qualified* path names to each line of the turbo.ins file, as shown:

```
/mnt/cdrom/image      0x0
/mnt/cdrom/initrd     0x00800000
/mnt/cdrom/parmline   0x00010480
```

**Note:** Be aware that the path names you append to the turbo.ins lines are *relative to the location of the turbo.ins file itself*, requiring that the file either reside on the same server as the CD/installation source files, or on a file system mounted on the server with the CD/installation source files.

### 8.4.2 dasdfmt

The Turbolinux `install.pl` installation script can do a `dasdfmt`, so when you are asked about this, you should take that option. Also, the partition LINUX0 201 (`/dev/dasda`) can be formatted manually *immediately prior* to running `install.pl`. For example:

```
insmod dasd_mod dasd=201,203
dasdfmt -vL -b 4096 -f /dev/dasda
```

### 8.4.3 Master Linux

VIF *requires* that a master Linux image be created. This master Linux is only used to access the `vif` command, which is the primary interface to the VIF hypervisor. As such, a minimum Linux system would normally be chosen in this case. However, the Turbolinux install script does not currently offer a “minimum system” selection, so we used the “FTP server” selection, and it appeared to work fine.

### 8.4.4 Other VIF-related issues

A few additional issues were observed while installing Turbolinux under VIF (which were probably more VIF-related than actually specific to Turbolinux), as follows:

- ▶ When a VIF command returns information (such as the output from the `hypervisor error` command), the returned information may be incorrectly displayed as *upper case*.
- ▶ While the VIF documentation states that you can skip the use of the `hypervisor volume initialize` command when using pre-initialized DASD types, such as RVA or Shark™, it does not mention that the command will *fail* if used, which could lead someone to incorrectly think there might be a problem with DASD.
- ▶ With VIF commands, the final output line returned (which is `Command Completed`) is the same regardless of *whether or not* the command was successful. It is therefore important that all VIF command output be read thoroughly.

**Attention:** There is currently a limitation using Turbolinux Server Version 6.0 under VIF in that, when you attempt to create new/additional Linux images beyond the “master Linux” itself, the new images *will not boot into the Turbolinux installation*, leaving you at a `CP` prompt. Tests of multiple image memory settings of up to 512 MB did not appear to fix the problem.

However, this symptom appears to have been fixed in the June 1, 2001 beta of Turbolinux Server V.6.5: you must first set the image memory to 128 MB, using the VIF command `IMAGE SET STO 128`.

## 8.5 Building the target system

In the following section, we describe the steps needed to build your target system.



## 8.5.1 Running the Turbolinux installation script

Once the network values have been customized for the bootstrap or target system, a telnet session to it should be started.

At the login prompt, enter: `root`. No password is needed. Then enter: `install.pl` to start the install script and complete the installation of the target system.

```
login: root
# install.pl
```

You will be asked which DASD are to be made available to Linux, as follows:

```
-----
Which DASD's do you want to have available in Linux?
-----
```

```
Example: 192, 194-196
Example: fd02, fd04-fd06
```

```
DASD's: 201,203
insmod dasd_mod dasd=201,203
Using /lib/modules/2.2.16/misc/dasd_mod.o
```

Then, depending on whether the DASD are already formatted, you may be asked a question requesting a format of the DASD, as follows:

```
One or more dasds need low level formatting
```

```
-----
      Number Type   Name  Status  Blksize Blocks FS
1      0201  ECKD  dasda  n
2      0203  ECKD  dasdb  n
-----
```

```
Format disk[# or q]: 1
dasdfmt -b 4096 -f /dev/dasda
```

If you reply 1, the following messages are received:

```
I am going to format the device /dev/dasda in the following way
Device number of device :0x0201
  Major number of device : 81
  Minor number of device : 0
  Labelling of device    : yes
  Disk Label              : LNX1 80405200
  Blocksize               : 4096
--->> ATTENTION! <<---
```

All data in the specified range of that device will be lost.  
Type "yes" to continue, no will leave the disk untouched: **yes**

Reply `yes` and expect to wait some time, depending on the size of the device (it takes up to about 60 minutes to format a 2600 cylinder minidisk). Once this completes, you will be asked for a swap device to be chosen, as follows.

Choose a swap device

```
-----  
      Number Type   Name  Status  Blksize Blocks FS  
1      0201  ECKD   dasda  n  
2      0203  ECKD   dasdb  n  
-----  
Swap_disk[#_or_q]: 2  
mkswap /dev/dasdb1  
init_swap: Setting up swap space version 1, size = 820576256 bytes  
chmod 600 /dev/dasdb1  
swapon /dev/dasdb1
```

In our case, we replied 2 to set up `dasdb` as the swap device.

The next question asks for a device for the root file system:

```
Which disk should be the root filesystem?
```

```
-----  
      Number Type   Name  Status   Blksize Blocks FS  
1      0201  ECKD   dasda active   4096   601020  
-----
```

```
Root_disk[#]: 1
```

```
Creat_a_new_filesystem_on_/dev/dasda1?_[y/n]: y
```

In our case, we replied 1 to choose dasda. Then you are asked to choose either FTP or NFS as an install method; this is the method by which the build process will access the Linux files for installation:

```
Please choose an install method:
```

- 1) NFS
- 2) FTP
- 3) quit

```
Which_method? 2
```

In our case, we replied 2 to choose FTP. Choosing FTP will cause prompts for the FTP host name, and the path where the install files reside, as follows:

```
FTP_hostname: itsolinux2.itso.ibm.com
```

```
FTP_path: /mnt/cdrom
```

After you provide these values, the next question asks for a package selection based on six models or general purposes for a system. The number of packages installed will vary with this choice. Option 5 will include all of the first 4, and option 6 includes everything:

- 1 Web server
- 2 FTP server
- 3 SMB file server
- 4 Web/Ftp Proxy Server
- 5 All servers
- 6 Development (All-in-one)

```
Package selection: 6
```

In our case, we replied 6 to install all packages. This step may take up to 60 minutes or more. As the packages are installed, the progress is displayed as follows:

```
1/515: setup-6.--10.noarch.rpm
```

```
2/515: filesystem-1.3.5-3.noarch.rpm
```

```
...
```

```
515/515: libungif-devel-4.1.0-3.s390.rpm
```

When this task completes successfully, the next question asks for the number of a time zone to be chosen. (We found that the Enter key had to be pressed quite a few times to page through the list.)

```
-----  
Please select your time zone  
-----  
    0 - Africa/Abidjan      |    1 - Africa/Accra  
    2 - Africa/Addis_Ababa |    3 - Africa/Algiers  
    ...  
    ...  
    30 - Africa/Libreville |    31 - Africa/Lome  
-----  
Timezone ([Enter] for more options) :
```

After you select an appropriate time zone, you will receive prompts for network settings to be used on the target system. The network settings used when setting up the bootstrap system are remembered by the system, and it will use them as defaults when no value is entered—making this final setup easier by requiring much less data entry.

After the network settings are accepted, you will receive prompts for the setup of additional users, the password for root and, finally, setting the boot disk.

**Note:** Turbolinux will *not* allow a telnet login by root. This means root can only login at the HMC console, or at the VM console, or at the VIF console. However, none of these is ideal for use with Linux because of the 3270 interface.

Therefore, you need to define at least one new user so that the normal telnet interface can be used. We recommend you create a new user as follows:

```
Do_you_want_to_add_a_new_user?(Y/n)Y  
  
User_Name:_[neo]: linuser  
Please set a password for linuser:  
New UNIX password:  
  
Retype new UNIX password:  
passwd: all authentication tokens updated successfully
```

Now you are prompted for the root password. You may receive messages about the password being bad; though these can be ignored, it is better to choose a proper password (refer to Chapter 22, “Overview of security on Linux” on page 435 for more information). By reentering the password, it will be accepted:

```
Please enter a password for the root user:  
  
Changing password for user root
```

```

New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully

```

The following final question is also important. The answer will determine on which volume the bootstrap record, parameter file and kernel image will be placed:

```

-----
SILO Configuration
-----

```

	Number	Type	Name	Status	Blksize	Blocks	FS
1	0201	ECKD	dasda	active	4096	601020	/
2	0203	ECKD	dasdb	active	4096	200340	swap

```

-----
Boot_Disk_[#] 1

```

In our case we choose 1, to place the boot information on the root file system. This causes the **sil** command to be run, as shown:

```

chroot /mnt/root silo -f /boot/image-2.2.16-122-tape -d /dev/dasda -p
/boot/parm -b /boot/ipleckd.boot -t2
Silo: o->image set to /boot/image-2.2.16-122-tape
Silo: o->ipldevice set to /dev/dasda
Silo: o->parmfile set to /boot/parm
Silo: o->bootsect set to /boot/ipleckd.boot
Silo: o->Testonly flag is now 0
Silo: o->Testlevel is set to 0
Silo: o->IPL device is: '/dev/dasda'
Silo: o->bootsector is: '/boot/ipleckd.boot' ...ok...
Silo: o->bootmap is: '/boot/image-2.2.16-122-tape' ...ok...
Silo: o->Kernel image is: '/boot/ipleckd.boot' ...ok...
Silo: o->original parameterfile is: '/boot/parm' ...ok...final parameter
parm is: '/boot/parm' ...ok...
Silo: ix 0: offset: 078523 count: 0c address 0x00000000
Silo: ix 1: offset: 078530 count: 80 address 0x0000c000
Silo: ix 2: offset: 0785b0 count: 80 address 0x0008c000
Silo: ix 3: offset: 078630 count: 6f address 0x0010c000
Silo: ix 4: offset: 07a8de count: 01 address 0x00008000
Silo: Bootmap is in block no: 0x00001ed7

```

The target system is now ready to be IPLed from DASD. Shut down the bootstrap system with the **halt** command or a **shutdown -h now** command.

To IPL the target system from VM, use the following CP command:

```
#CP IPL 201 CLEAR
```

To IPL the target system in an LPAR or native on a S/390 or zSeries, use the same procedure used to IPL the bootstrap system—except this time from DASD, instead of from tape.

Following the current example, from the HMC select the processor and then select an image which represents the LPAR to be used for Linux. Drag the icon onto the Load icon in the right-hand column. This will display the load dialog box. Select the clear button and enter 0201 as the address of the load device, then press **OK**.

You can begin customizing the target system once the Linux target system has completed IPL. This is described in Chapter 9, “Customizing and using Turbolinux” on page 141.

## 8.6 The Version 6.5 “beta”

While most of the testing for this redbook was done using Version 6.0 of the Turbolinux Server for zSeries and S/390, we did receive an early “beta” of Version 6.5, dated 06/01/01, which we were able to install under VM.

The following sections describe the obvious differences we observed in Version 6.5 installation and distribution, under VM.

### 8.6.1 The Version 6.5 installation

We noted the following differences related to the installation process.

#### 8.6.1.1 The introductory screen

The introductory installation screen was slightly redesigned and included new verbiage related to networking, as shown:

Welcome to

TURBOLINUX

for IBM S/390 and zSeries

z/Linux (TM) 6.5

July 2001

---

If you are using x3270 terminal emulator,  
under 'Options', choose Character Set "U.S.International"

In order to install you have to configure at least one network device:

- ctc0 - CTC Channel to Channel
- escon0 - ESCON Enterprise Systems Connection
- iucv0 - IUCV Inter-User Communication Vehicle (only for VM or VIF)
- eth0 - Ethernet
- tr0 - Token Ring

Please have the following information ready:

- assigned IP address,
- Peer address,
- Hostname,
- DNS address,
- MTU size (or just accept the default).

To accept the defaults, press Enter twice

### 8.6.1.2 New/additional networking selections - phase one

Networking options in Version 6.0 were individual prompts, in contrast to this more “menu-like” screen. Also, options for iucv and escon had been added:

Which of the following devices do you want to configure:

- [1] - ctc0
- [2] - iucv0
- [3] - eth0
- [4] - escon0
- [5] - tr0

[0] None

Enter the number of network device to configure []

### 8.6.1.3 Completion screen - phase one

We noted a redesigned phase one completion screen, as shown:

```
-----  
----- Congratulations! The first installation stage is finished. -----  
-----  
MANUALLY you can start ROUTED or GATED  
  
We suggest using TELNET for the remaining configuration steps.  
Use configured IP address and login as a root.  
There is no password for now, but it will be set.  
-----
```

To start the second stage type at the shell prompt:

```
install.pl
```

```
-----  
bash-2.04#
```

#### 8.6.1.4 DASD formatting options

We noted the addition of (recommended) to this screen:

```
One or more dasds need low level formatting (recommended).
```

```
-----  
      Number  Type   Name   Status  BlkSize  Blocks  FS  
-----  
1      0201   ECKD   dasda   active   4096    468000  
2      0203   ECKD   dasdb   active   4096     18000  
-----
```

```
Format disk [1 or 2 or etc. or q]
```

#### 8.6.1.5 LVM

Logical Volume Manager (LVM) support had been added to the distribution, as well as to the installation process:

```
*****
```

```
LVM has installed successfully!
```

```
You have to modify /etc/rc.d/rc.sysinit to start LVM in boot time  
Read /usr/share/doc/packages/lvm-0.9.1_beta7/lvm.TURBOLINUX_s390 file  
To start LVM type:
```

```
modprobe lvm-mod  
/sbin/vgscan  
/sbin/vgchange -a y
```

```
515/515: lvm-0.9.1_beta7-1.s390.rpm
```

#### 8.6.1.6 New/additional networking selections - phase two

This appeared to be basically a duplication of the screen in phase one of the installation:

```
Which of the following devices do you want to configure:
```

```
[1] - ctc0  
[2] - iucv0  
[3] - eth0  
[4] - escon0  
[5] - tr0
```



[0] None

Enter the number of network device to configure

### 8.6.1.7 New/additional Apache verbiage

The inclusion of Apache is not new, depending upon the installation option selected. However, the following text had been added:

```
Apache, the TurboLinux web server, comes configured to serve a
demonstration page. You may want to configure Apache first to fit
your needs and turn it on afterwards.
Turn on the web server now? (y/n) [n]:
```

### 8.6.1.8 Installation completion message

Additional verbiage related to VM had been added at the end of the installation:

```
-----
Installation is complete.
Type halt to shutdown the system. Do not use any other commands.
Then IPL the root disk.
If you are on VM use #cp i xxx where xxx is boot disk number.
-----
bash-2.04#
```

## 8.6.2 The Turbolinux V6.5 distribution

Two of the differentiators of Linux distributions are the installation process and the contents/packaging. In this section, we examine the three most obvious content additions in the June 1, 2001, beta of Turbolinux Server Version 6.5 for zSeries and S/390:

- ▶ The kernel version
- ▶ The tape driver
- ▶ Logical Volume Manager (LVM)

### 8.6.2.1 The kernel version

The June 1, 2001, beta of Turbolinux Server Version 6.5 for zSeries and S/390 is based upon version 2.2.19 of the Linux kernel, versus the 2.2.16 version included in Turbolinux Server Version 6.0 for zSeries and S/390.

### 8.6.2.2 The tape driver

Support for the tape driver comes pre-compiled into Turbolinux Server Version 6.5 for zSeries and S/390. However, no actual tape device files are included and they must therefore be manually created as documented in Chapter 23, “Backup and restore” on page 463.

### 8.6.2.3 LVM

Support for the LVM comes pre-compiled into Turbolinux Server Version 6.5 for zSeries and S/390. The LVM driver is loaded as follows:

```
# /sbin/modprobe lvm-mod
```

Further information on using LVM can be found in Chapter 17, “Logical Volume Manager” on page 301.



## Customizing and using Turbolinux

In this chapter we describe many aspects of customizing and using Turbolinux for S/390 and zSeries by examining the following topics:

- ▶ How to customize an installed Turbolinux for S/390 and zSeries system
- ▶ Administrative commands and tools available under Turbolinux for S/390 and zSeries

**Note:** Turbolinux for zSeries and S/390 disables telnet by root. In order to telnet and have root authority, you need to use either the `su` command or the `su - root` command, followed by the root user password.

**Note:** Access by user root via telnet can be enabled by either deleting or renaming the `/etc/securetty` file. However, due to the inherent insecurities associated with telnet (such as its use of clear-text passwords), a better solution is to use `ssh`.

## 9.1 Customizing Turbolinux

All of the packages discussed in the following section were already installed as part of the initial installation. When installing the system, we chose option **6** (Development All-in-one), which was the most comprehensive installation option available. This option resulted in 515 packages being installed.

However, not all packages started automatically after boot, even though they had scripts in the `/etc/rc.d/init.d` directory. Some of them required further steps, using the `chkconfig` command.

For example, `chkconfig --add nfs` adds run-level information for the NFS startup script to system services. By using defaults, NFS is defined as a daemon to be started automatically after boot (that is, it will run in run levels 3 to 5).

Executing a `chkconfig --list|sort|more` command will list all services that have been defined for automatic start, and at what run levels. In order for startup scripts to be added or defined using this method, the scripts need two special comment lines (which are described in the man pages of the `chkconfig` command).

Because Turbolinux does not have a specific customizing tool, we chose to use Webmin, which is described in 9.2.1, “Webmin” on page 148 and in 21.2.3, “Webmin” on page 403.

### 9.1.1 Secure shell (SSH)

Secure shell (SSH) is already installed and set up to use. There many packages beginning with the name `openssh` that comprise the full range of SSH services. Following is a summary of the files used for SSH on Turbolinux:

Name	<code>openssh-server</code>
daemon	<code>/usr/sbin/sshd</code>
startup script	<code>/etc/rc.d/init.d/sshd</code>
configuration file	<code>/etc/ssh/sshd_config</code>
secure ftp	<code>/usr/bin/sftpserv</code>
log files	<code>/var/log/messages</code>

Webmin allows for an SSH or telnet login. Webmin uses telnet for communication rather than SSH, but this is easily changed.

To make this change, bring up Webmin (see 9.2.1, “Webmin” on page 148) via a browser and select the lower left-hand icon **SSH/Telnet Login** on the main screen. Then double-click **Module Config** in the upper left-hand corner. On the configuration page, enter the host name, click the **Secure Shell** button, and then click **save**.

You should now get another window prompting you for the user ID and password. You can send root and the root password with confidence that your session is encrypted.

## 9.1.2 FTP server

On Turbolinux, the FTP daemon is typically set to start at boot time. FTP server sessions are initiated via `inetd`. Because of this, you will only see an FTP server process when someone has a session running. It is installed under RPM as the package name `proftpd`. The usual RPM queries will list files and so on. Following is a summary of the files used for `proftpd` on Turbolinux:

Name	ProFTP
daemon	/usr/sbin/proftpd
configuration files	/etc/proftpd/proftpd.conf /etc/proftpd/ftputers {a user exclusion list}
startup script	/etc/rc.d/init.d/inet
lock file	/var/lock/subsys/inet
log files	/var/log/messages

The FTP server configuration file is `/etc/proftpd/proftpd.conf`. By default, it has most options commented out. We uncommented them and tried an FTP session, but the FTP home directory could not be found. We then did a `mkdir /home/ftp` to create the directory, and the FTP server worked as expected.

## 9.1.3 Web server

On Turbolinux, the Web server, `httpd`, is typically set up to start automatically at boot. It is installed under RPM as the package name `apache`. Following is a summary of the files used for `httpd` on Turbolinux:

Name	Apache
daemon	/usr/sbin/httpd
configuration files	/etc/httpd/conf/httpd.conf /etc/httpd/conf/srm.conf /etc/httpd/conf/access.conf /etc/mime.types or /etc/httpd/conf/mime.types
startup script	/etc/rc.d/init.d/httpd
lock file	/var/lock/subsys/inet
log files	/var/log/httpd/access_log /var/log/httpd/access_log.1 /var/log/httpd/error_log /var/log/httpd/error_log.1

The setup for `httpd` is well-documented in the chapters covering the other distributions, as well as in *Turbolinux server 6 for zSeries and S/390 User Guide*.

Figure 9-1 shows the sample Web page that is loaded by default when a Web browser is pointed to the host name or IP address of the Turbolinux host.



Figure 9-1 Turbolinux Web server default page

## 9.1.4 Samba

On Turbolinux, Samba is typically installed and ready to run. Also, the Samba Web Administration Tool (SWAT) is installed but is commented out in the `inetd` configuration file, `/etc/inetd.conf`. To enable SWAT, edit the file and uncomment the following line:

```
#swat      stream tcp      nowait.400      root    /usr/sbin/swat swat
```

Then restart `inetd` with the command `/etc/rc.d/init.d/inet restart`. After it is restarted, you can point your Web browser to:

```
http://<Turbolinux_hostname>:901/
```

You will be prompted for a user ID and password. A screen shot of SWAT is shown in Figure 6-4 on page 101.

Following is a summary of the files used for Samba on Turbolinux:

Name	Samba
daemon	/usr/sbin/smbd /usr/sbin/nmbd
configuration file	/etc/smb.conf
startup script	/etc/rc.d/init.d/smb /etc/rc.d/init.d/nmb
log files	/var/log/samba/log.smb /var/log/samba/log.nmb

### 9.1.5 NFS server

The RPM package name for NFS in Turbolinux is knfsd. The portmap daemon is also required to be running for NFS, and the **exportfs** command is useful for checking and implementing any new changes to /etc/exports (which is used to define the access to directories). The **exportfs** command can be used instead of restarting the NFS daemons.

Following is a summary of the files used by knfsd, **exportfs** and **portmap**:

Name	knfsd
daemons	/usr/sbin/rpc.nfsd /usr/sbin/rpc.mountd /usr/sbin/rpc.rquotad /sbin/portmap
commands	/usr/sbin/nfsstat /usr/sbin/exportfs
configuration files	/etc/exports /etc/sysconfig/network
startup scripts	/etc/rc.d/init.d/nfs /etc/rc.d/init.d/portmap
lock files	/var/lock/subsys/nfs /var/lock/subsys/portmap

We found that the only thing required to get NFS started was to actually define an entry in the /etc/exports configuration file, and then start the port mapper and NFS. They were started via the scripts /etc/rc.d/init.d/nfs and /etc/rc.d/init.d/portmap. (However, the *Turbolinux server 6 User Guide* incorrectly lists the portmap script as /etc/rc.d/init.d/portmap.init).

### 9.1.6 The domain name server

The Turbolinux domain name server is (BIND).

This server has its required components installed using these four packages:

```
bind
bind-contrib
bind-devel
bind-utils
```

The packages can be listed by using the `rpm` command. Following is a summary of the files used by BIND:

Name	BIND
daemon	/usr/sbin/named
configuration files	/etc/resolv.conf /etc/named.conf /etc/named.boot /etc/host.conf
startup script	/etc/rc.d/init.d/named
lock file	/var/lock/subsys/named

BIND was already set up by the installation process for cache-only server mode; this only requires the `/etc/resolv.conf` configuration file, which specifies the DNS domain and the default domain name server:

```
$ cat /etc/resolv.conf
search itso.ibm.com
nameserver localhost
```

The other configuration file is `/etc/named.boot`:

```
$ cat /etc/named.boot
;
; a caching only nameserver config
;
directory /var/named
cache . named.ca
primary 0.0.127.in-addr.arpa named.local
```

## 9.1.7 XDM

The X window display manager (XDM) was already set up on this system. All that is needed is an X Window server on the workstation to test it.

Linux desktops will typically have an X server configured, but we had Windows desktops, which do not have an X server. We used Microimages MI/X for this purpose. Here is the URL to download the latest MI/X from MicroImages:

<http://www.microimages.com/>

Older and free versions may be available at mirror sites, which are listed at:

<http://www.microimages.com/freestuf/mix/mirrorlist/>



For installation, you should use IP addresses that are relevant to the local network and environment in which MI/X will be installed. This information should already be available from the install of Turbolinux. The only new data is the terminal's IP address.

Figure 9-2 shows an example of the MI/X X server running on MS Windows/2000, accepting displays from the xterm and XEDIT clients running on the Turbolinux host.

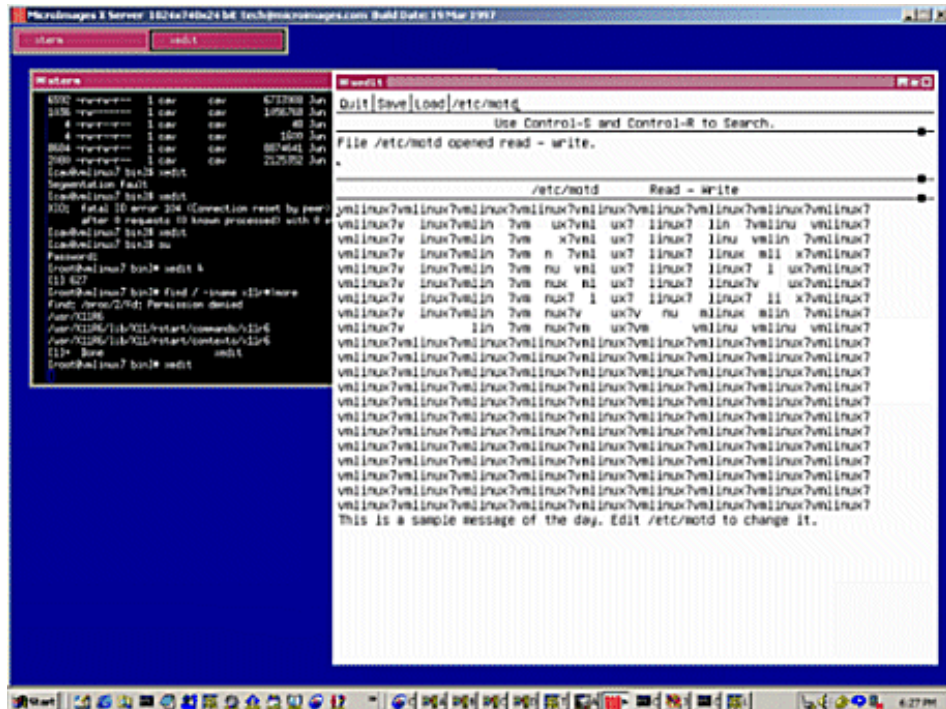


Figure 9-2 Turbolinux XDM being displayed to an MI/X window

## 9.2 Administration

As of version 6.0, Turbolinux for S/390 and zSeries has no distribution-specific administration utilities. Of course, common administrative commands such as **useradd**, **userdel**, **passwd**, **groupadd**, **groupdel**, **ifconfig** and **rpm** can be used.

## 9.2.1 Webmin

Webmin is a very useful, free administrative package that is distributed under the BSD licence and allows for the complete administration of practically any Linux or UNIX system. Webmin is basically comprised of Perl scripts that provide both a basic Web server (for use by Webmin only), as well as control of most of the standard configuration files and utilities on your Linux system.

### Prerequisites

Webmin requires perl5 to be present on the target system. This can generally be verified by looking for *either* of the following directory locations:

- ▶ /usr/bin/perl
- ▶ /usr/local/bin/perl

### Downloading the Webmin source

Webmin was not installed on our Turbolinux system, nor did we find it available as an RPM. We found the Webmin documentation and source code package at the following URL:

<http://www.webmin.com/webmin/download.html>

We downloaded the file webmin-0.86.tar.gz.

### Installation of Webmin

The installation of Webmin was straightforward. We installed it as follows:

```
# cp webmin-0.86.tar.gz /usr/local
# cd /usr/local
# tar -xvzf webmin-0.86.tar.gz
# cd webmin-0.86
# ./setup.sh
```

**Note:** The installation script allows for the customization of Webmin installation, such as allowing for non-default file locations.

However, in our testing, all we specified was the Linux distribution we were installing on and an alternative Webmin user/password. We accepted all the other defaults.

### Usage of Webmin

As its name implies, Webmin is accessed via a browser. It listens on port 10000. Therefore, you can test your Web browser by pointing it to the following URL:

<http://yourhostname:10000/>

The Webmin GUI interface is very intuitive and is used by simply clicking the particular function you're interested in (for example **Users and Groups**). The Webmin main screen is shown in Figure 9-3.

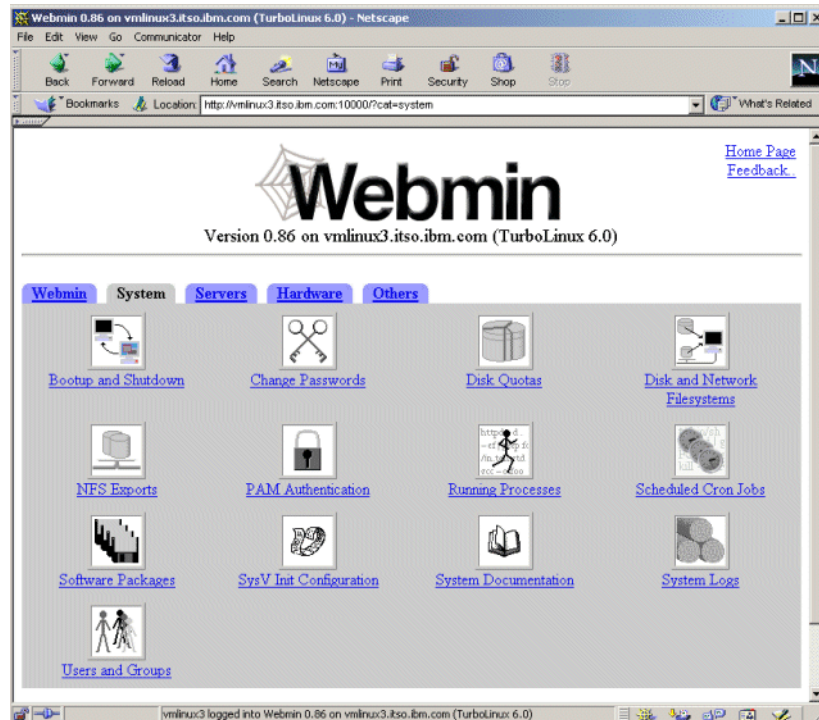


Figure 9-3 Webmin main screen





## Overview of Red Hat Linux

This chapter describes the Red Hat Linux for S/390 and zSeries distribution. A beta version of Red Hat Linux was originally available in February 2001. As of the time of writing, it is not yet generally available (GA).

Red Hat is based on a 2.2.19 kernel. The version we tested had a compilation date of Tue Apr 24 15:17:30 EDT 2001. As well as the 2.2.19 kernel, a 2.4.3 kernel is available for installation. Just about all the other Linux distributions are based on one of the 2.2 kernels.

Contrary to what other Linux for zSeries or S/390 distributions have done, Red Hat has incorporated glibc 2.2, with no provision for installing glibc 2.1.3. This caused us problems when we tried to install IBM's DB2 product, since that package requires glibc 2.1.3.

Red Hat provides an administration tool called setup, but most users install linuxconf for administering their system.

## 10.1 Obtaining the Red Hat distribution

Since Red Hat was still in beta testing at the time of writing, the only way to get the software was to download it from one of their FTP servers:

```
ftp://ftp.redhat.com/pub/redhat/linux/rawhide/s390
ftp://ftp.redhat.de/pub/s390
```

The three “starter” installation files you will need can be retrieved from:

```
ftp://ftp.redhat.com/pub/redhat/linux/rawhide/s390/images
ftp://ftp.redhat.de/pub/s390/images
```

The software packages (RPMs) will need to be retrieved from:

```
ftp://ftp.redhat.com/pub/redhat/linux/rawhide/s390/RedHat/RPMS
ftp://ftp.redhat.de/pub/s390/RedHat/RPMS
```

The ftp site ftp.redhat.com is usually extremely busy, and you may find it nearly impossible to access it via anonymous FTP. You should seriously consider using the ftp.redhat.de site, or one of the other Red Hat FTP mirrors that are listed at:

```
http://www.redhat.com/download/mirror.html
```

Excluding the source RPMs, you will need about 1.1 GB of disk space to hold the distribution.

There are no .iso images available yet, just the individual RPMs and source RPMs. There are only 45 Linux for S/390 SRPMS on the FTP servers at this time.

Because there are 1163 RPMs, you will need a convenient way to download them all. You may first have to socksify your Linux system if your company has a firewall between your desktop and the Internet. One package that accomplishes this for Linux systems is named *tsocks*. We found the package on the Internet at:

```
http://ftp1.sourceforge.net/tsocks/tsocks-1.7.tar.gz
```

We followed the instructions on how to socksify our Linux system. We tested a **ping** to an Web site server to verify our system was socksified.

After the Internet is accessed, there are several ways to get all the Red Hat Linux files, depending on what platform you are using. One option for Windows systems would be to use a GUI FTP client, such as WS\_FTP from Ipswitch, Inc., at:

```
http://www.ipswitch.com/Products/WS\_FTP/index.html
```

While there is a freely downloadable version of this product, commercial use requires a purchased license. This tool can retrieve an entire FTP site, creating the appropriate directory structure on the fly. This is relatively important, because the Red Hat installation process expects the installation files and directories to be in a certain structure. WS\_FTP can also be configured to only download files that have been changed since the last time they were downloaded. This makes keeping a mirror copy of a site relatively painless.

A second option, for Windows clients, would be to use the command line FTP client in conjunction with the FTP **mget** and **prompt** subcommands. These subcommands allow you to retrieve all of the files in a directory that match the specified pattern. The important pieces of an FTP session follow:

```
# ftp ftp.redhat.de
ftp> cd /pub/s390/RedHat/RPMS/
ftp> binary
ftp> prompt
ftp> mget *.rpm
150 Opening BINARY mode data connection for 4Suite-0.11-2.s390.rpm (1520605
bytes).
150 Opening BINARY mode data connection for Canna-3.5b2-43.s390.rpm
(1065846 bytes).
150 Opening BINARY mode data connection for Canna-devel-3.5b2-43.s390.rpm
(465980 bytes).
465980 bytes received in 9.95 seconds (46.84 Kbytes/sec)
...
ftp> quit
```

A final option for MS Windows systems would be to use the **wget** command from the Cygwin environment for Windows, which is available at:

<http://sources.redhat.com/cygwin>

The **wget** command is a very powerful, flexible command from GNU and the Free Software Foundation that has been ported from the UNIX/Linux world. Like WS\_FTP, **wget** can be used to retrieve an entire FTP site (or sub-tree thereof) while creating the directory structure dynamically. For example, the following command will retrieve the entire FTP server from the /pub/s390 directory down, including the files that are in /pub/s390:

```
# wget -m ftp://ftp.redhat.de/pub/s390
```

If all you want to retrieve are the contents of the documentation directory, then the following command will do that:

```
# wget -m ftp://ftp.redhat.de/pub/s390/docu/
```

For Linux/UNIX systems, using **wget** to retrieve the distribution is probably the best option available.

The directory structure that the Red Hat installer expects to be below the s390 directory is as follows:

```
s390
|
+---images
|
+---SRPMS
|
+---RedHat
|
+---RPMS
|
+---base
```

Note that the names of the directories leading up to and including “s390” in this example are of no concern to the installer. All the installer cares about are the directories below that. For example, the following directory structure will work:

```
/home
|
+---userid
|
+---ftp.mirror.site.com
|
+---pub
|
+---RedHat_Mirror
|
+---linux
|
+---rawhide
|
+---s390
|
+---images
|
+---SRPMS
|
+---RedHat
|
+---RPMS
|
+---base
```



## 10.2 Documentation available

Red Hat has not yet written any documentation for their distribution. The only documentation that has been available during the beta is what was written for the platform by other people and organizations. Red Hat has placed this documentation on their FTP servers, and it can be downloaded from:

<ftp://ftp.redhat.de/pub/s390/docu/>

The index.html file there points to these documents:

### **General information about Linux**

- ▶ *Linux questions and answers / A Linux Whitepaper* (l-faq.pdf)

### **General information manuals on Linux for S/390**

- ▶ This IBM Redbook *Linux for zSeries and S/390: Distributions* (sg246264.pdf)
- ▶ The IBM Redbook *Linux for S/390* (sg244987.pdf)
- ▶ *The Dinosaur and the Penguins: S/390 Qualities of Service for Linux* (dinopenguin.pdf)
- ▶ Frequently asked questions (FAQ) for Linux for zSeries/ S/390 (209faq.pdf)
- ▶ *S/390 Virtual Image Facility for Linux - Guide and Reference* (vifsl500.pdf)

The following specific installation guide is available:

- ▶ *Installation of IBM WebSphere 3.5 on Red Hat Linux for S/390* (WebSphere-HOWTO.html)

The following developer guides are available on the documentation site:

- ▶ *ELF Application Binary Interface Supplement* (l390abi0.pdf)
- ▶ *Linux for S/390 Device Drivers and Installation* (l390dd24.pdf)
- ▶ *Using the Dump tools (Kernel 2.2)* (l39dmp22.pdf)
- ▶ *Using the Dump tools (Kernel 2.4)* (l39dmp24.pdf)
- ▶ The redpaper *Building Linux Systems under IBM VM* (redp0120.pdf)

The list of software RPMs that are installed on Red Hat Linux are listed in an appendix entitled *Linux Software Packages*, which is available separately on the Web at:

<ftp://www.redbooks.ibm.com/redbooks/SG246264/RPMappendix.pdf>

## 10.3 How Red Hat Linux differs

When selecting software to include on their distributions, Red Hat tends to be more “bleeding edge” than others. For example, Red Hat’s Linux beta software includes a default kernel at the 2.2.19 level, and glibc 2.2, while others are sticking with what IBM is still calling the “GA” versions of 2.2.16, and 2.1.3. From our experience, this may cause problems when trying to install other software, such as IBM’s DB2.

Red Hat provides only a very rudimentary administration tool named *setup*. Linuxconf (an open software tool) is the software package that Red Hat recommends for this task.

Red Hat has provided the means to put a lot of installation/setup information in the parameter file, to allow a quicker and easier installation. For more information on this, see 11.2, “Preparing for installation” on page 158.

Red Hat’s installation scripts support more installation types than the other distributions:

- ▶ NFS
- ▶ FTP
- ▶ HTTP
- ▶ SMB server (Microsoft file and print sharing)

Most other distributions allow NFS and FTP installation, but not HTTP nor SMB.



# Installing Red Hat Linux

This chapter describes installation of Red Hat Linux for S/390 and zSeries under the following scenarios:

- ▶ Under VM
- ▶ In an LPAR
- ▶ Under VIF

For all these scenarios, make sure you have a current copy of the Red Hat installation files and packages. During our testing, we encountered problems because of a version mismatch between the installation files and some of the RPM packages.

## 11.1 Before you install

No matter which method you use to install, you will need to have the Red Hat installation files available to your system via one of four methods:

- ▶ NFS server
- ▶ FTP server
- ▶ HTTP server
- ▶ SMB server (Microsoft file and print sharing)

Our testing showed that the Red Hat installer will work with an FTP or HTTP server that is either Linux/UNIX-based *or* MS Windows-based. This is because it does not try to parse the output from an `ls` command to the server. Instead, there is a file in the installation `/base` directory that contains the names of all the RPM files that are available.

The RAMdisk has another set of files with predetermined packages in them, which are used to look up the RPM names in the `/base/list` file. A retrieval request for the specific RPM name is then sent to the server via the `wget` command. (This makes changing the contents of the predetermined packages awkward, and could probably be improved. There is no reason the file needs to be in the RAMdisk; it could just as easily be on the file server.)

We did not test the installation from an SMB server.

## 11.2 Preparing for installation

The amount of disk space needed for the default and full installations is listed in Table 11-1. At the time of our testing, these were the only choices available.

Table 11-1 Red Hat disk requirements

Directory	Installation type	
	default	full
/bin	5.4 MB	6 MB
/boot	1.7 MB	1.7 MB
/dev	104 KB	104 KB
/etc	8.2 MB	12 MB
/lib	17 KB	17 MB
/opt	0	0

Directory	Installation type	
	default	full
/root	32 KB	44 KB
/sbin	5.1 MB	9 MB
/tmp	140 KB	288 KB
/usr	630 MB	2,300 MB

Red Hat has named their IPL parameter file `inst.parm`. (The README file still refers to `inst22.parm`.) It allows many more variables than that of other distributions. Edit the `inst.parm` file to match your installation requirements. We used the following file for all three Red Hat installation scenarios:

```

root=/dev/ram0 ro ip=off DASD=200-20f
HOST=vmlinux4.itso.ibm.com:eth0:9.12.6.80:1500
NETWORK=9.12.6.0:255.255.255.0:9.12.6.255:9.12.6.75
LCS=0x2926,0
DNS=9.12.14.7
SEARCHDNS=itso.ibm.com:ibm.com
RPMSEVER=ftp://9.12.2.120
MOUNTS=/dev/dasdb1:/,/dev/dasdc1:/usr
SWAP=/dev/dasdd1
ROOTPW=root11
INSTALL=default

```

Note the following regarding the parameter file. When IPLing from the VM reader, the parameter file is limited to 80 bytes per line, with a maximum of 11 lines. When IPLing from tape, the parameter file is limited to one line of 1024 bytes.

Red Hat has apparently made some extensions to the parameters, which can be specified to allow quicker and easier installation. If all the values shown in the example are filled out for your installation, you will not be prompted for any information at the console, and will only have to verify the values given during the rest of the installation process. This avoids having to re-type the information over and over again if you need to restart the process.

One parameter we recommend that you do *not* specify is `ROOTPW`. This causes a syntax error in the installation script, and the password is not set.

If you do not specify this parameter, you will be prompted for a password, and it will be set correctly.

Some parameter values (for example `INSTALL=`) are not documented in the `README` file in the root directory of the FTP server. The only valid values for this parameter are `all` and `default`.

The remaining parameters were documented as follows:

If you want to, customize the parameter file (`inst22.parm`). It takes the following parameters:

```
root=/dev/ram0 ro ip=off
```

These values are given directly to the linux kernel and should be left as provided

```
HOST=<...>
```

The values for the `HOST` statement depends, whether you want to install this machine via the ethernet driver or via `ctc/iucv` devices:

```
HOST=fqdn:device:ipaddr[:mtu] (for ethernet devices)
```

`fqdn`: your full qualified domain name of the virtual machine  
`device`: `eth0`

`ipaddr`: the fixed IP-address for this virtual machine  
`mtu`: maximum transfer units (optional, should be 1492 or 1500)

```
HOST=fqdn:device:ipaddr:gateway[:mtu] (for ctc devices)
```

`fqdn`: your full qualified domain name of the virtual machine  
`device`: `ctc0`

`ipaddr`: the fixed IP-address for this virtual machine  
`gateway`: IP-address of the gateway

`mtu`: maximum transfer units (optional, should be 1492 or 1500)

```
HOST=fqdn:device:ipaddr:host:gateway[:mtu] (for iucv devices)
```

`fqdn`: your full qualified domain name of the virtual machine  
`device`: `iucv0`

`ipaddr`: the fixed IP-address for this virtual machine  
`host`: VM host name of IUCV partner

`gateway`: IP-address of the IUCV partner  
`mtu`: maximum transfer units (optional)

```
DASD=x-y
```

Range of addresses of your DASD devices.

200-20f should be sufficient in most cases.

```
DTZ=timezone
```

default time zone as returned by the `tzselect`-utility,

eg: `DTZ=Europe/Berlin` or `DTZ=America/Chicago`

```
LCS=base,type
```

`type` defines an OSA-2 with LCS (1) or an OSA-Express with QDIO/QETH (2) device with the base address of `base`,

e.g. `LCS=0xfc20,1` for and OSA-2 with LCS at `0xfc20`

or `LCS=0xf000,2` for an OSA-Express with QDIO/QETH at `0xf000`

```
NETWORK=IP:netmask:broadcast[:gw]
```

`IP`: Your IP

`netmask`: the netmask

`broadcast`: the broadcast address

`gw`: the gateway-IP for your eth device (for eth-device only)

```
DNS=list:of:dns:servers
```

the list of DNS servers, separated by colons  
e.g. DNS=10.0.0.1:10.0.0.2  
will use the DNS servers 10.0.0.1 and 10.0.0.2  
SEARCHDNS=list:of:search:domains  
the list of the search domains, separated by colons  
eg: SEARCHDNS=redhat.de:redhat.com  
RPMSEVER=ftp://your.ftp.server/your.s390.rpm.dir  
RPMSEVER=http://your.http.server/your.s390.rpm.path  
RPMSEVER=IP:/dir  
Your Red Hat Linux for S/390 binary RPM packages may be located on  
any of your ftp- or http-servers in the given directories, or you may  
give an IP-address and a directory of a NFS server (e.g. a mount of  
your Red Hat Linux for S/390 CD-ROM).  
MOUNTS=dev:mountpoint,dev2:mountpoint2  
defines a comma-delimited list of DASD device-partitions  
and where they should be mounted by default.  
eg: MOUNTS=/dev/dasda1:/,/dev/dasdb1:/usr,/dev/dasdc1:/tmp  
SWAP=list:of:swap:devices  
the list of initial swap devices, delimited by colons  
ROOTPW=my\_secret\_password  
the default unencrypted root password.  
eg: ROOTPW=sEcrEt  
DEBUG=<value>  
if set to any value you will get a shell during  
installation to debug if you encounter problems.

**Note:** The DASD keyword, and all Red Hat extensions, need to be in all upper case letters. However, the *values* of those parameters are case sensitive in the normal way, so take care that whatever editor you use is in its mixed case mode.

The installer script will take whatever you specify for RPMSEVER, and append /RedHat/base/list to it to find the list of all RPMs that are available. Keep this in mind when determining what to specify for this value.

As can be seen from the installation worksheet that follows, a number of IP networking parameters are required. Your IP network group/administrator or system programmer will have to provide these.

When specifying the device numbers for a Linux guest running under VM, use the virtual device addresses that are defined for the guest in the user directory (see 3.1.1, “The VM user directory” on page 28).

Table 11-2 Installation worksheet

1. VM guest user ID (if installing under VM)	
2. VM guest password (if installing under VM)	
3. Device: 1st DASD/vdev(VM) (/ file system)	
4. Device: 2nd DASD/vdev(VM) (swap space)	
5. Device: tape unit *optional (eg. 0181)	
6. Host name (eg. vmlinux.itso.ibm.com)	
7. Network device type (eg. eth0, ctc0)	
8. Host IP address (eg. 9.12.6.80)	
9. IP address of CTC/IUCV point-to-point partner	
10. Netmask (eg. 255.255.255.0)	
11. Broadcast IP address (eg. 9.12.6.255)	
12. Network IP address (eg. 9.12.6.0)	
13. Default gateway IP address (eg. 9.12.6.75)	
14. OSA-2 with LCS or OSA-E with qdio/qeth	
15. Network device and port 1st of pair (eg. 2926,0)	
16. Domain name search list (eg. itso.ibm.com,ibm.com)	
17. DNS IP address (eg, 9.12.14.7)	
18. MTU size (eg. 1500)	
19. FTP server IP address or hostname	
20. FTP path (eg. /mnt/cdrom)	
21. FTP user ID (eg. TOT32)	
22. FTP password	
23. Linux root user password	

## 11.3 Installation of Red Hat under VM

In this section, we cover the steps needed to install the Red Hat distribution as a guest operating system under VM.



### 11.3.1 Red Hat under VM checklist

1. Read the preparation section for installing Linux under VM.  
Make sure your VM guest has enough disk capacity on its 191 disk to hold a minimum of 6 MB of data. More would be preferable (perhaps 10 to 20 MB); refer to “Prepare the VM guest for Linux” on page 28.
2. Plan your DASD configuration:
  - a. Which volumes are going to be used.
  - a. What file systems are going to be on what volumes.
  - a. How big those volumes need to be.
3. Fill out the installation worksheet (see Table 11-2 on page 162).
4. Logon to the VM guest you will be using as your Linux system.
5. Make sure the following files are available to your VM/CMS guest and stored on the 191 disk (or another disk that is accessible to CMS):
  - INST EXEC - the REXX exec used to write the following three files to the VM virtual reader.
  - VMKRNL22 BIN - the kernel to be used for IPLing under VM.
  - INST PARM - the parameter file to be used for IPLing under VM.
  - VMINRD22 BIN - the RAMdisk containing the system to install Linux under VM.

This can be done in several ways, but using the VM/CMS FTP client is probably the easiest (refer to “Getting the installation files for VM” on page 30).
6. Edit the kernel parameter file, `inst.parm`, according to your installation’s requirements. Putting as much information as possible in this file will make the rest of the installation much easier.
7. Run `inst exec` to punch the three files to your virtual reader and IPL. You’ll see output similar to Example 11-1 on page 166.
8. Continue with the rest of the installation as documented in “Common installation steps” on page 165.

## 11.4 Installation of Red Hat in an LPAR

In this section, we cover the steps needed to install the Red Hat distribution in an LPAR. These same steps would be followed if you install on the “bare metal.”

### 11.4.1 Red Hat in an LPAR checklist

1. Read the preparation section for installing Linux in an LPAR (refer to “Prepare the LPAR for Linux” on page 31).
2. Plan your DASD configuration:
  - a. Which volumes are going to be used.
  - a. What file systems are going to be on what volumes.
  - a. How big those volumes need to be.
3. Fill out the installation worksheet (see Table 11-2 on page 162).
4. Edit the kernel parameter file, `lpar.prm`, according to your installation’s requirements. Putting as much information as possible in this file will make the rest of the installation much easier.
5. Prepare media, depending on one of the following initial IPL methods:
  - a. From a tape (see 3.4.3, “Booting from tape” on page 44).
  - b. From the HMC CD-ROM or an FTP server (see 3.4.2, “Booting from HMC CD-ROM or FTP server” on page 35).
  - c. From an ICKDSF bootstrap (see 3.4.1, “Using the ICKDSF bootstrap” on page 34).
6. IPL Linux, using the appropriate media depending on the IPL method.
7. Continue with the rest of the installation as documented in 11.6, “Common installation steps” on page 165.

### 11.4.2 Comments on LPAR installation

Aside from the inherent differences between installing under VM and in an LPAR, the installation proceeded exactly the same. There were no problems encountered in one scenario and not the other.

## 11.5 Installation of Red Hat under VIF

In this section, we cover the steps needed to install the Red Hat distribution under VIF.

The installation of Red Hat Linux under VIF is similar to that under VM, with one exception: because you do not have the VM console available, the *only* way to access your linux0 system is via SSH, or via the HMC. Telnet access is disabled, by default.

## 11.5.1 Red Hat under VIF checklist

1. Read the preparation section for installing Linux in an LPAR, “Prepare the LPAR for Linux” on page 31.
2. Plan your DASD configuration:
  - a. Which volumes are going to be used.
  - a. What file systems are going to be on what volumes.
  - a. How big those volumes need to be.
3. Fill out the installation worksheet (see Table 11-2 on page 162 ).
4. In the root directory of the Red Hat FTP server, there is a file named `redhat.ins`. This file points to the initial RAMdisk, kernel, and kernel parameter files to be used by VIF for setting up the master Linux system, `Linux0`.  
  
Edit the kernel parameter file, `lpar.prm`, according to your installation’s requirements. Putting as much information as possible in this file will make the rest of the installation much easier.
5. Start the IPL of the VIF tape. Follow the process that is documented in 16.5.3, “Load VIF tape onto DASD” on page 278.
6. When the Red Hat installation system is IPLed by VIF, continue with the rest of the installation as documented in “Common installation steps” on page 165.
7. Edit `/etc/fstab` to make the 203 volume (which contains the `vif` command) read-only:

```
/dev/dasdb1    /vif    ext2    defaults,ro    0 0
```

## 11.5.2 Comments on VIF installation

Aside from the inherent differences between installing under VM and under VIF, the installation proceeded exactly the same. There were no problems encountered in one scenario and not the other.

## 11.6 Common installation steps

In this section we provide examples which illustrate the steps that are common to both types of installations.

If you're installing Red Hat in an LPAR, or under VIF, some minor details in the examples that follow will differ. In particular, the VM console More/Holding/Running indicator will not be seen on the HMC, and CP commands will not be available for IPLing.

Otherwise, the input and output shown should be identical.

*Example 11-1 Beginning of the IPL process at the console*

---

```
CP IPL OOC CLEAR
0000003 FILES CHANGED
Linux version 2.2.19-0.07B00Tvrdr (laroche@rawhide.redhat.de) (gcc version
2.95. 3 20010319 (prerelease Red Hat Linux S/390)) #1 SMP Tue Apr 24
15:31:32 EDT 2001

Command line is:root=/dev/ram0 ro ip=off DASD=200-20f
HOST=vmlinux4.itso.ibm.com:eth0:9.12.6.80:1492
NETWORK=9.12.6.0:255.255.255.0:9.12.6.255:9.12.6.75
LCS=0x2926,0
DNS=9.12.14.7
SEARCHDNS=itso.ibm.com:ibm.com
RPMSEVER=ftp://9.12.2.120/
MOUNTS=/dev/dasdb1:/,/dev/dasdc1:/usr
SWAP=/dev/dasdd1
ROOTPW=root11
INSTALL=default

We are running under VM
This machine has an IEEE fpu
Initial ramdisk at: 0x02000000 (3586960 bytes)
Detected device 2926 on subchannel 0000 - PIM = 80, PAM = 80, POM = FF
Detected device 2927 on subchannel 0001 - PIM = 80, PAM = 80, POM = FF
Detected device 0191 on subchannel 0002 - PIM = F0, PAM = F0, POM = FF
Detected device 0201 on subchannel 0003 - PIM = F0, PAM = F0, POM = FF

HOLDING  VMLINUX
```

---

1. If you chose not to completely specify all the parameters in `inst.parm/lpar.prm`, answer the questions asked by the setup script:
  - a. The fully qualified domain name of your Linux system, for example:  
`vmlinux4.itso.ibm.com`
  - b. Which communications device are you going to use, that is: `ctc0`, `eth1`, `icuv2`, and so on
  - c. The IP address of your Linux system.
  - d. The network mask for this system, for example: `255.255.255.192`

- e. The broadcast address of this system. This will be dependent on your network mask. For a subnet mask of 255.255.255.0, there is only one broadcast address, at xx.xx.xx.255.
- f. The network address of this system. This is the result of a logical AND of your IP address, and subnet mask.  
For a subnet mask of 255.255.255.0, this simply means that the value of the last octet will always be 0. For any other subnet mask, it's best to break out the hexadecimal calculator.
- g. The IP address of your default gateway. This is the address to which your system will send all its IP traffic if it doesn't know of a specific route to the destination. Some people use the lowest address in the subnet for their gateways. Others use the next-to-highest address. Still others pick something else.

There is no way to calculate or guess at this address; you'll have to ask your IP network administrator/group for the value.

- i. Depending on what kind of communications device you said you had, you will be asked further questions about it.
- ii. For a CTC, it will ask for the "IP of your ctc/iucv point-to-point partner:" (which means the system on the other end of the virtual CTC).
- iii. For iucv, you will be asked to "Enter iucv kernel module options (usually iucv=HOST):" where "HOST" is the VM user ID of the system on the other end of the iucv connection.
- iv. If you said you were using eth0, you will be asked whether you are using an OSA-2 or OSA-Express. If you reply OSA-2 (and perhaps for OSA-Express), you will be prompted for the "OSA Device address (e.g. fc20,1 - or "auto" (may not work)):"

In our test case, the value we entered was: 2926,0.

- 2. Once the network setup is complete, you will be instructed to telnet into your Linux system and finish the installation, as in Example 11-2. You will *not* be required to specify a password to logon to the system.

*Example 11-2 End of the first phase*

---

```
eth0      Link encap:Ethernet  HWaddr 00:06:29:6C:CB:CE
          inet addr:9.12.6.80  Bcast:9.12.6.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1492  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:3924  Metric:1
```

```
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
```

Kernel IP routing table

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
127.0.0.1	0.0.0.0	255.255.255.255	UH	0	0	0	lo
9.12.6.0	0.0.0.0	255.255.255.0	U	0	0	0	eth0
9.12.6.0	0.0.0.0	255.255.255.0	U	0	0	0	eth0
0.0.0.0	9.12.6.75	0.0.0.0	UG	0	0	0	eth0

Starting portmap.

Starting telnetd to allow login over the network.

Please telnet now to 9.12.6.80 and start 'rhsetup'!

Bringing up debugging shell in 3270...

/bin/sh: No controlling tty (open /dev/tty: No such device or address)

/bin/sh: warning: won't have full job control

#

RUNNING VMLINUX

---

3. After you have logged onto the system, enter the command **rhsetup** to begin the rest of the installation:

```
Linux 2.2.19-0.07B00Tvrdr ((none)) (tty0)
```

```
Welcome to Red Hat Linux on S/390.
```

```
Please run 'rhsetup' to start the installation.
```

```
# rhsetup
```

```
Debug: this machine has IP = 9.12.6.80 and device = eth0
```

```
Loading DASD kernel module. This can take a while.
```

4. Verify that the DASD volumes to be formatted are correct. The screen should look similar the one shown in Figure 11-1 on page 169.

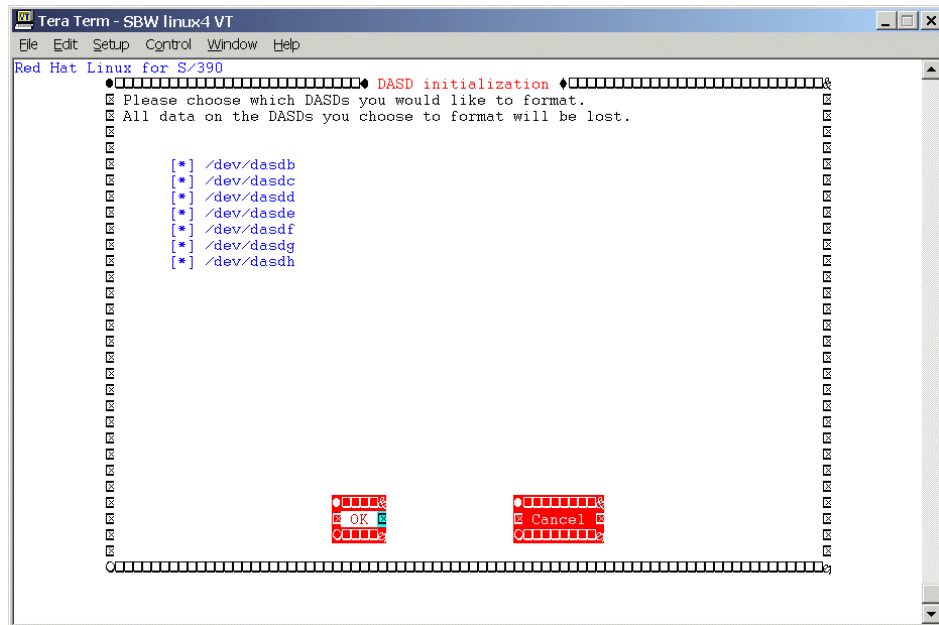


Figure 11-1 Red Hat DASD initialization panel

5. Start the formatting process by tabbing to the **OK** button and pressing Enter. You will see a succession of status boxes with the heading DASD Initialization, telling you what volume is currently being processed (depending on the size of your volumes, this can take quite a while).

6. Verify that the DASD volumes are being assigned to the file systems and mount points you want. This is done via the dialog box, as shown in Figure 11-2.

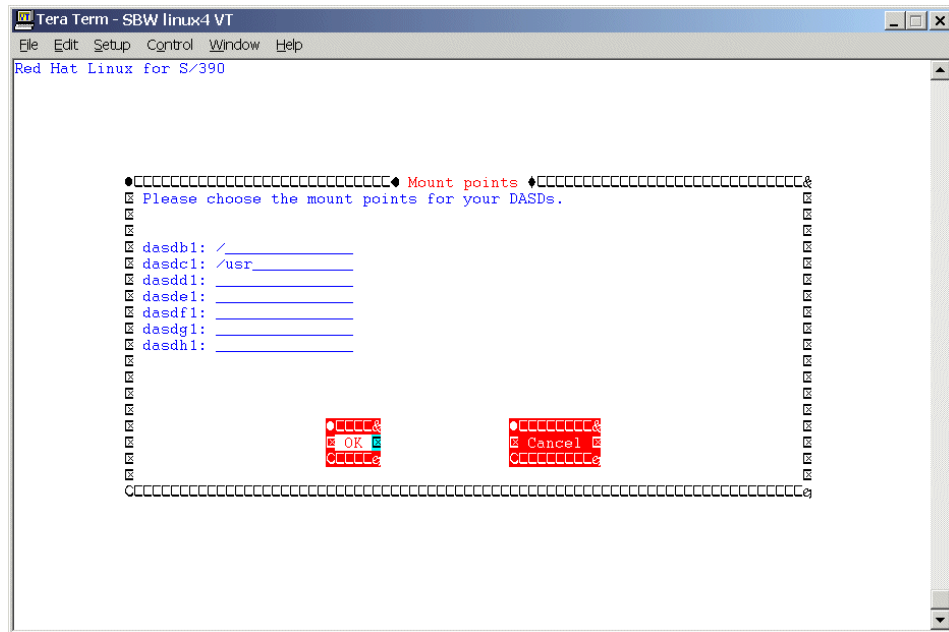


Figure 11-2 Red Hat mount points panel

7. Verify that the network information is correct, as shown in the panel with the title *Network Setup*.



- Verify the protocol to be used to retrieve the packages, and their location on the server, as shown in Figure 11-3.

If you use an FTP server that does not allow anonymous access, you can specify a user ID and password that is a valid account on that machine as follows:

```
ftp://userid:password@ftp.server.my.com/path/to/packages
```

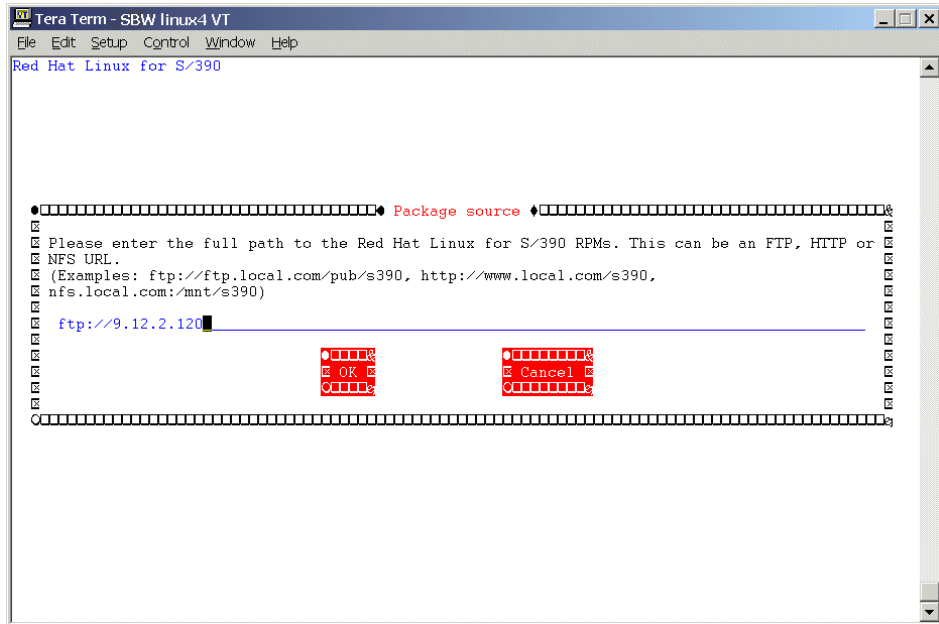
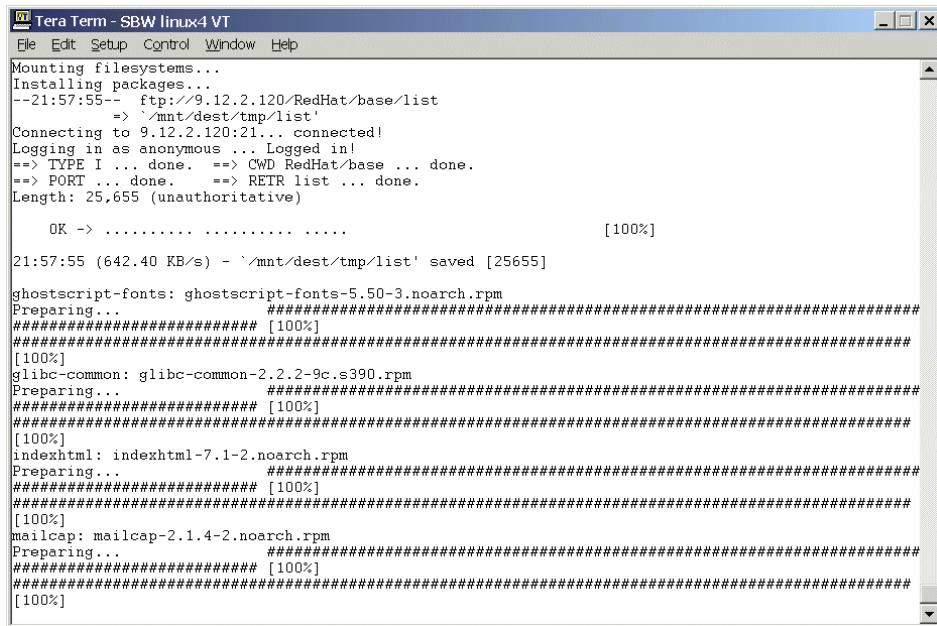


Figure 11-3 Red Hat package sources panel

9. The installation process will begin, and you will see the progress of packages added as they installed, as shown in Figure 11-4.



```
Tera Term - SBW linux4 VT
File Edit Setup Control Window Help
Mounting filesystems...
Installing packages...
--21:57:55-- ftp://9.12.2.120/RedHat/base/list
=> `/mnt/dest/tmp/list'
Connecting to 9.12.2.120:21... connected!
Logging in as anonymous ... Logged in!
==> TYPE I ... done. ==> CWD RedHat/base ... done.
==> PORT ... done. ==> RETR list ... done.
Length: 25,655 (unauthoritative)

OK -> ..... [100%]

21:57:55 (642.40 KB/s) - `/mnt/dest/tmp/list' saved [25655]

ghostscript-fonts: ghostscript-fonts-5.50-3.noarch.rpm
Preparing... ##### [100%]
##### [100%]
##### [100%]
glibc-common: glibc-common-2.2.2-9c.s390.rpm
Preparing... ##### [100%]
##### [100%]
##### [100%]
indexhtml: indexhtml-7.1-2.noarch.rpm
Preparing... ##### [100%]
##### [100%]
##### [100%]
mailcap: mailcap-2.1.4-2.noarch.rpm
Preparing... ##### [100%]
##### [100%]
##### [100%]
```

Figure 11-4 Red Hat installation progress panel

10. At the end of the installation, the `rhsetup` script will run `zilo`, add your network device driver parameters to `/etc/modules.conf`, and then set the password for the root user ID.

In our initial test, we had specified `/dev/dasdb` as the root file system, and used the `R00T=` parameter in the `inst.parm` file. As a result, we experienced failures with `zilo` and setting the root password, as shown in Example 11-3:

Example 11-3 Unsuccessful end of the installation process

```
sh-2.04# /sbin/zilo
Testlevel is set to 0
IPL device is: '/dev/dasda'
/boot/ipleckd.boot is not on device (94/0) but on (94/4)
zilo.c (line:400) 'verify_file (o->bootsect, dev)' returned 22='Invalid
argument'
bootsector is: '/boot/ipleckd.boot'Usage:
zilo -d ipldevice [additional options]
-d /dev/node : set ipldevice to /dev/node
-f image : set image to image
-F conffile : specify configuration file (/etc/zilo.conf)
-p parmfile : set parameter file to parmfile
```

```

-b bootsect : set bootsector to bootsect
Additional options
-B bootmap:
-v: increase verbosity level
-v#: set verbosity level to #
-t: decrease testing level
-h: print this message
-?: print this message
-V: print version
sh-2.04# /usr/sbin/pwconv
sh-2.04# # Allow root logins over 3270
sh-2.04# echo "console" >> /etc/securetty
sh-2.04# # Handle the LCS and QETH module if necessary
sh-2.04# if [ -n "noauto=1 devno_portno_pairs=0x2926,0" ]; then
> echo "options lcs noauto=1 devno_portno_pairs=0x2926,0" >>
/etc/modules.conf
> fiecho "options lcs noauto=1 devno_portno_pairs=0x2926,0" >>
/etc/modules.conf
sh-2.04# if [ -n "" ]; then
> echo "alias eth0 qeth" >> /etc/modules.conf
> echo "options qeth " >> /etc/modules.conf
> fi
sh-2.04# # Handle the IUCV module if necessary
sh-2.04# [ -n "" ] && echo "options netiucv " >> /etc/modules.conf
sh-2.04# # enable default Timezone if available
sh-2.04# [ -n "" ] && /usr/sbin/timeconfig
sh-2.04# exit
passwd: This option requires a username
root password set to bootparameter-provided token

```

Congratulations

You have successfully installed Red Hat Linux on S/390 !

Next, set your linux user in vm to boot into linux and re-login.

---

11. If you encounter similar problems, you can correct them as described in 11.7, “Problems we encountered” on page 174.

12. If your inst.parm file was set up to avoid these problems with the install script, you should see output identical to that of Example 11-3 (without the highlighted error messages).

13. Ensure that your DASD volumes are unmounted:

```

# mount
/dev/root on / type ext2 (rw)
none on /proc type proc (rw)

```

14. Re-IPL from DASD:

```

#CP IPL 201 CLEAR

```

## 11.7 Problems we encountered

We encountered a few problems along the way which are worth mentioning here:

- ▶ `nfslockd` appears to start, but vanishes without a trace. If the `nfsd` daemon is started, `nfsd` will start `lockd`, and `lockd` will remain running.
- ▶ The Red Hat kernel has support for serial devices in it. This results in seeing the following error message during IPL:

```
modprobe: modprobe: Can't locate module char-major-4.
```

This can be corrected by adding the following entry to `/etc/modules/conf`:

```
alias char-major-4 off
```

- ▶ The documentation for the `LCS=` parameter is incorrect. It says to use a value of `'address,1'` if you are using an OSA-2 card, and a value of `'address,2'` if you are using an OSA-Express.

In our testing using an OSA-2 card, this did not work. We had to specify `'address,portnumber'` as is done when passing parameters directly to the `lcs.o` module.

- ▶ The `INSTALL=` parameter in `inst.parm` is not documented. The *keywords* must be in upper case. The values must be in appropriate case (usually all lower case).
- ▶ When trying a full install with a 4100 cylinder disk volume for `/usr`, `dasdfmt` seemed to have a problem with the volume. The `tune2fs` command also seemed to have a problem. The complete error messages were hidden by the installation script.

Manually issuing the `dasdfmt` and `mke2fs` commands allowed us to proceed normally.

- ▶ The installation script assumes that you will be booting from your `/dev/dasda` volume, and gives you no way to change that assumption. It then writes that information into `/etc/zilo.conf`.

If this is not the case at your site, there will be problems with `zilo`. However, the default package installs the `vi` editor, so you should be able to correct this problem as described here (keep in mind that, if you define your own packages, there is no editor—other than `sed`—on the RAMdisk, so make sure you install at least one editor on your system).

Remount your disks and then do the following:

```
chroot /mnt/dest
cd /etc
vi zilo.conf
Edit /etc/zilo.conf with the parameters that fit your installation.
Save the file and exit. (:x will do this for vi)
/sbin/zilo
```

- ▶ If you use the parameter file to specify the initial password for root, it does not get set due to a syntax error in the `rhsetup` script.

The way to avoid this is to not specify `ROOTPW=` in the parmfile. You will then be prompted for an initial password by the `rhsetup` script, and it will be set correctly.

If you forget, and specify it in the parmfile, you can set it before re-IPLing the system as follows:

```
mount /dev/dasdb1 /mnt/dest/
mount /dev/dasdc1 /mnt/dest/usr
mount /dev/dasdg1 /mnt/dest/etc
chroot /mnt/dest
/usr/bin/passwd
enter password once
enter password again
exit
umount /mnt/dest/etc
umount /mnt/dest/usr
umount /mnt/dest/
```

At this point, you can re-IPL from DASD.

- ▶ The Mozilla package does not work at all. It appears to start, but then disappears. There is a platform-dependent module (called an `xpcom` module) that needs to be written for the S/390 architecture, and this has not yet been done.
- ▶ The `kmail` program tries to locate the `libssl.so.1` and `libcrypto.so.1` libraries, but what is actually installed is `libssl.so.2` and `libcrypto.so.2`. Creating a symbolic link for each of them is an effective workaround until a permanent fix is put out:

```
ln -s /usr/lib/libssl.so.0.9.6a libssl.so.1
ln -s /usr/lib/libcrypto.so.0.9.6a libcrypto.so.1
```
- ▶ The `reiserfs` package is included in the distribution, but the official source tree for `reiserfs` does not yet support big-endian machines such as the S/390. Do not try to install this RPM, as it will not work.
- ▶ When trying to boot the 2.4.3 kernel using 3270 console support, we ran into consistent system hangs. If we were quick on the PA2 key to clear the screen when it filled up, we could get to this point:

*Example 11-4 System hang with 2.4.3 kernel and 3270 console support*

---

```
dasd(eckd): /dev/dasdb(94:4),0201 IRQ0x3:(4kB blks): 1872000kB at 48kB/trk
  classic disk layout
Partition check:
dasdb:(nonl)/      : dasdb dasdb1
dasd(eckd): /dev/dasdc(94:8),0202 IRQ0x4:(4kB blks): 1872000kB at 48kB/trk
  classic disk layout
```

```
dasdc:(nonl)/      : dasdc dasdc1
dasd(eckd): /dev/dasdd(94:12),0203 IRQ0x5:(4kB blks): 72000kB at 48kB/trk
  classic disk layout
dasdd:(nonl)/      : dasdd dasdd1
dasd(eckd): /dev/dasde(94:16),0204 IRQ0x6:(4kB blks): 144000kB at 48kB/trk
  classic disk layout
dasde: LNX1/ x0204: dasde dasde1
dasd(eckd): /dev/dasdf(94:20),0205 IRQ0x7:(4kB blks): 144000kB at 48kB/trk
  classic disk layout
dasdf: LNX1/ x0205: dasdf dasdf1
dasd(eckd): /dev/dasdg(94:24),0206 IRQ0x8:(4kB blks): 144000kB at 48kB/trk
  classic disk layout
dasdg: LNX1/ x0206: dasdg dasdg1
dasd(eckd): /dev/dasdq(94:64),0191 IRQ0x2:(4kB blks): 36000kB at 48kB/trk
  classic disk layout
dasdq: CMS1/MY191 : dasdq dasdq1
dasd: initialization finished
NET4: Linux TCP/IP 1.0 for NET4.0
IP Protocols: ICMP, UDP, TCP, IGMP
IP: routing cache hash table of 2048 buckets, 32Kbytes
TCP: Hash tables configured (established 65536 bind 65536)
NET4: Unix domain sockets 1.0/SMP for Linux NET4.0.
VFS: Mounted root (ext2 filesystem) readonly.
Freeing unused kernel memory: 14k freed
```

Linux Running

---

It was at this point that the system consistently hung up. By removing the support for the 3270 console, this problem was avoided. However, trying to SSH in to the system caused this error *immediately*:

```
HCPGIR450W CP entered; disabled wait PSW 000A0000 80022E3C
```

Rebooting with the same kernel produced an illegal opcode exception in **sshd**.

Rebooting again and logging on from the console produced another disabled wait:

```
HCPGIR450W CP entered; disabled wait PSW 000A0000 80022E3C
```

Rebooting yet again produced the same error at a different time.

At this time, we can safely recommend that you stay away from the Red Hat 2.4.3 kernel.



Chapter 12.

# Customizing and using Red Hat Linux

In this chapter we describe many aspects of customizing and using Red Hat Linux for S/390 and zSeries.

## 12.1 Creating multiple IPL volumes

Red Hat includes a module named `zilo` (instead of `silos`). Both `zilo` and `silos` require that all files used to write out the IPL/boot information physically reside on the same volume as is to be IPLed. Normally, this is not a problem; simply copying the kernel, parameter file, and boot sector file to any directory on that volume, then running `silos` from that directory, will work.

However, with `zilo`, there is a further restriction: we could not get the usual technique to work, unless the files were in a directory named `/boot`. In our case, to satisfy this restriction, we temporarily renamed `/boot` to `/bootold`, and then created a symbolic link to our directory of choice, as follows:

```
# cd /
# mv /boot /bootold
# mkdir /path/to/new/boot
# ln -s /path/to/new/boot /boot
# cp -p /usr/src/linux/arch/s390/boot/image /boot
# cp -p /usr/src/linux/System.map /boot
# cd /boot
# cp -p /bootold/ipleckd.boot /bootold/parmfile .
# zilo -d /dev/dasd? -f image -p parmfile -b ipleckd.boot
```

Once this was done, we deleted the symbolic link to `/boot`, and then decided to rename `/bootold` back to `/boot`.

Because Linux looks at the `System.map` file in `/boot` during IPL processing, you should ensure that `/boot` contains the correct files for the kernel you are IPLing (while this is not required, it will avoid having some error messages generated).

## 12.2 Installing linuxconf

`linuxconf` is *not* installed as part of the default package, so if you want to use `linuxconf` instead of `setup` and the various command line utilities, you will have to install it once your initial IPL from DASD is complete.

### ***linuxconf***

`linuxconf` comes in several varieties, and can be invoked in a number of ways:

- ▶ `linuxconf`, run from a non-X telnet/ssh session
- ▶ `linuxconf`, run from an xterm, either via telnet/ssh or X
- ▶ `linuxconf-web`, run from a browser
- ▶ `gnome-linuxconf`, run from an xterm, either via telnet/ssh or X



### ***linuxconf-web***

linuxconf-web is installed as part of the base linuxconf RPM. To enable it, you must edit `/etc/xinetd.d/linuxconf-web` and change the “disabled = yes” to “disabled = no,” or comment out the line by putting a pound sign (#) in column one.

However, we discovered a major problem with linuxconf-web; whenever we tried to connect to linuxconf-web from a browser, it would immediately start using as much CPU as was available, but would never send anything to the browser. We would have to issue a **kill** command to get it to stop. We do *not* recommend trying to run linuxconf-web at this time.

### ***gnome-linuxconf***

gnome-linuxconf is a separate RPM package from linuxconf, and can only run in an X environment. Once it is installed, this version will be invoked any time you have a terminal type of “xterm” and the DISPLAY environment variable is set. This behavior can be bypassed by specifying the `--text` option.

We also discovered problems with gnome-linuxconf involving the OSA card. Whenever we would try to set the time zone, it would immediately drive the OSA card as hard as it could, and never perform the update. Again, we had to issue a **kill** command to get it to stop.

The text-based **linuxconf** command seems to have problems with setting the time zone as well. Because the main dialog box overlays the error messages that are generated, we could not see what they were trying to tell us.

Although this is the standard system administration tool for Red Hat (and other) systems, we strongly recommend caution in its use. Take backups of your system before invoking this tool, and inspect the changes afterwards. The range of tasks that linuxconf will handle can be seen by the menu selections that are shown in Figure 12-1 and Figure 12-2 on page 181, which are screen shots of a linuxconf telnet session.

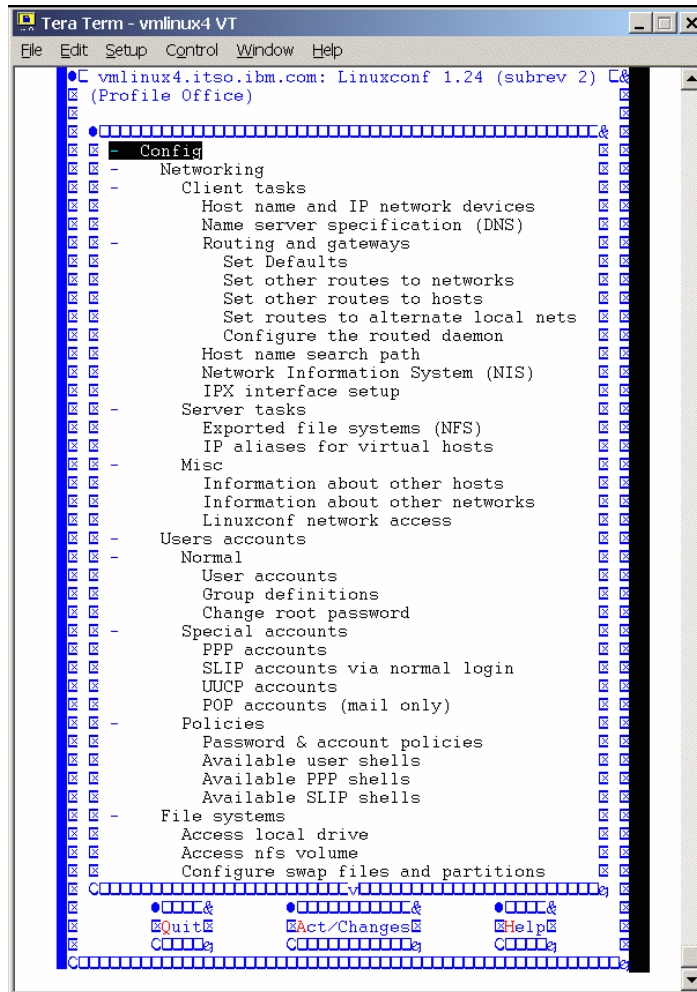


Figure 12-1 Linuxconf menu selections, screen 1

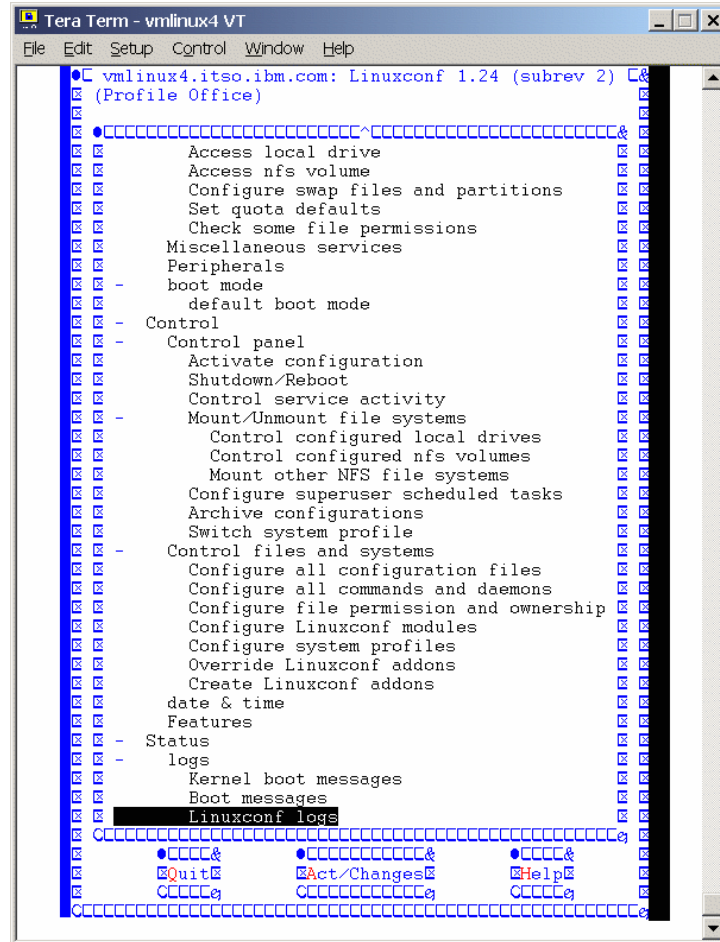


Figure 12-2 Linuxconf menu selections, screen 2

Figure 12-3 shows an xterm session, using `linuxconf --text` to invoke the tool. It looks nicer than the plain telnet text version, but doesn't have nearly the graphics/network load that a complete graphical desktop environment would have.

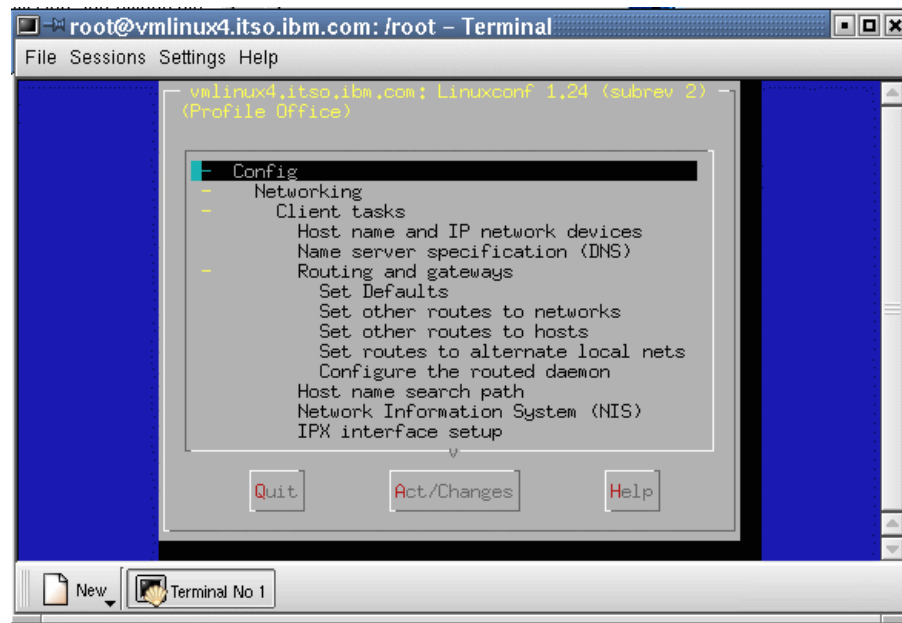


Figure 12-3 *Linuxconf via an xterm with the --text flag*

Figure 12-4 shows a running `gnome-linuxconf` session. It is invoked with just the `linuxconf` command and no parameters from an `xterm`. This tool has all the nice features of a graphical interface, but does create more network traffic as a result.

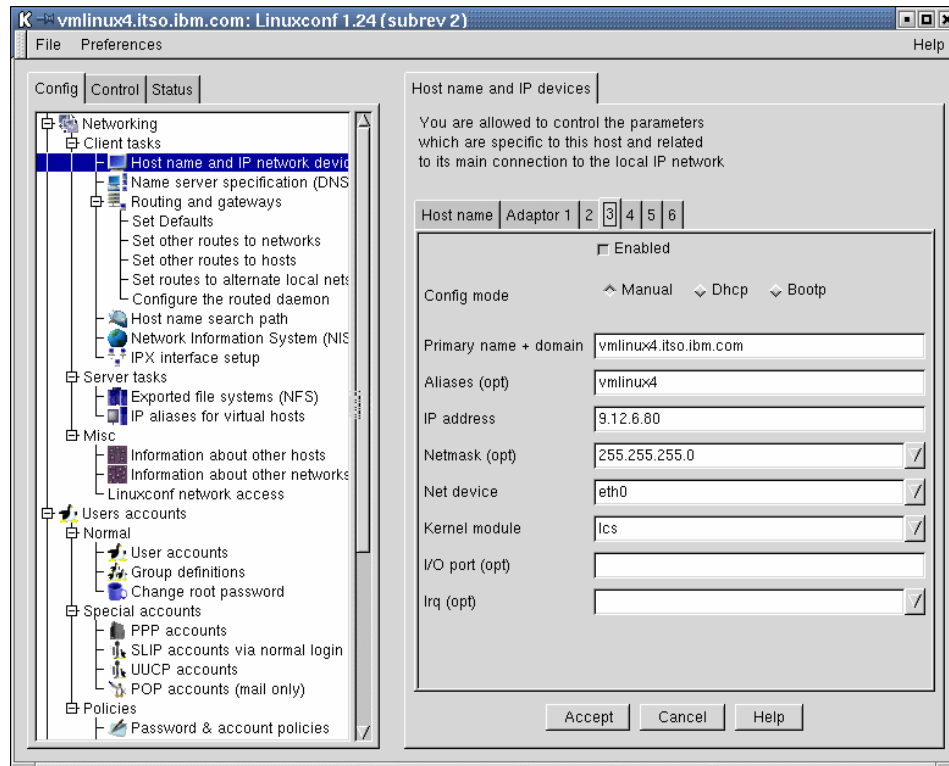


Figure 12-4 A `gnome-linuxconf` session

## 12.3 Adding a non-root user

`linuxconf` can be used to do add a non-root user, but it can also be done from the command line. Issue the command `useradd -g users username` to add a user that has only normal user privileges.

## 12.4 Managing the network

In general, networking is managed via the script `/etc/rc.d/init.d/network`. A useful command to check the status of the network is as follows:

```
]# /etc/rc.d/init.d/network status
Configured devices:
lo ctc0 ctc1 eth0 iucv0 iucv1
Devices that are down:
ctc0 ctc1
Devices with modified configuration:
iucv0 iucv1
```

Secondary DNS servers and additional search domains can be added to the file `/etc/resolv.conf`. Add systems that don't have DNS records to the file `/etc/hosts`.

## 12.5 Configuring Apache

By default, Apache was installed on our system. Edit the file `/etc/httpd/conf/httpd.conf` to have the values you want for your installation.

Issue the **chkconfig** command to set the run-levels in which you want Apache to be active:

```
$ chkconfig --level 2345 httpd on
```

Issue the command `/etc/init.d/httpd start` to start it the first time. Optionally, install the tool **apacheconf** to configure Apache. Install the `apache-manual` RPM if you want the Web-accessible manual documentation.

The configuration files for Apache (and php) are in `/etc/httpd/conf/`, especially in the file `httpd.conf`.

The log files for Apache (and php) are in `/var/log/httpd`, especially in the files `access_log` and `error_log`.

The default location for Apache's "Document Root" is `/var/www/html`. This is the directory from which your documents will be served. Symbolic links and aliases may be used to point to other locations.

## 12.6 Managing the FTP Server

In order to enable the FTP server, edit the file `/etc/xinetd.d/wu-ftp` and change `disable=yes` to `disable=no` and then issue command **service xinetd restart** to pick up the change.

Since the anonymous FTP package is part of the default installation, this will also enable anonymous FTP access. If you want to disable anonymous FTP, simply uninstall the anonftp RPM as follows:

```
$ rpm -e anonftp-4.0-4
```

Unfortunately, this is likely to produce some amount of confusion. Users who attempt to connect anonymously will see these results:

```
$ ftp vmlinux4
Connected to vmlinux4.itso.ibm.com.
220 vmlinux4.itso.ibm.com FTP server (Version wu-2.6.1-16) ready.
User (vmlinux4.itso.ibm.com:(none)): anonymous
331 Guest login ok, send your complete e-mail address as password.
Password: userid@mydomain.com
530 Can't set guest privileges.
Login failed.
```

We were unable to determine any way to eliminate the Guest login ok message completely. This is most likely because wu-ftp was designed with anonymous access as the preferred method of use.

The configuration files for wu-ftp are all in the /etc directory:

- ▶ ftpaccess
- ▶ ftpconversions
- ▶ ftpgroups
- ▶ ftphosts
- ▶ ftpusers

The default location for files accessible by anonymous FTP is /var/ftp.

Note that wu-ftp writes its logs to the files /var/log/ftpaccess and /var/log/messages.

## 12.7 Managing Samba

By default, Samba was installed on our system. To customize it, edit the Samba configuration file, /etc/samba/smb.conf, to set the values you want for workgroup, security, password encryption, etc. Issue the **chkconfig** command to set the run-levels in which you want Samba to be active. For example:

```
# chkconfig --level 2345 smb on
```

Issue the command **/etc/init.d/smb start** to start Samba services the first time.

If the security setting in the configuration file is not set to `security = server` or `security = domain`, the value will default to `security = user`. When this is the case, you must create user accounts on the Linux system. If you also want to handle DES-encrypted passwords, you must also create Samba accounts as follows:

```
# useradd -g users username
# smbpasswd -a username
```

The very first time you issue the **smbpasswd** command on a Linux system, you will get an error message about the `smbpasswd` file not being found:

```
start smbfilepwent_internal: unable to open file /etc/samba/smbpasswd. Error
was No such file or directory
Added user username.
```

This is really just a warning message, since the file is then created, and the first user added. You can avoid this error message by first invoking the command:

```
touch /etc/passwd
```

The configuration files for Samba are in `/etc/samba`. Samba writes its log files to the directory `/var/log/samba`.

### 12.7.1 Enabling SWAT

The Samba Web Administration Tool (SWAT) is not installed by default. We install the package `samba-swat-2.0.7-36.s390.rpm`.

To enable SWAT, edit the file `/etc/xinetd.d/swat` and change `disable = yes` to `disable = no`, or comment out the line. If you want to be able to access SWAT from your desktop, you will also have to comment out the line that reads `only_from = 127.0.0.1`.

To tell `inetd` to invoke this change, enter the command **service xinetd restart**. SWAT should now be accessible via a browser with the URL:

```
http://<your.server.name>:901
```

## 12.8 Mail server

Red Hat Linux uses `sendmail` as the mail transfer agent (MTA). It can either be configured with `linuxconf`, or you can edit the configuration files directly. The following example shows you how to configure `sendmail` with an external mail host in order to be able to send and receive e-mails between your Linux system and the outside world.



**Note:** DNS must be set up with MX records to have mail for your domain delivered to your Linux system.

Edit the file `/etc/sendmail.cf`. Lines 92 and 93 (approximately) should contain a comment and a configuration parameter:

```
# "Smart" relay host (may be null)
DS
```

Put in the DNS name of one of your company's mail forwarding systems. If we use `smtp.mycompany.com` as an example, the file would be changed to this:

```
# "Smart" relay host (may be null)
DSsmtp.mycompany.com
```

Send a message to `sendmail` to tell it to re-read its configuration file:

```
Determine the process id (PID) by doing ps ax | grep sendmail.
Kill -1 ### where “###” is the PID from the previous command.
```

At this point, `sendmail` is configured and you should be able to send to, and receive mail from, other hosts.

## 12.9 Enabling xdm

In order to enable the X Display Manager (`xdm`), edit the file `/etc/X11/xdm/Xaccess`. Uncomment the line that has a single asterisk and the trailing comment “#any host can get a login window.”

Also, edit the file `/etc/X11/xdm/xdm-config`. Comment out the last line of the file by putting an exclamation point in column one. The line should then read:

```
! DisplayManager.requestPort: 0
```

Start `xdm` via the command: `/usr/X11R6/bin/xdm`

The configuration files for `xdm` are in the directory `/etc/X11/xdm`. The log files for `xdm` are written to the file `/var/log/xdm-errors`.

## 12.10 Installing KDE

Although we do not recommend that you run a graphical desktop environment on Linux, some people will still want to.

To do so, you must install the following RPMs in the order given:

1. kdesupport-2.1-3a.s390.rpm
2. kdelibs-2.1.1-5.s390.rpm
3. kdelibs-sound-2.1.1-5.s390.rpm
4. kdebase-2.1.1-8.s390.rpm

This will allow you to login and have the KDE desktop available. If you want to go further, you can install the following RPMs for more functionality. They can be installed in just about any order.

- ▶ kdeadmin-2.1.1-3.s390.rpm
- ▶ kdegames-2.1.1-1.s390.rpm
- ▶ kdegraphics-2.1.1-1.s390.rpm
- ▶ kdenetwork-2.1.1-1.s390.rpm
- ▶ kdepim-2.1.1-1.s390.rpm
- ▶ kdetoys-2.1.1-2.s390.rpm
- ▶ kdeutils-2.1.1-1.s390.rpm
- ▶ kdevelop-1.4.1-2.s390.rpm
- ▶ switchdesk-kde-3.9.5-1.s390.rpm

## 12.11 Installing GNOME

Although we do not recommend that you run a graphical desktop environment on Linux, some people will still want to. To do so, you must install the following RPMs in the order given:

1. librepp-0.13.3-1.s390.rpm
2. rep-gtk-0.15-3.s390.rpm
3. rep-gtk-gnome-0.15-3.s390.rpm
4. aumix-2.7-2.s390.rpm
5. control-center-1.2.2-8.s390.rpm
6. sawfish-0.36-7.s390.rpm
7. gnome-core-1.2.4-16.s390.rpm

This will allow you to login and have the GNOME desktop available. If you want to go further, you can install the following RPMs for more functionality. They can be installed in just about any order.

- ▶ gnome-applets-1.2.4-3.s390.rpm
- ▶ gnome-games-1.2.0-10.s390.rpm
- ▶ gnome-kerberos-0.2.2-2.s390.rpm
- ▶ gnome-linuxconf-0.64-1.s390.rpm
- ▶ gnome-lokkit-0.43-7.s390.rpm
- ▶ gnome-objc-1.0.2-11.s390.rpm
- ▶ gnome-pim-1.2.0-9.s390.rpm

- ▶ libgnomeprint11-0.25-9.s390.rpm and gnome-print-0.25-9.s390.rpm
- ▶ gnome-users-guide-1.2-3.noarch.rpm
- ▶ openssh-askpass-gnome-2.5.2p2-5.s390.rpm
- ▶ switchdesk-gnome-3.9.5-1.s390.rpm
- ▶ switchdesk-gnome-3.9.5-1.s390.rpm
- ▶ up2date-gnome-2.5.2-1.s390.rpm

## 12.12 Miscellaneous

Add `alias char-major-4 off` to the file `/etc/modules.conf` to turn off the serial port.

## 12.13 Installing maintenance on Red Hat

Red Hat, like nearly all Linux distributors, posts maintenance on their FTP sites for packages that require it. You will need to download the RPMs and install them.

**Important:** You may need to “socksify” your TCP/IP stack to get through your company’s firewall. We do not address this topic, but the `tsocks` package is a possible solution.

Red Hat also provides a utility called **up2date**. The man page for it says:

```
Update Agent provides a complete system for updating the RPM packages
installed on a Red Hat Linux system. Both command line and graphical
interfaces are included.
```

```
When you run Update Agent, you will be prompted for the computer's root
password. This is because Update Agent needs read/write access to the
RPM database, and it needs to be able to install packages when so
requested. It also needs read/write access to the up2date systemid file
in /etc/sysconfig/rhn/systemid , and it's configuration in
/etc/sysconfig/rhn/up2date
```

Before you can run **up2date**, you will need to register with the Red Hat Network by running the `rhn_register` command.





## Part 3

# Other distributions

In this part we describe other Linux distributions for zSeries and S/390. They are:

- ▶ The Marist “distribution” or file system
- ▶ The Caiman distribution by LinuxKorea Inc.
- ▶ The Think Blue 64-bit distribution by Millenux Inc.





## The Marist File System

This chapter provides a history and overview of the Marist College Linux for S/390 file system. It is not called a distribution because it was not distributed on a CD with many RPM packages.

This chapter also describes the installation and customization of the updated (May 1, 2001) Marist College Linux for S/390 file system under VM.

## 13.1 History and overview of the Marist File System

The Marist File System was the first Linux system that was available for download. Created by the IBM Linux for S/390 developers, it was made available on the Marist College Linux FTP server on January 5, 2000. There were two versions, a small file system and a big file system. Both versions would allow you to bring up a basic Linux system, but the “big” file system had enough packages to compile and install new software.

This was not a “distribution” in the usual sense of the word, in that it was simply a tarred and gzipped copy of a running Linux system. Installing it consisted of un-gzipping and un-tarring the file onto the DASD where it was to run, running silo to make it bootable, and configuring the network parameters. Other than “big” or “small,” there were no choices available as to what software would be installed.

Second and third versions, only slightly different from the original, were made available in February and May of 2000.

A fourth version, referred to as the “updated” file system, was made available on May 1, 2001. The initial test version of this release is what we installed. It is based on a Linux 2.2.16 kernel that was compiled on Thu Apr 19 11:10:14 EDT 2001. Unlike the previous versions, the May 1, 2001 update only comes in one “size.” Although fairly large in absolute terms, it is still much smaller than typical “default” installations of other distributions.

The list of software RPMs that are installed on the updated Marist file system are listed in an appendix entitled “Linux Software Packages”. Due to the size of this document, it is not included in this redbook, but can be found separately on the Web at:

<ftp://www.redbooks.ibm.com/redbooks/SG246264/RPMappendix.pdf>

Note that although XFree86 is installed, none of the more popular desktop environments (such as KDE and Gnome) are installed. Also note that this list will not reflect changes that may be made to the file system after publication.

### 13.1.1 Getting the Marist File System

The Marist File System consists of five files, and it can only be downloaded from:

<ftp://linux390.marist.edu/pub/update>

The files are:

- ▶ A README file



- ▶ A Linux kernel to be used when booting from a tape drive (image.tape)
- ▶ A Linux kernel to be used when booting from a VM virtual reader (image.vm)
- ▶ An initial RAMdisk to support the installation of the file system (initrd.gz)
- ▶ A tarred and gzipped copy of the file system (marist\_fs.tgz)

### 13.1.2 Documentation

*LINUX ® for S/390 ® Installation, Configuration and Use*, by IBM, was written with the original Marist File System as the model for installation, and is available at:

<http://linux390.marist.edu/download/inst.pdf>

*LINUX ® for S/390 ® LCS Device Driver*, by IBM, is available at:

<http://linux390.marist.edu/download/lcsdd.pdf>

### 13.1.3 How the Marist File System differs

The most important difference between the Marist File System and other Linux distributions is that it is a tarred and gzipped copy of a running file system, and not a collection of software packages to be installed. As such, it is not a distribution in the usual sense of the word, since there are no options for selecting what software to install. Whatever the file system maintainer decided to install is what you will wind up with on your system.

This provides the following advantages:

- ▶ It is very easy to get running on your hardware.
- ▶ It does not require much knowledge of Linux, or Linux for S/390.

The disadvantages are:

- ▶ The number of installed packages is limited and fixed, and finding binary RPMs that will install properly may be difficult or impossible.
- ▶ Upgrading to a complete new file system will involve saving copies of all configuration files—wherever they are located, and whatever they are named—and then restoring them afterwards.

The system was built using RPM, and therefore has a full RPM database. Because finding binary RPMs may be difficult, if there is anything you want to upgrade before the next update to the entire file system, you should probably be prepared to install source RPMs and build your own binaries.

Because it is intended as a minimalist, easy-to-get-started system, there are no system administration tools installed.

## 13.2 Installing the Marist File System

No matter which method you use to install, you will need to have the tarred and gzipped file system (marist\_fs.tgz) available to your system via one of three methods:

- ▶ NFS server
- ▶ On DASD accessible to the system being installed
- ▶ FTP server

**Note:** *Installation* from an FTP server is not supported by the Marist File System. If all you have is an FTP server, you must IPL the initial kernel and RAMdisk, and then FTP the marist\_fs.tgz file to the target system to use it.

If you do not have an NFS server available, you will need to have enough *additional* DASD space on your system to hold the marist\_fs.tgz file, as well as the three “starter” files. This additional space will be a minimum of 135 MB. Having even more additional space would be preferable.

The amount of disk space needed for the installation (*not including* the marist\_fs.tgz file) is listed in Table 13-1.

Table 13-1 Marist disk requirements

Directory	Size
/bin	5.8 MB
/boot	1.6 MB
/dev	80 KB
/etc	1.2 MB
/home	2.0 MB
/lib	4 MB
/opt	0
/root	0
/sbin	3.6 MB
/usr	445 MB
/var	4.1 MB
Total	468 MB

As can be seen from the installation worksheet that follows, a number of IP networking parameters are required. Your IP network group/administrator or system programmer will have to provide these.

When specifying the device numbers for a Linux guest running under VM, use the virtual device addresses that are defined for the guest in the user directory; see 3.1.1, “The VM user directory” on page 28.

Table 13-2 Installation worksheet

1. VM guest user ID (if installing under VM)	
2. VM guest password (if installing under VM)	
3. Device: 1st DASD/vdev(VM) (/ file system)	
4. Device: 2nd DASD/vdev(VM) (swap space)	
5. Device: tape unit *optional (eg. 0181)	
6. Network device type (eg. CTC, OSA-E)	
7. Network device 1st of pair (eg. 292C)	
8. Network device port 1st of pair (eg. 0)	
9. Host name (eg. vmlinux.itso.ibm.com)	
10. Host IP address (eg. 9.12.6.73)	
11. Netmask (eg. 255.255.255.0)	
12. IP address of CTC/ESCON "Peer"	
13. Broadcast IP address (eg. 9.12.6.255)	
14. Gateway IP address (eg. 9.12.6.75)	
15. Network IP address (eg. 9.12.6.0)	
16. DNS IP address (eg. 9.12.14.7)	
17. DNS search domain (eg., itso.ibm.com,ibm.com)	
18. MTU size (for Marist the default is taken)	
19. FTP server IP address or hostname	
20. FTP path (eg. /mnt/cdrom)	
21. FTP user ID (eg. TOT82)	
22. FTP password	
23. Linux root user password	

## 13.3 Installing the Marist File System under VM

In this section, we cover the steps needed to install the Marist File System under VM.

### 13.3.1 Marist File System under VM checklist

1. Make sure your VM guest has enough disk capacity on its 191 disk to hold a minimum of 6 MB of data. More would be preferable, perhaps 10 to 20MB.
2. Logon to the VM guest you will be using as your Linux system.
3. Since it is likely that you will be IPLing your Linux guest from the VM reader more than once, it is advisable to create a REXX exec as follows:

```
/* */
'cl rdr'
'purge rdr all'
'spool punch * rdr'
'PUNCH image vm a (NOH'
'PUNCH parmfile vm a (NOH'
'PUNCH initrd gz a (NOH'
'ch rdr all keep nohold'
'ipl 00c clear'
```

4. Make sure the following files are available to your VM/CMS guest and stored on the 191 disk, or another disk that is accessible:
  - IMAGE VM - the kernel to be used for IPLing under VM
  - INITRD GZ - the RAMdisk containing the system to install Linux

This can be done in several ways, but using the VM/CMS FTP client is probably the easiest.

5. Create the PARMFILE VM file to match your installation. Since DASD support is compiled into the kernel, you *must* specify which DASD addresses you will need to access. The file that we used for our installation was:

```
ramdisk_size=32768 root=/dev/ram0 ro dasd=200-20f
```

6. Run the starter EXEC to punch the three files to your virtual reader and IPL. You should see output similar to this:

```
Linux version 2.2.16 (root@lxbuid1) (gcc version 2.95.2 19991024
(release)) #3 SMP Thu Apr 19 11:10:14 EDT 2001
Command line is: ramdisk_size=32768 root=/dev/ram0 ro dasd=200-20f
We are running under VM
This machine has an IEEE fpu
Initial ramdisk at: 0x02000000 (8388640 bytes)
Detected device 2926 on subchannel 0000 - PIM = 80, PAM = 80, POM = FF
Detected device 2927 on subchannel 0001 - PIM = 80, PAM = 80, POM = FF
...
```

7. After the IPL is complete, you will be asked a series of questions about your network setup, as shown in Example 13-1 on page 201. Answer them according to the information given to you by your IP network person or group. It's a good idea to have these values written down on the worksheet, as you

will be asked for them again the first time you IPL your Linux system from DASD.

a. Whether your Linux guest is attached to a network.

It is not required that your system be connected, but working with it will be rather painful and relatively pointless. If you answer “no,” the script will skip all the network configuration, and put out a prompt for the root password, which is: pass4root.

b. Which communications device you are going to use.

For example: OSA token ring; OSA ethernet; channel-to-channel; ESCON channel. The Marist file system does not offer IUCV as an option.

c. Depending on what kind of communications device you said you had, you will be asked further questions about it.

- If you said you were using OSA token ring, or OSA Ethernet, you will be prompted:

```
Please type in the options for the lcs module, e.g. to select
relative port 1 on device 0xfd00 you should enter:
noauto=1 devno_portno_pairs=0xfd00,1
lcs parameter:
```

In our test case, the value we entered was:

```
noauto=1 devno_portno_pairs=0x2926,0
```

- For a CTC and ESCON channel, it will ask for the “the IP address of your peer:” This means the IP address of the system on the other end of the virtual CTC.

d. Your host name, for example: vmlinux4.itso.ibm.com

e. The IP address of your Linux system, for example: 9.12.6.80

f. The network mask for this system, for example: 255.255.255.192.

(Note that this is *not* the value we used for our testing. It is different to show that other values are possible, and provide an example for the subnet discussion that follows.)

g. The broadcast address of this system.

This will be dependent on your network mask. It will be the highest address in the subnet. If your network mask is 255.255.255.192, as given in the previous example, your broadcast addresses will be xx.xx.xx.63, xx.xx.xx.127, xx.xx.xx.191, and xx.xx.xx.255.

For a different subnet mask, these values will be different also. For a subnet mask of 255.255.255.0, there is only one broadcast address, at xx.xx.xx.255.

- h. The IP address of your default gateway.

This is the address to which your system will send all its IP traffic if it doesn't know of a specific route to the destination. Some people use the lowest address in the subnet for their gateways. Others use the next-to-highest address. Still others pick something else. There is no way to calculate or guess at this address. You will *have* to ask your IP networking person for the value.

- i. The network address of this system.

This is the result of a logical AND of your IP address, and subnet mask. For a subnet mask of 255.255.255.0, this simply means that the value of the last octet will always be 0. For any other subnet mask, it's best to break out the hexadecimal calculator.

- j. The IP address of your DNS server.

You will need to get this information from your IP network person or group.

- k. The DNS search domain.

Again, you should get this information from your IP network person. Simply put, though, it is the value that gets appended to a host name if an initial name search fails.

Using `vmlinux4` as an example, and `itso.ibm.com` as the DNS search domain, if you were to type in the command `ping vmlinux4`, then the name resolution would fail. The system would then re-issue the request as `vmlinux4.itso.ibm.com`. In this case, the name resolution would succeed, since that is a name that the ITSO DNS server knows.

The DNS search domain can be *multiple* values, separated by blanks, with no commas. Therefore, `ibm.com itso.ibm.com` would cause a failed name resolution to be tried twice: once with `ibm.com` appended to the host name, and once with `itso.ibm.com`.

- 8. You will be shown the values you entered, and asked to confirm them. If you reply "no", you will be thrown all the way back to being asked if your system is connected to a network. The script does *not* remember any of the values you entered before, so you will have to enter them all over again.

If you reply "yes," the network will be started, and you will be prompted to enter the root password, which is: `pass4root`.

*Example 13-1 Providing network parameters to the installation script*

---

```
Enabling swap space [ OK ]
/etc/rc.d/rc.sysinit: /boot/kernel.h: No such file or directory
INIT: Entering runlevel: 3
Entering non-interactive startup
```

Welcome to Linux for S/390

```

Is your machine connected to a network (Yes/No) ? yes
yes
Select the type of your network device
1) for osa token ring
2) for osa ethernet
3) for channel to channel
4) for escon channel
Enter your choice (1-4): 2
2
Please type in the options for the lcs module, e.g. to select
relative port 1 on device 0xfd00 you should enter:
noauto=1 devno_portno_pairs=0xfd00,1
lcs parameter: noauto=1 devno_portno_pairs=0x2926,0
noauto=1 devno_portno_pairs=0x2926,0
Please enter your host name: vmlinux4.itso.ibm.com
vmlinux4.itso.ibm.com
Please enter your IP address: 9.12.6.80
9.12.6.80
Please enter the net mask: 255.255.255.0
255.255.255.0
Please enter the broadcast address: 9.12.6.255
9.12.6.255
Please enter the gateway address: 9.12.6.75
9.12.6.75
Please enter the net address: 9.12.6.0
9.12.6.0
Please enter the IP address of the DNS server: 9.12.14.7
9.12.14.7
Please enter the DNS search domain: itso.ibm.com
itso.ibm.com

Configuration will be:
LCS parameter      : noauto=1 devno_portno_pairs=0x2926,0
Host name          : vmlinux4.itso.ibm.com
IP address         : 9.12.6.80
Net mask           : 255.255.255.0
Broadcast address  : 9.12.6.255
Gateway address    : 9.12.6.75
Net address        : 9.12.6.0
DNS IP address     : 9.12.14.7
DNS search domain  : itso.ibm.com
Is this correct (Yes/No) ? yes
yes
Bringing up interface lo [ OK ]
Bringing up interface eth0 [ OK ]
Starting portmapper: [ OK ]
Initializing random number generator [ OK ]
Starting INET services: [ OK ]
Give root password for maintenance

```



(or type Control-D for normal startup):

---

9. Once the network setup is complete, you can either logon to the console, or telnet in (recommended) to your Linux system and finish the installation. You will be required to specify a password to logon to the system.
10. Start formatting each of the DASD volumes you will be using for your system. You need to be careful here, since **dasdfmt** and **mke2fs** have different defaults for the DASD blocksize, and having them not match will cause problems. You can format the volumes using either their device number, or their Linux device name:

```
# dasdfmt -b 4096 -n 200
# dasdfmt -b 4096 -f /dev/dasda
# dasdfmt -b 4096 -n 204
# dasdfmt -b 4096 -f /dev/dasde
```

11. You will be prompted to verify the command; make sure you are formatting the proper volume with the proper blocksize, and then reply yes. You should get this response:

```
Formatting the device. This may take a while (get yourself a coffee).
```

12. Create an ext2 file system on each DASD volume you formatted, excluding any swap volumes. For example:

```
# mke2fs -b 4096 /dev/dasdb1
```

13. Format and activate your swap volumes. For example:

```
# mkswap /dev/dasdd1
Setting up swap space version 0, size = 73711616 bytes
# chmod 600 /dev/dasdd1
# swapon /dev/dasdd1
```

14. Decide which DASD volumes you want mounted on what mount points. In our test, we used /dev/dasdb as our root file system, and /dev/dasdc as our /usr file system.

15. Mount the DASD volumes on their correct mount points. This will require a **mkdir** command be issued for each volume other than the root file system.

```
# mount /dev/dasdb1 /mnt
# mkdir /mnt/usr
# mount /dev/dasdc1 /mnt/usr
# df
```

Filesystem	1k-blocks	Used	Available	Use%	Mounted on
/dev/ram0	7931	7774	157	98%	/
/dev/dasdb1	1842540	24	1748920	0%	/mnt
/dev/dasdc1	1842540	20	1748924	0%	/mnt/usr

16. Change to the /mnt directory:

```
cd /mnt
```

17. Access the `marist_fs.tgz` file. If you do this by FTP, then there must be enough room on your root file system for the `marist_fs.tgz`, and all the files that will be installed there. We recommend that you use NFS instead, and access `marist_fs.tgz` over the network. If you do use NFS, you must create a temporary mount point for the NFS mount, as shown here:

```
# mkdir nfs
# mount 9.12.9.183:/home/redbooks/linuxSystems/marist /mnt/nfs
# df
Filesystem                1k-blocks      Used Available Use% Mounted on
/dev/ram0                   7931          7774         157   98% /
/dev/dasdb1                 1842540         28    1748916    0% /mnt
/dev/dasdc1                 1842540         20    1748924    0% /mnt/usr
9.12.9.183:/home/redbooks/linuxSystems/marist
                          7098092    4796928    1940592    71% /mnt/nfs
```

18. Untar the file system onto your DASD.

Note the use of the “-p” parameter to preserve file permissions

```
# tar -zxvpf /mnt/nfs/marist_fs.tgz
./
lost+found/
.bash_history
bin/
bin/mktemp
bin/bash
...
```

19. During our testing, the following error message was seen:

```
tar: usr/bin/cpp: Could not link to `lib/cpp': No such file or directory
```

Which resulted in this error message at the very end:

```
tar: Error exit delayed from previous errors
```

This error did not cause any problems, but should be fixed in a future update to the file system.

20. If you are using an OSA Ethernet or OSA token ring communications device, you must copy the `lcs.o` module to your DASD:

```
# mkdir /mnt/lib/modules/2.2.16/net
# cp -p /lib/modules/2.2.16/net/lcs.o /mnt/lib/modules/2.2.16/net/
```

21. We found there were some extraneous directories left on the file system which should be deleted:

```
# rm -rf /mnt/lib/modules/2.2.15
```

22. Use the `chroot` command to set the `/mnt` directory to the root file system. Update `/etc/fstab` with the appropriate entries. This is what our test system used:

```
# chroot /mnt
# vi /etc/fstab
```

```

...
# cat /etc/fstab
/dev/dasdb1      /          ext2  defaults,errors=remount-ro 0 1
/dev/dasdc1     /usr       ext2  defaults      1    2
/dev/dasdd1     swap       swap  defaults      0    0
none            /proc      proc  defaults      0    0

```

23. Change to the /boot directory, update the parmline file, and run silo.

```

# cd /boot
# ls
System.map image ipldump.boot ipleckd.boot iplfba.boot kernel.h
parmline
# vi parmline
...
cat parmline
dasd=200-20f root=/dev/dasdb1 ro noinitrd
# silo -d /dev/dasdb -f image -p parmline -b ipleckd.boot -t2
o->ipldevice set to /dev/dasdb
o->image set to image
o->parmfile set to parmline
o->bootsect set to ipleckd.boot
Testonly flag is now 0
Testlevel is set to 0
IPL device is: '/dev/dasdb'
bootsector is: 'ipleckd.boot'...ok...
bootmap is set to: './boot.map'...ok...
Kernel image is: 'image'...ok...
original parameterfile is: 'parmline'...ok...final parameterfile is:
'parmline'...ok...
ix 0: offset: 0401f0 count: 0c address: 0x00000000
ix 1: offset: 0401fd count: 80 address: 0x0000c000
ix 2: offset: 04027d count: 80 address: 0x0008c000
ix 3: offset: 0402fd count: 5f address: 0x0010c000
ix 4: offset: 040392 count: 01 address: 0x00008000
Bootmap is in block no: 0x00040393

```

24. There was a slight glitch with the test Marist File System: a configuration file was left in /etc/sysconfig/network-scripts. It should be removed before IPLing from DASD.

```

cd /etc/sysconfig/network-scripts
rm ifcfg-ctc0

```

25. There was a file misnamed in the /etc directory. Correct it as follows:

```

cd /etc
mv conf.modules modules.conf
ln -s modules.conf conf.modules (optional)

```

26. The permissions on the /tmp directory need to be changed:

```
chmod 1777 /tmp
The permissions on /dev/null need to be changed:
chmod 666 /dev/null
```

27. The setuid bit on the xterm binary needs to be set:

```
chmod 4711 /usr/X11R6/bin/xterm
```

28. Get out of the chroot environment with the exit command, change to the root directory, and unmount the DASD volumes, and the NFS mount:

```
# exit
# cd /
# umount /mnt/nfs
# rmdir /mnt/nfs
# umount /mnt/usr
# umount /mnt
```

29. You can now reboot from DASD. Go back to the console, and boot from the device you specified in the silo command:

```
#CP IPL 201 CLEAR
```

You should see a result similar to this:

```
Linux version 2.2.16 (root@1xbuid1) (gcc version 2.95.2 19991024
(release)) #3 SMP Thu Apr 19 11:10:14 EDT 2001
Command line is: dasd=200-20f root=/dev/dasdb1 ro noinitrd
```

```
We are running under VM
This machine has an IEEE fpu
Initial ramdisk at: 0x02000000 (8388608 bytes)
Detected device 2926 on subchannel 0000 - PIM = 80, PAM = 80, POM = FF
Detected device 2927 on subchannel 0001 - PIM = 80, PAM = 80, POM = FF
...
```

30. Repeat the answering of the networking questions.

31. Logon as root.

32. Change the password for root with the **passwd** command.

## 13.4 Customization

In this section we discuss various customizations you can make to your Linux system.

### 13.4.1 Installing SSH

Get an SSH RPM package, install it and use it!

This will actually require three packages: bc, SSL, and SSH. We recommend having a current version of each to install. These can be found a number of places. A commonly used site is Thinking Objects; it has source RPMs that are intended to go on top of a Red Hat-based Linux system. It is on the Web at:

<http://linux.s390.org/download/>

We still had to modify the spec file to match our environment better:

<http://linux.zSeries.org/download/ftp/SRPMS/bc-1.05a-13.src.rpm>

<http://linux.zSeries.org/download/ftp/SRPMS/openssl-0.9.5a-23.src.rpm>

<http://linux.zSeries.org/download/ftp/SRPMS/openssh-2.5.2p2-1.7.2.src.rpm>

Download the source packages; you can install them with the following commands:

```
# rpm -i bc-1.05a-13.src.rpm
# rpm -i openssl-0.9.5a-23.src.rpm
# rpm -i openssh-2.5.2p2-1.7.2.src.rpm
# cd /usr/src/redhat/SPECS
# rpm -ba bc.spec
# rpm -i ../RPMS/s390/bc-1.05a-13.s390.rpm
# rpm -ba openssl.spec
# rpm -i ../RPMS/openssl-0.9.5a-23.s390.rpm
# rpm -i ../RPMS/openssl-devel-0.9.5a-23.s390.rpm
# vi openssh.spec
Change "%define no_gnome_askpass 1" to "%define no_gnome_askpass 0"
Change "PreReq: initscripts >= 5.20" to "PreReq: SysVinit"
Save and exit
# rpm -ba openssh.spec
# rpm -i ../RPMS/s390/openssh-2.5.2p2-1.7.2.s390.rpm
# rpm -i ../RPMS/s390/openssh-clients-2.5.2p2-1.7.2.s390.rpm
# rpm -i ../RPMS/s390/openssh-askpass-2.5.2p2-1.7.2.s390.rpm
# rpm -i ../RPMS/s390/openssh-server-2.5.2p2-1.7.2.s390.rpm
```

## 13.4.2 Creating multiple IPL volumes

If you want to create another IPLable volume, the silo program requires that all the files it needs be on the same physical DASD volume. The easiest way to accomplish this is to use `mkdir` to create a directory somewhere on that volume.

What you name the directory, and where in the directory tree it resides does not matter—all that matters is that it is on the physical volume that you want to make IPLable.

As an example, consider this structure, and assume that `/dev/dasdb` is our IPL device:

```
# df
Filesystem          1k-blocks      Used Available Use% Mounted on
/dev/dasdb1         1842468        79944  1668928   5% /
/dev/dasdc1         1842468        741972  1006900  42% /usr
```

If we now want to make `/dev/dasdc` an alternate IPL device, we first need to make a directory somewhere on that volume to hold a copy of the files in `/boot`. Since `/dev/dasdc1` is mounted on `/usr`, that directory has to be somewhere under `/usr`. In our case, we kept things simple:

```
# mkdir /usr/boot
# cp -a /boot/* /usr/boot
# cd /usr/boot
# silo -d /dev/dasdc -f image -p parmfile -b ipleckd.boot -B boot.map -t2
```

We were then able to IPL from `/dev/dasdc`, as well as `/dev/dasdb`.

### 13.4.3 Adding a non-root user

Issue the command `useradd -m username` to add a user that has only normal user privileges. Set the password for the user with the `passwd username` command.

Update `/etc/suauth` if multiple people need root access.

### 13.4.4 Using Apache

Apache is started by default, with default parameters.

Edit `/etc/httpd/conf/httpd.conf` to have the values you want for your installation. Restart the daemon with the following command:

```
# /etc/rc.d/init.d/httpd restart
```

Issue the `chkconfig` command to see what run levels will have Apache active:

```
# chkconfig --list
```

If you want to adjust the run levels, reissue the `chkconfig` command:

```
# chkconfig --level 2345 httpd on
```

## 13.4.5 FTP server

The FTP server is enabled by default. The home directory for FTP does not exist. If you want to enable anonymous FTP access, you will have to create it as follows:

```
# mkdir /home/ftp
```

## 13.4.6 xdm

Because the Xaccess and Xservers files are shipped as read-only files, you will need to do a **chmod** command to make them writeable by root *before* you edit them. It's up to you whether you change the permissions back to 444 afterwards.

Create a file named `/etc/pam.d/xdm`. Put these entries in it:

```
##PAM-1.0
auth    required /lib/security/pam_unix.so      nullok #set_secrcp
account required /lib/security/pam_unix.so
password required /lib/security/pam_unix.so    #strict=false
session required /lib/security/pam_unix.so     debug # trace or none
```

Edit `/usr/X11R6/lib/X11/xdm/Xaccess`. Uncomment the line that has a single asterisk and this trailing comment:

```
#any host can get a login window.
```

Edit `/usr/X11R6/lib/X11/xdm/Xservers`. Comment out the last line that contains the following:

```
:0 local /usr/X11R6/bin/X.
```

Create a symbolic link from where xdm thinks its configuration files exist, to where they really do exist:

```
# cd /etc/X11
# ln -s /usr/X11R6/lib/X11/xdm/ xdm
```

Start xdm with the command `/usr/X11R6/bin/xdm`.

To start xdm with every IPL, edit file `/etc/rc.d/rc.local` and add the `/usr/X11R6/bin/xdm` command to the very end (after the last "fi" statement).

## 13.4.7 Network

Add secondary DNS servers to: `/etc/resolv.conf`

Add additional search domains to: `/etc/resolv.conf`

Add systems to `/etc/hosts` for systems that don't have DNS records.

## 13.4.8 Misc

Add alias `char-major-4 off` to the `/etc/modules.conf` - serial port.

Edit `/etc/profile`, and change the following:

```
if [ "$TERM" = "linux" ]; then
```

to this:

```
if [ "$CONSOLE" = "/dev/console" ]; then
```

## 13.5 Problems we encountered

All of the following problems have been reported to the Marist File System maintainer. We believe they will be corrected before the final release of the updated version.

- ▶ The leftover `ifcfg-ctc0` file in the `/etc/sysconfig/network-scripts` directory caused long delays while the I/O to the non-existent CTC device timed out. Once we deleted the file, startup experienced with no delays.
- ▶ The `/etc/conf.modules` was named `/etc/modules.conf`, which caused some confusion and apparent problems. Merging the contents of the two files and renaming the result to `modules.conf` fixed it. Just as a precaution (in case other scripts somewhere on the system were looking to read or write the file), we created a symbolic link to it with the old name.

- ▶ At boot time, we noticed the message:

```
modprobe: can't locate module char-major-4
```

Since `char-major-4` is for serial port support, it should be aliased 'off' in `/etc/modules.conf`, which we did.

- ▶ When we examined the error logs for Apache, we saw that there was a missing image file, `poweredby.png`, for the graphic entitled Powered by Red Hat Linux. We got a copy of it from another location to eliminate the error.
- ▶ During the boot process, we saw these error messages:

```
/etc/rc.d/rc.sysinit: /proc/sys/kernel/sysrq: No such file or directory  
/etc/rc.d/rc.sysinit: /etc/sysconfig/keyboard: No such file or directory
```

While these won't cause any problems, we did report them to the file system maintainer.

- ▶ The `xfs` task was starting at boot time, then immediately terminating. We tracked down the problem to the directory permissions for `/tmp` not being `1777`. After we did a `chmod 1777 /tmp`, the problem went away.



- ▶ The old message Give root password for maintenance (or type Control-D for normal startup) is still there at the end of the boot process. While this isn't a problem per se, it does confuse a lot of people.
- ▶ In testing anonymous FTP, we discovered that /home/ftp does not exist, so anonymous FTP fails after providing the e-mail address for the password. Simply doing a `mkdir /home/ftp` fixes the problem.
- ▶ Apparently the xdm binary is expecting to find its configuration files in /etc/X11/xdm/, when they're actually in /usr/X11R6/lib/X11/xdm/. We set up a symbolic link as a temporary workaround.
- ▶ To get xdm working, we had to create an /etc/pam.d/xdm file. We put in the following for the contents:

```

#%PAM-1.0
auth    required    /lib/security/pam_unix.so    nullok #set_secrcp
account required    /lib/security/pam_unix.so
password required    /lib/security/pam_unix.so    #strict=false
session required    /lib/security/pam_unix.so    debug # trace or none

```

- ▶ The permissions on /dev/null needed to be changed from 644 to 666.
- ▶ To get the xterm command to work, we had to turn on the setuid bit for it:

```
# chmod 4711 /usr/X11R6/bin/xterm
```

- ▶ Whenever we logged onto the system, either through telnet or ssh, we noticed that our TERM environment variable was being set to "dumb," which caused warning messages whenever we tried to use the `less` command, or an editor such as `vi`. We tracked this down to the /etc/profile script. In there is a line:

```
if [ "$TERM" = "linux" ]; then
```

We fixed the problem by changing this to:

```
if [ "$CONSOLE" = "/dev/console" ]; then
```





## Caiman Linux for zSeries

Caiman Linux for zSeries is a Debian-based distribution developed by the Korean company LinuxKorea. The Debian project is a Linux distribution committed to free software, created and maintained by thousands of developers all over the world. So, the Debian project uses the same scheme for developing a distribution as the Linux kernel developers. For more information about the Debian project visit:

<http://www.debian.org>

Caiman Linux for zSeries is not an official Debian distribution. It first became available in November 2000. Because it is Debian-based, it inherits most of the features of the Debian distribution (see 14.2, “What’s specific about Caiman Linux” on page 214). We used version 1.0 of Caiman Linux for zSeries, which is based on the 2.2.16 Linux kernel.

## 14.1 Learning more about Caiman

Further information on Caiman Linux for zSeries is available at:

<http://linux390.linuxkorea.co.kr>

The Caiman Linux distribution can be obtained via FTP from:

<ftp://linux390.linuxkorea.co.kr/caiman>

The documentation for the distribution is *Caiman Linux for zSeries Installation Guide*, July 2001. It is available at:

<ftp://linux390.linuxkorea.co.kr/caiman/doc>

We used version 1.0 of Caiman Linux for zSeries, which is based on the 2.2.16 Linux kernel.

## 14.2 What's specific about Caiman Linux

Above all, Caiman Linux provides deb-formatted packages and all the advanced tools for package management found in the Debian distribution. It also supports the Advanced Package Tool (apt) acquisition via **apt-get**. This allows easy installation and updates of software packages for Caiman Linux. It also has a very sophisticated dependency check between packages, to avoid conflicts.

There are different versions of the distribution which represent the development state. The current stable version is referred as *potato*, and the unstable (that is, still under development) version is referred as *woody*. The default installation is minimal, which means you have to choose which software packages to install.

## 14.3 System requirements

Before you start installing Caiman Linux, you should check the system requirements. During installation, you will be asked many questions about your system. Therefore, we suggest that you fill out a worksheet before you start.

- ▶ Hardware requirements
  - 9672 G5 and newer processor or Multiprise 3000
  - Minimum of 64 MB main memory
  - Access to the Hardware Management Console (HMC) for LPAR or VIF installation
  - At least one volume of 3390 model 3 DASD
  - Access to a tape drive 3480 or 3490 for LPAR or VIF installation

- Network adapter: OSA-2 (required for VIF), OSA-Express or a CTC connection
- FTP server for holding the Caiman distribution
- ▶ Software requirements
  - For VIF, installation tape of VIF
  - For z/VM, a guest machine user ID and password (it is assumed that VM is already installed and thus no tape unit is required)

### 14.3.1 Installation worksheet - Caiman

We suggest that you complete the worksheet shown in Table 14-1 before you start the installation.

*Table 14-1 Caiman installation worksheet*

VM guest user ID (if installing under VM)	
VM guest password (if installing under VM)	
FTP server IP address or host name	
FTP path to Caiman directory (CD)	
FTP user ID	
FTP password	
DASD devices to use for Caiman Linux	
Tape unit *optional (if installing in LPAR)	
Network device type	
Network device (and port)	
Full qualified host name	
Host IP address	
Netmask	
Broadcast address	
Gateway IP address	
DNS IP address	
DNS search domain	
MTU size	

## 14.3.2 Caiman disk requirements

The initial installation only extracts a minimal root file system onto DASD, which takes up about 100 MB. Later on, the DASD requirements are completely up to you, depending on how many (and which) packages you add.

Table 14-2 lists the requirements for a reasonable installation with Internet services like HTTP, FTP and mail.

Table 14-2 Caiman disk requirements

Directory	Size
/bin	2.3 MB
/boot	1.8 MB
/etc	4.6 MB
/lib	5.1 MB
/sbin	1.6 MB
/usr	196 MB
/var	8.5 MB

## 14.4 Installation

This section explains the steps needed to install Caiman Linux on a z/VM virtual machine. (For LPAR installation, tape production and IPL from tape are needed.)

The installation steps are the same as for the other distributions, except that you have to substitute the names of the kernel and RAMdisk image accordingly. The Caiman distribution comes with a detailed documentation for a installation under VIF.

For this installation, we downloaded the CD ISO image from the Caiman FTP server. This image was put on a local Linux for S/390 system, mounted on the loopback device, and was accessible via FTP. You can certainly also put the image or the discrete distribution files on other FTP server types that are available to you. Or, you can install directly from the Caiman FTP server (which may require additional settings if you are behind a firewall).

In the following example we installed Caiman Linux on a z/VM guest machine, running on a z/900 with the following guest machine settings:

- Memory: 512 MB
- DASD: VM minidisks 1x 2600 cylinder (/) and 1x 100 cylinder (swap)

- Network: OSA-Express in non-QDIO mode

These examples assume that you already have a VM guest machine with similar or sufficient capabilities.

### 14.4.1 Preparing the first IPL of the install system

After logging onto your VM guest machine, you need to transfer the initial files that are needed for the minimal RAMdisk installation system:

VMRDR IMG	The kernel image (in binary)
INITRD IMG	The initial RAMdisk (in binary)
CAIMAN PARM	The parameter file (in EBCDIC)
CAIMAN EXEC	The REXX script to IPL from the virtual reader (in EBCDIC)

Example 14-1 shows how you can use the VM FTP client to get the files from the FTP server:

*Example 14-1 Transferring the initial files to VM*

---

```
ftp 9.12.9.183
...
Command:
locsite fix 80
Command:
bin
>>>TYPE i
200 Type set to I.
Command:
cd /home/redbooks/linuxSystems/caiman/images      <- the files for initial
                                                    IPL are in images

get initrd.img
get vmrdr.img
ascii
get caiman.exec
get caiman.prm
quit
```

---

The parameter file may need some customizing, depending on your configuration. You can use XEDIT to change the original parameter file. The file which comes on the Caiman CD is written for use with a VIF installation:

```
ramdisk_size=32768 root=/dev/ram0 ro iucv=$TCPIP
```

So, we changed the file to match our setup:

```
ramdisk_size=32768 dasd=201,203 root=/dev/ram0 ro
```

The REXX script for IPLing from the virtual reader can be used as is:

```
/* */
'clear rdr'
'purge rdr all'
'spool punch * rdr'
'punch vmrdr img a (noh'
'punch caiman prm a (noh'
'punch initrd img a (noh'
'ch rdr all keep nohold'
'ipl 00c'
```

## 14.4.2 Initial IPL and network setup

You start the first IPL by simply executing the CAIMAN EXEC from the VM console, as shown in Example 14-2:

*Example 14-2 IPL from the virtual reader*

---

```
caiman          <- execute the EXEC
...
Linux version 2.2.16 (root@map)(gcc version 2.95.2 20000220 (Debian
GNU/Linux))
#1 SMP Tue May 15 11:53:02 KST 2001
Command line is: dasd=0201,0203 root=/dev/dasda1 noinitrd
root=/dev/dasda1 ro
We are running under VM
This machine has an IEEE fpu
Initial ramdisk at: 0x02000000 (16777216 bytes)
Detected device 2920 on subchannel 0000 - PIM = 80, PAM = 80, POM = FF
...          <- you will see a lot of boot messages here

Caiman Installation System is starting...
====,
===
===
===
=====' a i m a n Linux for zSeries 1.0 For your successful e-business!

First, select the type of your network device:
0) no network
1) OSA Token Ring
2) OSA Ethernet
3) OSA-Express Gigabit Ethernet (experimental)
4) Channel To Channel
5) Escon
6) IUCV
Enter your choice (1-6):
2          <- we are using a OSA FE adapter
```



To set up the network, you have to read and confirm the license information of the network device driver provided by IBM.

Do you want to see the license (Yes/No) ?

yes

... <- you are required to read the license page by page

Do you agree with this license (Yes/No) ?

yes

Ok, now we can set up the network configuration.

Please enter the device address of the network device

Ask your system administrator for the correct settings.

If there is only ONE OSA network device attached to your machine, you may type "auto" for automatic detection; use this option carefully.

Network device address (e.g. FC20):

**2920**

Please enter the relative port number on device address 2920

Relative port:

**0**

Unloading LCS module if active...

rmmod: module lcs is not loaded

Trying to start the LCS module now...

insmod -v lcs noauto=1 devno\_portno\_pairs=0x2920,0 :

... <- in our version there are a lot of debugging messages here  
but finally the lcs driver gets loaded:

user function /lib/modules/2.2.16/net/lcs.o

Using /lib/modules/2.2.16/net/lcs.o

Symbol version prefix 'smp\_'

Starting lcs

lcs: eth0 configured as follows read subchannel=0 write subchannel=1

read\_devno=2920 write\_devno=2921

hw\_address=00:06:29:6C:CB:CE rel\_adapter\_no=0

lcs configured to use sw statistics,

ip checksumming of received packets is off.

autodetection is off.

configured to detect

cu\_model 0x01,15 rel\_adapter(s)

cu\_model 0x08,15 rel\_adapter(s)

cu\_model 0x60,1 rel\_adapter(s)

cu\_model 0x1F,15 rel\_adapter(s)

lsmod now shows all loaded modules:

lcs 14896 0 (unused) <- this line shows success

Was the start of the LCS module successful (Yes/No) ?

yes <- answer yes because lcs loaded cleanly

Please enter your full host name (e.g. linux3.linuxkorea.co.kr):

**vmlinux1.itso.ibm.com**

Please enter your IP address:

**9.12.6.98**

Please enter the net mask:

**255.255.255.0**

Please enter the broadcast address:

**9.12.6.255**

Please enter the gateway address:

**9.12.6.75**

Please enter the IP address of the DNS server (leave blank for none):

**9.12.14.7**

Please enter the DNS search domain (e.g. linuxkorea.co.kr):

**itso.ibm.com**

Please enter the MTU (Maximum Transfer Unit, leave blank for default: 1492):

Configuration for eth0 will be:

Full host name : vmlinux1.itso.ibm.com

IP address : 9.12.6.98

Net mask : 255.255.255.0

Broadcast address: 9.12.6.255

Gateway address : 9.12.6.75

DNS IP address : 9.12.14.7

DNS search domain: itso.ibm.com

MTU size : 1492

Is this correct (Yes/No) ?

**yes**

For security reasons you have to set an temporary installation system password for the user "root".

You'll be asked for it only when you telnet into this installation system to limit the access to it and it will be cleared as soon as you shut down or reset the installation system

Please enter the temporary installation password:

**vmlinux1** <- only valid as long as the RAMdisk system runs

ifconfig eth0 9.12.6.98 netmask 255.255.255.0 broadcast 9.12.6.255 mtu 1492

/sbin/ifconfig eth0 :

```
eth0      Link encap:Ethernet  HWaddr 00:06:29:6C:CB:CE
          inet addr:9.12.6.98  Bcast:9.12.6.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1492  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
```

Trying to ping my IP address:

...

Trying to ping the IP address of the Gateway:

...

Trying to ping the IP address of the DNS Server:

```
...          <- pinging of loopback, gateway and DNS server successful
Network Setup finished, running inetd...
Caiman vmlinux1:/ # <- at this point we are ready to telnet into the system
```

---

### 14.4.3 Installation of the base system

To complete the basic installation, a telnet session is needed. You can use your favorite telnet client on your workstation to connect to the system. In our case, we tested the freely available telnet client PuTTY and the TeraTerm client from our Windows 2000 PCs. However, the best results in terms of graphic presentation we achieved by working from a Linux client.

```
Welcome to Caiman Linux for zSeries 1.0!
Kernel 2.2.16 on an S/390 (vmlinux1.itso.ibm.com)
login: root
Password:

  -----,
  ---
  ---
  ---
  ---
  -----' a i m a n Linux for zSeries 1.0 For your successful e-business!

  For your system installation, enter 'install.sh'.
```

Enjoy it.

```
# install.sh
```

The Caiman installer is a single script that steps through all the necessary tasks, such as DASD specification and formatting, and FTP settings. The installer comes with a menu-driven interface, as shown in Figure 14-1 on page 222.

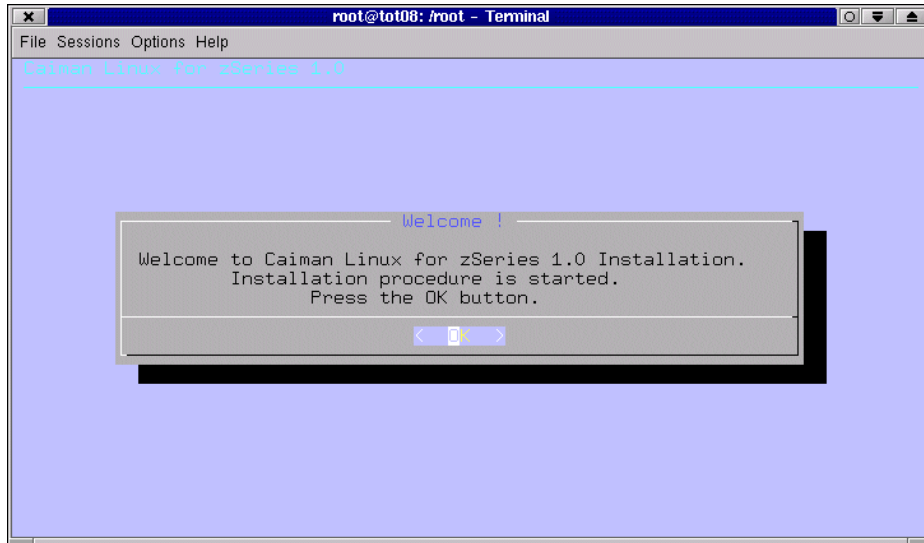


Figure 14-1 Caiman installer start screen

1. Press **OK** to start the installation.
2. In the first step, the installer checks all the DASD volumes which are online to your VM guest machine. Press **OK** to start the check.
3. After finishing the check, the installer gives you a window with all recognized volumes. Press **Exit** to step further.
4. The next window asks for the DASD module parameters. Here you specify all the volumes you want to use for Linux. Press **OK** to load the module.
5. When the module load succeeds, you will get a message box saying so. Press **OK** to go on.
6. The next window (shown in Figure 14-2 on page 223) lets you assign the volumes to the different file systems you want to create for your Linux system.

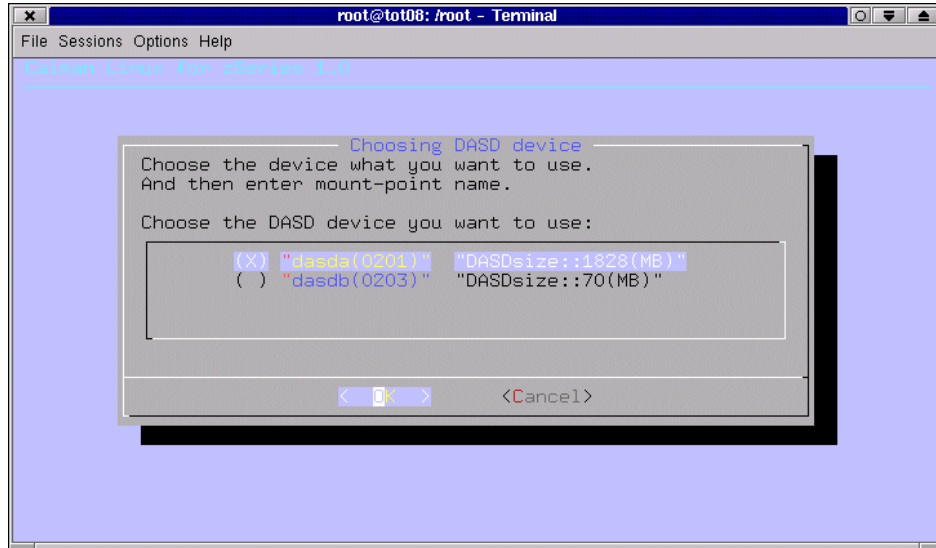


Figure 14-2 Caiman installer DASD choice screen

7. You select a volume by highlighting the entry, pressing **Space** and then **Enter**. This will give you an input box for the volume where you specify the mount point, such as `/`, `/usr`, or `/home`. For a swap partition, just insert **swap** instead of a mount point. After finishing the volumes, press **OK** to proceed.
8. Next, you will be asked if you want to format the volumes. Press **Yes** to format and create the file system, or press **No** to leave the volume untouched. You have to go through all volumes you assigned in the previous step and wait until the formatting is finished. This may take a while for each volume, depending on the size (meanwhile the screen stays blue - be patient).

**Note:** This procedure was changed by Caiman developers due to our request during the residency.

Before that, the formatting of multiple volumes ran parallel in the background. And while this offered the advantage of speed, it meant that you had to pay *close* attention to the console messages in order to recognize when the formatting was finished, before proceeding further with the installation.

Consequently, if you have multiple volumes to format (perhaps for later use with LVM), we recommend that you only choose necessary volumes to format during installation. You can format the other volumes later on in parallel and in the background by using `dasdfmt` and `mke2fs`.

9. After formatting the volumes, you can check the message log file. Select **Yes** to do so, or select **No** to move on.
10. Specify the IP address of the FTP server which contains your Caiman distribution CD (or ISO image or files). Press **OK** for the next window.
11. Provide the absolute path to your Caiman distribution on the FTP server (see Figure 14-3) to download the base root file system. After that, press **OK**.

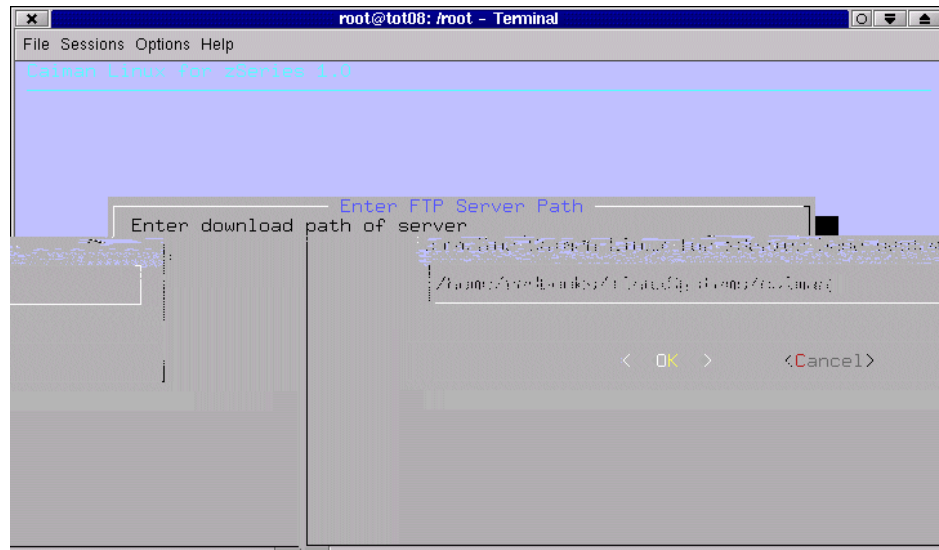


Figure 14-3 Caiman installer FTP server path

**Note:** The installer automatically appends /images to the path when looking for the file system base<Caimanversion>.tgz. So if you just mounted a CD-ROM on the FTP server, you would specify, for example, /mnt/cdrom as path.

12. Provide a user name for the FTP server, and press **OK** when done.
13. Supply the password for the user, and press **OK** when done.
14. The installer first checks if the server is online. Press **OK** to start the download.
15. The installer gets the base file system from the FTP server, unpacks it, and configures the system. This usually takes only a few minutes. When done, you will get a screen telling you that the installation is now complete. Pressing **OK** returns you to the command prompt.

16. In your directory is a file created named `silolog`. Review the output of `silolog` with `cat silolog` to verify that the IPL record was written correctly. Then you can close the telnet session.

#### 14.4.4 Booting the base system from DASD and adding packages

After finishing the base system install, go to the VM console of your Linux guest system and re-IPL from DASD. At this point you don't need to do a clean shutdown of the running install system since it is only in memory and all DASD is automatically unmounted by the installer script when done.

```
#cp i 201 clear      <- the device number depends on your configuration

Linux version 2.2.16 (root@map) (gcc version 2.95.2 20000220(Debian
GNU/Linux))
#1 SMP Tue May 15 11:53:02 KST 2001
Command line is: dasd=0201,0203 root=/dev/dasda1 noinitrd
root=/dev
...
```

On the VM console, you'll receive Linux boot messages which will quickly fill up the console screen. The screen will be automatically updated after a while, so if you want to read the messages at your own pace, press **Ctrl** (hold) during the boot process.

If everything was successful during the previous installation steps, your system should get to the point where it asks for the root password, ready to telnet into. Connect to the system as root via telnet to add more packages with the Debian package manager front-end **dselect**.

**Note:** The password you specified for root during the initial IPL from the virtual reader is gone with the purge of the RAMdisk system from memory.

The base file system has a default password "pass4root" for the user root. You should change that immediately with the **passwd** command after the first login.

At this point we will just focus on how you can add more software packages to the system quickly. For more information on the powerful Debian package tools, refer to 14.5.9, "Software package management" on page 234.

Start **dselect** from the command prompt. You will see a divided screen, with a menu at the upper half and explanations in the lower half.

The following examples only show the upper half of the screen, where you select actions.

```
Debian `dselect' package handling frontend.
```

- \* 0. [A]ccess     Choose the access method to use.
- 1. [U]pdate     Update list of available packages, if possible.
- 2. [S]elect     Request which packages you want on your system.
- 3. [I]nSTALL    Install and upgrade wanted packages.
- 4. [C]onfig     Configure any packages that are unconfigured.
- 5. [R]emove     Remove unwanted software.
- 6. [Q]uit       Quit dselect.

The asterisk (\*) next to the menu items selects the menu item. At this point, we'll simply follow the numbered items in order.

Select **Access** and press **Enter** to specify the access method to the distribution packages.

```
dselect - list of access methods
Abbrev.      Description
cdrom        Install from a CD-ROM.
nfs          Install from an NFS server (not yet mounted).
harddisk     Install from a hard disk partition (not yet mounted).
mounted      Install from a filesystem which is already mounted.
floppy       Install from a pile of floppy disks.
* apt        APT Acquisition [file,http,ftp]
```

Select **apt** and press **Enter**. You will be asked if you want to change the configuration. In our case we choose yes, because we had the distribution on a local FTP server and wanted to get done quickly. So we put in our FTP server address, including user name and password, as follows:

```
ftp://tot08:<ftp_passwd_here>@9.12.9.183/home/redbooks/linuxSystems/caiman
```

On the next questions we took the default options `stable` and `main contrib non-free`, and completed this step by answering **n** to the question: Would you like to add another source?.

Once back on the main menu, select **Update** and press **Enter**. This downloads a list of available packages from the FTP server. After the download is finished, you will automatically be back in the main menu.

Next, choose **Select** and press **Enter**. This will bring up a help screen, which you should carefully read; then leave by pressing **Space**. This will bring up the select screen, as shown in Figure 14-4 on page 227.



```

root@tot08: /root - Terminal
File Sessions Options Help
-----
EIOM Pri Section Package Inst.ver Avail.ver Description
-----
- All packages -
  - Newly available packages -----
    - New Standard packages -----
      - New Standard packages in section devel -----
n* Std devel gdb <none> 5.0-caiman3 The GNU Debugger
      - New Standard packages in section editors -----
n* Std editors emacs20 <none> 20.7-caiman The GNU Emacs editor.
      - New Optional packages -----
      - New Optional packages in section admin -----
l* Opt: admin: lvm <none> 0.8i-caiman The Logical Volume Manage
-----
lvm packages
-----
line you have highlighted represents many packages; if you ask to
install, remove, hold, &c it you will affect all the packages which match
the criterion shown.

If you move the highlight to a line for a particular package you will see
information about that package displayed here. You can use 'o' and 'O' to
change the sort order and give yourself the opportunity to mark packages in
different kinds of groups.

description
-----

```

Figure 14-4 Caiman dselect select screen

In this screen, you can select items by pressing the up and down arrows, and view a short description of the package in the lower half of the screen.

Note the following under the column heading EIOM in the upper left:

- ▶ An asterisk (\*) in the “I” column means installed.
- ▶ An asterisk (\*) in the “M” column marks it as a package *to be* installed.
- ▶ You select a package for installation by highlighting it and pressing the plus sign (+).
- ▶ The packages are grouped into sections like — All packages —, which you can select as a whole by highlighting and pressing +.

When you select packages or groups, dselect will automatically check the dependencies. If there are any unfulfilled dependencies, it will give you another window where you can select or deselect related packages to meet the requirements. To step further after selecting all the packages you want, press **Enter**.

Back in the main menu, select **Install** and press **Enter**. The installation will start and you can follow its progress on the command line. Several packages need to be configured during this process, so depending on your package selection, you may have to answer some questions (for example, to configure Apache, if you selected it).

After the installation is finished, you will be returned to the main menu. You can quit `dselect` now and further configure and explore your new installed Caiman Linux for zSeries!

## 14.5 System configuration and administration

In this section we describe the basics of how to configure your system using Debian tools and configuration files. For more in-depth information, refer to Caiman Linux documentation or to the Debian project home page.

Caiman Linux does not provide a graphical or menu-driven system configuration and administration program. However, there are command line tools which perform the necessary tasks, and the system configuration files are editable with any text editor (which in some situations may be more efficient than using tools). There are also tools like Webmin available on the Internet (see 21.2.3, “Webmin” on page 403).

### 14.5.1 File system layout

The Debian policy strictly implements the File system Hierarchy Standard (FHS), and so does Caiman Linux.

Caiman Linux uses the System-V initialization style. The `/etc/init.d` directory contains the scripts executed by `init` at boot time and when `init` state (or run level) is changed. These scripts are referenced by symbolic links in the run level directories `/etc/rc<runlevel>.d/`.

The Debian home page provides further information:

<http://www.debian.org/doc/debian-policy/>

Here are some important file/directory locations for the Caiman Linux distribution:

File	Description
<code>/etc/samba/smb.conf</code>	Configuration of smb (Samba) daemon
<code>/etc/apache/httpd.conf</code>	Web server configuration file
<code>/etc/modules.conf</code>	Aliases and options for loadable kernel modules

<i>/etc/fstab</i>	File systems mounted or available for mounting
<i>/etc/group</i>	Group information
<i>/etc/hosts</i>	Map of IP numbers to host names
<i>/etc/hosts.allow</i>	Hosts allowed to access Internet services
<i>/etc/hosts.deny</i>	Hosts forbidden to access Internet services
<i>/etc/inetd.conf</i>	Configuration for the inetd daemon, which controls access to Internet services
<i>/etc/inittab</i>	Configuration for the init daemon, which controls executing processes
<i>/etc/login.defs</i>	Options for useradd and related commands
<i>/etc/init.d</i>	Scripts for system and process startup and shutdown
<i>/etc/rc*.d</i>	Symbolic links to the startup scripts
<i>/etc/rc.boot</i>	Scripts for system boot
<i>/etc/skel</i>	Skeleton files used to establish new user accounts
<i>/var/log/apache/access.log</i>	Log of Web server access
<i>/var/log/apache/error.log</i>	Log of Web server errors
<i>/var/log/messages</i>	System log
<i>/var/spool/cron</i>	Directory for at and cron configuration files

## 14.5.2 Network setup

The network setup for Caiman Linux is stored in a set of configuration files. These files are all plain text files, which you can edit to change the settings.

Your host name is stored in */etc/hostname*.

```
# cat /etc/hostname
vmlinux1
```

The IP address, your specified host name, and the domain are stored in */etc/hosts*.

```
# cat /etc/hosts
```

```

#
# /etc/hosts
#
127.0.0.1    localhost
9.12.6.98   vmlinux1.itso.ibm.com  vmlinux1
The IP address of the DNS server and the search domain are stored in the
file /etc/resolv.conf.
# cat /etc/resolv.conf
nameserver 127.0.0.1
#
# /etc/resolv.conf
#
# Automatically generated by Caiman Linux/390 base system installer
#
search itso.ibm.com
nameserver 9.12.14.7

```

The configurations for your network interfaces are stored in the file */etc/network/interfaces*.

```

# cat /etc/network/interfaces
auto lo eth0
iface lo inet loopback

iface eth0 inet static
    address 9.12.6.98
    netmask 255.255.255.0
    broadcast 9.12.6.255
    gateway 9.12.6.75
    up ifconfig eth0 mtu 1492

```

The settings for the network device driver (lcs, in our case) are stored in */etc/modutils/arch/s390*. These settings get integrated into */etc/modules.conf* by running **update-modules**.

```

# cat /etc/modutils/arch/s390
alias eth0 lcs
options lcs noauto=1 devno_portno_pairs=0x2920,0
# update-modules
# cat /etc/modules.conf
...
### update-modules: start processing /etc/modutils/arch/s390
alias eth0 lcs
options lcs noauto=1 devno_portno_pairs=0x2920,0
...

```

You activate the settings you made in the configuration files by starting or restarting the network activation script.

```

# /etc/init.d/networking restart

```

```
Reconfiguring network interfaces: done.  
#
```

For further information on how to configure other network devices and command syntax, refer to the man pages and the Caiman Linux for zSeries installation guide.

### 14.5.3 Managing DASD

To add new DASD to your system you need to specify them in the parameter file, which gets passed to the kernel at boot time. So add your new devices to the parameter file `/boot/parmfile`. This new information needs to be written onto the IPL record, so you have to re-run **sil**.

```
# cat /boot/parmfile  
dasd=0201,0203 root=/dev/dasda1 noinitrd  
# vi /boot/parmfile  
# cat /boot/parmfile  
dasd=0201,0203-206 root=/dev/dasda1 noinitrd  
# sil  
Testlevel is set to 0  
IPL device is: '/dev/dasda'  
bootsector is: '/boot/ipleckd.boot'...ok...  
bootmap is set to: '/boot/boot.map'...ok...  
Kernel image is: '/boot/image'...ok...  
original parameterfile is: '/boot/parmfile'...ok...tempfile is  
./parm.t4Kb7U  
final parameterfile is: './parm.t4Kb7U'...ok...  
ix 0: offset: 02021e count: 0c address: 0x00000000  
ix 1: offset: 02022b count: 80 address: 0x0000c000  
ix 2: offset: 0202ab count: 80 address: 0x0008c000  
ix 3: offset: 02032b count: 80 address: 0x0010c000  
ix 4: offset: 0203ab count: 0d address: 0x0018c000  
ix 5: offset: 052112 count: 01 address: 0x00008000  
Bootmap is in block no: 0x00023c87  
#
```

After a reboot, the new volumes must be formatted with **dasdfmt**. After that is done, you can create file systems with **mke2fs** and mount the volumes where you want. To automatically mount the new DASDs at boot time, you need to add a entry in `/etc/fstab`.

Caiman Linux includes the Logical Volume Manager (LVM), so you can set up volume groups and logical volumes out of a pool of physical volumes. Refer to Chapter 17, “Logical Volume Manager” on page 301 for information on that task.

## 14.5.4 Managing users

There are several command line tools you can use to manage users and groups on your Caiman Linux system:

```
useradd
userdel
adduser
groupadd
groupdel
```

For detailed descriptions of using these tools, you can refer to the man pages and also see Chapter 22, “Overview of security on Linux” on page 435.

For adding users you would utilize the **adduser** program, which is actually a Perl script that automatically creates a new home directory for the user and asks for the new password and additional information.

### ***adduser***

```
# adduser ron
Adding user ron...
Adding new group ron (1000).
Adding new user ron (1000) with group ron.
Creating home directory /home/ron.
Copying files from /etc/skel
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for ron
Enter the new value, or press return for the default
    Full Name []: ron
    Room Number []: 123
    Work Phone []: 1-2345
    Home Phone []: -
    Other []: -
Is the information correct? [y/n] y
```

You can delete a user and his home directory with the **userdel** command.

### ***userdel***

```
# userdel -r ron
```

If you omit the **-r** parameter, the users home directory will not be deleted.

## 14.5.5 Configure mail

Caiman Linux uses **exim** as the Mail Transfer Agent (MTA). It comes with a command line configuration tool known as **eximconf**. The following example shows how to configure **exim** with an external mail host, in order to be able to send and receive e-mails between with your Caiman system and the outside world.

**Note:** In order to send and receive e-mails using a external mail host, this external mail host needs to be configured properly and your Linux system needs to have a DNS entry in your DNS server.

```
# eximconf
```

You have to answer a set of questions at the command line. At the beginning, choose option **2**:

- (2) Internet site using smarthost: You receive Internet mail on this machine, either directly by SMTP or by running a utility such as fetchmail. Outgoing mail is sent using a smarthost. optionally with addresses rewritten. This is probably what you want for a dialup system.

After that, you'll need to answer other questions, and for most of them you can just take the default. At the point where it asks for the smarthost, you have to insert your mail server:

```
=====
Which machine will act as the smarthost and handle outgoing mail?

Enter value (`x' to restart): d01mlc96.pok.ibm.com
=====
```

At the end, you'll get a summary:

```
=====
Mail generated on this system will have `vmlinux1.itso.ibm.com' used
as the domain part (after the @) in the From: field and similar places.

The following domain(s) will be recognised as referring to this system:
localhost, vmlinux1.itso.ibm.com

Mail for postmaster, root, etc. will be sent to root.

Local mail is delivered.

Outbound remote mail is sent via d01mlc96.pok.ibm.com.

Is this OK ? Hit Return or type `y' to confirm it and install,
```

or `n' to make changes (in which case we'll go round again, giving you your previous answers as defaults. (Y/n) y

At this point, **exim** is configured and you should be able to send to, and receive mail from other hosts.

## 14.5.6 Using Apache

The main configuration file for the Apache Web server is `/etc/apache/httpd.conf`. You can edit this file directly to configure Apache. For the basic settings, use the **apacheconfig** tool which comes with Caiman Linux. The start/stop script for Apache is `/etc/init.d/apache`.

There are many documents available on the Internet that explain how to configure Apache, and it also supplies online documentation that you can refer to for further information.

## 14.5.7 Using FTP

Caiman Linux comes with the proftpd server. It is a powerful replacement for wu-ftpd. This File Transfer Protocol daemon supports hidden directories, virtual hosts, and per-directory ".ftppass" files. It uses a single main configuration file, with a syntax similar to Apache. The configuration file is `/etc/proftpd.conf`.

You can run proftpd either as a standalone server or with **inetd**. For more information, refer to the man pages.

## 14.5.8 Using Samba

There are several tools available to configure Samba for Linux systems. Within Caiman Linux, you can use SWAT or other graphical tools, for example the Webmin Samba plug-in (see Appendix 21.2.3, "Webmin" on page 403). For the basic configurations, there is also a command line tool known as **sambaconfig** which configures Samba to be started as either a standalone server, or to be invoked by **inetd**.

The main configuration file is `/etc/samba/smb.conf`.

## 14.5.9 Software package management

In this section we provide a brief overview of the Debian package management tools used in Caiman Linux, and explain how to use them. A useful, more detailed explanation can be found in the Caiman Linux for zSeries installation guide.



### ***dselect***

As introduced in 14.4.4, “Booting the base system from DASD and adding packages” on page 225, **dselect** is a graphical tool that can be used to manage packages on your system. You can browse, install and upgrade packages with this tool. It also provides automatic dependency check, search functionality, and other useful functions. For more information about **dselect**, refer to the Caiman Linux for zSeries installation guide or the man pages.

### ***dpkg***

One of the command line tools for package management is **dpkg**. With **dpkg** you can install, remove and query information on **deb** packages in a similar way to **rpm**. For more information, refer to the man pages.

As an example of using **dpkg**, we show here the installation of OpenSSL and OpenSSH. (We got them as extra packages from the Caiman developers, since they were not included in the CD ISO image at the time of writing).

```
# dpkg -i openssl_0.9.6a-3_s390.deb
Selecting previously deselected package openssl.
(Reading database ... 29846 files and directories currently installed.)
Unpacking openssl (from openssl_0.9.6a-3_s390.deb) ...
Creating directory /etc/ssl
Setting up openssl (0.9.6a-3) ...

# dpkg -i ssh_2.5.2p2-2.1_s390.deb
...
      <- you will get a configuration dialog for ssh
...
Setting up ssh (2.5.2p2-2.1) ...
Starting OpenBSD Secure Shell server: sshd.

You have mail in /var/spool/mail/root      <- you also get a warning mail
      regarding insecure telnet
```

You can remove packages by invoking the **-r** parameter instead of the **-i** parameter.

```
# dpkg -r ssh
# dpkg -r openssl
```

### ***apt-get***

With the command line tool **apt-get** you can easily receive and install packages from FTP servers you specify. The server configurations are stored in `/etc/apt/sources.list` and are processed in the same order as they appear in the file. The tool **apt-get** is also used by **dselect**, so the configuration file should show the settings you made during the installation.

```
# cat /etc/apt/sources.list
```

```
deb ftp://tot08:tot08@9.12.9.183/home/redbooks/linuxSystems/caiman stable
main contrib non-free
deb ftp://linux390.linuxkorea.co.kr/caiman stable main contrib non-free
deb http://http.us.debian.org/debian stable main contrib non-free
```

To retrieve and install the package for **at** (executes delayed cron jobs), simply do:

```
# apt-get install at
Reading Package Lists... Done
Building Dependency Tree... Done
The following NEW packages will be installed:
  at
0 packages upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
1 packages not fully installed or removed.
Need to get 39.2kB of archives. After unpacking 209kB will be used.
Get:1 ftp://9.12.9.183 stable/main at 3.1.8-10 [39.2kB]
Fetched 39.2kB in 0s (94.8kB/s)
Selecting previously deselected package at.
(Reading database ... 30277 files and directories currently installed.)
Unpacking at (from ../archives/at_3.1.8-10_s390.deb) ...
Setting up at (3.1.8-10) ...
Starting deferred execution scheduler: atd.
```

For more information on how to use **apt-get** from the command line, refer to the man page.

### **alien**

The tool **alien** is a set of scripts that converts other binary software packages. For example, **rpm** packages can be converted into the Debian package format and vice versa.

As an example we pick the **pico** editor package from the SuSE distribution (see Figure 14-3).

*Example 14-3 Convert an rpm package to deb*

---

```
# ls pico*
pico-4.30-13.s390.rpm
# alien --to-deb pico-4.30-13.s390.rpm
-- Examining pico-4.30-13.s390.rpm
-- Unpacking pico-4.30-13.s390.rpm
344 blocks
----
-- Automatic package debianization
-- Building the package pico_4.30-14_s390.deb
dh_testdir
# Nothing to do.
dh_testdir
dh_testroot
dh_clean -k
```

```

dh_installdirs
cp -a `ls |grep -v debian` debian/tmp
dh_installdocs
dh_installexamples
dh_installmenu
dh_installcron
dh_installchangelogs
dh_compress
dh_suidregister
dh_installdeb
dh_shlibdeps
dh_gencontrol
dh_makeshlibs
dh_md5sums
dh_builddeb
dpkg-deb: building package `pico' in `../pico_4.30-14_s390.deb'.

```

```

Generation of pico_4.30-14_s390.deb complete.
-- Successfully finished
# ls pico*
pico-4.30-13.s390.rpm  pico_4.30-14_s390.deb
# dpkg -i pico_4.30-14_s390.deb
Selecting previously deselected package pico.
(Reading database ... 30170 files and directories currently installed.)
Unpacking pico (from pico_4.30-14_s390.deb) ...
Setting up pico (4.30-14) ...

```

---

For further information on how to use **alien**, refer to the man page.

### ***Building a package from source***

You can also build a binary **deb** package from source using the **dpkg** toolset. To get the source files, either use **apt-get** or simply do an FTP download from one of the Debian source sites.

There are usually three files which you have to download for a package:

<package>.dsc	The Debian package description file
<package>.orig.tar.gz	The original source as zipped tar archive
<package>.diff.gz	Patches as zipped tar archive (if applicable)

After downloading the files in a local directory, you change to that directory and issue a **dpkg-source -x <package>.dsc**. This will extract the source into a directory, check the PGP signature, and apply the patches.

After that, you change into the newly created source directory and issue a **dpkg-buildpackage -b**. This will build the binaries and create a new deb package in superior directory. The -b parameter prevents **dpkg-buildpackage** from creating the source archives again. After that, you have a new binary **deb** package which you can install on your local (or any other) Caiman Linux system.

Following is an example for the **whowatch** package:

```
# ls who*
whowatch_1.3-1.diff.gz whowatch_1.3-1.dsc whowatch_1.3.orig.tar.gz
# dpkg-source -x whowatch_1.3-1.dsc
dpkg-source: extracting whowatch in whowatch-1.3
# cd whowatch-1.3/
/whowatch-1.3# dpkg-buildpackage -b
...
dpkg-deb: building package `whowatch' in `../whowatch_1.3-1_s390.deb'.
dpkg-genchanges -b
dpkg-genchanges: binary-only upload - not including any source code
dpkg-buildpackage: no source included in upload
/whowatch-1.3# cd ..
# dpkg -i whowatch_1.3-1_s390.deb
Selecting previously deselected package whowatch.
(Reading database ... 14916 files and directories currently installed.)
Unpacking whowatch (from whowatch_1.3-1_s390.deb) ...
Setting up whowatch (1.3-1) ...
```

And now you can use the **whowatch** tool:

```
# whowatch
2 users: (0 local , 2 telnet , 0 ssh , 0 other)

(in.telnetd) root pts/0 tot08.itso.ibm.com whowatch
(in.telnetd) root pts/1 tot08.itso.ibm.com dselect

enter - proc tree, i - init tree, t - idle/cmd, x - refresh, q - quit
```



## Think Blue 64 Linux for zSeries

In this chapter we discuss the Millenux Think Blue 64-bit distribution. Millenux was founded in March 2000 by the managing directors of the Thinking Objects software GmbH, Fritz Elfert and Werner Gold. Now Millenux has taken over the past Linux division of Thinking Objects and continues to develop the distribution.

The Think Blue distribution is Red Hat-based and is available in two versions:

- ▶ Think Blue Linux for S/390 (31 bit)
- ▶ Think Blue Linux for zSeries (64-bit)

For further information visit:

<http://www.millenux.com>

The distributions for zSeries can be obtained on the Web at:

<ftp://linux.zseries.org>

For S/390, they can be obtained at:

<ftp://linux.s390.org>

This chapter only covers the 64-bit version. The distribution we used is the Think Blue 64 7.1 version and is based on the 2.4.3 Linux kernel.

## 15.1 What's specific about Think Blue

The Think Blue distribution was the second Linux distribution available for S/390 after the Marist distribution (see Chapter 13, “The Marist File System” on page 193).

It is based on Red Hat 7.1 and it comes as a 31-bit version and as a 64-bit version. The Think Blue 64-bit distribution is the first 64-bit Linux for zSeries distribution that is available for download. It includes support for tape devices (see Chapter 23, “Backup and restore” on page 463).

You can either download the Think Blue distributions as CD ISO images, or download the installation images and rpm files separately.

## 15.2 System requirements

Before you start, you should check the system requirements. The Think Blue 64-bit distribution only runs on a z900 system. You should also prepare a worksheet (see Table 15-1 on page 241) with all the information you will have to provide during installation.

- ▶ Hardware requirements
  - eServer zSeries - z900
  - Minimum of 64 MB main memory
  - Access to the Hardware Management Console (HMC) for LPAR installation
  - At least one volume of 3390 model 3 DASD
  - Access to a tape drive 3480 or 3490 for LPAR installation
  - Network adapter: OSA-2, OSA-Express, or a CTC connection
  - FTP server for holding the Think Blue distribution
- ▶ Software requirements
  - z/VM guest machine with user ID and password (for z/VM installation)

**Note:** This list only provides the requirements for the installation of Linux for zSeries. For Linux installation under VM, it is assumed that VM is already installed and thus no tape unit is required.

## 15.2.1 Installation worksheet

We suggest that you complete the following worksheet before you start the installation.

Table 15-1 Installation worksheet

VM guest user ID (if installing under VM)	
VM guest password (if installing under VM)	
FTP server IP address or host name	
FTP path to Think Blue directory (CD)	
FTP user ID	
FTP password	
DASD devices to use for Think Blue Linux	
Tape unit *optional (if installing in LPAR)	
Network device type	
Network device (and port)	
Full qualified host name	
Host IP address	
Netmask	
Broadcast address	
Gateway IP address	
DNS IP address	
DNS search domain	
MTU size	

You can also use an HTTP or NFS server as your installation server; simply substitute the FTP sections in the worksheet.

## 15.2.2 Think Blue disk requirements

The amount of DASD space required depends on your installation and what packages you want to add. Table 15-2 lists the sizes of the larger directories for the default installation plus KDE2.

Table 15-2 Think Blue disk requirements

Directory	Size
/bin	6.3 MB
/boot	4.8 MB
/etc	8.4 MB
/lib	13 MB
/opt	0
/sbin	8.5 MB
/usr	630 MB
/var	21.2 MB

## 15.3 Installation

In this section we discuss the installation of the Think Blue 64-bit distribution in a z/VM guest machine. For installation in a LPAR, a few different steps are necessary at the beginning of the installation. For information on how to create a IPL tape or install from FTP or HMC for LPAR installations, refer to 3.4, “Booting the installation kernel in an LPAR” on page 34.

For our installation, we downloaded the Think Blue 64 distribution files onto a local FTP server, and since we used the VM FTP client to get the initial files transferred to the VM guest, we also used an FTP server for installation. (However, as previously mentioned, note that there also are CD ISO images available for download, and that the installer also supports installation via NFS or HTTP.)

In the following example, we installed Think Blue 64-bit on a z/VM guest machine, running on a z/900 with the following guest machine settings:

- Memory: 512 MB
- DASD: VM minidisks 1x 2600 cylinder (/) and 1x 100 cylinder (swap)
- Network: OSA-Express in non-QDIO mode



In these examples, it is assumed that you already have a VM guest machine with similar or sufficient capabilities.

### 15.3.1 Preparing the first IPL of the install system

After logging onto your VM guest machine, you need to transfer the initial files that are needed for the minimal RAMdisk installation system:

VMKRNL24 IMG      - the kernel image  
VMINRD24 IMG      - the initial RAMdisk  
THINK PRM          - the parameter file  
THINK EXEC         - the REXX script to IPL from the virtual reader

Example 15-1 shows how you can use the VM ftp client to get the files from the FTP server.

*Example 15-1 Transferring the initial files to VM*

---

```
ftp 9.12.9.183
...
cd /home/redbooks/linuxSystems/ThinkBlue/boot      <- the files for initial
                                                    IPL are in boot

locsite fix 80
get think.exec
get think.prm
bin
get vminrd24.bin
get vmkrnl24.bin
quit
```

---

You can use the supplied parameter file as is:

```
root=/dev/ram0 ro ip=off cio_msg=yes init=/linuxrc
```

Or you can specify more information in the parameter file, so that you don't have to type so much during the initial setup:

```
root=/dev/ram0 ro cio_msg=yes init=/linuxrc
DASD=202,203
HOST=vmlinux1.itso.ibm.com:eth0:9.12.6.98:1496
NETWORK=9.12.6.0:255.255.255.0:9.12.6.255:9.12.6.75
LCS=0x2920,0
DNS=9.12.14.7
SEARCHDNS=itso.ibm.com:ibm.com
INSTALL=default
```

These options and more are described in the README file which comes with the distribution. Also refer to 11.2, “Preparing for installation” on page 158. For this example we used the default parameter file.

The REXX script to IPL from the VM virtual reader can be used as is:

```
/* */
'cl rdr'
'purge rdr all'
'spool punch * rdr'
'PUNCH VMKRNL24 BIN A (NOH'
'PUNCH THINK PRM A (NOH'
'PUNCH VMINRD24 BIN A (NOH'
'ch rdr all keep nohold'
'i 00c'
```

### 15.3.2 Initial IPL and network setup

From your VM console, issue the command think to start the first IPL as shown in Example 15-2.

*Example 15-2 IPL from the virtual reader*

---

```
think
NO FILES PURGED
...
Linux version 2.4.3-0.4.19B00Tvrdr (root@z02.millenux.com) (gcc version 2.95.2
19991024 (release)) #1 SMP Wed May 23 14:56:38 CEST 2001
Command line is: root=/dev/ram0 ro ip=off cio_msg=yes init=/linuxrc
We are running under VM
...

Starting the S390 initrd to configure networking. Version is 0.22.
Please enter the FQDN of your new virtual machine (e.g. z01.millenux.com):
vmlinux1.itso.ibm.com
Please enter the network device you intend to use (e.g. ctc0, iucv0, eth0, tr0)
eth0
Please enter the IP address of your new virtual machine:
9.12.6.98
Please enter your netmask (e.g. 255.255.255.0):
255.255.255.0
Please enter your broadcast address:
9.12.6.255
Please enter your network address:
9.12.6.0
Please enter your default gateway:
9.12.6.75
(1) for OSA-2 with LCS or (2) for OSA-Express with QDIO/QETH
2920,0
Warning: kernel-module version mismatch
~/lib/lcs.o was compiled for kernel version 2.4.3
while this kernel is version 2.4.3-0.4.19B00Tvrdr
```

```

Starting lcs
debug: reserved 8 areas of 2 pages for debugging lcs
debug: lcs: new level 0
eth0: ip v6 supported no enabled no
eth0: multicast supported yes enabled yes

lcs: eth0 configured as follows read subchannel=0 write subchannel=1
read_devno=2920 write_devno=2921
hw_address=00:06:29:6C:CB:CE rel_adapter_no=0

lcs configured to use sw statistics,
ip checksumming of received packets is off.
autodetection is off.
configured to detect
cu_model 0x01,15 rel_adapter(s)
cu_model 0x08,15 rel_adapter(s)
cu_model 0x60,1 rel_adapter(s)
cu_model 0x1F,15 rel_adapter(s)
SIOCADDRT: No such device
eth0      Link encap:Ethernet  HWaddr 00:06:29:6C:CB:CE
          inet addr:9.12.6.98  Bcast:9.12.6.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16176  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0

Kernel IP routing table
Destination    Gateway         Genmask         Flags Metric Ref    Use Iface
9.12.6.0       0.0.0.0         255.255.255.0  U   0      0      0 eth0
0.0.0.0        9.12.6.75       0.0.0.0         UG  0      0      0 eth0
Starting portmap.

Starting telnetd to allow login over the network.

Please telnet now to 9.12.6.98 and start 'tbsetup!'
Bringing up debugging shell in 3270...
/bin/sh: No controlling tty (open /dev/tty: No such device or address)
/bin/sh: warning: won't have full job control
#

```

---

At this point the RAMdisk installation system is running and ready to telnet into.

### 15.3.3 Installation with tbsetup

1. Telnet into the running install system and start the install script:

**Note:** There is no password protection at this point!

```
Linux 2.4.3-0.4.19B00Tvrdr ((none)) (tty0)
```

```
Welcome to Think Blue 64 Linux for zSeries.
```

```
Please run 'tbsetup' to start the installation.
```

```
# tbsetup
```

2. Specify which DASD to use and press **Enter**. You have to wait for the DASD module load, and then the menu driven interface starts:

```
Debug: this machine has IP = 9.12.6.98 and device = eth0
```

```
If you know the address range of your DASDs, please enter it now.
```

```
If you don't, just press return and autodetection will be attempted.
```

```
While autoprobng usually works, it may not give the correct results.
```

```
(200-20f is usually a good setting for vm, fd00-fd0f for LPARs)
```

```
202,203
```

```
Loading DASD kernel module. This can take a while.
```

3. The first screen gives you the list of DASD that are now available for Linux. Choose the DASD you want to format by selecting it and pressing **OK**. You have to wait until the formatting is finished, which can take a while if you have large volumes.
4. After finishing the DASD format, you have to specify the mount points like /, /usr or /home, as shown in Figure 15-1 on page 247. If you want to set up a volume as swap device, just put in swap instead of a mount point.  
Press **OK** when you have assigned all volumes.

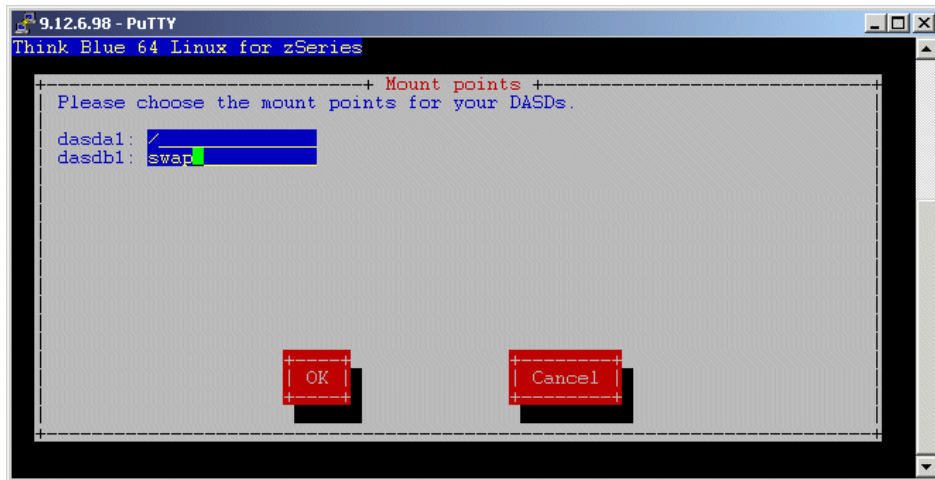


Figure 15-1 Think Blue 64 installer mount point selection

5. The next screen asks for the network settings such as the host name (fully-qualified), the domain search list, and the IP address of your DNS server. After filling in all the information, press **OK** to step further.
6. This screen asks for the RPM server that holds the Think Blue distribution files. Depending on your configuration, you either have to specify an FTP, an HTTP or an NFS server address. In any case you have to specify the full path to the Think Blue directory (see Figure 15-2 on page 248).

The installer automatically appends ThinkBlue/base to the path when it is looking for the package list on the server. In our example we used an FTP server with user authentication where the syntax is:  
ftp://user:password@host/

Press **OK** to start the installation.

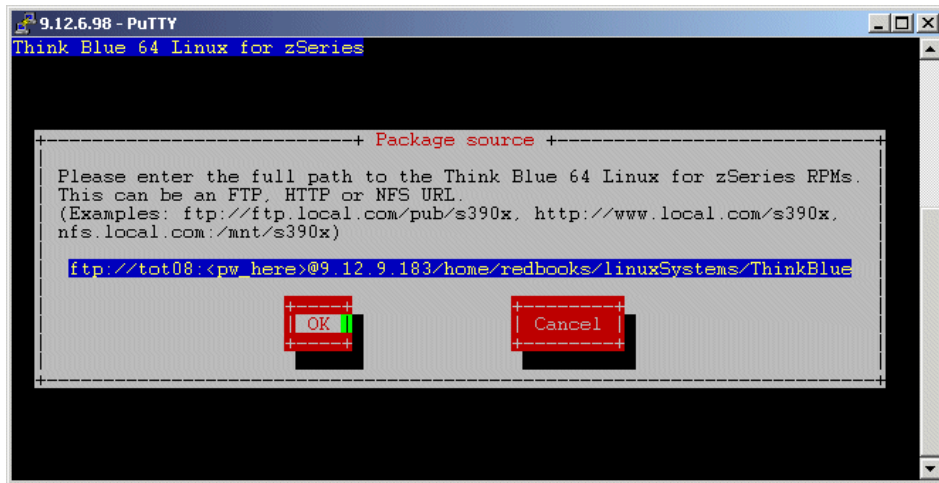


Figure 15-2 Think Blue 64 installer specify ftp server

7. Now the installer automatically receives the rpm packages from the FTP server, configures and installs them. This can take while, depending on the network speed.
8. At the end of the install process, the installer asks for a new root password. When supplying the password, the installation is finished. You should review the output of the **zilo** command, by scrolling up a little in your telnet session. (On most telnet clients, this works with [Shift]+[Pg-Up].)

You can now close the telnet session and IPL from DASD. Since the installer has automatically unmounted all DASD volumes, a clean shutdown is not needed at this point.

```
#CP IPL 202 CLEAR
```

```
Linux version 2.4.3-0.4.19vrdr (root@z02.millinux.com) (gcc version 2.95.2
19991024 (release)) #1 SMP Wed May 23 14:52:32 CEST 2001
Command line is: root=/dev/dasda1 ro dasd=202,203 noinitrd
```

```
We are running under VM
```

```
...
```

The system should boot up to the point where it presents a Linux login at the VM console. To read the boot messages at your leisure, press **Ctrl** to prevent a page update.

**Note:** The telnet daemon on Think Blue is disabled by default, and only ssh connections are allowed. The reason for this is because telnet is insecure and there are many ssh clients for all operating systems available (like PuTTY for Windows).

However, if for some reason you can only use telnet, you can enable the telnet daemon from the VM console with the following commands:

```
chkconfig telnet on
service xinetd restart
useradd dummy
passwd dummy
```

You can then login via telnet as user dummy and then also su to root.

## 15.4 System configuration and administration

In this section we discuss the basic tasks needed to configure and administrate your Think Blue system. Since the Think Blue distribution is Red Hat-based, the administrative tasks and tools are equivalent to Red Hat. So, in most cases you should also refer to Chapter 12, “Customizing and using Red Hat Linux” on page 177.

### 15.4.1 File system layout

Here are the locations of some important files and directories in the Think Blue distribution:

so that you

File	Description
<i>/etc/samba/smb.conf</i>	Configuration of smb (Samba) daemon
<i>/etc/httpd/conf/httpd.conf</i>	Web server configuration file
<i>/etc/modules.conf</i>	Aliases and options for loadable kernel modules
<i>/etc/fstab</i>	Filesystems mounted or available for mounting
<i>/etc/group</i>	Group information
<i>/etc/hosts</i>	Map of IP numbers to hostnames
<i>/etc/hosts.allow</i>	Hosts allowed to access Internet services

<i>/etc/hosts.deny</i>	Hosts forbidden to access Internet services
<i>/etc/xinetd.d</i>	Configuration file directory for the xinetd daemon, which controls access to Internet services
<i>/etc/inittab</i>	Configuration for the init daemon, which controls executing processes
<i>/etc/login.defs</i>	Options for useradd and related commands
<i>/etc/rc.d/rc.sysinit</i>	Main startup script
<i>/etc/rc.d/init.d</i>	Scripts for system and process startup and shutdown
<i>/etc/rc.d/rc*.d</i>	Symbolic links to the startup scripts
<i>/etc/rc.local</i>	Scripts for execution after init scripts
<i>/etc/skel</i>	Skeleton files used to establish new user accounts
<i>/etc/sysconfig</i>	Directory with configuration files for system services and facilities
<i>/var/log/httpd/access_log</i>	Log of Web server access
<i>/var/log/httpd/error_log</i>	Log of Web server errors
<i>/var/log/messages</i>	System log
<i>/var/spool/cron</i>	Directory for cron configuration files

## 15.4.2 Configure services/daemons

To enable and disable services like HTTP or NFS for different run levels, system boot, you can use one of the different tools like **chkconfig**, **setup** or **tksysv** that are provided in the Think Blue distribution. Some service daemons are started as standalone servers and have start/stop scripts in */etc/rc.d/init.d*, which are referenced by symbolic links in */etc/rc.d/rc<#>.d*.

Other services, like *telnetd*, get invoked by the *xinetd*, where the configuration files reside in */etc/xinetd.d/*. For example, you can check for which run levels the Apache server is enabled by using **chkconfig**:

```
# chkconfig --list httpd
```



```
httpd          0:off  1:off  2:off  3:on   4:on   5:on   6:off
```

To disable the Web server in run level 3, you can do the following:

```
# chkconfig --level 3 httpd off
```

Refer to the man page for `chkconfig` for further information about the options. Alternatively, you can use the `setup` (or `ntsysv`) command to enable or disable system services, as shown in Figure 15-3.

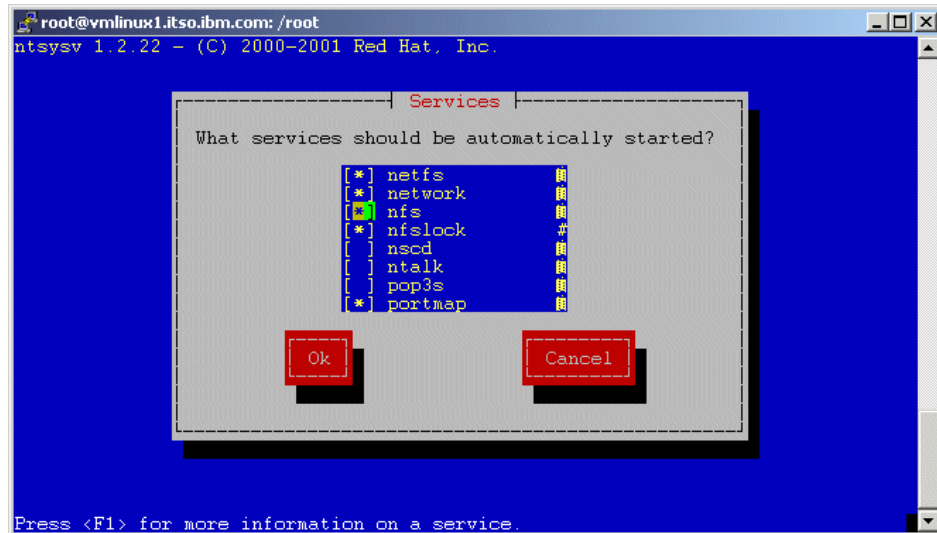


Figure 15-3 Setting up system services

### 15.4.3 Linuxconf

The Think Blue distribution includes the system administration tool `linuxconf`. The approach of `linuxconf` is to provide one single administrator interface for common tasks and services. There are many modules available which can be plugged into and used within `linuxconf` or can be used standalone. However, since the development is done on PC-based systems, some of the modules may not work correctly or (even if included) may not be applicable for a S/390 or zSeries system. Examples are the boot or the PPP dialup configuration.

**Note:** Because of these problems and the testing that still had to be done, `linuxconf` was not included by the Think Blue developers in the default installation. The example settings in this chapter can also be done without using `linuxconf`.

Since it is not included in the default installation, you would have to install the rpm package `linuxconf-1.24r2-11.s390x.rpm` afterwards if you want to use it. If you want to use a graphical interface, you can install the `gnome-linuxconf-0.64-1.s390x.rpm` package.

For additional information about installation and configuration of `linuxconf`, also refer to 12.2, “Installing `linuxconf`” on page 178.

The default settings for `linuxconf` do not include many modules. You can add modules under **Control->Control files and systems->Configure `linuxconf` modules**. In order to illustrate the capabilities of `linuxconf`, some of the examples in this chapter show the usage of different modules.

## 15.4.4 Network setup

The network configuration is stored in different files on the Think Blue system.

The basic settings about your system are in `/etc/sysconfig/network`:

```
# cat /etc/sysconfig/network
NETWORKING=yes
HOSTNAME=vmlinux1.itso.ibm.com
GATEWAY=9.12.6.75
```

The host name and aliases are stored in the file `/etc/hosts`. Information about DNS servers and search domains are stored in the file `/etc/resolv.conf`:

```
# cat /etc/resolv.conf
search itso.ibm.com
nameserver 9.12.14.7
```

The configuration for your network interfaces is stored in `/etc/sysconfig/network-scripts`. In our example, where we used a OSA adapter, it looks like this:

```
# cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
BOOTPROTO=static
IPADDR=9.12.6.98
NETMASK=255.255.255.0
NETWORK=9.12.6.0
ONBOOT=yes
```

You can edit these files to change the network configuration. After any changes, you would have to reload the configuration or restart the network:

```
# /etc/rc.d/init.d/network restart
```

As an alternative, you can also use graphical (menu-driven) tools like **netcfg** or the linuxconf module **netconf**. To configure the network with linuxconf, start the program and go to **Config->Networking->Client tasks**. Or you can start the netconf module directly and get a configuration dialog as shown in Figure 15-4.

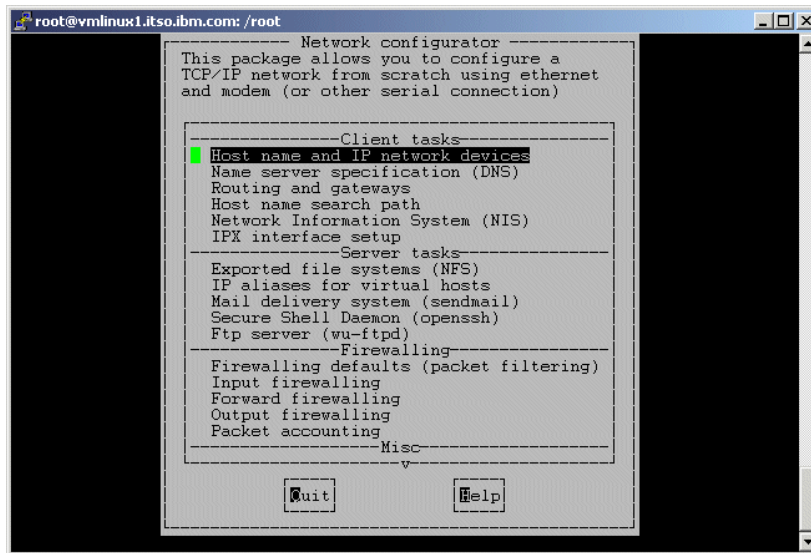


Figure 15-4 netconf dialog

There you can select the different settings and make the changes, as shown in Figure 15-5 on page 254.

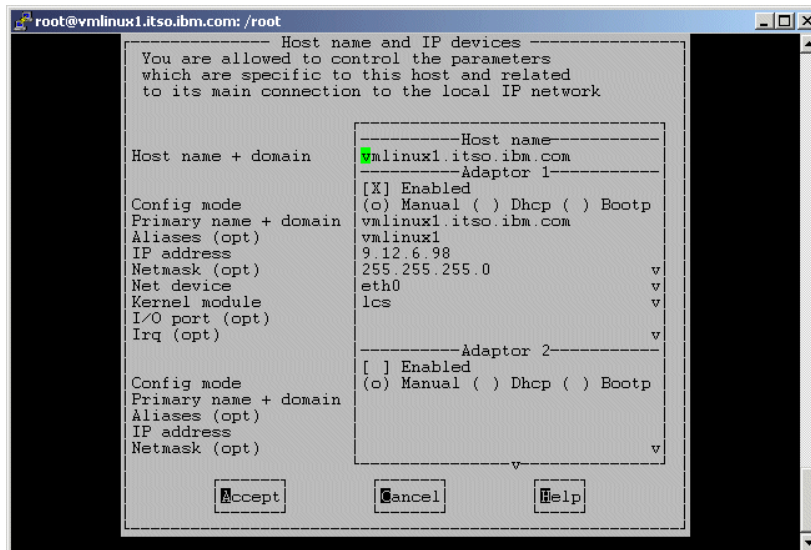


Figure 15-5 netconf host name and IP setup

## 15.4.5 Managing DASD

The procedure for adding DASD volumes is similar to the other distributions, except that you have to add the DASD device numbers in the `zilo` configuration file `/etc/zilo.conf`. When the `zilo` command is run, it reads the `/etc/zilo.conf` file and actually generates a new parameter file in `/boot`; therefore, any changes you made there get overwritten. To add a new DASD, you add an entry in `/etc/zilo.conf`:

```
# cat /etc/zilo.conf
append="dasd=202,203,207 noinitrd"
ipldevice=/dev/dasda
image=/boot/image
root=/dev/dasda1
readonly
```

After that, you have to run `zilo`. The new device can be formatted and mounted only after a reboot.

Think Blue 64 includes the Logical Volume Manager (LVM), so that you can set up volume groups and logical volumes out of a pool of physical volumes. Refer to Chapter 17, “Logical Volume Manager” on page 301 for further information.

## 15.4.6 Managing users

In the Think Blue distribution you have several command line tools for adding and deleting users and groups:

```
useradd
userdel
adduser
groupadd
groupdel
```

To add a normal user and automatically create a home directory, you would use:

```
# useradd ron
# passwd ron
```

To delete this user and his home directory issue:

```
# userdel -r ron
```

For additional information, refer to the man pages and to Chapter 22, “Overview of security on Linux” on page 435.

As an alternative, you could also use `linuxconf` to manage user accounts and groups. To do that, start `linuxconf` and go to **Config->Users accounts->Normal**. Or you can start the `userconf` module directly and get a dialog as shown in Figure 15-6.

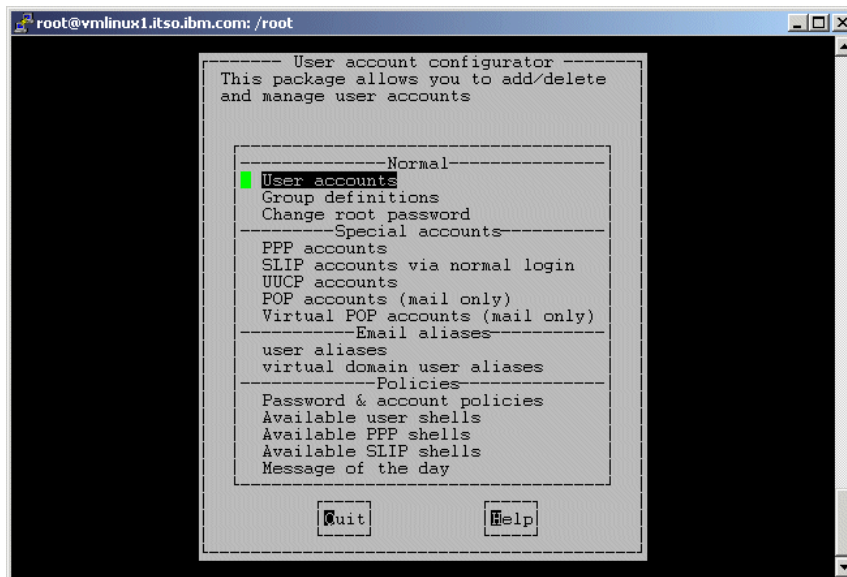


Figure 15-6 `userconf` dialog

To add a new user, go to **User accounts->Add** and you will get a configuration dialog, as shown in Figure 15-7.

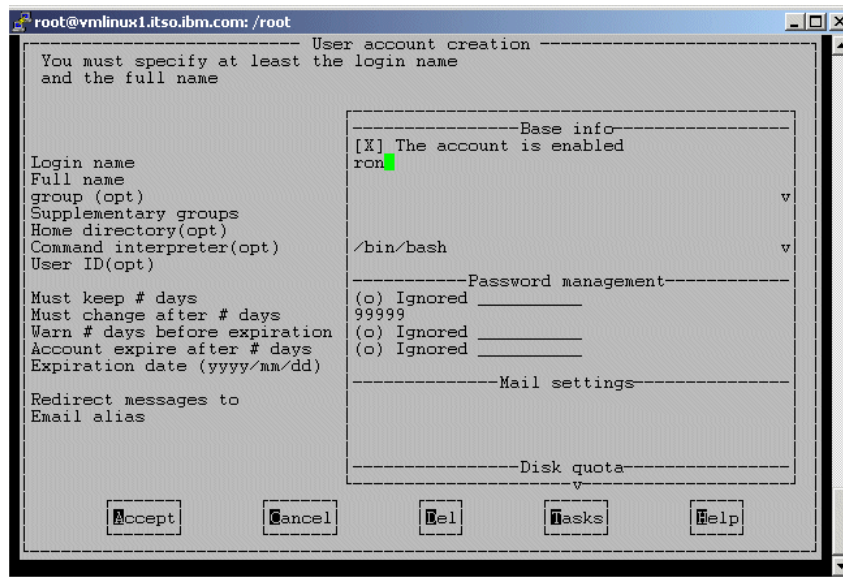


Figure 15-7 userconf user creation dialog

**Note:** On the Think Blue system (as on Red Hat), the default settings are that each user is created with its own group if no special group is specified. This schema has advantages in assigning permissions when sharing files or directories between users and groups. However, you can change these defaults in userconf Password and account policies.

## 15.4.7 Configuring the mail server

Think Blue uses **sendmail** as the Mail Transfer Agent (MTA). The main configuration file for sendmail is `/etc/sendmail.cf`. To be able to send and receive mail with your Think Blue Linux system, you can configure sendmail to use an external mail server.

**Note:** In order to send and receive e-mails using a external mail host, this external mail host needs to be configured properly and your Linux system needs to have a DNS entry in your DNS server

To do this, we provide a very basic example using the **m4** macro processor to generate the sendmail configuration file. First create a backup of the original sendmail.cf file:

```
# cp /etc/sendmail.cf /etc/sendmail.cf.orig
```

Then create the input configuration file for m4, with your mail server as smart host:

```
# cat /etc/mail/mymailcfg.mc
include(`~/usr/share/sendmail-cf/m4/cf.m4')
OSTYPE(`linux')
define(`SMART_HOST', `esmtplib.com')
MAILER(local)
MAILER(smtp)
```

Then process the input file with m4 to generate the new sendmail configuration file:

```
# m4 /etc/mail/mymailcfg.mc > /etc/sendmail.cf
```

After the changes, restart sendmail:

```
# /etc/rc.d/init.d/sendmail restart
or by using the service command (which works for all services)
# service sendmail restart
```

Now you should be able to send and receive mail with your Think Blue Linux system, using your favorite e-mail client. Refer to the sendmail documentation and man pages to find out more about the different options you can specify using m4 macros.

Alternatively, you can configure sendmail using linuxconf under **Config->Networking->Server tasks->Mail delivery system->Basic**. Or you can use the mailconf module directly and get a configuration dialog as shown in Figure 15-8 on page 258.

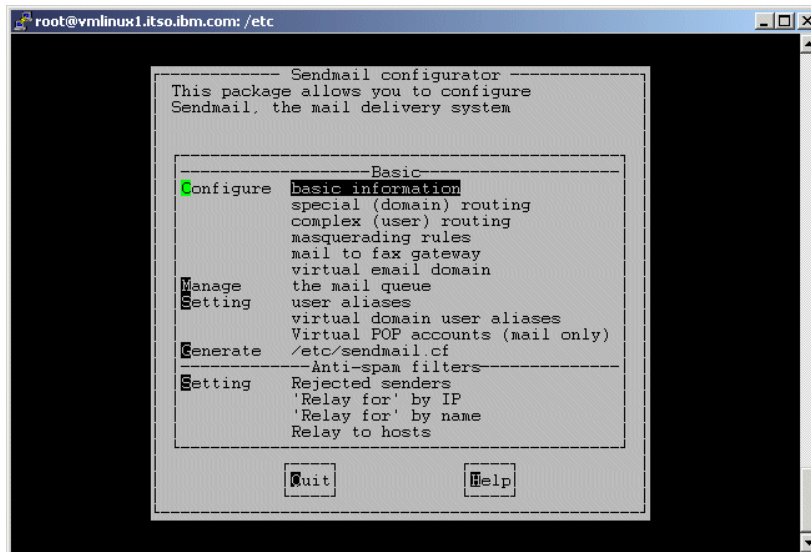


Figure 15-8 mailconf dialog

There you can configure the basic settings, as shown in Figure 15-9.

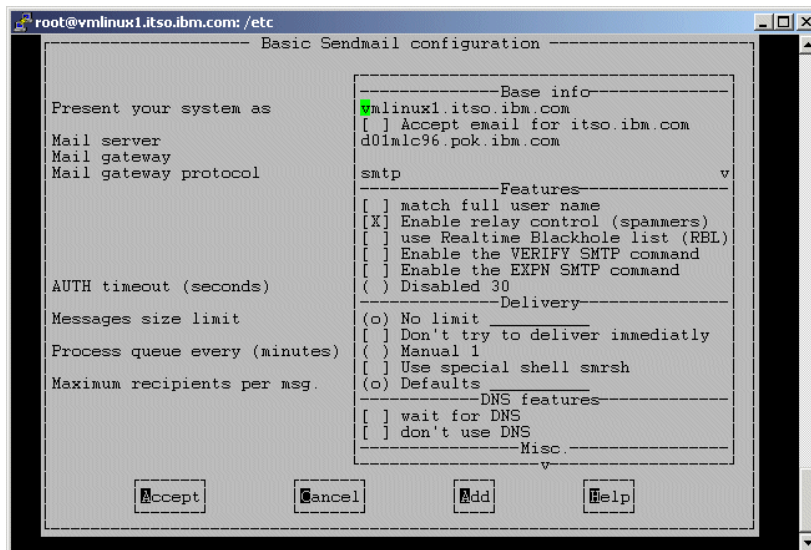


Figure 15-9 mailconf basic sendmail settings



**Note:** To configure your system as a mail server for other systems, more and different steps are needed; refer to the sendmail documentation and man pages for more detail.

## 15.4.8 Using Apache

The Apache Web server is included in the default installation of Think Blue. The main configuration file is `/etc/httpd/conf/httpd.conf`. You can edit this file directly to configure Apache. The start/stop script is `/etc/rc.d/init.d/httpd`.

For more information, please refer to the documentation and man pages.

## 15.4.9 Using FTP

The Think Blue distribution includes the `wu-ftp` (Washington University FTP) package in the default installation. The configuration files are located in `/etc`:

```
# ls /etc/*ftp*
ftpaccess  ftpconversions  ftpgroups  ftphosts  ftpusers
```

For information about the syntax and use of the configuration files, refer to the man pages. To enable anonymous FTP, install the `anonftp-4.0-4.s390x.rpm` package.

Within `linuxconf`, there is also a module where you can configure `wu-ftp`. Start `linuxconf` and go to **Config->Networking->Server tasks->FTP server**. Or start the `netconf` module directly, go to **FTP server->Basic configuration**, and you'll see a configuration dialog as shown in Figure 15-10 on page 260.

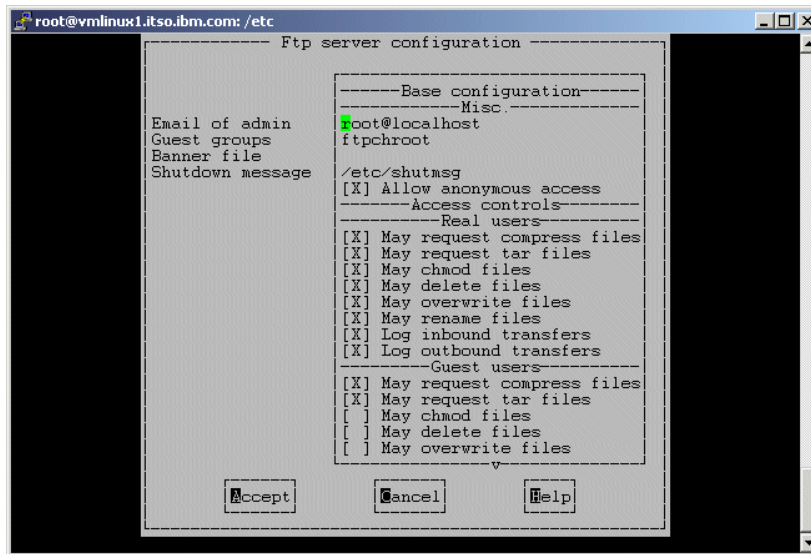


Figure 15-10 netconf FTP server configuration

## 15.4.10 Using Samba

The Samba package is included in the default installation of Think Blue. The main configuration file is `/etc/samba/smb.conf`. The Web-based graphical administration interface SWAT is included in the distribution, but not installed by default. To use it, you have to install the `samba-swat-2.0.8-1.7.1.s390x.rpm` package.

For more information on how to configure and use Samba, refer to the documentation and man pages.

## 15.4.11 RPM

Since the Think Blue distribution is Red Hat-based, it uses `rpm` for package management. The version used in this distribution is 4.0.2. Extensive documentation and man pages are available if you need further information. The naming convention for rpm packages is:

```
name-version-release.architecture.rpm
```

Following are some examples for very basic commands. To install, upgrade, or freshen packages, use the following commands:

```
rpm -ivh <package>.rpm
rpm -Uvh ftp://linux.zseries.org/pub/ThinkBlue64/RPMS/s390x/<package>.rpm
```

```
rpm -Fvh http://linux.zseries.org/download/ftp/RPMS/s390x/<package>.rpm
```

To query all packages, or query the information or signature of a single package, use the following commands:

```
rpm -qa                list all installed packages
rpm -qip <package>.rpm  display package information
rpm --checksig <package>.rpm  check the signature/md5sum
```

To build a package from source, use the following command:

```
rpm --rebuild <package>.src.rpm  build from a source package
```

The man page contains detailed explanations of all options for the **rpm** command. There are also various graphical front-end tools available, such as **gnorpm** and **kpackage**.

## 15.4.12 Tape support

Think Blue 64 comes with the tape driver for channel-attached tape drives that are compatible with IBM 3480 or IBM 3490 magnetic tape subsystems. Refer to Chapter 23, “Backup and restore” on page 463 for more information about how to configure and use the tape devices.

## 15.4.13 Linux 2.4 kernel specific enhancements

### 64-bit support

The Think Blue 64 distribution fully supports the 64-bit architecture of the zSeries hardware.

### The /proc file system

The /proc file system contains additional sections with information compared to the 2.2 kernel. We do not cover all new directories here; instead, you should browse and explore the /proc file system yourself. Example 15-3 shows the section with statistical information about DASD usage.

*Example 15-3 DASD usage statistics*

---

```
# cat /proc/dasd/statistics
43659 dasd I/O requests
<4  8 16 32 64 128 256 512 1k 2k 4k 8k 16k 32k 64k 128k
256 512 1M 2M 4M 8M 16M 32M 64M 128M 256M 512M 1G 2G 4G 8G
Histogram of sizes (512B secs)
 0 5457 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Histogram of I/O times
 0 0 0 0 0 0 0 0 0 336 238 283 258 1314 406 681 123
 44 63 100 160 248 316 384 497 0 0 0 0 0 0 0 0 0
Histogram of I/O times per sector
 0 0 0 0 0 336 238 283 258 1314 406 681 123 44 63 100
160 248 316 384 497 0 0 0 0 0 0 0 0 0 0 0 0
Histogram of I/O time till ssch
```

```

1298 102 194 21 23 8 22 9 162 162 155 161 361 512 395 92
32 53 94 157 242 315 379 496 0 0 0 0 0 0 0 0
Histogram of I/O time between ssch and irq
0 0 0 0 0 0 0 0 839 799 317 269 1288 303 1466 166
5 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Histogram of I/O time between ssch and irq per sector
0 0 0 0 0 839 799 317 269 1288 303 1466 166 5 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Histogram of I/O time between irq and end
1055 550 24 1257 324 234 275 1102 142 102 22 17 23 45 75 108
44 29 18 2 0 0 0 0 0 0 0 0 0 0 0 0

```

---

For example, you can use the data for creating graphs about the statistical usage. We provide a very brief example CGI script here, which makes use of the **gnuplot** tool; the script is listed in Example 15-4.

Note that there are no tests for file existence or similar tests; in our example, we simply assume that we can write our temporary files to /tmp.

*Example 15-4 dasdusage cgi script*

---

```

# cat dasdusage
#!/bin/sh

cat /proc/dasd/statistics > /tmp/du.stat
dem="/tmp/du.dem";dat="/tmp/du.dat";stat="/tmp/du.stat"
host=`uname -n`

echo "#" > $dat
tmp=`sed -n -e '7,8p' $stat`
for tmp2 in $tmp
do
    echo $tmp2 >> $dat
done

echo "set title \"DASD usage statistics for $host\"" > $dem
echo "set terminal png color" >> $dem
echo "set xtics (\<4\" 1, \"16\" 3, \"64\" 5, \"256\" 7, \"1k\" 9, \"4k\"
11, \"16k\" 13, \"64k\" 15, \"256\" 17, \"1M\" 19, \"4M\" 21, \"16M\" 23,
\"64M\" 25, \"256M\" 27, \"1G\" 29, \"4G\" 31)" >> $dem
echo "set data style histeps" >> $dem
echo "set size 1,0.5" >> $dem
echo "plot '/tmp/du.dat' t \"Histogram of I/O times\"" >> $dem

echo Content-type: image/png
echo
gnuplot $dem

```

---

After creating and editing the file, make it executable and copy it into the Web server's CGI directory and make it executable.

```
# chmod +x dasdusage
# cp dasdusage /var/www/cgi-bin/
```

After that, you connect with your browser ; you should see a graphic like Figure 15-11.

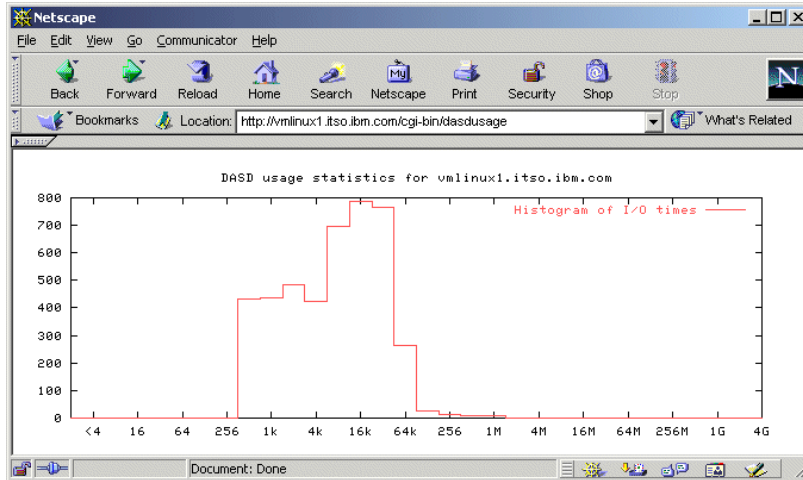


Figure 15-11 dasdusage plot

## The channel device layer

The channel device layer provides a consistent interface for configuration and default machine check handling (devices appearing and disappearing) channel devices. The current setup can be read from the file `/proc/chandev` as shown in Figure 15-5. The configuration file is `/etc/chandev.conf`.

*Example 15-5 Channel device information in /proc*

```
# cat /proc/chandev

chan_type key bitfield ctc=0x1,escon=0x2,lcs=0x4,osad=0x8,qeth=0x10,claw=0x20

*'s for cu/dev type/models indicate don't cares

use_devno_names: off

Channels enabled for detection
chan  cu  cu  dev  dev  max  auto  recovery
type  type model type model port_no.  type
-----
0x08  0x3088  0x62  *  *  0  not_operational,no_path,revalidate,device_gone
0x10  0x1731  0x01  0x1732  0x01  15  not_operational,no_path,revalidate,device_gone
0x04  0x3088  0x60  *  *  1  not_operational,no_path,revalidate,device_gone
0x06  0x3088  0x1f  *  *  15  not_operational,no_path,revalidate,device_gone
0x05  0x3088  0x08  *  *  15  not_operational,no_path,revalidate,device_gone
0x04  0x3088  0x01  *  *  15  not_operational,no_path,revalidate,device_gone

channels detected
      chan  cu  cu  dev  dev  in  chandev
```

irq	devno	type	type	model	type	model	use	reg.
0x0000	0x2920	0x04	0x3088	0x60	0x0000	0x00	yes	no
0x0001	0x2921	0x04	0x3088	0x60	0x0000	0x00	yes	no

---

## Large UIDs

With the 2.4 Linux kernel, you can have user IDs and group IDs greater than 65534. Following is an example of adding a user with a large UID.

```
# useradd -u 123456789 ron
# passwd ron
...
# ssh -l ron localhost
ron@localhost's password:
[ron@vmlinux1 ron]$ id
uid=123456789(ron) gid=123456789(ron) groups=123456789(ron)
[ron@vmlinux1 ron]$ exit
```

## Files greater than 2 GB

With the 2.4 kernel, the (theoretical) file size limit has shifted into the terabyte arena (as compared to the 2.2 kernel, where you could only have a maximum of 2 GB files on an ext2 file system).

```
# dd if=/dev/zero of=big_zero count=3000 bs=1M
3000+0 records in
3000+0 records out
# ls -lh
total 3.0G
-rw-r--r--  1 root    root      2.9G Jun  5 11:54 big_zero
drwxr-xr-x  2 root    root      16k Jun  5 11:27 lost+found
```

## Other topics

In this part of the book we discuss topics outside of specific distributions, but pertinent to running Linux on the mainframe. The topics presented are as follows:

- ▶ Virtual Image Facility (VIF)
- ▶ File systems
- ▶ Managing DASD
- ▶ Logical Volume Manager (LVM)
- ▶ High Availability
- ▶ Debugging
- ▶ LDAP
- ▶ Systems Management
- ▶ Security
- ▶ Backup and restore
- ▶ DB2
- ▶ WebSphere Application Server







## VIF

In this chapter we discuss the Virtual Image Facility for Linux (VIF) mainly using SuSE Linux. Any flavor of Linux that runs on S/390 should be supported, though besides SuSE, we only tested Turbolinux and Red Hat. You can find additional related information in 11.5, “Installation of Red Hat under VIF” on page 164 and 8.4, “Installation of Turbolinux under VIF” on page 129.

## 16.1 Overview of VIF

The following description is copied from the VIF Web site:

<http://www.ibm.com/servers/eserver/zseries/os/linux/vif/>

The S/390 Virtual Image Facility enables you to run tens to hundreds of Linux server images on a single S/390 server. It is ideally suited for those who want to move Linux and/or UNIX workloads deployed on multiple servers onto a single S/390 server, while maintaining the same number of distinct server images.

This provides centralized management and operation of the multiple image environment, reducing complexity, easing administration and lowering costs.

Deploying Linux workloads on Virtual Image Facility is particularly attractive if the workload interacts with S/390 servers, applications, or data located on the same S/390 server.

Figure 16-1 shows a block diagram of VIF. It is important to understand the need for two IP addresses; the VIF hypervisor can be thought of as a built-in TCP/IP stack, while the *master Linux* is the interface used to issue commands to VIF.

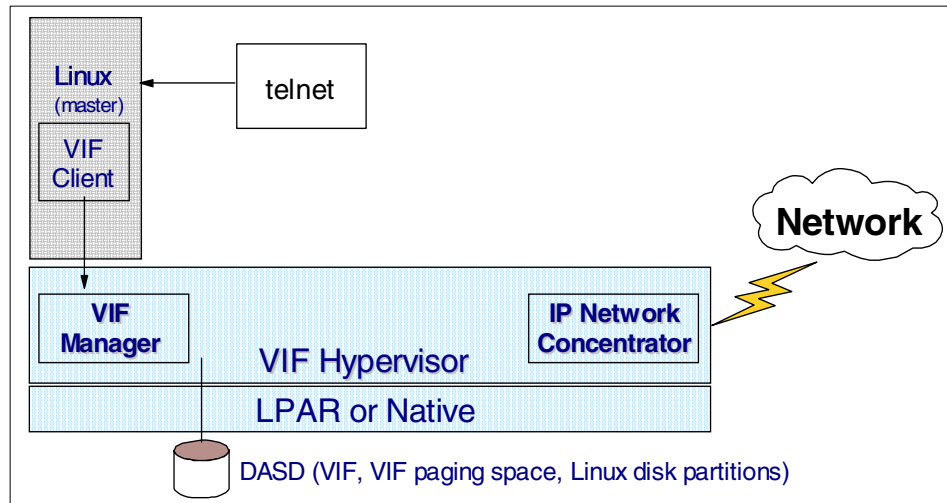


Figure 16-1 VIF block diagram

The base documentation for VIF is *S/390 Virtual Image Facility for LINUX; Guide and Reference*, GC24-5930; a link can be found at the bottom of the VIF Web site.

The -04 level of that publication was available at the time of writing this redbook, and some of the information is repeated here for convenience; this chapter should augment that book.

## Terminology

First we'll discuss the terminology used by VIF, because some of the terms align more closely with Linux than with traditional S/390 parlance.

A VIF *partition* is a piece of a disk. This is based on the PC term used to describe how hard drives are portioned off. In S/390, terms it can be described as DASD or a VM minidisk. Typically, in S/390 terms, a partition refers to a logical partition, or a "slice of a mainframe".

A VIF *image* is a virtual Linux server. The term image is used for many different meanings, but in general it is used to mean an electronic version of a picture.

The VIF *hypervisor* makes hardware virtual and manages the assignment of it to each VIF image.

VIF *paging* is analogous to Linux swap space. Though Linux uses the term swap, it is more correctly the paging of virtual memory between physical memory and other media.

Also, the term *master Linux* is used to designate a special Linux image that has the VIF client. This is the only Linux image that can communicate with, and thus control, the VIF hypervisor.

## 16.2 Planning for VIF

VIF requires the following resources:

- ▶ A G5 processor or later or Multiprise 3000 (typically an LPAR on such a processor is used), and access to the HMC
- ▶ A 3390-3 DASD volume for the VIF install
- ▶ The VIF installation tape and a 3480 or 3490 tape drive
- ▶ One or two token-ring or Ethernet ports on an OSA, IBM 3172, or equivalent device operating in LCS mode, connected to a LAN (OSAs in QDIO mode are not supported)
- ▶ An IP address and associated network info for the VIF hypervisor
- ▶ A second IP address used by the master Linux image
- ▶ A Linux distribution and the IP address of an FTP server with access to it

- ▶ The user name password for the FTP server
- ▶ The file path of the VIF install file

VIF can be used with Integrated Facility for Linux (IFL) engines. These are CPUs for G5 or later processors which can either be turned on for existing machines, or purchased with new machines. IFL engines enable an installation to purchase additional processing capacity, exclusively for Linux workloads, with no additional software charges for software running on the other existing processors. When IFL engines are used in an LPAR, it is rendered with a penguin icon on the HMC. This icon is shown on the right side of Figure 16-2 on page 283.

**Important:** VIF currently does not support tape drives because there is way to assign a drive to a VIF image.



Figure 16-2 HMC icons for standard and IFL engines

There are other, more abstract considerations when planning for VIF. We address some of these in the next two sections.

## 16.2.1 VIF networking options

VIF can do two types of networking, which are referred to as *internal* or *external*. Internal can be thought of as virtual networking. It is based on the interuser communication vehicle (IUCV), which is a point-to-point protocol. This allows multiple Linux images to share a single network interface. It uses high-speed cross-memory data transfers and thus does not require virtual device I/O functions.

Internal or IUCV networking requires that the Linux images be on a different subnet than VIF (often referred to as *the hypervisor*). This is because point-to-point networking is used and therefore the Linux images are not truly on the LAN. Therefore, TCP/IP packets destined for the Linux images must be routed through the VIF hypervisor. It is also why, when using the IUCV driver as a network interface, a subnet mask of 255.255.255.255 is necessary. With IUCV networking there are limitations due to its point-to-point nature. For example, you cannot do DHCP, Ethernet bridging or proxy ARP.

External networking requires a physical network connection. OSA cards can share a number of physical connections, so if the number of Linux images is relatively small, an external network is viable and recommended over IUCV due to the previous limitations. See 2.2, “Planning networking” on page 15 for information on OSA cards.

## 16.2.2 VIF partitions and paging space

For each Linux image, VIF requires space on DASD for partitions (disks) and for paging. This allows real storage to be shared among the Linux images. The amount of space for partitions depends on the amount of read/write space each Linux image requires and how much can reasonably be shared read-only. The amount of paging space depends on the virtual memory assigned to each Linux partition. There is about two and a half times as much VIF paging space as virtual memory set aside for each Linux image. For example, if each Linux image is given the default value of 64 MB of memory, then VIF will allocate approximately 160 MB of paging space.

## 16.3 Installation of VIF with Turbolinux

With Turbolinux 6.0, we were able to install VIF but only bring up a master Linux. It is documented that this release does not work with the IUCV driver, so an external network (direct OSA connection) was used. After VIF and the master Linux were installed, we could create additional Linux images, but could not get a response from them when telnetting to the hypervisor.

We were able to get VIF working with the beta code drop of Turbolinux 6.5. It required that the storage of the newly created VIF image be set to 128 MB via the command `vif image set 'name' storage 128`. For more details, see 8.4, “Installation of Turbolinux under VIF” on page 129.

## 16.4 Installation of VIF with Red Hat

The install of Red Hat with VIF worked fine. See 11.5, “Installation of Red Hat under VIF” on page 164 for a few more details.

## 16.5 Installation of VIF with SuSE Linux

This section discusses installing VIF with SuSE Linux for S/390, which is documented with much detail. For all distributions we worked with VIF V1 R1 M1, service level 0500 which became generally available June 1, 2001.

The VIF install with SuSE Linux for S/390 is tricky. The main reason for this is that you're effectively installing two different products (VIF, and Linux for S/390) from two different vendors. It just happens that the first product requires at least one flavor of the second product.

For convenience, a high-level checklist of the install steps follow; the step number is the same as the section suffixes after the list. Steps 5 and 6 documented in this chapter are specific to SuSE and will differ among distributions. In general, all other steps are common to VIF.

1. \_\_ Prepare an FTP server with the VIF install file and Linux.
2. \_\_ Format a clean S/390 3390-3 DASD.
3. \_\_ Load VIF tape onto the DASD.
4. \_\_ IPL the DASD and answer VIF questions.
5. \_\_ Begin master Linux installation from HMC.
6. \_\_ Telnet to master Linux and finish installation.
7. \_\_ Reset master Linux boot partition and reboot.

### **16.5.1 Prepare an FTP server with the VIF install file and Linux**

An integral part of the VIF install is accessing a VIF file and many Linux files via FTP. Only one file is needed by VIF: the VIF install file, which points to a Linux kernel, a RAMdisk, and a parameter file. Example 16-3 shows the relationship of the VIF install file to the Linux initialization files. In this section we discuss how to do the following tasks:

- ▶ Prepare an FTP server
- ▶ Prepare the VIF install file
- ▶ Prepare a Linux distribution

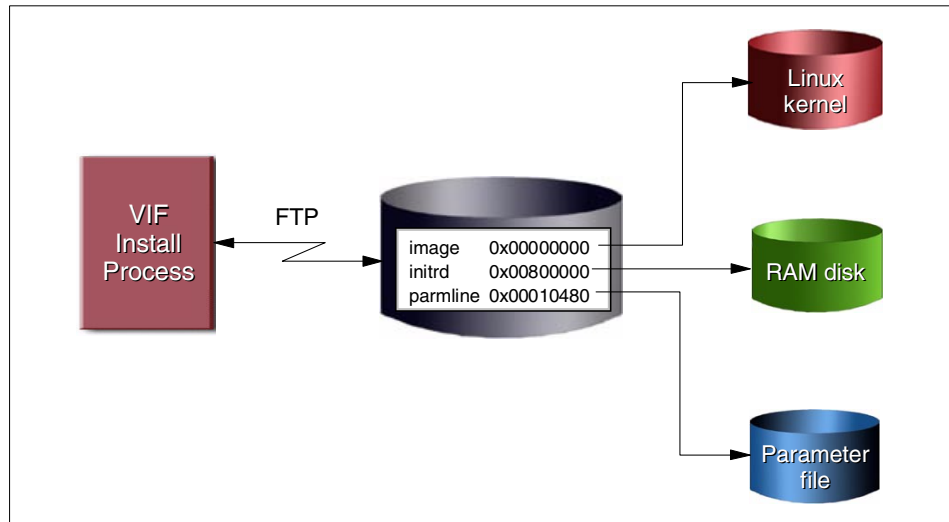


Figure 16-3 VIF install file relationship to Linux

## Prepare an FTP server

When installing on traditional mainframe hardware, a PC server running Linux works fine. Many Linux distributions don't allow root and other user IDs to access FTP via inclusion of the disallowed user IDs in the `/etc/ftpusers` file. Simply using a conventional user ID is recommended.

When installing VIF on a Multiprise 3000, we were able to use an OS/2 FTP server, but only after we copied the three critical files from CD to disk. VIF seems to reject the colon (`:`) character as part of a valid FTP path. However, this occurred while using an early release of VIF (0300), and may now be fixed.

## Prepare the VIF install file

If you are using the SuSE distribution and VIF 0500 or later, you can use one of the VIF install files included on CD1 named `suse.ins` or `susev.ins`:

```
# cat suse.ins
* SuSE Linux for S/390 Installation/Rescue System (default)
suse/images/tapeipl.ikr 0x00000000
suse/images/initrd 0x00800000
suse/images/parmfile 0x00010480
# cat susev.ins
* SuSE Linux for S/390 Installation/Rescue System with IUCV
suse/images/tapeipl.ikr 0x00000000
suse/images/initrd 0x00800000
suse/images/parmfile.v 0x00010480
```

These files tell VIF where to load Linux into memory. The only difference between these two files is that `suse.ins` points to the file named `parmfile`, and `susev.ins` points to the file named `parmfile.v`, which have the following differences:

```
# cat suse/images/parmfile
ramdisk_size=32768 root=/dev/ram0 ro
# cat suse/images/parmfile.v
ramdisk_size=32768 root=/dev/ram0 ro iucv=$TCPIP
```

The extra parameter `iucv=$TCPIP` is necessary when the VIF hypervisor is communicating with the Linux images via IUCV. So, if you are using an internal network, you will want to use `susev.ins` as the VIF install file name.

It seems with VIF 0400 or earlier there is a bug in which relative file names are not handled correctly. Also, other distributions may not include a VIF install file.

If either of these is the case, you should create your own VIF install file with fully qualified file names. For example, we created our own parameter file and pointed to the kernel and RAMdisk on a mounted CD:

```
# cat vifbeta.txt
/mnt/cdrom/suse/images/tapeipl.i kr 0x00000000
/mnt/cdrom/suse/images/parmfile.v 0x00010480
/mnt/cdrom/suse/images/initrd      0x00800000
```

The VIF installation file will be read during the VIF install process. It specifies where to load the Linux kernel, the parameter file, and the initial RAMdisk. Fully qualified path names are recommended.

## Prepare a Linux distribution

The GA level of Turbolinux for S/390 comes on a single CD, but the SuSE GA version comes on three. If you are doing a SuSE minimal or default install, you will only need the first CD, but for some installation types, YaST will need to get packages off multiple CDs.

If the distribution you are using occupies more than one CD, it can make installation tricky. However, there are a few ways you make them accessible:

- ▶ Mount CD1 and start the install. When packages from other CDs are needed, the install process will fail. You can then remount additional CDs (from another telnet session, perhaps) and continue with the install process. This has been tested and it does work with SuSE's YaST; however, it is a less-than-elegant solution and is not recommended.
- ▶ Mount three loopback .iso images. A desirable feature of Linux or UNIX is the ability to mount an ISO image of a CD such that you don't need a physical CD.



If you have access to the ISO images of the CDs, you could specify a directory and under that directory mount the three images in the directories named `cd1`, `cd2`, and `cd3`. We used this option to access the early copy of the SuSE distribution for which we had the three `.iso` files (you can create an iso file from a CD by simply using the `dd` command). Assuming you have the three `.iso` files in the top level directory `/suse`, you can mount them using loopback with the following commands:

```
# cd /suse
# mkdir cd1 cd2 cd3
# mount /suse/cd1.iso cd1 -o loop -r
# mount /suse/cd2.iso cd2 -o loop -r
# mount /suse/cd3.iso cd3 -o loop -r
```

- ▶ Copy the three CDs to disk. We did this for the SuSE GA (7.0) version because it enabled us to use the kernel and RAMdisk with the updated `iucv` driver. You will need about 1.8 GB of free disk space for the GA version of SuSE Linux for S/390.

We did this as follows:

```
# mkdir /suse
# cd /suse
# mkdir cd1 cd2 cd3
(...put CD1 in drive...)
# mount /dev/cdrom /mnt/cdrom
mount: block device /dev/cdrom is write-protected, mounting read-only
# cp -r /mnt/cdrom/* cd1
# umount /mnt/cdrom
(...put CD2 in drive...)
# mount /dev/cdrom /mnt/cdrom
mount: block device /dev/cdrom is write-protected, mounting read-only
# cp -r /mnt/cdrom/* cd2
# umount /mnt/cdrom
(...put CD3 in drive...)
# mount /dev/cdrom /mnt/cdrom
mount: block device /dev/cdrom is write-protected, mounting read-only
# cp -r /mnt/cdrom/* /cd3
# umount /mnt/cdrom
```

## 16.5.2 Format a clean S/390 DASD

We recommend that you format the 3390-3 DASD with ICKDSF. Typically, this is done from another system with access to the DASD such as OS/390 or VM. However, to save time in formatting, we learned that only the first 500 cylinders have to be formatted. This formatting can also be accomplished using a standalone ICKDSF tape.

If VIF has ever been installed on the target DASD, a new install of VIF will detect the previous install and will try to use the same parameters and master Linux that were previously installed. If you've never installed VIF on the target DASD before, you could omit this step. We have done so with success.

To format the first 500 cylinders of the DASD at address 0996, we used the JCL job from OS/390 shown in Figure 16-4 on page 277.

```

//TRKFMT JOB (999,POK),'VIF PREP',NOTIFY=&SYSUID,
// CLASS=A,MSGCLASS=T,MSGLEVEL=(1,1)
//*****//
//*                                     *//
//* INITIALIZE VOLUME PRIOR TO INSTALLATION OF VIF          *//
//*                                     *//
//* STEP 1: CLEAR RESIDUAL DATA FROM PREVIOUS INSTALL      *//
//*                                     *//
//*****//
//STEP1      EXEC PGM=ICKDSF,PARM='NOREPLYU'
//SYSPRINT DD SYSOUT=*
//SYSIN      DD *
TRKFMT UNITADDRESS(996) ERASEDATA -
          CYLRANGE(000,499) VERIFY(*NONE*)
/*
//

```

Figure 16-4 Using JCL on OS/390 to initialize VIF volume

Alternatively, you could also format the DASD from VM. To format the DASD at address 0996, we the VM **cpfmtxa** command as shown in Figure 16-5.

```

DASD 0996 ATTACHED TO MIKEM 0996 WITH DEVCTL
cpfmtxa
ENTER FORMAT, ALLOCATE, LABEL, OR QUIT:
format
ENTER THE VDEV TO BE PROCESSED OR QUIT:
996
ENTER THE CYLINDER RANGE TO BE FORMATTED ON DISK 0996 OR QUIT:
0-499
ENTER THE VOLUME LABEL FOR DISK 0996:
vif996
CPFMTXA:
FORMAT WILL ERASE CYLINDERS 00000-00499 ON DISK 0996
DO YOU WANT TO CONTINUE? (YES | NO)
yes
...

```

Figure 16-5 Using CPFMTXA on VM to initialize a VIF volume

Either way, the format took 3 to 4 minutes. The standalone **ICKDSF** with **CPVOLUME** format command could have also been used.

### 16.5.3 Load VIF tape onto DASD

We loaded the VIF tape into the tape drive at address 0B30. We used a load parameter of AUTO0996, as shown in Figure 16-6. This loads the VIF code from the tape onto the DASD volume at address 0996. The systems programmers assured us that the tape drive at address 0B30 and the 3390-3 at address 0996 were not online to any other systems. Though the VIF manual states you should do a load type of “clear”, we did a load type of “normal” and it worked just the same. However, a load type of clear is still recommended.

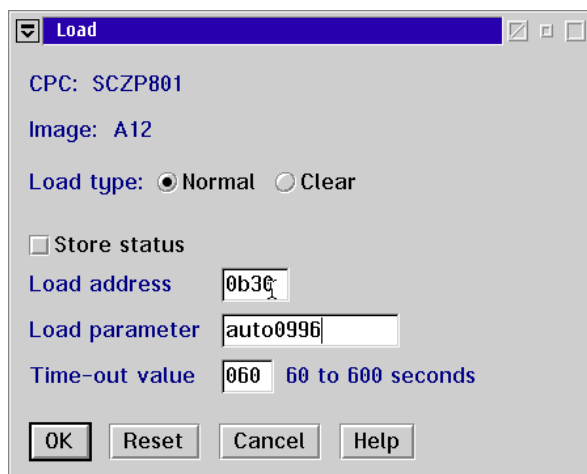


Figure 16-6 HMC load panel

The icon representing the VIF LPAR remained green for about 10 to 15 minutes as the VIF tape was being copied to the DASD volume (the tape drive indicated the tape was being actively read early in the process, but then became inactive though VIF was still initializing on DASD).

The icon then turned red and a hardware message appeared stating: Disabled wait state. Clicking **Details** revealed a program status word (PSW) of 000A0000 00000000, as shown in Figure 16-7 on page 279.

**Note:** Ensure that you receive this PSW, because we had problems with channel paths to the tape drive which sometimes resulted in a different PSW. When this was the case, VIF had not been cleanly installed to DASD and would not subsequently IPL.

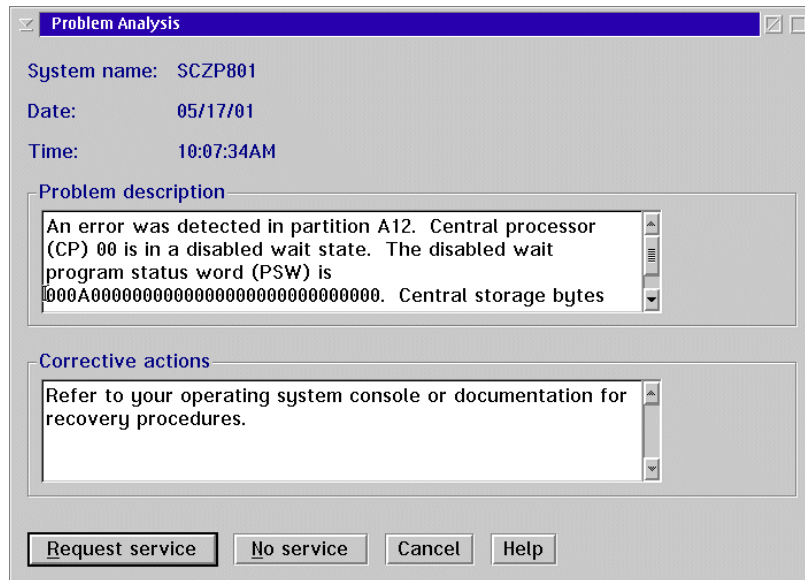


Figure 16-7 Disabled wait state message with proper PSW

Alternatively, VM can be used to load the VIF tape onto the target DASD. In addition to using the HMC load panel, we also loaded VIF onto DASD by attaching a tape drive to a VM ID as device 181 and attaching the DASD (99A, in this example) that VIF was to be loaded onto. Then we issued the CP IPL command as follows:

```
IPL 181 LOADPARM AUTO099A
...
HCPGIR450W CP entered; disabled wait PSW 000A0000 00000000
```

This IPLs the VIF tape. The parameter `LOADPARM AUTO099A` tells the IPL to load the image on device 99A. When the tape is done loading onto DASD (about 15 minutes again), CMS can be re-IPLed with this command:

```
IPL CMS
```

#### 16.5.4 IPL the DASD and answer VIF questions

Now you can IPL from the VIF image that has been written to DASD. Before installation, you will want to know the following information:

Table 16-1 Information needed for VIF installation

1. <i>Sysres valid for VIF</i> - this is the label that will be written to the VIF volume.	
2. <i>Hypervisor device address and port number</i> - this is the even address of the network device pair is needed. The port number is typically 0, though it may be 1 for OSA2 cards.	
3. <i>Network type</i> - Ethernet, Token-Ring, 802.3 or FDDI.	
4. <i>MTU size</i> - 576,1492, 1500, 2000 or 4096. For Ethernet, the maximum size is 1500.	
5. <i>VIF hypervisor IP address and subnet mask</i> - this is the network information for the hypervisor; this must be accessible via a physical device.	
6. <i>Gateway IP address</i> - this is the machine that will route IP traffic off the subnet; it is often referred to as the <i>router</i> .	
7. <i>Master Linux network type</i> - this is either internal or external; see 16.2.1, "VIF networking options" on page 270.	
8. <i>Master Linux network type</i> - this is either internal or external; see 16.2.1, "VIF networking options" on page 270.	
9. <i>Master Linux IP address and subnet mask</i> - if you are using an external network, this will typically be on the same subnet as the VIF hypervisor. If you are using an internal network, it will have to be on a different subnet.	
10. <i>FTP server IP address</i> - see "Prepare an FTP server" on page 273.	
11. <i>FTP user ID, password and account</i> - typically the user ID will not be root and the account will be null (simply press <b>Enter</b> ).	
12. <i>VIF installation file name</i> - see "Prepare the VIF install file" on page 273.	

We IPLed from the HMC by selecting the proper LPAR and double-clicking **LOAD**. This time, the DASD was specified as the load address with no load parameter. The process took a few minutes, but then began asking the aforementioned questions. We answered the questions and were asked to confirm.

In your case, after you enter `yes`, VIF will first attempt to get the install file described in 16.5.1, “Prepare an FTP server with the VIF install file and Linux” on page 272. Then it will try to get and load the Linux kernel, parameter file, and RAMdisk specified in the install file. If these steps are successful, a Linux image will boot to RAMdisk.

If you get a message stating: `Error 17 installing Linux`, this probably means that VIF has found the install file, but not the associated Linux files. If this is the case, create a custom install file as described in 16.5.1, “Prepare an FTP server with the VIF install file and Linux” on page 272.

## 16.5.5 Begin master Linux installation from HMC

When a Linux for S/390 kernel first boots, you are typically presented with questions whose answers are required to start networking. SuSE Linux for S/390 is no exception; when the kernel first boots, a script should ask you the following question:

```
=
==--                               Welcome to SuSE Linux S/390 7.0          ==
=
```

First, select the type of your network device:

- 0) no network
  - 1) OSA Token Ring
  - 2) OSA Ethernet
  - 3) OSA-Express Gigabit Ethernet
  - 4) Channel To Channel
  - 5) Escon
  - 6) IUCV
  - 7) CLAW (Cisco Mainframe Channel Connection)
- Enter your choice (1-7):

If you are using IUCV communications (internal network), enter `6` and answer the SuSE networking questions. If you are using an external network, you will want to enter `1` or `2`, depending on whether you are using Token Ring or Ethernet (other network types are not supported by VIF).

One option is to answer 0 for “no network” and start networking manually. This is not recommended, but we tried it with the following commands using an internal network, where 9.12.8.150 is the master Linux, and 9.12.14.196 is the hypervisor TCP/IP address (which must be associated with the OSA address and port):

```
$ ifconfig iucv0 9.12.8.150 pointopoint 9.12.14.196 mtu 1492 up
$ route add default gw 9.12.14.196
$ inetd &
```

Whether you answer the questions or enter the commands manually, you should have networking going, have a default route out of the VIF Hypervisor TCP/IP stack, and have `inetd` started. The new Linux image will subsequently be accessible via telnet.

### Issues with telnet clients and SuSE YaST

You need a telnet session with at least 80 rows and 25 columns in order to use YaST under SuSE. (Turbolinux and Red Hat install tools are text-based, rather than curses-based, so this is not an issue for those distributions.) It is easier to telnet from a Linux PC. Using telnet in either command line mode or in X Window mode will give you a good interface to YaST.

If you have a Windows desktop, there can be problems with YaST. Using the telnet client that comes with Windows is virtually impossible - graphics characters simply don't map and therefore you can't make sense out of the panels. We used TeraTerm, a free telnet client for Windows, which was usable.

A problem with TeraTerm is the use of function keys. Pressing a function key does not map properly from Windows to YaST. If you have this problem, you can press **Ctrl-F** and then the number of the function key you want; for example, if you want to press F4, you can press **Ctrl-F** and then **4**.

Another increasingly popular telnet client for Windows is PuTTY. It is a free implementation of Telnet and SSH for Win32 platforms, written and maintained primarily by Simon Tatham. It is included on the SuSE GA CD in the directory `/dosutils/puttytel/`. Simply copy the `puttytel.exe` file to a directory in your Windows PATH (C:\WINNT, for example) and you can invoke the command:

```
C:\WINNT>puttytel 9.12.6.95
```



## Master Linux DASD

When installed, VIF creates two virtual DASD as shown in Table 16-2.

Table 16-2 VIF default DASD

Address	Device	Size (formatted)	Description
201	/dev/dasda	830MB	Empty DASD for the root file system
203	/dev/dasdb	956KB	Configured DASD with the VIF client

You have to let the new Linux image know that it owns two DASD devices at virtual addresses 201 and 203. Then you can verify they are recognized by querying the proc file system. We did this as follows:

```
# insmod dasd dasd=201,203
Using /lib/modules/2.2.16/block/dasd.o
# cat /proc/dasd/devices
0201(ECKD)at (94:0) is dasda:active at blocksize: 4096,216000 blocks,843 MB
0203(ECKD)at (94:4) is dasdb:active at blocksize: 1024, 2475 blocks, 2 MB
```

We've successfully allowed YaST to format the 201 disk. However, you should *not* format the 203 disk—because you will erase the `vif` client. Instead, we recommend that you format and make file systems “by hand”, as follows:

```
# dasdfmt -b 4096 -f /dev/dasda -y
# mke2fs -b 4096 /dev/dasda1
...
Writing superblocks and filesystem accounting information: done
```

### 16.5.6 Telnet to master Linux and finish installation

Now that `/dev/dasda` was properly prepared, it was time to finish the installation of the master Linux. With Suse, this is done via YaST. After installation there are a few more steps to be performed. We invoke YaST as follows:

```
# yast
```

In our case, we chose the following values:

- ▶ SELECT LANGUAGE panel - **English**
- ▶ SELECTION OF INSTALLATION MEDIUM panel - **Installation from an FTP site**
- ▶ TYPE OF INSTALLATION panel - **Install Linux from scratch**
- ▶ SELECT SWAP PARTITION panel - **<no swap-partition>**
- ▶ PARTITION HARDDRIVES panel - **< Do not partition >**

From this point, you can do the following:

- ▶ `CREATING FILESYSTEMS` panel - Using **F4**, set `/dev/dasda1` to mount over `/`, and `/dev/dasdb1` to mount over `/vif` (you could choose another directory such as `/usr/local/vif`). Because `/vif` is not a choice, you have to choose **<Other entry>** and enter `/vif` in the panel that comes up.

If it were possible to set the `/vif` file system to be read-only, you would want to do that here. However, this does not appear to be an option.

- ▶ `FTP Settings` panel - Enter the FTP server IP address (in our example, it was `9.12.0.88`) and the Server directory (in our example, it was `/suse/suse`). Unless you've set up your FTP server for anonymous FTP, uncheck `Anonymous FTP`; then you can enter the FTP user ID and password.

Press **F10** to begin (with Linux telnet or putty, the **F10** key works fine; however, with other telnet clients, you may have to press **Ctrl-F**, then **0**).

- ▶ `Installation` panel - It should be noted that YaST goes from a "wizard" model, which takes you through the proper order of panels, to a menu-driven model, where you have to know which panels to choose.

- Choose `Load Configuration` - On the new panel, it is recommended that you take the default. For the version of SuSE that we used, the default configuration used 735 MB of the 830 MB root file system. This does not leave you with much space to work with, but it is recommended that the master Linux just be used for the `vif` command.

- Choose `start installation` - On the resulting panel, you may see a message stating: Search for installed or ready to be installed packages whose dependencies are NOT given!..., followed by a list of packages.

If this happens, select **< Auto >**. Installation should then begin and run for 10 to 15 minutes. Finally you should see the message: `INSTALLATION COMPLETE`.

- Choose `Main Menu` - YaST now takes you back to the wizard model, as all remaining panels are presented to you sequentially.

- ▶ `SELECT KERNEL` panel - Choose `Default kernel for S/390`.
- ▶ `TIME ZONE CONFIGURATION` panel - Choose your time zone.
- ▶ `ADJUSTMENT OF HARDWARE CLOCK` panel - Choose `local time` unless your hardware clock is set to GMT.
- ▶ `ENTER YOUR HOSTNAME` panel - We recommend a host name of `linux0`, so it is the same as the VIF image name.
- ▶ `TCP/IP CONFIRMATION` panel - Choose **< Real network >**.
- ▶ `DHCP CONFIRMATION` panel - Choose **< No >**.

- ▶ ENTER THE NETWORK ADDRESSES panel - Choose **iucv0**, or **eth0**, or **tr0**, depending on the type of network you are using. All other values should be correct.
- ▶ START INETD? panel - Choose < **Yes** >.
- ▶ START THE PORTMAPPER panel - We recommend choosing < **No** >.
- ▶ ADJUST NEWS FROM-ADDRESS panel - Take the default value.
- ▶ Access a nameserver? panel - Choose < **yes** > and enter your name server information.
- ▶ SELECTION OF NETWORKING DEVICE panel - For “Network type”, press F3 to get a selection list and choose your network type.
- ▶ SENDMAIL CONFIGURATION panel -> We recommend that you choose **Do not install /etc/sendmail.cf**.
- ▶ Now the tool SuSEconfig is automatically run. This tool takes all the settings entered into YaST and populates the values in the correct places in the root file system. You should see the following messages:
 

```

Started the SuSE-Configuration Tool.
Running in full featured mode.
Reading /mnt/etc/rc.config and updating the system...
Installing new /etc/HOSTNAME
Installing new /etc/resolv.conf
Installing new /etc/nntpserver
Installing new /etc/inews_mail_gateway
Installing new /var/lib/news/mailname
Installing new /var/lib/news/whoami
Installing new /etc/SuSEconfig/profile
Installing new /etc/SuSEconfig/csh.cshrc
      
```
- ▶ Finally, YaST should terminate and show the message: YaST finished.

## Set VIF file system to be read-only

With other distributions, you should try to set the VIF file system (/dev/dasdb at the address 203) to be read-only. There does not appear to be a way do this in YaST, so it must be done manually. If it is left to the default of read/write, error messages will appear on the hardware console.

To modify the master Linux to mount the VIF file system as read-only, we mount what will become the new root file system and modify the file /etc/fstab:

```

# mount /dev/dasda1 /mnt
# cd /mnt/etc
# cat fstab
/dev/dasda1    /                ext2            defaults    1    1
/dev/dasdb1    /vif             ext2            defaults    1    2

```

```
none          /proc          proc          defaults 0 0
# End of YaST-generated fstab lines
```

Modify the second line so it looks like this:

```
/dev/dasdb1   /vif          ext2          defaults,ro 0 0
```

The last two integers are set to 0 0, so the VIF file system is not dumped—nor is the `fsck` command run against it.

## 16.5.7 Reset master Linux boot partition and reboot

We now had to set VIF so it would boot from the new root file system. Assuming the root file system was still mounted from the previous step, we mounted the VIF file system over it:

```
# mount /dev/dasdb1 /mnt/vif
```

We saw from the VIF query image command that the master Linux (LINUX0) booted *default*, which was the RAMdisk:

```
# cd /mnt/vif
# ./vif 9.12.6.97 q image linux0
LINUX0 has 1 CPUs and 128M of storage and boots default
LINUX0 has an external network connection via devices 295E-295F
LINUX0 has a 843 MB read/write partition as device 201 on volume VIF996
LINUX0 has a 3 MB read-only partition as device 203 on volume VIF996
Command Complete
#./vif 9.12.6.97 image set linux0 boot 201
LINUX0 boot changed to 201
Command Complete
```

We now unmounted the DASD and rebooted the master Linux as follows:

```
# cd /
# umount /mnt/vif
# umount /mnt
# reboot
```

The master Linux should shut down and reboot. For SuSE Linux, you will have to go back to the HMC to finish the install (this will probably differ for other distributions). Before networking comes back up, you will have to enter the new root password. This HMC screen is shown in Figure 16-8 on page 287.

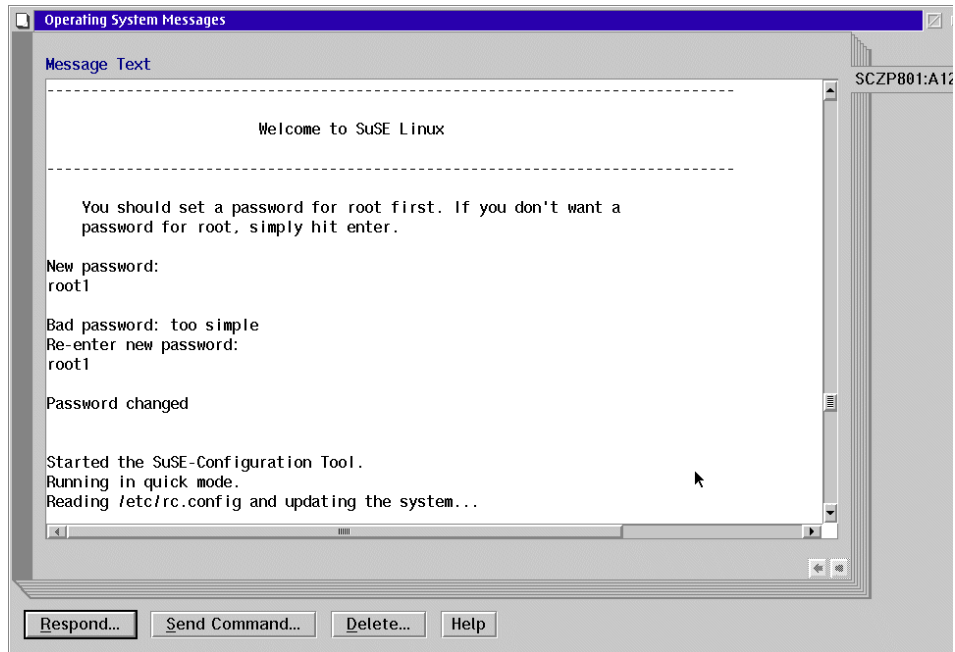


Figure 16-8 Setting root password for master Linux from HMC

If you forget this step and the master Linux boots the RAMdisk, don't despair. You can still perform all the previous steps by booting the RAMdisk, adding the 203 DASD via the `insmod` command, and mounting the disk to get access to the `vif` command.

## 16.6 Using VIF

In these sections, we describe some of the more common usage scenarios with VIF.

First we describe how to create a simple script to make the VIF client command more convenient, how to apply service to an existing VIF installation, and how to change the Linux distribution.

Then we show how to install and customize a "blueprint" or template Linux image, and finally we provide shell scripts that enable you to clone the Linux template.

## 16.6.1 Writing a simple v script

For convenience, we started by creating a simple script named `v` in `/usr/local/bin`, which is in the default `PATH`. (In your case, you should first ensure that `v` is not an alias, and if it is, choose another name for your script):

```
# alias v
bash: alias: `v' not found
```

We hardcode the hypervisor IP address to follow the `vif` command because the hypervisor is always the same. The bash built-in `$@` expands to all parameters, so that all text that follows the new `v` command is sent to the VIF hypervisor:

```
# cd /usr/local/bin
# vi v
...
# chmod +x v
# cat v
/vif/vif 9.12.6.97 $@
```

## 16.6.2 Applying service to VIF

If you have an existing VIF installation and there is an upgrade, you will want to use the VIF service tape. This allows you to upgrade your level of VIF without reinstalling.

We performed this task as follows from a tape attached as device 560:

```
# /mnt/vif 9.12.6.97 hypervisor service 560
Command may take up to 2 minutes; please wait
Command is still processing; please wait
HYPERVISOR Service Complete.
Command Complete
```

## 16.6.3 Changing the Linux distribution

When VIF is installed, it copies the Linux kernel, RAMdisk, and parameter file from the FTP server to a location on the VIF install volume. These files are used when a new VIF image is created and started.

These files can be thought of as a Linux “boot floppy for PCs”: you only have one floppy, which you can rewrite—but you cannot have a second copy of it.

### ***Maintaining multiple VIF images***

With VIF, you can change these files, but you cannot maintain multiple sets. So in order to have VIF images from more than one distribution, we modified the Linux kernel RAMdisk and parameter file with the following command:

```
# v hyp install 9.12.9.183 mikem nunya /linuxSystems/suse/cd1/suse.ins
Transferring Linux from 9.12.9.183 /linuxSystems/suse/cd1/suse.ins
...
Command Complete
```

## **16.6.4 Allocating image and paging space**

We decided to allocate two 3390-3 volumes for images and one for paging space.

```
# v hyp vol add image 996 VIF996
IMAGE volume VIF996 added
Command Complete
# v hyp vol add image 998 VIF998
IMAGE volume VIF998 added
Command Complete
# v hyp vol add paging 9AC VIFPAG
Command may take up to 22 minutes; please wait
...
```

## **16.6.5 Installing a template Linux image**

We created a new image named `vif1` to be used as a template to create additional Linux images:

```
# v image create vif1
Image vif1 created successfully
Command Complete
```

Now we carved out a 200 MB partition for the root file system at address 201, a 1.2 GB partition for a sharable `/usr` at address 202, and a 200 MB partition for `/modelroot`:

```
# v part create vif1 201 200
PARTITION CREATE completed successfully
Command Complete
# v part create vif1 202 1200
PARTITION CREATE completed successfully
Command Complete
# v part create vif1 203 200
PARTITION CREATE completed successfully
Command Complete
```

Now a networking device must be added. This is done differently, depending on whether you are using internal or external networking. If using internal networking, the hypervisor must be told to add an IUCV connection:

```
# v hyp net add vif1 9.12.8.151 255.255.255.255
Command Complete
# v q net
Hypervisor uses IP address 9.12.14.196 via device 21C0
Linux image vif1 uses IP address 9.12.8.151 via the internal network
Linux image LINUX0 uses IP address 9.12.8.150 via the internal network
```

If using external networking, the networking device address assigned to the IP address must be known; for example, during this project we had the network addresses 2942 and 2943 assigned to the IP address 9.12.6.81. To specify this network address, only the even value is supplied with the command:

```
# v image network vif1 add 2942
NETWORK ADD completed successfully
Command Complete
```

Now we were ready to start the image:

```
# v image start vif1
Linux image vif1 started
Command Complete
```

We found we had to telnet to the hypervisor and log in as vif1. The hypervisor can only *speak* telnet in line mode. We found it difficult to communicate with the hypervisor, as most telnet clients do not seem to speak the same dialect.

At first, our keystrokes were not echoed. Setting the telnet client to echo locally made the session was much more usable; however, characters were not always rendered correctly. From VM, you can telnet to the hypervisor with the “linemode” parameter:

```
telnet 9.12.6.97 (linemode
```

There is a usability issue when telnetting to the hypervisor: unlike VM, where you can disconnect with the command #CP DISC, VIF has no such mechanism. So once you have telnetted to the hypervisor, the only apparent “way out” is by escaping to the telnet prompt and typing: quit. To get to the telnet prompt, you normally type: Ctrl-], but from VM telnet ,you press **F4**.

At this point, you should be presented with the networking questions, to which you can answer 6 for iucv:

```
S/390 Virtual Image Facility for LINUX --VIF      --PRESS BREAK KEY TO BEGIN
SESS
ION. logon vif1
```



```
LOGMSG - 21:02:27 UTC Wednesday 06/14/00
*** S/390 Virtual Image Facility for LINUX ready!
FILES:  NO RDR,  NO PRT,  NO PUN
RECONNECTED AT 15:33:32 UTC THURSDAY 01/11/01
!
.

First, select the type of your network device:
...
```

For internal network (iucv, choice 6), we answered the questions as follows:

```
Configuration for iucv0 will be:
Full host name   : vif1.itso.ibm.com
IP address      : 9.12.8.151
Peer IP address  : 9.12.14.196
DNS IP address   : 9.12.14.7
DNS search domain: itso.ibm.com
MTU size        : 1492
Is this correct (Yes/No) ?
```

For external networking (ethernet, choice 2), we answered the questions as follows:

```
Configuration for eth0 will be:
Full host name   : vif1.itso.ibm.com
IP address      : 9.12.6.81
Net mask        : 255.255.255.0
Broadcast address: 9.12.6.255
Gateway address  : 9.12.6.75
DNS IP address   : 9.12.14.17
DNS search domain: itso.ibm.com
MTU size        : 1492
Is this correct (Yes/No) ?
```

In your case, if networking starts cleanly, you can leave the hypervisor session and telnet normally to vif1 (9.12.8.151), and then install the flavor of SuSE you want to be cloned. The first step is to tell Linux about the three new partitions:

```
# insmod dasd dasd=201-203
Using /lib/modules/2.2.16/block/dasd.o
# cat /proc/dasd/devices
0301(ECKD) at (94:0) is dasda:act at blksz: 4096, 51300 blocks, 200 MB
0302(ECKD) at (94:4) is dasdb:act at blksz: 4096, 307260 blocks, 1200 MB
0303(ECKD) at (94:8) is dasdc:act at blksz: 4096, 51300 blocks, 200 MB
```

Again we recommend formatting manually with `dasdfmt` and `mke2fs`. Then complete the install with `yast` as with LINUX0.

In our case, after a successful install we gave the command **shutdown -h now** to shutdown the new Linux image.

Then we went over to the master Linux, LINUX0, and invoked the following commands to switch the boot location from RAMdisk to DASD 201:

```
# v image stop vif1
Linux image vif1 stopped
# v image set vif1 boot 201
vif1 boot changed to 201
Command Complete
# v image start vif1
Linux image vif1 started
Command Complete
```

Again, we first had to telnet to the hypervisor, logon to vif1, and set the root password. Then the new Linux booted successfully and we were able to telnet in and verify that the new DASD are mounted correctly:

```
# df -h
Filesystem                Size  Used Avail Use% Mounted on
/dev/dasda1                194M   53M  131M  29% /
/dev/dasdb1                1.2G   1.0G   84M  92% /usr
/dev/dasdc1                194M   20k  184M   0% /modelroot
```

## 16.6.6 Customizing the template Linux image

There are many customizations that can be done to vif1. In fact, you may want to create many Linux template images, based on the function they will perform. For example, you may want to add certain users, set up anonymous FTP, set up paging space, or start Samba. If you are going to use VIF in production, it is suggested that you plan the Linux templates carefully.

In our case, we added a swap *file* (not a swap partition):

```
# mkdir /swap
# cd /swap
# dd if=/dev/zero of=swapfile bs=1M count=64
64+0 records in
64+0 records out
# mkswap -c swapfile
Setting up swapspace version 1, size = 67104768 bytes
# chmod 700 /swap
# chmod 600 swapfile
# swapon swapfile
# cat /proc/swaps
Filename                                Type           Size          Used          Priority
/swap/swapfile                          file           65528         0             -1
```

We then added the swap file to the `/etc/fstab` file, so it is set up at IPL time:

```
# cd /etc
# vi fstab
...
# cat fstab
/dev/dasda1 / ext2 defaults 1 1
/dev/dasdb1 /usr ext2 defaults 1 2
/dev/dasdc1 /modelroot ext2 defaults 1 2
/swap/swapfile swap swap defaults 0 0
none /proc proc defaults 0 0
# End of YaST-generated fstab lines
```

We also added a user, since it is helpful to have a non-root user:

```
# useradd mikem
# passwd mikem
New password:
Re-enter new password:
Password changed
# mkdir /home/mikem
```

We next shut down the image `vif1` and deleted the 203 partition, and then copied the 201 partition to a new 203 partition. This has the effect of copying the root file system, which can be used for cloning. By having a copy of the root file system, we will not have to manipulate the existing root file system. We did not have to create a 203 partition (`/modelroot`) in the first place; however, we felt it was easier to create one so YaST would populate the parameter file and `/etc/fstab` files and run `sil` properly.

From the customized `vif1` image we did the following:

```
# shutdown -h now
```

Then, from the master Linux we did the following:

```
# v image stop vif1
Linux image VIF1 stopped
Command Complete
# v part del vif1 203
PARTITION DELETE completed successfully
Command Complete
# v part copy vif1 201 to vif1 203
Command may take up to 5 minutes; please wait
PARTITION COPY completed successfully
Command Complete
# v image start vif1
Linux image VIF1 started
Command Complete
```

When we telnetted back to vif1, we saw that we could log in as mikem and that /dev/dasdc1 looked just as it did before—however, it was now a copy of the root file system:

```
vif1 login: mikem
Password:
Have a lot of fun...
# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/dasda1     194M  120M   64M  65% /
/dev/dasdb1     1.2G  1.0G   84M  92% /usr
/dev/dasdc1     194M  120M   64M  65% /modelroot
# ls /modelroot
bin      cdrom  floppy  lost+found  opt          proc  swap  var
boot    dev    home    mnt          parm.RMnPvq  root  tmp
boot.map etc    lib     modelroot   parm.TA92Zo  sbin  usr
```

A few changes must be made to the modelroot file system: the file that will become the /etc/fstab, and the parameter file, need to have the modelroot file system removed from them:

```
# cd /modelroot/etc
# vi fstab
... (delete line with modelroot)
# cd /modelroot/boot
# vi parmfile
... (delete device 203)
# silo -b ./ipleckd.boot -p ./parmfile -d /dev/dasda -f ./image -t2
```

One final addition will help to automate the cloning process later:

```
# cd /root
# vi .rhosts
... add the line:
9.12.6.97
```

Creating the file .rhosts in the root home directory allows the **rsh** command to work as root from the specified machine, which is the master Linux in this case. It should be noted that many security specialists consider the existence of a .rhosts file anywhere on any machine to be a security issue.

Now vif1 should be rebooted; assuming it comes back up successfully, it is ready to be cloned.

## 16.6.7 Cloning the template Linux image

Now that a template has been created, it can be cloned relatively easily with VIF commands. First we want to modify the files in the modelroot such that when it is copied to another Linux image, that image's networking will come up cleanly. To do this, the following files have to be modified:

- ▶ /etc/hosts
  - host name
  - IP address
- ▶ /etc/rc.config (for SuSE only); the variables:
  - IPADDR\_0 - IP address
  - IFCONFIG\_0 - IP address
  - FQHOSTNAME - host name
  - FROM\_HEADER - host name
- ▶ /etc/route/conf
- ▶ /etc/conf.modules (for external network only) - network device address

Then, from the master Linux, the following VIF commands can be used to clone the Linux image:

```
# v image create vif2                                // create new image
# v part copy vif1 203 to vif 201                    // copy modelroot
# v part share vif1 201 with vif2 201                // share /usr
# v hyp net add vif2 9.12.6.82 255.255.255.255      // add iucv network
# v image set vif2 boot 202                          // set boot partition
# v image start linux02                              // start new image
```

Refer to Figure 16-9 on page 296 for a look at the structure of our VIF system.

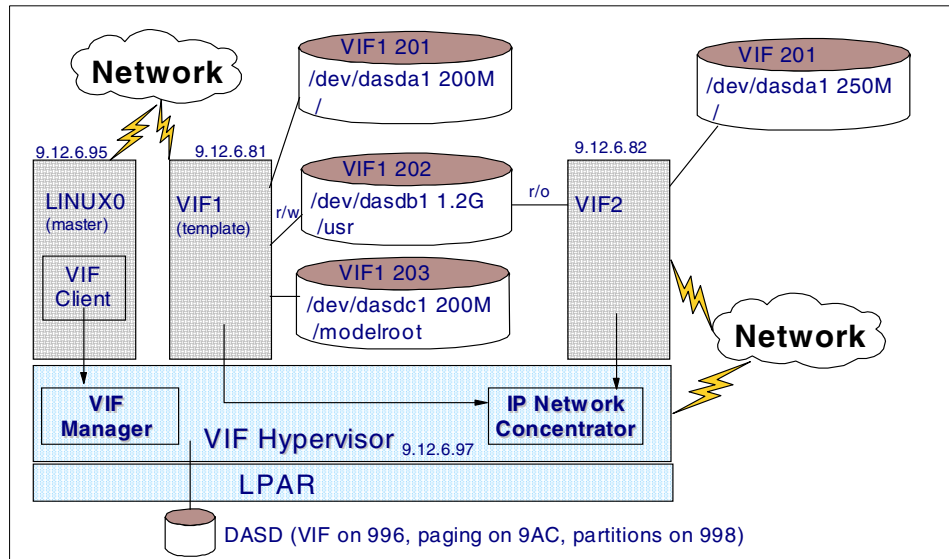


Figure 16-9 VIF system after template and cloned Linux image

## 16.6.8 Scripts to clone Linux images

Now that we manually created a Linux image for cloning, we wanted to automate its cloning.

We wrote a pair of scripts: one to run on the master Linux and issue VIF commands, and one to run on the template Linux and modify the *model root* file system with new TCP/IP address, the new host name, etc. These scripts were written to run on a SuSE Linux image and to use an internal (IUCV) network. Many other assumptions were also made.

The script shown in Example 16-1 on page 297 is run on the template Linux image. Before it is run, copies of the files `/etc/hosts`, `/etc/rc.config` and `/etc/route.conf` are made with the suffix `.iucv`.

This is so the `sed` command can modify the contents to the files that will become the actual `/etc/hosts`, `/etc/rc.config` and `/etc/route.conf` (because you never want to try to use `sed` to modify a file in place - it is often deleted). When the script completes, it returns the string: success.

Note that functions are defined at the start of the script. The first line of that is interpreted and executed is after the comment with the text **main()**.

*Example 16-1 cloneiucv.template script*

---

```
function modifyFiles
{
    cat /modelroot/etc/hosts.iucv | sed \
        -e "s/9.12.6.2/$ipaddr/g" \
        -e "s/linux2/$hostName/g" > /modelroot/etc/hosts
    cat /modelroot/etc/rc.config.iucv | sed \
        -e "s/9.12.6.2/$ipaddr/g" \
        -e "s/linux2/$hostName/g" > /modelroot/etc/rc.config
    cat /modelroot/etc/route.conf.iucv | sed \
        -e "s/9.12.6.2/$ipaddr/g" > /modelroot/etc/route.conf
}

function setDefaults
{
    hostName="vif$suffix"
    ipaddr="9.12.6.$suffix"
    mask="255.255.255.255"
    netaddr="9.12.6.0"
    gateway="9.12.17.173"
}

# main()
if [ $# = 1 ]; then
    suffix=$1
else
    echo "failure"
fi
mount /dev/dasdc1 /modelroot
setDefaults
modifyFiles
umount /modelroot
echo "success"
```

---

Now the script named `cloneiucv.master` is written; see Example 16-2 on page 298. It uses the `rsh` command to invoke the `cloneiucv.template` on the Linux template image VIF1. If that script succeeds, the template is ready to be cloned.

This is done with the `vif` commands shown in bold font (the first three `vif` commands ensure that you do not try to create an image with an existing name).

Each vif command is preceded with a function call to the function anyCmd, so the command that is being executed is echoed to the screen (stdout).

*Example 16-2 cloneiucv.master script*

---

```
function usage
{
    echo; echo "Usage: $scriptName hostNameSuffix"; echo
    echo "Clone a new VIF Linux image from the existing VIF1"
    echo "The following assumptions are made:"
    echo "VIF1 is the VIF template image whose IP address is 9.12.6.81"
    echo "VIF1 has a script /usr/local/bin/cloneiucv.template"
    echo "rsh to VIF1 as root works"
    echo "VIF1 301 is sharable /usr"
    echo "VIF1 302 is modified root file system"
    exit
}

function anyCmd
{
    echo "Executing: $@"
    eval $@
}

# main()
scriptName=`basename $0`
if [ $# != 1 ]; then
    usage
fi

retVal=`rsh 9.12.6.81 /usr/local/bin/cloneiucv.template $1`
if [ $retVal = "success" ]; then
    anyCmd v image stop vif$1
    anyCmd v hyp net del vif$1
    anyCmd v image del vif$1
    anyCmd v image create vif$1
    anyCmd v part copy VIF1 201 to vif$1 201
    anyCmd v part share VIF1 202with vif$1 202
    anyCmd v hyp net add vif$1 9.12.6.$1 255.255.255.255
    anyCmd v image set vif$1 boot 201
    anyCmd v image start vif$1
else
    echo "Error: retVal = $retVal"
fi
```

---

We then tried using the VIF system and the scripts to carve out 17 more Linux images (named linux2 to linux18). It would be advisable to first have the host names and associated IP addresses defined to DNS.



This task took about 3 minutes per image—with most of the time being spent copying the root file system—which illustrates the power of VIF to create virtual Linux servers easily.

```
# time for i in 2 3 4 5 6 7 8 8 10 11 12 13 14 15 16 17 18
> do
>   cloneiucv.master $i
> done
...
Linux image VIF18 started
Command Complete

real    42m57.002s
user    0m2.390s
sys     0m12.660s
```

### 16.6.9 Shutting down VIF

Shutting down `vif` cleanly presents somewhat of a “chicken and egg” problem (that is, which comes first?). If you shut down the master Linux, you no longer have a `vif` command to shut down the hypervisor. If you shut down the hypervisor, the master linux comes down *hard* (that is, it is not a clean shutdown via the `shutdown -h now` command).

Perhaps the best compromise is to unmount the master Linux’s root file system first, then issue the `vif hyp shutdown` command. Because the master Linux is designed to only run the `vif` command, you can put almost all of the file system on a R/O partition, eliminating the long `fsck` times when VIF comes back up. In this situation, ensure that you provide a `r/w` “current directory” so that `vif` can write its output files.





## Logical Volume Manager

In this chapter we discuss Logical Volume Manager (LVM), which is a set of kernel patches and utilities that allows you to combine multiple physical disks into one or more logical volumes that may be larger than any single physical volume. LVM can also stripe its logical volumes across multiple physical volumes. Possibly LVM's most interesting feature is that you can dynamically add space to a logical volume without having to back up the data, repartition, and then restore the data. (However, the logical volume has to be unmounted to perform an `fsck` against it, so it's not completely nondisruptive.)

Table 17-1 lists the support for LVM in the distributions we cover in this redbook:

*Table 17-1 LVM in various distributions*

Distribution	Kernel patches installed	LVM utilities installed
SuSE	Yes	Yes
Turbolinux 6.0	No	No
Turbolinux 6.5	Yes	Yes
Red Hat	No	No
Caiman	Yes	Yes
Marist	No	No
ThinkBlue	Yes	Yes

## 17.1 Problem support for LVM

One note of caution about LVM - the Linux kernel maintainers did not merge the LVM code into the official kernel source until Linux 2.4. As a result, they do not view LVM in the Linux 2.2 kernels as something they support. If your Linux distributor includes LVM in their distribution, and you are paying them for support, they should provide you with that support. If you want *free* support for LVM, you will only be able to get that for Linux 2.4 kernels and above.

## 17.2 LVM basics

Suppose you want to use more space than is available on a single volume—how about using the space of several DASDs as if it were one big volume? This method is known as *striping* in regard to RAID methods. LVM is able to make one or more big volumes out of several smaller volumes.

In LVM terms, a DASD volume is called a *physical volume* (PV), because that's the volume where the data is physically stored. It can be a model-3 or a model-9 DASD, or even a VM minidisk of any size. The PV is divided into several *physical extents* (PE) of the same size. The PEs are like blocks on the PV.

Several PVs make up a *volume group* (VG), which is a pool of PEs available for the *logical volume* (LV). The LVs appear as normal devices in /dev/ directory. You can add or delete PVs to/from a VG, and increase/decrease your LVs. Figure 17-1 shows how LVM works.

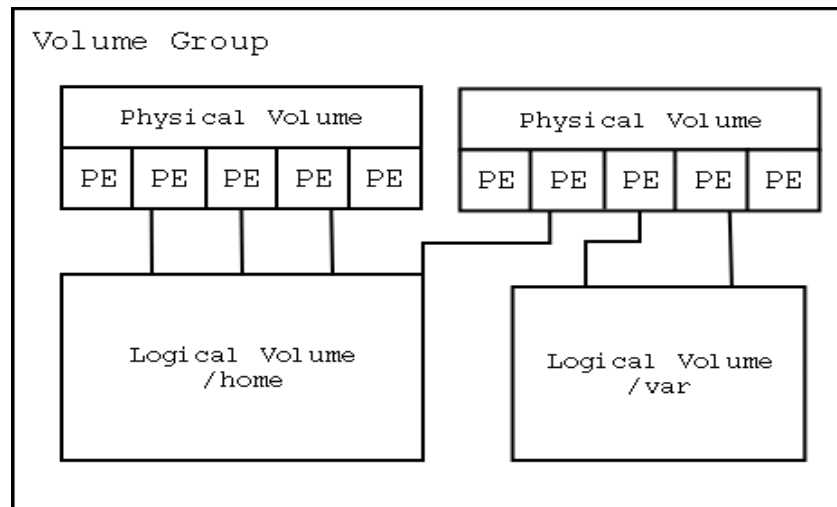


Figure 17-1 LVM block diagram

**Note:** Currently LVM can only handle up to 256 PVs and LVs.

## 17.3 Sample LVM session

The sample process flow that follows was modeled from Richard Higson's write-up on LVM at:

<http://pax.gt.owl.de/~higson/LVM/>

To create a logical volume, you must start with formatting the DASD of all the real volumes that will make it up.

In our case, we first formatted the four DASD that will be used to create a physical volume:

```
# dasdfmt -y -b 4096 -n 204 &
# dasdfmt -y -b 4096 -n 205 &
# dasdfmt -y -b 4096 -n 206 &
# dasdfmt -y -b 4096 -n 207 &
```

### 17.3.1 Using the pvcreate and vgscan commands

You will also need to create a *physical volume* entry for each of these, so that LVM will know about them. We attempted to create physical volumes:

```
# pvcreate /dev/dasd[efg]1
pvcreate -- ERROR: "/etc/lvmtab" doesn't exist; please run vgscan
```

We got an error, because LVM had not been initialized. If this is the first time you've used LVM, you'll have to run **vgscan** first, which we did as follows:

```
# vgscan
vgscan -- reading all physical volumes (this may take a while...)
vgscan -- no volume groups found
```

This command scans all disks for volume groups and builds the files `/etc/lvmtab` and `/etc/lvmtab.d/*` which are the database for all other LVM commands. Now we ran **pvcreate** again:

```
# pvcreate /dev/dasd[efg]1
pvcreate -- reinitializing physical volume
pvcreate -- physical volume "/dev/dasde1" successfully created
pvcreate -- reinitializing physical volume
pvcreate -- physical volume "/dev/dasdf1" successfully created
pvcreate -- reinitializing physical volume
pvcreate -- physical volume "/dev/dasdg1" successfully created
```

You will then need to create a volume group. You can have multiple volume groups (or pools). They can have meaningful names, such as *sales*, *developers*, and so on, if you choose to divide your space pool in that manner.

Taken together, all the volume groups can contain no more than 256 physical volumes. Using the largest device mapping currently available, that of a 3390-9, this means that there can only be .7TB (256\*7GB) of LVM space for any one Linux system.

### 17.3.2 Using the `vgcreate` command

We ran the `vgcreate` command to create a volume group:

```
# vgcreate -s 1m testvg /dev/dasd[efg]1
vgcreate -- INFO: maximum logical volume size is 64 Gigabyte
vgcreate -- doing automatic backup of volume group "testvg"
vgcreate -- volume group "testvg" successfully created and activated
```

You are now at the point where you can create what you really want, a logical volume.

### 17.3.3 Using the `lvcreate` command

We tried to create a logical volume with the `lvcreate` command:

```
# lvcreate -i 6 -I 8 -L 50 testvg -v
lvcreate -- checking volume group name "testvg"
lvcreate -- checking volume group existence
lvcreate -- checking volume group activity
lvcreate -- checking stripe count
lvcreate -- checking stripe size
lvcreate -- locking logical volume manager
lvcreate -- getting volume group status from VGDA in kernel
lvcreate -- checking stripe size against volume group physical extent size
lvcreate -- rounding 51200 KB to stripe boundary size 55296 KB / 54 PE
lvcreate -- reading volume group data of "testvg"
lvcreate -- checking logical volume maximum size
lvcreate -- checking volume group free space
lvcreate -- checking stripe count against physical volume count
lvcreate -- too many stripes for "testvg" with 3 physical volumes
```

However, the command failed because we specified the wrong value with the `-i` parameter; it must equal the number of volumes to stripe over (which is three, not six). The next attempt worked much better:

```
# lvcreate -i 3 -I 8 -L 50 testvg -v
lvcreate -- checking volume group name "testvg"
lvcreate -- checking volume group existence
```

```

lvcreate -- checking volume group activity
lvcreate -- checking stripe count
lvcreate -- checking stripe size
lvcreate -- locking logical volume manager
...
"/etc/lvmconf/testvg.conf"
lvcreate -- unlocking logical volume manager
lvcreate -- logical volume "/dev/testvg/lvol1" successfully created

```

The `-I 8` parameter specifies the stripe size in KB. The `-L 50` parameter specifies the size in MB to allocate for the new logical volume.

You can now create a file system on the logical volume. This does not have to be `ext2`; it can be something else, such as `ext3`, `JFS`, `GFS`, etc. However, due to the issues noted in 2.3.1, “Planning file systems” on page 24, we choose `ext2`. We created a file system on the logical volume:

```

# mke2fs -b 4096 -m2 /dev/testvg/lvol1
mke2fs 1.19, 13-Jul-2000 for EXT2 FS 0.5b, 95/08/09
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
13056 inodes, 13056 blocks
261 blocks (2.00%) reserved for the super user
First data block=0
1 block group
32768 blocks per group, 32768 fragments per group
13056 inodes per group

Writing inode tables: done
Writing superblocks and filesystem accounting information: done

```

**Note:** If there are any entries in `/etc/fstab` with the volumes you just used, *remove* them. If you do not, the next time you IPL, you will probably wind up in single user mode, being prompted to run `fsck` on `/dev/dasd?1`. These physical volumes now belong to the logical volume group.

We now checked the new allocations in the virtual `/proc` file system:

```

# cat /proc/lvm
LVM driver version 0.8i (02/10/1999)

Total: 1 VG 3 PVs 1 LV (0 LVs open)
Global: 5420 bytes vmalloced IOP version: 5 0:14:29 active

VG: testvg [3 PV, 1 LV/0 open] PE Size: 1024 KB
Usage [KB/PE]: 430080 /420 total 52224 /51 used 377856 /369 free
PVs: [AA] /dev/dasde1 143360 /140 17408 /17 125952 /123

```

```

      [AA] /dev/dasdf1          143360 /140          17408 /17          125952 /123
      [AA] /dev/dasdg1        143360 /140          17408 /17          125952 /123
LV: [AWDS3 ] lv011                      52224 /51          cclose

```

At this point, 51 out of 420 *extents* were now in use.

We then tried to create another logical volume with 400 extents. This command failed because there were only 369 free extents:

```

# lvcreate -i 3 -I 8 -L 400 testvg -v
lvcreate -- checking volume group name "testvg"
lvcreate -- checking volume group existence
...
lvcreate -- only 369 free physical extents in volume group "testvg"

```

So we created a second logical volume with 40 extents:

```

# lvcreate -i 3 -I 8 -L 40 testvg -v
lvcreate -- checking volume group name "testvg"
...
lvcreate -- unlocking logical volume manager
lvcreate -- logical volume "/dev/testvg/lv012" successfully created

```

Again we created a file system from the new logical volume:

```

# mke2fs -b 4096 -m2 /dev/testvg/lv012
mke2fs 1.19, 13-Jul-2000 for EXT2 FS 0.5b, 95/08/09
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
10752 inodes, 10752 blocks
215 blocks (2.00%) reserved for the super user
First data block=0
1 block group
32768 blocks per group, 32768 fragments per group
10752 inodes per group

Writing inode tables: done
Writing superblocks and filesystem accounting information: done

```

We now created mount points for the volumes, and mounted them:

```

# mkdir -p /testvg/lv1 /testvg/lv2
# mount /dev/testvg/lv011 /testvg/lv1
# mount /dev/testvg/lv012 /testvg/lv2
# df -h

```

Filesystem	Size	Used	Avail	Use%	Mounted on
/dev/dasdb1	1.8G	123M	1.5G	7%	/
/dev/dasdc1	1.8G	1007M	701M	59%	/usr
/dev/testvg/lv011	49M	20k	48M	0%	/testvg/lv1



```
/dev/testvg/lvol2      41M   20k   40M   0% /testvg/lv2
```

We unmounted the logical volumes:

```
# umount /testvg/lv1 /testvg/lv2
```

### 17.3.4 Using the `lvremove` command

We deleted the second logical volume to exercise the `lvremove` command:

```
# lvremove /dev/testvg/lvol2
lvremove -- do you really want to remove "/dev/testvg/lvol2"? [y/n]: y
lvremove -- doing automatic backup of volume group "testvg"
lvremove -- logical volume "/dev/testvg/lvol2" successfully removed
```

### 17.3.5 Using the `lvextend` command

It is easy to enlarge a logical volume as long as you have the file system unmounted and have enough extents. This is accomplished via the `lvextend` command:

```
# lvextend /dev/testvg/lvol1 -l +42
lvextend -- extending logical volume "/dev/testvg/lvol1" to 93 MB
lvextend -- doing automatic backup of volume group "testvg"
lvextend -- logical volume "/dev/testvg/lvol1" successfully extended
```

We now remounted the logical volume and displayed it. However, we saw that the `df` command reports the wrong size:

```
# mount /dev/testvg/lvol1 /testvg/
# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/dasdb1     1.8G  124M  1.5G   7% /
/dev/dasdc1     1.8G 1007M  701M  59% /usr
/dev/testvg/lvol1  49M   20k   48M   0% /testvg
```

Though we did use `lvextend`, we still must reformat the `ext2` file system.

### 17.3.6 `resize2fs`

If you want to enlarge an `ext2` volume group, you must use the `resize2fs` command on extended logical volumes. We tried it without first running `e2fsck`, but received an error:

```
# resize2fs /dev/testvg/lvol1
resize2fs 1.19 (13-Jul-2000)
Please run 'e2fsck -f /dev/testvg/lvol1' first.
```

So we ran `e2fsck` to check the file system and then reran `resize2fs`:

```

# e2fsck -f /dev/testvg/lvol1
e2fsck 1.19, 13-Jul-2000 for EXT2 FS 0.5b, 95/08/09
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/dev/testvg/lvol1: 11/13056 files (0.0% non-contiguous), 417/13056 blocks
# resize2fs /dev/testvg/lvol1
resize2fs 1.19 (13-Jul-2000)
The filesystem on /dev/testvg/lvol1 is now 23808 blocks long.

```

We again mounted and displayed it, and this time the correct size was shown:

```

# mount /dev/testvg/lvol1 /testvg/lv1
# df -h
Filesystem              Size  Used Avail Use% Mounted on
/dev/dasdb1             1.8G  125M  1.5G   7% /
/dev/dasdc1             1.8G 1007M  701M  59% /usr
/dev/testvg/lvol1       91M   20k   88M   0% /testvg/lv1

```

## 17.4 Creating a 20 GB logical volume

In this example, we created a 20 GB logical volume from three minidisks, each of which were created from entire 3390-9 volumes. The volumes being used had already had **dasdfmt** run against them (that step is not shown here).

We identified the physical volumes to be used by LVM, which are **/dev/dasdh**, **/dev/dasdi**, and **/dev/dasdp** in this example:

```

# pvcreate /dev/dasd[hip]1
pvcreate -- reinitializing physical volume
pvcreate -- physical volume "/dev/dasdh1" successfully created
pvcreate -- reinitializing physical volume
pvcreate -- physical volume "/dev/dasdi1" successfully created
pvcreate -- reinitializing physical volume
pvcreate -- physical volume "/dev/dasdp1" successfully created

```

We created the volume group to be used:

```

# vgcreate -s 1m testvg /dev/dasd[hip]1
vgcreate -- INFO: maximum logical volume size is 64 Gigabyte
vgcreate -- doing automatic backup of volume group "testvg"
vgcreate -- volume group "testvg" successfully created and activated

```

We created the logical volume:

```

# lvcreate -i 3 -L 21126m testvg
lvcreate -- INFO: using default stripe size 16 KB

```

```
lvcreate -- doing automatic backup of "testvg"  
lvcreate -- logical volume "/dev/testvg/lvol1" successfully created
```

We checked to see what the system thinks so far, and found all the space is being used:

```
# cat /proc/lvm  
LVM driver version 0.8i (02/10/1999)  
Total: 1 VG 3 PVs 1 LV (0 LVs open)  
Global: 342625 bytes vmalloced IOP version: 5 2:36:11 active  
VG: testvg [3 PV, 1 LV/0 open] PE Size: 1024 KB  
Usage [KB/PE]: 21633024 /21126 total 21633024 /21126 used 0 /0 free  
PVs: [AA] /dev/dasdh1 7211008 /7042 7211008 /7042 0 /0  
[AA] /dev/dasdi1 7211008 /7042 7211008 /7042 0 /0  
[AA] /dev/dasdp1 7211008 /7042 7211008 /7042 0 /0  
LV: [AWDS3 ] lvol1 21633024 /21126 close
```

We created an ext2 file system on the logical volume:

```
# mke2fs /dev/testvg/lvol1 -b 4096  
mke2fs 1.19, 13-Jul-2000 for EXT2 FS 0.5b, 95/08/09  
Filesystem label=  
OS type: Linux  
Block size=4096 (log=2)  
Fragment size=4096 (log=2)  
2709120 inodes, 5408256 blocks  
270412 blocks (5.00%) reserved for the super user  
First data block=0  
166 block groups  
32768 blocks per group, 32768 fragments per group  
16320 inodes per group  
Superblock backups stored on blocks:  
32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,  
4096000  
  
Writing inode tables: done  
Writing superblocks and filesystem accounting information: done
```

We mounted the logical volume and displayed the space:

```
# mount /dev/testvg/lvol1 /lvm1  
# df  
Filesystem 1k-blocks Used Available Use% Mounted on  
/dev/dasdb1 1842468 87892 1660980 5% /  
/dev/dasdc1 1842468 782012 966860 45% /usr  
/dev/testvg/lvol1 21292924 20 20211256 0% /lvm1
```

The `-h` flag to the `df` command displays sizes using “human” units:

```
# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/dasdb1     1.8G   86M  1.6G   5% /
/dev/dasdc1     1.8G  764M  944M  45% /usr
/dev/testvg/lvol1 20G   20k   19G   0% /lvm1
```

## 17.5 Striping

By default, LVM creates one stripe over several PVs. This means that the PEs within this LV are filled up in a row, as shown in Figure 17-2. LVM jumps to the next PE when the space on the PE before is exhausted.

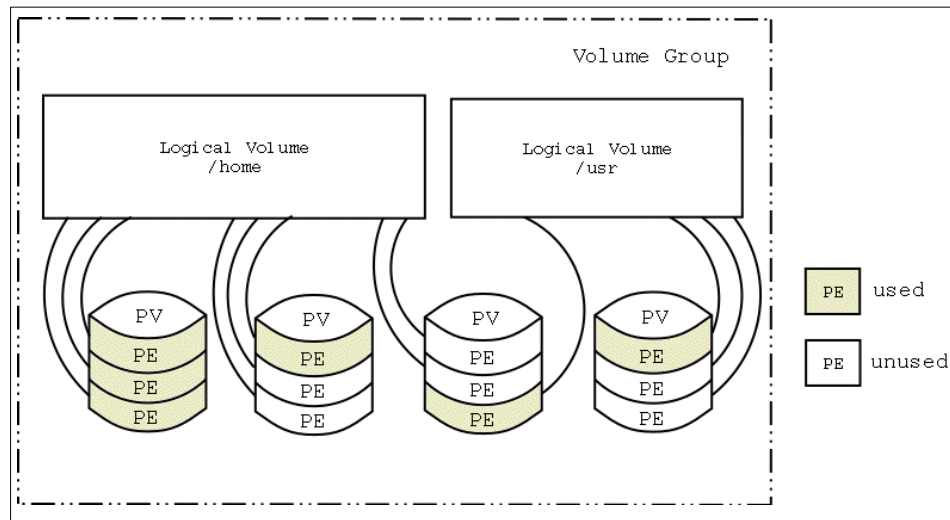


Figure 17-2 LVM - striping (1 stripe)

It is also possible to use more than one stripe over several PVs.

To increase the performance (depending on the channels and control unit), you can use more stripes—but you can only use as many stripes as you have PVs. Additionally, all PVs must be the same size.

With more stripes, the PEs are divided into smaller blocks. If a block of data is written to the PEs, LVM uses the next PE on the next PV to write the next data block. This can increase performance significantly, but the size of the performance increase depends on your hardware configuration—if the PVs (DASDs) in a VG are handled by only one control unit, the performance impact is not as big as if every PV is handled by a different control unit.

See Figure 17-3 for an overview. It shows a LV with three PVs. These PVs are configured with three stripes.

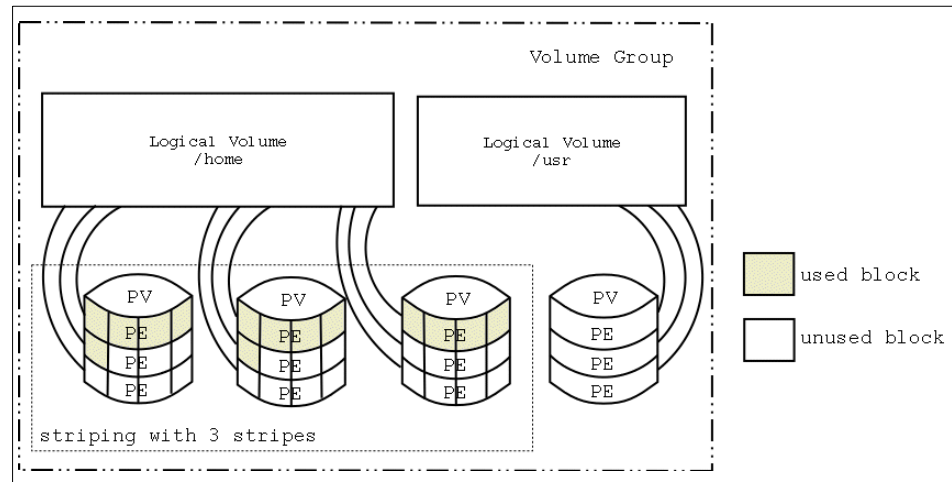


Figure 17-3 LVM - striping with three stripes

## 17.6 Using pvmove to migrate PEs to other PVs

If you have to release some DASDs from your VG, you can move the PEs to other PVs. Consider the following scenarios:

- ▶ A file system is running out of space because you are using only 3390-3 volumes in your VG.

One possible solution is to add 3390-9s to your VG and increase the size. But why not use all 3390-9s and release the 3390-3s for other tasks, without unmounting the file system?

- ▶ A similar scenario occurs if you want to migrate your file system from a RVA to a ESS (Shark) without copying the entire file system, as it will cause system downtime.

The layer model of LVM allows this type of transfer while the file system on the LVs is mounted and in use, as shown in Figure 17-4 on page 312.

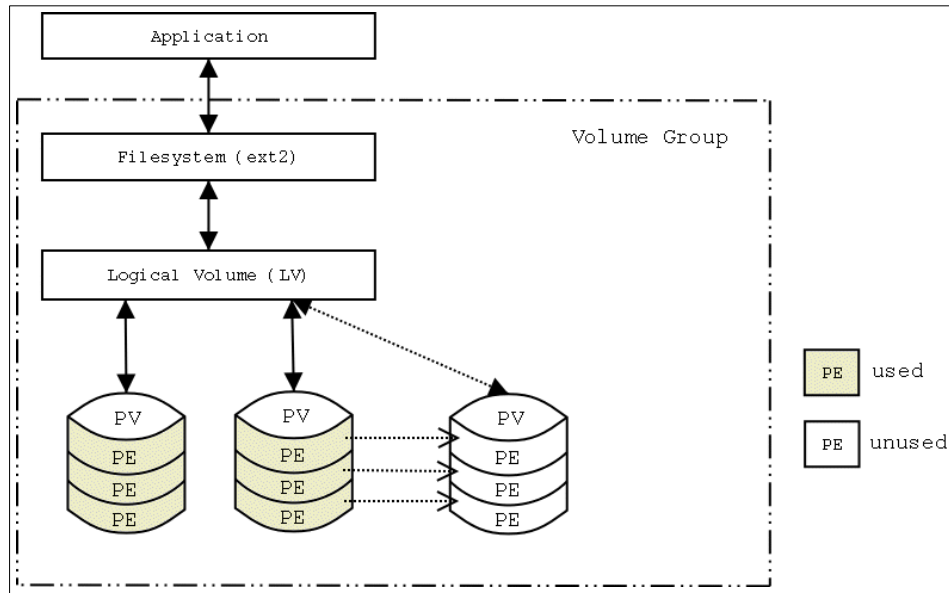


Figure 17-4 LVM layer - showing a pvmove

**Attention:** We recommend you back up your system before using pvmove.

The configuration of the target VG is as follows:

```
# cat /proc/lvm
LVM driver version 0.8i (02/10/1999)

Total: 1 VG 2 PVs 1 LV (1 LV open 1 times)

Global: 5161 bytes vmallocd IOP version: 5 3:12:15 active

VG: vg1 [2 PV, 1 LV/1 open] PE Size: 4096 KB
Usage [KB/PE]: 286720 /70 total 286720 /70 used 0 /0 free
PVs: [AA] /dev/dasdc1 143360 /35 143360 /35 0 /0
      [AA] /dev/dasde1 143360 /35 143360 /35 0 /0
LV: [AWDS2 ] lv1 286720 /70 1x open
```

There are two PVs with one stripe and they are used for one LV.

First add one or more new PVs to your VG:

```
vmlinux2:~ # vgextend vg1 /dev/dasddd1
vgextend -- INFO: maximum logical volume size is 255.99 Gigabyte
vgextend -- doing automatic backup of volume group "vg1"
vgextend -- volume group "vg1" successfully extended
```

**Note:** Make sure the DASD is a PV; use **pvcreate** to create one.

The resulting configuration is shown:

```
# cat /proc/lvm
LVM driver version 0.8i (02/10/1999)

Total: 1 VG 3 PVs 1 LV (1 LV open 1 times)

Global: 5731 bytes vmlalloced IOP version: 5 2:17:37 active

VG: vg1 [3 PV, 1 LV/1 open] PE Size: 4096 KB
Usage [KB/PE]: 430080 /105 total 286720 /70 used 143360 /35 free
PVs: [AA] /dev/dasdc1 143360 /35 143360 /35 0 /0
      [AA] /dev/dasde1 143360 /35 143360 /35 0 /0
      [AA] /dev/dasdd1 143360 /35 0 /0 143360 /35
LV: [AWDS2 ] lv1 286720 /70 1x open
```

The PV /dev/dasde1 will now be replaced by the added PV:

```
# pvmove /dev/dasde1
pvmove -- moving physical extents in active volume group "vg1"
pvmove -- WARNING: moving of active logical volumes may cause data loss!
pvmove -- do you want to continue? [y/n] y
pvmove -- doing automatic backup of volume group "vg1"
pvmove -- 35 extents of physical volume "/dev/dasde1" successfully moved
```

The configuration after the PV is moved is shown:

```
# cat /proc/lvm
LVM driver version 0.8i (02/10/1999)

Total: 1 VG 3 PVs 1 LV (1 LV open 1 times)

Global: 5731 bytes vmlalloced IOP version: 5 2:46:26 active

VG: vg1 [3 PV, 1 LV/1 open] PE Size: 4096 KB
Usage [KB/PE]: 430080 /105 total 286720 /70 used 143360 /35 free
PVs: [AA] /dev/dasdc1 143360 /35 143360 /35 0 /0
      [AA] /dev/dasde1 143360 /35 0 /0 143360 /35
      [AA] /dev/dasdd1 143360 /35 143360 /35 0 /0
LV: [AWDS2 ] lv1 286720 /70 1x open
```

The released PV can now be removed from the VG:

```
# vgreduce vg1 /dev/dasde1
vgreduce -- doing automatic backup of volume group "vg1"
vgreduce -- volume group "vg1" successfully reduced by physical volume:
vgreduce -- /dev/dasde1
```

And the final configuration is:

```
# cat /proc/lvm
LVM driver version 0.8i (02/10/1999)

Total: 1 VG 2 PVs 1 LV (1 LV open 1 times)

Global: 5161 bytes vmlalloced IOP version: 5 3:26:51 active

VG: vg1 [2 PV, 1 LV/1 open] PE Size: 4096 KB
Usage [KB/PE]: 286720 /70 total 286720 /70 used 0 /0 free
PVs: [AA] /dev/dasdc1 143360 /35 143360 /35 0 /0
      [AA] /dev/dasdd1 143360 /35 143360 /35 0 /0
LV: [AWDS2 ] lv1 286720 /70 1x open
```

The PV `/dev/dasde1` is now replaced by `/dev/dasdd1`—and this was all done without unmounting the LV. You only have to reboot if you add new DASDs to your system, or if you have to take DASDs out of your system.

For SuSE systems, LVM can be managed by `yast`; see 6.5, “Setting up LVM with YaST” on page 108.





## High Availability using clusters

High Availability (HA) is a term that has been given many meanings. One such meaning is to utilize clustering techniques to maintain high availability in case of server failure. In this chapter we will show how to create Linux virtual servers to set up a High Availability Linux cluster, and describe what is still needed to complete a viable solution in Linux for S/390.

Much of the information in this chapter was adapted from multiple documents found on the Linux Virtual Server project Web site:

<http://linuxvirtualserver.org>

## 18.1 What is a Linux cluster

There are several categories of clustering applications or technologies, as discussed in the following sections.

### ***Parallel performance clusters***

With *parallel performance clusters*, a large number of individual machines are put together to act as a single powerful machine. This kind of clustering technique is commonly seen in scientific research where a very complex problem is divided into many parts in order to speed up the compute process. This type of parallelism is seen most often in applications such as weather forecasting or matrix calculations. It requires communication between the servers and complex algorithms.

One open source solution for this type of clustering technology is Beowulf. For more information, refer to the IBM Redbook *Linux HPC Cluster Installation*, SG24-6041, on the Web at:

<http://www.redbooks.ibm.com/abstracts/sg246041.html>

### ***Workload balancing cluster***

Another type of clustering technology is the use of *workload balancing clusters*, sometimes also called *high availability* or *redundancy clusters* because they can be designed in a way that provides very high uptimes for a single application. A good example of this type of clustering is high availability Web servers.

This type of cluster can be implemented by running duplicate copies of an application like Web servers, and having a monitoring system that detects the failure of a member of the cluster. Additionally, since many servers are handling the load, the failure of one server is not considered catastrophic. We discuss this kind of clustering technology in more detail in 18.2, “HA workload balancing Linux cluster for S/390” on page 317.

### ***Grid computing***

The last clustering technology is a modification of parallel performance clusters. This is sometimes referred as *grid computing* because it uses a large network (like the Internet) to allow many individual machines to download data, work on it at their own pace, and transfer the results back. These clusters do not require communication between servers to process the data. A well-known example of this type of clustering is the SETI@home project.

Most file systems can be shared read-only between nodes in a cluster. However, certain file systems (such as GFS) have the ability to share file systems read-write.

## 18.2 HA workload balancing Linux cluster for S/390

One example of HA workload balancing clusters is the Linux Virtual Server (LVS) methodology. (LVS clusters are sometimes referred to as RAILS, or Redundant Arrays of Inexpensive Linux Servers.)

LVS is an open source project that provides a set of kernel modifications and drivers for setting up a load balancing cluster using Linux systems; to outside clients, this cluster of servers appears to be one single server.

This illusion is called a *virtual server*. Virtual servers are highly scalable and highly available servers. The architecture of the cluster is totally transparent to the clients, and the clients require no changes.

A typical configuration for HA is composed of 2 directors and 1 or more real servers, as shown in Figure 18-1.

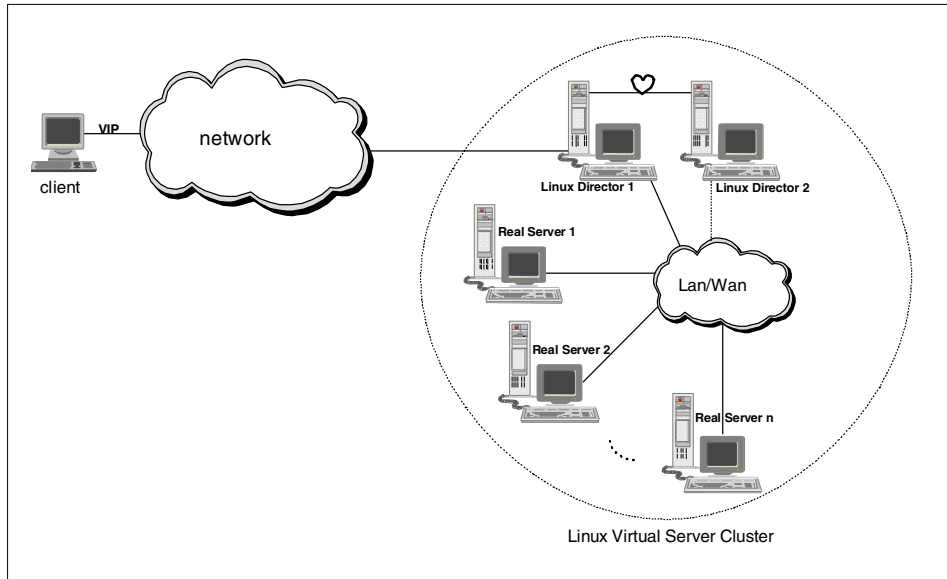


Figure 18-1 Typical workload balancing cluster

The Linux directors handle the incoming IP (VIP) traffic from the clients and, based on a set of workload balancing algorithms, the request is sent to the real servers. The Linux directors are constantly checking a heartbeat between them so that in case of failure, the other director can do an IP (VIP) takeover. Scalability is attained by allowing dynamic changes to the real server farm.

High availability is not provided by the LVS itself; instead, HA is provided by other software available (monitoring tools) used in conjunction with the LVS to detect node failures of either real servers or directors, and to dynamically change the configuration in order to remove or add nodes.

A Linux director is considered a layer 4 switch. The director makes decisions based on the IP information on the headers of packets going between the clients and the real servers. It does not look into the content of the packets. The director is basically a router, with routing tables set up for the LVS function. The real servers are the workhorses of the clusters; they provide the actual service to the clients.

## 18.3 Linux virtual server architectures

The Linux virtual server is currently available in three different network architectures.

### 18.3.1 Virtual server via network address translation

This type of server is sometimes abbreviated VS-NAT. It uses private IP addresses that cannot be seen by users outside of the network. It relies on the fact that you can manipulate the source and destination addresses contained in the headers—so that clients may think they are contacting one IP address when in fact they are being redirected to a different server—which in turn believes it is being contacted directly by the client. This kind of IP masquerading makes it possible to build a virtual server. A possible architecture of a virtual server via NAT is shown in Figure 18-2 on page 319.

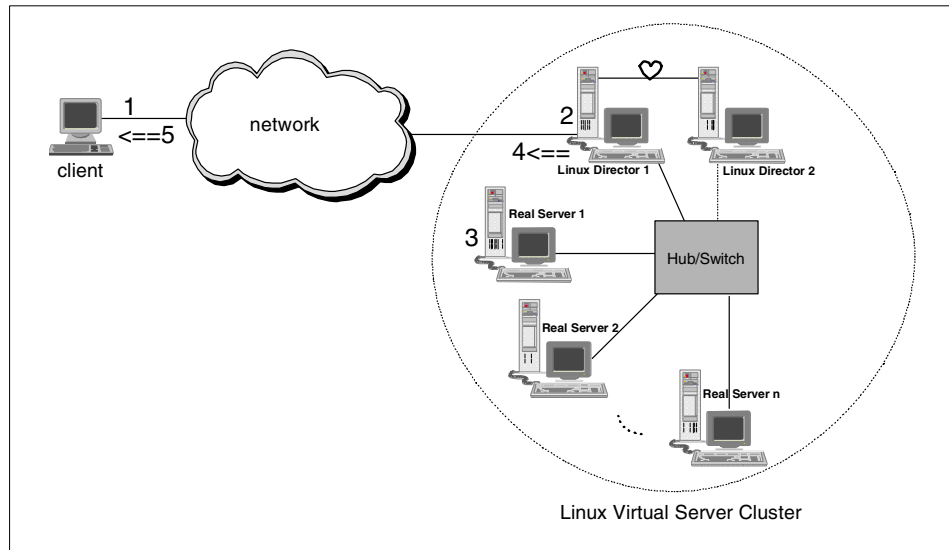


Figure 18-2 Virtual server via Network Address Translation (VS-NAT)

Real servers are usually running the same application, either sharing the file system, or with duplicate files, and the flow as follows:

- ▶ A request comes from a client into the Linux Director using a VIP (1).
- ▶ The Linux Director checks the package destination address and port (2). If there is a match for the Linux virtual server service based on the LVS rules, a real server is chosen via a workload balancing algorithm and the connection is added into the hash table, which maintains all connections.
- ▶ The Linux director then rewrites the destination address and the port of the packet to the chosen real server and forwards the packet to the server (3). The real server acts upon it and replies via the Linux director.
- ▶ The director (4) rewrites the source address of the packets to those of the virtual IP(VIP).
- ▶ It returns the request back to the client (5). Whenever the connection terminates or times out, the connection record is removed from the hash table.

The advantage of the NAT configuration is that only *one real IP* address is needed for the VIP; all other addresses can be private. Also, any real server that supports TCP/IP, regardless of operating system, can be supported as a real server.

The disadvantage of this configuration is that the scalability of the Linux director is not very good, due to possible bottlenecks: every request and response to the client has to be manipulated by the Linux director.

### 18.3.2 Virtual server via IP tunneling

This type of server is sometimes abbreviated VS-TUN. *IP tunneling* or *IP encapsulation* is the technique used to package an IP packet within another IP packet (IPIP encapsulation). In this LVS architecture, a packet is IPIP encapsulated and forwarded to the real server depending upon the workload algorithm. The real server then replies directly to the client. A possible configuration for tunneling is shown in Figure 18-3.

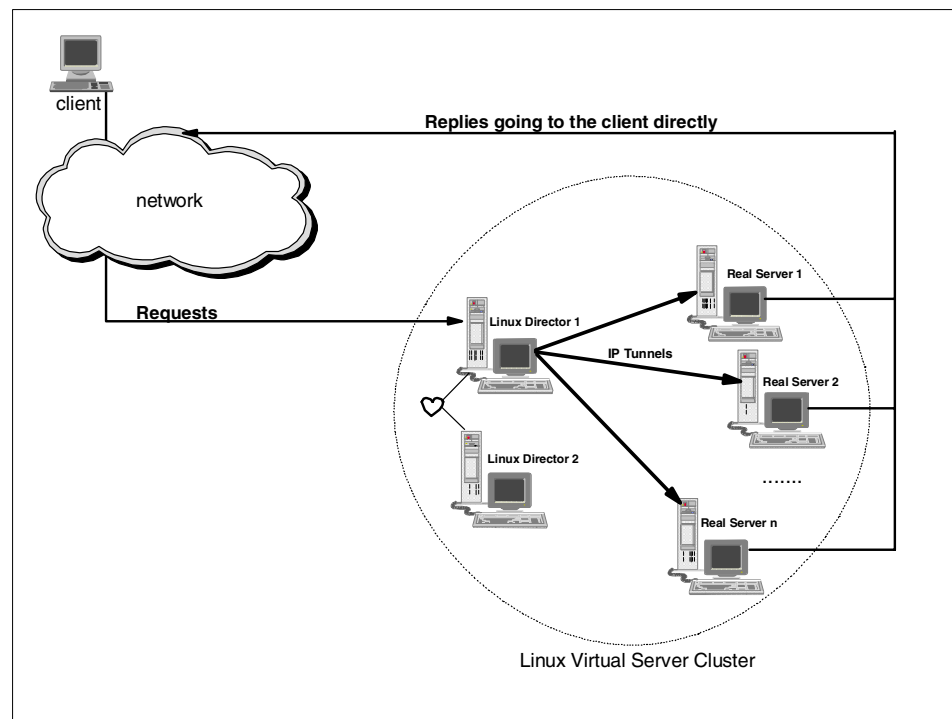


Figure 18-3 Virtual Server via IP tunneling (VS-TUN)

The flow in this case is as follows:

- ▶ A client sends a request to the VIP address, which in our case is Linux Director 1.
- ▶ Linux Director 1 receives the request and checks the IP address and port against the virtual server services.

- ▶ If it matches, the Director adds the connection to the hash table (where all connections are maintained) and, based on a workload balanced algorithm, encapsulates the packet and forwards the request to a real server.  
Any other incoming packets are checked against the hash table and, if found, the packet is encapsulated and forwarded to the server.
- ▶ When the real server receives the packet, it processes the request and replies directly to the client.

The IP tunneling flow has the advantage that the Linux Director only schedules the incoming requests and the real servers return the replies directly to the clients. This reduces the bottleneck seen in the NAT architecture. Also, with this technique, real servers can be geographically disperse, in remote networks.

The only disadvantage is that IP tunneling is not yet available in all operating systems.

### 18.3.3 Virtual server via direct routing

This type of server is sometimes abbreviated VS-DR. The direct routing technique is similar to the one implemented by the IBM NetDispatcher, where the VIP is shared by the real servers and the Linux directors.

The director changes the MAC address on the incoming packets and forwards them to the real server. A possible direct routing architecture is shown in Figure 18-4 on page 322.

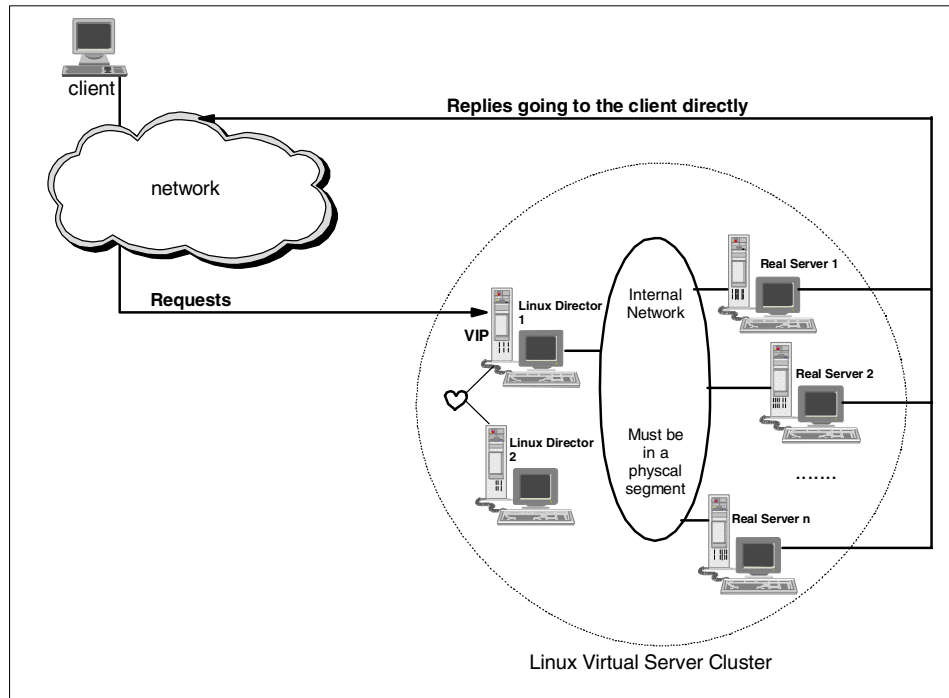


Figure 18-4 Virtual server via direct routing (VS-DR)

This flow proceeds as follows:

- ▶ The Linux director has an interface configured to the VIP (possibly as an alias to the eth0 device) where it receives the packets from the clients. The director, based on a workload balancing algorithm, selects a real server.
- ▶ The real servers can redirect the VIP address packets to a local socket (possibly the loopback device lo0) and process the packet. The real server replies directly to the client. The Linux director and the real servers must be physically connected via a hub or switch.

The advantage of the direct routing technique is that the real servers can be almost any OS. This technique has the same high scalability and throughput as that of VS-TUN.

The biggest disadvantage VS-DR is that the directors and real servers must both be in the same physical segment in order to arp each other.



## 18.4 Scheduling algorithms

The LVS project has implemented four scheduling algorithms under the 2.2.16 kernel for selecting real servers:

- ▶ **Round-Robin Scheduling** - This algorithm directs the connection to a different server in a round-robin manner. All servers are considered equal.
- ▶ **Weighted Round-Robin Scheduling** - This algorithm can treat real servers with a weight depending on their capacity. For example, if you have three servers ABC and you give them weights 4,3,2 respectively, a sample scheduling sequence would be ABCABCABA... This algorithm is not as taxing as the dynamic scheduling algorithms.
- ▶ **Least-Connection Scheduling** - This algorithm directs network connections to the real servers with the least number of connections. As one of the dynamic scheduling algorithms, it must count active connections for each real server.
- ▶ **Weighted Least-Connection Scheduling** - This algorithm allows you to assign a weight to the real servers in order to favor a particular server. For example, if you have three servers ABC and you give them weights of 4,3,2 respectively, a sample scheduling sequence would be ABCABACBA... The formula is that the next server to be scheduled is the one with the minimum ratio of active connections/weight.

## 18.5 High availability testing

Because of the brief amount of time available and to hardware resource constraints, you should be aware that the tests we ran during this residency were simplistic and not exhaustive. We tested with the VS-DR architecture. This architecture only supports Ethernet devices. We used two LPARS to do all our testing, which meant we had to test only one Linux director and one real server.

Prior to this test in Linux for S/390, we did a test using the exact same technique under SuSE Linux 7.1 for Intel. In that test we were able to see the workload balancing algorithms at work.

Our configuration is shown in Figure 18-5 on page 324.

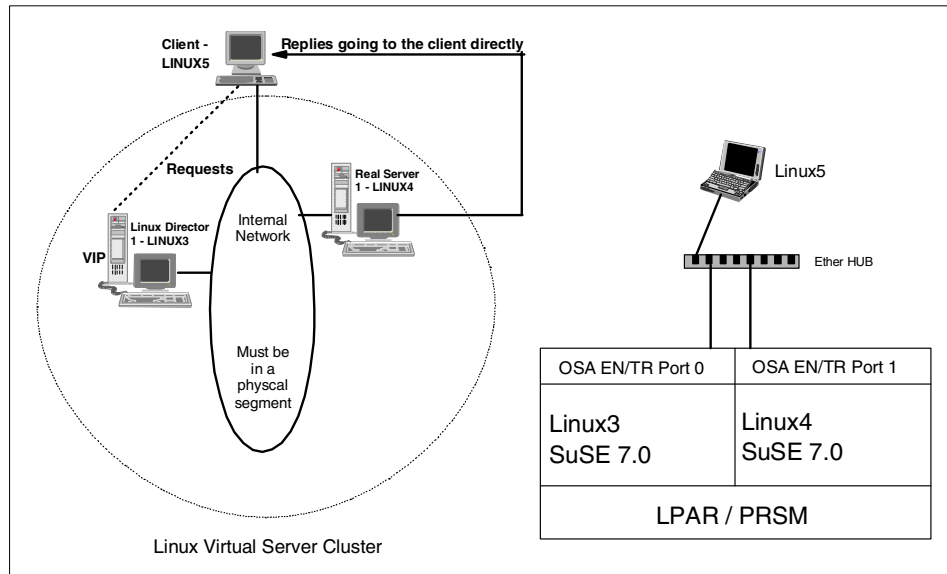


Figure 18-5 Linux Virtual Server test configuration

We used the following addresses:

Host name	IP address	Role
Linux3	192.168.2.3	Director
Linux4	192.168.2.4	Real server
Linux5	192.168.2.5	Client

### 18.5.1 The Linux virtual server patch and test

We downloaded the kernel patch from:

<http://linuxvirtualserver.org/>

We untarred it, changed directory to ipvs, and checked the README file:

```
linux3: # tar xzfBp /root/ipvs-0.9.15-2.2.16.tar.gz
linux3: # cd ipvs-0.9.15-2.2.16/
linux3: # ls
COPYING ChangeLog contrib ipvsadm
CREDITS README ipvs-0.9.15-2.2.16.patch net-tools-1.54-ipvs.patch
```

Based on the README file, we changed directory to /usr/src/linux and applied the patch; however, an error was encountered:

```
linux3: # patch -sp1 < /usr/src/ipvs-0.9.15-2.2.16/ipvs-0.9.15-2.2.16.patch
```

```
1 out of 2 hunks FAILED - saving rejects to file include/linux/sysctl.h.rej
```

After analyzing the rejects we were able to build a new patch (which can also be done by changing the original LVS patch as stated below):

```
linux3: # diff -urN linux-2.2.16/include/linux/sysctl.h
linux-2.2.16-vs-0.9/include/linux/sysctl.h
--- linux-2.2.16/include/linux/sysctl.hFri Jun 16 11:19:41 2000
+++ linux-2.2.16-vs-0.9/include/linux/sysctl.hWed Jun 14 17:37:18 2000
@@ -274,6 +275,29 @@
     <== line numbers changed
     NET_IPV4_CONF_BOOTP_RELAY=10,
     NET_IPV4_CONF_LOG_MARTIANS=11,
     NET_IPV4_CONF_HIDDEN=12,      <== we added the comma
     NET_IPV4_CONF_ARPFILTER=13   <== we added this line
+};
+
+/* /proc/sys/net/ipv4/vs */
+
+enum
+{
+ NET_IPV4_VS_AMEMTHRESH=1,
+ NET_IPV4_VS_AMDROPRATE=2,
+ NET_IPV4_VS_DROP_ENTRY=3,
+ NET_IPV4_VS_DROP_PACKET=4,
+ NET_IPV4_VS_SECURE_TCP=5,
+ NET_IPV4_VS_TO_ES=6,
+ NET_IPV4_VS_TO_SS=7,
+ NET_IPV4_VS_TO_SR=8,
+ NET_IPV4_VS_TO_FW=9,
+ NET_IPV4_VS_TO_TW=10,
+ NET_IPV4_VS_TO_CL=11,
+ NET_IPV4_VS_TO_CW=12,
+ NET_IPV4_VS_TO_LA=13,
+ NET_IPV4_VS_TO_LI=14,
+ NET_IPV4_VS_TO_SA=15,
+ NET_IPV4_VS_TO_UDP=16,
+ NET_IPV4_VS_TO_ICMP=17,
+ };
+
+/* /proc/sys/net/ipv6 */
```

We applied the new patch successfully:

```
linux3: # patch -sp1 < /root/sysctl-2.2.16.patch
```

Now that we had a patched kernel, we had to change the kernel compile options via the commands **make menuconfig** or **xconfig**.

We wanted to include the new Linux virtual server options by selecting at least the following options:

```
Code maturity level options ---
  [*] Prompt for development and/or incomplete code/drivers
Networking options ---
  [*] Network firewalls
  ....
  [*] IP: firewalling
  ....
  [*] IP: masquerading
  ....
  [*] IP: masquerading virtual server support
(12) IP masquerading table size (the Nth power of 2)
<M> IPVS: round-robin scheduling
<M> IPVS: weighted round-robin scheduling
<M> IPVS: least-connection scheduling
<M> IPVS: weighted least-connection scheduling
....
  [*] IP: aliasing support
```

Once we exited and saved the new options, we had to recompile the kernel and reboot with the new kernel:

```
linux3: # make dep image modules modules_install
linux3: # cp System.map to /boot/System.map
linux3: # cd arch/s390/boot
linux3: # cp image /boot/image-ipvs
linux3: # cd /boot
linux3: # mv System.map-2.2.16 System.map-old
linux3: # ln -s System.map System.map-2.2.16
linux3: # silo -f image-ipvs -b ipleckd.boot -p parmfile -d/dev/dasda
```

This puts the new image into /dev/dasda, and the system can be rebooted:

```
linux3: # shutdown -r now
```

After the system came up, we logged in and compiled the **ipvsadm** command. After that was successful, we tested it by simply issuing the command.

**Note:** In your case, if the response you receive does not look like the one in the example, you might not have booted up with the new kernel image:

```
linux3: # cd /usr/src/cd ipvs-0.9.15-2.2.16/ipvsadm
linux3: # make
linux3: # make install
linux3: # ipvsadm
IP Virtual Server version 0.9.15 (size=4096)
Prot LocalAddress:Port Scheduler Flags
-> RemoteAddress:Port          Forward Weight ActiveConn InActConn
```

Now we were ready to add a service to the virtual server. The service we tested was Telnet, because it is easy to identify (if logged in to the director or the real server).

First we configured the Virtual IP address in the director, turned on IP forwarding, and then added the actual service with the `ipvsadm` command.

```
linux3: # ifconfig eth0:0 192.168.2.10 netmask 255.255.255.255 broadcast
        192.168.2.10 up
linux3: # route add -host 192.168.2.10 dev eth0:0
linux3: # echo 1 > /proc/sys/net/ipv4/ip_forward
linux3: # ipvsadm -A -t 192.168.2.10:23 -s wlc
linux3: # ipvsadm -a -t 192.168.2.10:23 -r 192.168.2.4 -g
linux3: # ifconfig
eth0      Link encap:Ethernet  HWaddr 10:00:5A:D1:1C:8C
          inet addr:192.168.2.3  Bcast:192.168.2.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1492  Metric:1
          RX packets:1547 errors:0 dropped:0 overruns:0 frame:0
          TX packets:958 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100

eth0:0    Link encap:Ethernet  HWaddr 10:00:5A:D1:1C:8C
          inet addr:192.168.2.10  Bcast:192.168.2.10  Mask:255.255.255.255
          UP BROADCAST RUNNING MULTICAST  MTU:1492  Metric:1

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:3924  Metric:1
          RX packets:20 errors:0 dropped:0 overruns:0 frame:0
          TX packets:20 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0

linux3: # route
Kernel IP routing table
Destination Gateway      Genmask          Flags Metric Ref  Use Iface
linux3.demo.ibm *            255.255.255.255 UH    0     0   0  eth0
192.168.2.0 *            255.255.255.0   U     0     0   0  eth0
loopback   *            255.0.0.0       U     0     0   0  lo
```

The real server needs to be configured with the VIP also, because the director only changes the MAC address of the header and forwards the request—while the real server needs to configure the VIP on its loopback alias interface to process the packet locally.

```
linux4: # ifconfig lo:0 192.168.2.10 netmask 255.255.255.255 broadcast
        192.168.2.10 up
linux4: # route add -host 192.168.2.10 dev lo:0
linux4: # ifconfig
eth0      Link encap:Ethernet  HWaddr 10:00:5A:D1:1C:4C
```

```

        inet addr:192.168.2.4 Bcast:192.168.2.255 Mask:255.255.255.0
        UP BROADCAST RUNNING MULTICAST MTU:1492 Metric:1
        RX packets:507 errors:0 dropped:0 overruns:0 frame:0
        TX packets:299 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:100

lo      Link encap:Local Loopback
        inet addr:127.0.0.1 Mask:255.0.0.0
        UP LOOPBACK RUNNING MTU:3924 Metric:1
        RX packets:18 errors:0 dropped:0 overruns:0 frame:0
        TX packets:18 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0

lo:0    Link encap:Local Loopback
        inet addr:192.168.2.10 Mask:255.255.255.255
        UP LOOPBACK RUNNING MTU:3924 Metric:1

```

```
linux4: # route
```

```
Kernel IP routing table
```

Destination	Gateway	Genmask	Flags	Metric	Ref	Use
Iface						
192.168.2.10	*	255.255.255.255	UH	0	0	0 lo
192.168.2.0	*	255.255.255.0	U	0	0	0
eth0						
loopback	*	255.0.0.0	U	0	0	0 lo

Our test proceeded by logging in to the client 192.168.2.5 and telneting from that client to the VIP address service offering.

```

linux5: # telnet 192.168.2.10
Trying 192.168.2.10...
Connected to 192.168.2.10.
Escape character is '^]'.
Welcome to SuSE Linux 7.0 (s390) - Kernel 2.2.16 (0).

```

```

linux4 login: root
Password:
You have new mail in /var/spool/mail/root.
Last login: Thu Jun 21 11:39:20 from 192.168.2.5
Have a lot of fun...
linux4: # ifconfig
eth0    Link encap:Ethernet HWaddr 10:00:5A:D1:1C:4C
        inet addr:192.168.2.4 Bcast:192.168.2.255 Mask:255.255.255.0
        UP BROADCAST RUNNING MULTICAST MTU:1492 Metric:1
        RX packets:596 errors:0 dropped:0 overruns:0 frame:0
        TX packets:360 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:100

lo      Link encap:Local Loopback
        inet addr:127.0.0.1 Mask:255.0.0.0

```

```

UP LOOPBACK RUNNING MTU:3924 Metric:1
RX packets:18 errors:0 dropped:0 overruns:0 frame:0
TX packets:18 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0

lo:0      Link encap:Local Loopback
          inet addr:192.168.2.10 Mask:255.255.255.255
          UP LOOPBACK RUNNING MTU:3924 Metric:1

```

In your case, if you telnet to the Linux director (192.168.2.3) and issue an **ipvsadm** command, you should see something like the following:

```

linux3: # ipvsadm
IP Virtual Server version 0.9.15 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port          Forward Weight ActiveConn InActConn
TCP   linux3.demo.ibm.com:telnet wlc
  -> 192.168.2.4:telnet           Route    1      1          0

```

## 18.5.2 Testing LVS under z/VM using tunneling

After weeks of trying, we were finally successful in running the LVS under VM. We used the exact same technique to patch the kernel and to get the **ipvsadm** command installed.

Using the tunneling method, we were able to have Linux machines act as real servers connected via VCTC to VM's tcpip. Our configuration looked as shown in Figure 18-6 on page 330:

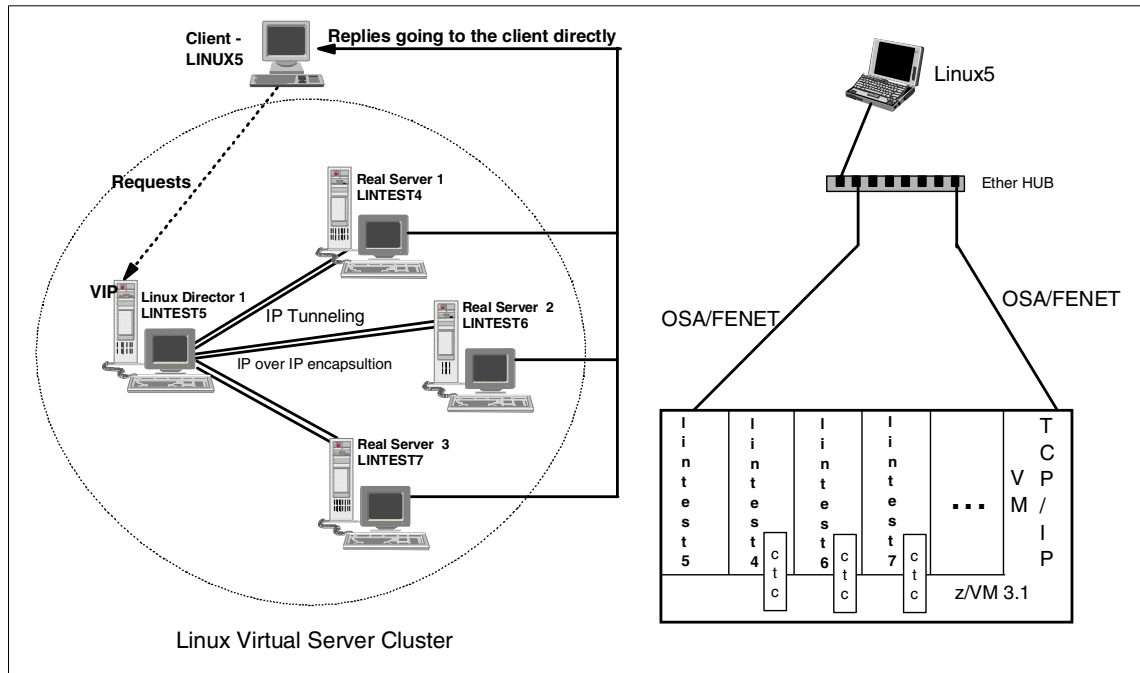


Figure 18-6 Linux virtual servers running under VM using tunneling

Host name	IP address	Role	Connection
lintest5	9.117.198.215	linux director	OSA Fenet
lintest4	9.117.198.214	real server 1	VCTC
lintest6	9.117.198.216	real server 2	VCTC
lintest7	9.117.198.217	real server 3	VCTC
linux5	9.117.200.101	client	Token Ring
VM51	9.117.198.25	VM	OSA Fenet

### Changes in configuration

We had to make two changes in the configuration of Linux.

We made the first change because the installation of iproute2 was needed only in the director. We did the following:



- ▶ We installed iproute2 software
  - yast -> Choose/Install packages --->
  - Change/create configuration --->
  - Scroll down to “n” Network-Support (TCP/IP, UCP, Mail, News) --->
  - Select the package iproute2

We made the second change because support for tunneling as part of the kernel was necessary for all servers involved, directors and real servers. We did the following:

- ▶ We recompiled the kernel with tunneling support. (You can follow the steps detailed in 18.5.2, “Testing LVS under z/VM using tunneling” on page 329). Note that this has to be done in the real servers as well, because you must configure the tunl0 device in each real server with the VIP address to accept the incoming requests.
- Networking options ---

```

[*] Network firewalls
....
[*] IP: firewalling
....
[*] IP: masquerading
....
[*] IP: masquerading virtual server support
(12) IP masquerading table size (the Nth power of 2)
<M> IPVS: round-robin scheduling
<M> IPVS: weighted round-robin scheduling
<M> IPVS: least-connection scheduling
<M> IPVS: weighted least-connection scheduling
....
[*] IP: tunneling
[*] IP: aliasing support

```

Once we rebooted with the new kernel, the steps that we followed were the same as those in 18.5.1, “The Linux virtual server patch and test” on page 324. The only exception was that the network definitions and the `ipvsadm` command must define the real server services as tunneling.

Notice that we used the `ip` command. There is no documentation included for the `ip` command (except for a help note on the parameters), so for more information, you can check the following:

<http://www.linuxgrill.com/iproute2-toc.html>

In the Linux director, we had to configure a tunnel connection to each of the real servers by issuing the following commands:

```

lintest5: # ip tunnel add lin7 mode ipip remote 9.117.198.217 local
9.117.198.200

```

```

lintest5: # ip link set lin7 up
lintest5: # ip tunnel add lin6 mode ipip remote 9.117.198.216 local
9.117.198.200
lintest5: # ip link set lin6 up
lintest5: # ip tunnel add lin4 mode ipip remote 9.117.198.214 local
9.117.198.200
lintest5: # ip link set lin4 up

```

On each of the real servers we had to recompile the kernel to add tunneling support, and then issue the **ifconfig** command:

```

lintest4: # ifconfig tunl0 9.117.198.200 netmask 255.255.255.255 broadcast
9.117.198.200 up
...
lintest6: # ifconfig tunl0 9.117.198.200 netmask 255.255.255.255 broadcast
9.117.198.200 up
...
lintest7: # ifconfig tunl0 9.117.198.200 netmask 255.255.255.255 broadcast
9.117.198.200 up

```

Now we had to add the LVS services to the Linux director—but this time we needed to specify that we were using tunneling instead of direct routing. This can be done by using the **-i** flag when adding a real server, instead of the **-g** flag:

```

lintest5: # ipvsadm -A -t 9.117.198.200:23 -s wlc
lintest5: # ipvsadm -a -t 9.117.198.200:23 -r 9.117.198.214 -i
lintest5: # ipvsadm -a -t 9.117.198.200:23 -r 9.117.198.216 -i
lintest5: # ipvsadm -a -t 9.117.198.200:23 -r 9.117.198.217 -i
lintest5: # ipvsadm
IP Virtual Server version 0.9.15 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port          Forward Weight ActiveConn InActConn
TCP  lin200.pok.ibm.com:telnet wlc
  -> lintest6.pok.ibm.com:telnet Tunnel 1      0      0
  -> lintest4.pok.ibm.com:telnet Tunnel 1      0      0
  -> lintest7.pok.ibm.com:telnet Tunnel 1      0      0

```

The tests were done in the same manner and everything was successful. We also tested the LVS cluster using WebSphere AE 3.5 running in the Linux real servers, and demonstrated that we could support a high availability WebSphere under Linux.

This test was done both using persistence, and without using persistence, from the LVS perspective. What this means is that once a client is connected to a specific address, it always returns to that address (this will expire after a specific idle time).

One odd result we experienced was that the tunnel devices really didn't need to be defined in the Linux director. In our case, with the tunl0 device iconfiged in the real servers and the services defined to ipvsadm as tunneling, everything worked. We don't know whether this was an anomaly resulting from running under VM, or the way it's supposed to work.

### 18.5.3 The mon tool and test

For this test we used an open source monitoring tool called *mon*. It's a good general tool that can be used to monitor availability of services. These services can be anything that can be tested with software, and they can be local or networked services.

Note that mon itself doesn't do the testing, but instead uses auxiliary programs to do it. In essence, mon is simply a scheduler which, at a given interval, runs a program that checks a service; if the service is deemed not available by the program, mon then executes an alert program.

mon can continue to monitor for the availability of the service and, when it becomes available, it executes an **upa!ert** program. This makes mon very easily extensible by just writing new scripts or code to monitor a service and handling the results. For our tests we downloaded the mon tarball from:

<http://www.kernel.org/software/mon>

The version we used was mon-0.38.16. We installed **mon** on the Linux director. The test required us to add another real server to our configuration. Our new configuration is shown in Figure 18-7.

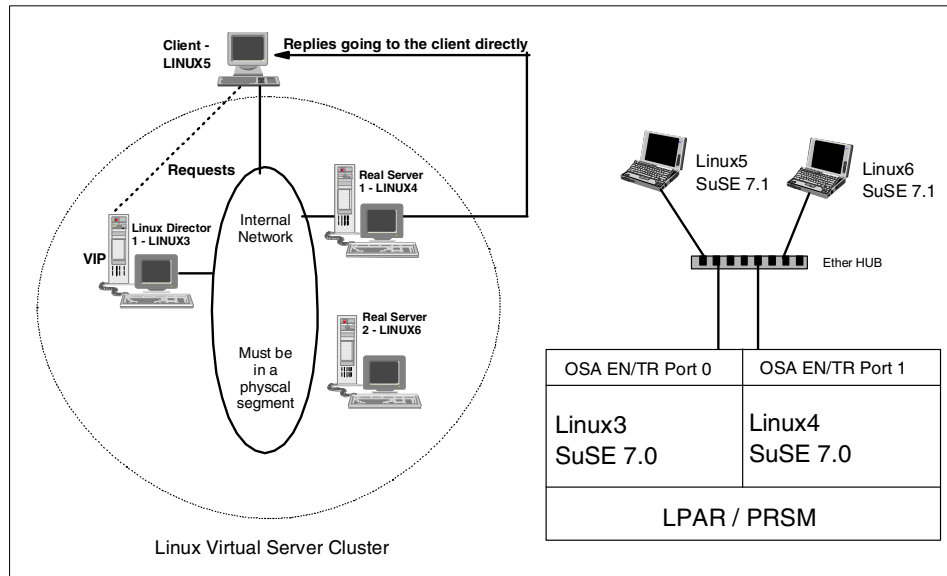


Figure 18-7 Test configuration with new server

We started by untaring mon and following the INSTALL document. We had to make small adjustments to make everything work.

Here is what we did:

```
linux3: # cd /usr/src
linux3: # tar xzfBp /root/mon-0.38.16.tar.gz
linux3: # cd mon-0.38.16
linux3: # less INSTALL
```

In your case, you should read the INSTALL document. In it you will find that you must get five extra Perl modules, which you can find at:

<http://search.cpan.org>

These modules are:

- ▶ Mon::<sup>\*</sup>
- ▶ Time::Period
- ▶ Time::HiRes
- ▶ Convert::BER
- ▶ Net::Telnet (if you want to use the telnet monitor)

We also had to convert to ph (Perl header files) the gcc lib “.h” files, in addition to the ones they specified. Here is our installation:

```

linux3: # cd /usr/include
linux3: # h2ph *.h sys/*.h asm/*.h
linux3: # cd /usr/lib/gcc-lib/s390-suse-linux/2.95.2/include
linux3: # h2ph *.h
linux3: # cd /usr/src
linux3: # tar xzfBp /root/Mon-0.11.tar.gz
linux3: # tar xzfBp /root/Net-Telnet-3.02.tar.gz
linux3: # tar xzfBp /root/Convert-BER-1.31.tar.gz
linux3: # tar xzfBp /root/Period-1.20.tar.gz
linux3: # tar xzfBp /root/Time-HiRes-01.20.tar.gz

```

In your case, you should then go into each of the directories of the Perl modules that you just untared and enter the following commands (remember to do this for each of the Perl modules):

```

linux3: # perl Makefile.PL
linux3: # make test install

```

The first command creates a makefile. The second makes (compiles and links), tests, and installs the Perl modules in the correct libraries.

**Note:** Before starting `mon`, a configuration file must be created. We based ours on the example found in the `etc` branch of the `mon` directory. The default location for the `mon` configuration file is the directory `/etc/mon/` which we created.

Because we wanted to control virtual server services, we found an `lvs.alert` sample in the Internet which we copied and moved to the `alert.d` directory. We also needed to make changes to the `S99mon` script in order to make it work in our environment. These are the files we used:

```

lvs.alert
#!/usr/bin/perl
#
# lvs.alert - Linux Virtual Server alert for mon
#
# It can be activated by mon to remove a real server when the
# service is down, or add the server when the service is up.
#
#
use Getopt::Std;
getopts ("s:g:h:t:l:P:V:R:W:F:u");

$ipvsadm = "/usr/sbin/ipvsadm";
$protocol = $opt_P;
$virtual_service = $opt_V;
$remote = $opt_R;

    if ($opt_u) {
        $weight = $opt_W;

```

```

        if ($opt_F eq "nat")
            { $forwarding = "-m"; }
        elsif ($opt_F eq "tun")
            { $forwarding = "-i"; }
        else
            { $forwarding = "-g"; }

        if ($protocol eq "tcp")
            { system("$ipvsadm -a -t $virtual_service -r $remote -w $weight
$forwarding"); }
        else
            { system("$ipvsadm -a -u $virtual_service -r $remote -w $weight
$forwarding"); }
    }
    else {
        if ($protocol eq "tcp")
            { system("$ipvsadm -d -t $virtual_service -r $remote"); }
        else
            { system("$ipvsadm -d -u $virtual_service -r $remote"); }
    };
};

mon.cf
#
# Very simple mon.cf file
#
# $Id: very-simple.cf,v 1.2 1999/08/16 00:25:06 trockij Exp $
#
alertdir   = /usr/src/mon-0.38.16/alert.d
mondirdir  = /usr/src/mon-0.38.16/mon.d
# maxprocs   = 20
histlength = 100
# randstart  = 60s

#
# define groups of hosts to monitor
#
hostgroup server1 192.168.2.4

hostgroup server2 192.168.2.6

watch server1
    service telnet
        interval 30s
        monitor telnet.monitor
        period wd {Mon-Fri}
        alert mail.alert root
        upalert mail.alert root
        alert lvs.alert -P tcp -V 192.168.2.10:23 -R 192.168.2.4 -W 1 -F dr

```

```

        upalert lvs.alert -P tcp -V 192.168.2.10:23 -R 192.168.2.4 -W 1 -F
dr -u

watch server2
    service telnet
        interval 30s
        monitor telnet.monitor
        period wd {Mon-Fri}
        alert mail.alert root
        upalert mail.alert root
        alert lvs.alert -P tcp -V 192.168.2.10:23 -R 192.168.2.6 -W 1 -F dr
        upalert lvs.alert -P tcp -V 192.168.2.10:23 -R 192.168.2.6 -W 1 -F
dr -u

```

### **auth.cf**

```

#
# authentication file
#
# entries look like this:
# command: {user|all}[,user...]
#
# THE DEFAULT IT TO DENY ACCESS TO ALL IF THIS FILE
# DOES NOT EXIST, OR IF A COMMAND IS NOT DEFINED HERE
#
list: all
servertime:all
reset: root
loadstate:root
savestate:root
term: root
stop: root
start: root
set: root
get: root
dump: root
disable:root
enable:root
test: root
ack: root
reload:root
clear: root

```

### **S99mon**

```

#!/bin/sh
## You probably want to set the path to include
# nothing but local filesystems. The script accepts two parameters,
# first parameter is either stop or start,
# the second parameter (if it is there) is the configuration file name

```

```

#
PATH=/bin:/usr/bin:/sbin:/usr/sbin
export PATH

M=/usr/src/mon-0.38.16
PID=/var/run/mon.pid
case "$1" in
    start)
        if [ -x $M/mon ]; then
            if [ $# = 2 ]; then
                $M/mon -c /etc/mon/$2 -a $M/alert.d -s $M/mon.d -f 2>/$M/errors
            else
                $M/mon -c /etc/mon/mon.cf -a $M/alert.d -s $M/mon.d -f 2>/$M/errors
            fi
        fi
        ;;
    stop)
        kill -TERM `cat $PID`
        ;;
    *)
        echo "Usage: mon {start|stop}"
        exit 1
esac

exit 0

```

In your case, when you copy the alert files into the directory `alert.d`, ensure that the owner and group are the same as the ones already in the directory (we had to execute a **chown** to correct them):

```
# chown 15473.948 *
```

In `mon.d`, the `monitors` directory, there are two monitors that require a makefile. If you are planning to use them, you must issue the `make` command. The **make install** command moves the executable to `/usr/lib/mon` (which doesn't exist). This directory is also used, by default, in some scripts and executables. In our testing, we simply created a symbolic link instead:

```
# ln -s mon-0.38.16 /usr/lib/mon
```

The command **ipvsadm** requires root authority. In order for `mon` to execute the `lvs.alert`, we had to change the owner and group of the `mon` script to root:

```
# chown root.root mon
```

The **mon** command does not move the man pages into the correct libraries; instead, they are found under the `doc` directory. You can view them by using the **nroff** command:

```
# nroff -man mon.1 | less
```



Our tests consisted of first checking the load balancing algorithm by using Telnet to login to the VIP (192.168.2.10). We telneted from the client (192.168.2.5) and made sure that we went to the least connection real server, or that it did a round robin between the real servers. This test was easy; we just did telnet twice to the VIP and we got to each of the real servers.

We tested the least connection algorithm by dropping a connection from the server that was not the next in line (in a round-robin style) and telneted again to see if we got the next real server (or the one with the fewest connections) and it worked.

After we tested that the director was working correctly, we shut down a real server (192.168.2.6) and verified, using the `ipvsadm` command, that we had drop the server from the virtual server tables. We then telneted to the VIP and we got into the real server that was available.

Here we checked our e-mail to corroborate that mon has sent the alert to root's mail id. Everything looked good, so we then re-booted the real server (192.168.2.6) and watched mon issue the upalert, indicating that the server was available. We verified, using the `ipvsadm` command, that it was added into the virtual server tables as an available real server. We retried our first test and again, everything worked.

As mentioned, these tests were done without any workload and were very simplistic in nature.

## 18.5.4 The IP takeover tool and test

The IP takeover tool we used was named *fake*. The fake utility is simply designed to enable a switch to a backup server by using arp spoofing to take over an IP address. The interface that is being taken over can be a physical or logical (alias) interface. This utility will add the interface via the `ifconfig` command, and it will do the route command for it.

The configuration that we used for testing the IP takeover procedure was the same as in the mon configuration, but in this case we used linux3 and linux4 as directors and linux6 as the real server (linux3 was the primary and linux4 was the backup).

Installing fake was really simple. You can download it from:

<http://www.ca.us.vergenet.net/linux/fake>

In this example, we used fake version 1.1.6. We untarred the file into `/usr/src/`. In your case, you should read the INSTALL document and follow the steps.

```
linux3: # tar xzfBp /root/fake-1.1.6.tar.gz
```

```
linux3: # cd fake-1.1.6
linux3: # less INSTALL
linux3: # make patch
linux3: # make
linux3: # make install
```

The next step was to configure fake to take over the VIP (192.168.2.10). The configuration files were placed into /etc/fake directory by the make install. We had to add a file for the VIP (192.168.2.10), and we changed the rate of the gratuitous arp to 10 seconds on the .fakerc file. These are the configurations files we used:

#### **.fakerc**

```
#####
# Set up basic environment for fake
# Variables are set as bash variables
# i.e. <VARIABLE>=<value>
#
# Must set:
# ARP_DELAY: Delay in seconds between gratuitous arp
# PID_DIR: Directory where pid files are kept
# INSTANCE_CONFIG_DIR: Directory where specific
# configuration files for an IP address takeover are kept
# LOG_DIR: Directory where logs are kept
# CLEAR_ROUTERS_FILE: New line delimited list of routers to rsh
# to and execute "clear arp-cache"
# FAKE_RSH: Programme to use to "rsh" to another machine
# to obtain macaddress by running ifconfig
#
# PATH can be set here to ensure that send_arp is in the
# path
#####

FAKE_HOME="/etc/fake"

#PATH=/sbin:/usr/sbin:/bin:/usr/bin

ARP_DELAY=10
CLEAR_ROUTERS_FILE="$FAKE_HOME/clear_routers"
PID_DIR="/var/run"
LOG_DIR="$FAKE_HOME/log"
INSTANCE_CONFIG_DIR="$FAKE_HOME/instance_config"

#Only needed if you wish to send gratuitous arp
#advertising the "real" mac address when turning fake off
#FAKE_RSH=ssh
```

### 192.168.2.10.cfg

```
SPOOF_IP=192.168.2.10
SPOOF_NETMASK=255.255.255.0
SPOOF_BROADCAST=192.168.2.255
TARGET_INTERFACE=eth0:0
```

```
#These are only needed if you wish to send gratuitous arp
#advertising the "real" mac address when turning fake off
#FOREIGN_INTERFACE=eth0
#FOREIGN_ARP=20
```

After modifying the configuration files, we were ready for some testing. We started our cluster using linux3 as our primary director. Both directors must have **fake**, **mon** and the lvs patch installed.

In our case, linux3 has fake and mon running, in addition to owning the cluster. The mon configuration for linux3 monitored the cluster and the backup director. The alert for the backup director simply sends a mail message. (This setup is similar to the mon test we did in the previous section.) linux4 had mon running to monitor the primary director and do an IP takeover in case it went down.

Here are the extra files for linux4 and commands we used.

#### secondary-mon.cf

```
#
# This file is to monitor the primary director from a backup director.
# It will inform root in system linux5.demo.ibm.com and it will execute
# the director.alert when the primary director is down.
#
# $Id: very-simple.cf,v 1.2 1999/08/16 00:25:06 trockij Exp $
#
alertdir  = /usr/src/mon-0.38.16/alert.d
mondir    = /usr/src/mon-0.38.16/mon.d
# maxprocs   = 20
histlength = 100
# randstart  = 60s

#
# define groups of hosts to monitor
#

hostgroup director 192.168.2.3
```

```

watch director
  service telnet
    interval 30s
    monitor ping.monitor
    period wd {Mon-Fri}
    alert mail.alert root@linux5.demo.ibm.com
    upalert mail.alert root@linux5.demo.ibm.com
    alert director.alert

```

#### **director alert**

```

#!/usr/bin/perl -d
#
# This alert is to just do an IP takeover.
# You can modify this alert to use options instead of
# ip numbers and services. I left the getopts and the
# assignment of fakeip as a sample.
#
#
use Getopt::Std;
getopts ("s:g:f:h:t:l:P:V:R:W:F:u");

$fakeip = $opt_f;

system("fake 192.168.2.10 &");
system("echo 1 > /proc/sys/net/ipv4/ip_forward");
system("ipvsadm -A -t 192.168.2.10:23 -s wlc");
system("ipvsadm -a -t 192.168.2.10:23 -r 192.168.2.6 -g");

```

Here are the commands we used to test the setup:

```

linux3: # fake 192.168.2.10 &
linux3: # echo 1 > /proc/sys/net/ipv4/ip_forward
linux3: # ipvsadm -A -t 192.168.2.10:23 -s wlc
linux3: # ipvsadm -a -t 192.168.2.10:23 -r 192.168.2.6 -g
linux3: # /etc/rc.d/S99mon start

linux4: # /etc/rc.d/S99mon start secondary-mon.cf

```

We tested the cluster by successfully telneting to the VIP. Next, we shut down linux3 and waited for the takeover process to occur. We checked linux4 to make sure that the IP takeover took place. We tested the cluster by telneting to the VIP. The test was successful. We then checked linux4 to make sure the session showed up in the lvs table by using the **ipvsadm** command.

## 18.6 Other tests needed and requirements

Because of the lack of time, the tests we did for this chapter were strictly function tests of each of the pieces. They were also done without any real workload. We did not consider what to do when the primary director came back up and the backup had done an IP takeover. That scenario is something that will be handled differently at each site.

A script is also needed that will re-start `mon` as the primary on the backup director when it takes over. This probably can be done by setting a new run level. We tested stopping `mon` from the alert and restarting it with a different configuration, but we encountered problems with reusing the IP port for `mon`.

There are other requirements for high availability. One major requirement missing in this configuration is for a file system that can support journaling and/or sharing in read/write mode. IBM's JFS just became available at the time this redbook was written. The other file system that promises to be of great service is Global File System (GFS) by Sistina. The file is open source and allows for read/write sharing of the file systems locally connected to different machines.

## 18.7 Using journaled file systems

We had difficulty getting journaled file systems to work on Linux for zSeries; however, we were able to get IBM JFS 1.0.1 working.

### 18.7.1 Using IBM JFS

Journaled file systems are very important for Linux; see 2.3.1, "Planning file systems" on page 24. We initially had difficulty getting any journaled file system to work and therefore had to use `ext2` file systems with LVM; however, as mentioned, we were able to get the IBM JFS 1.0.1 to work.

The installation was very clean under SuSE Linux 7.0 for S/390. We downloaded the kernel patch untarred and followed the `README` file to complete the installation:

```
# tar xzfBp /root/jfs-1.0.1-patch.tar.gz
# ls
README                jfs-2.4.0-v1.0.1-patch  linux-2.2.16.SuSE
jfs-2.2.14-v1.0.1-patch jfs-2.4.5-v1.0.1-patch  packages
jfs-2.2.16-v1.0.1-patch jfs-common-v1.0.1-patch
jfs-2.2.18-v1.0.1-patch linux
# less README
```

We needed to apply two patches: jfs-common-v1.0.1-patch and jfs-2.2.16-v1.0.1-patch. We obtained them from the jfs version 1.0.1 tarball:

```
# cd linux
# patch -sp1 < /usr/src/jfs-common-v1.0.1-patch
# patch -sp1 < /usr/src/jfs-2.2.16-v1.0.1-patch
```

We proceeded by configuring the Linux kernel and including the JFS support in the kernel (you have to prompt for code that is in development in order to see the JFS file system option). After saving the new configuration, we recompiled the kernel and rebooted with the new image:

```
# make menuconfig
```

```
Code maturity level options --->
```

```
 [*] Prompt for development and/or incomplete code/drivers
```

```
FileSystems --->
```

```
...
```

```
 [*] JFS filesystem support (experimental)
```

```
# make dep clean image modules modules_install
```

```
# cp System.map /boot/
```

```
# cd arch/s390/boot/
```

```
# cp image /boot/image-jfs
```

```
# cp ipleckd.boot /boot/ipleckd.boot-jfs
```

```
# cd /boot
```

```
# ls
```

```
System.map          image-jfs           ipldump.boot       parm.Bw16Vc
```

```
System.map-2.2.16  image.autoconf.h   ipleckd.boot       parmfile
```

```
boot.map           image.config       ipleckd.boot-jfs
```

```
image             image.version.h    iplfba.boot
```

```
# mv System.map-2.2.16 System.map-old
```

```
# ln -s System.map System.map-2.2.16
```

```
# silo -p parmfile -f image-jfs -b ipleckd.boot-jfs -d /dev/dasda
```

```
# shutdown -r now
```

After the system came up, we compiled and installed the jfs utilities:

```
# cd /usr/src/linux/fs/jfs/utils/
```

```
# ls
```

```
SPECS                extendfs  libfs     logredo   man_html  xchkdmp   xpeek
```

```
defrag              fsck      logdump   makefile  mkfs      xchklog
```

```
# make
```

```
# make install
```

We created a JFS file system as follows:

```
# mkfs -t jfs /dev/dasdc1
```

```
mkfs.jfs development version: $Name: v1_0_1 $
```

Warning! All data on device /dev/dasdc1 will be lost!

Continue? (Y/N) y

Format completed successfully.

```
143988 kilobytes total disk space.  
# mkdir /mnt/jfs1  
# mount -t jfs /dev/dasdc1 /mnt/jfs1  
# cd /mnt/jfs1
```

We ran some tests to make sure that the file system was sound by copying different directories and running the bonnie benchmark several times. We were also able to move the root file system to the JFS and boot from it successfully. Documentation on how to do that can be found at the JFS Web site.

Due to a lack of time, we were unable to do a serious test of how to recover a JFS. As of this writing, JFS version 1.0.2 was made available which contained fixes for SMP support (multiple CPUs). We were not able to test this version.

**Note:** Keep in mind that JFS version 1.0.2 just recently became available and the code is early on zSeries, so be sure to test your implementation thoroughly before moving the new file system into a production environment.





# Debugging and problem diagnosis

This chapter contains a discussion on debugging and problem diagnosis. There are a number of debugging tools that are suitable for use by application developers and system programmers. While there are overlaps, the needs of these two groups are treated separately, with the major focus of this chapter being on the needs of the system programmer. Support issues are also discussed, and collection of diagnostic data that may be requested by technical support personnel is described as well.

The purpose of this chapter is to provide a starting point for debugging and diagnosing errors. This topic alone is worthy of an entire redbook, and so only the fundamentals are covered within the space available.

The source of much of the information in this chapter was from a paper “Debugging on Linux for 390” by Joseph Barrow, which can be found on the Internet at:

<http://penguinvm.princeton.edu/notes.html#Debug390>

## 19.1 Linux diagnostic tools

For system programmers or system administrators, the main concern is system availability and availability of system services. System programmers will not be concerned if a program that has been written by a user has crashed or keeps crashing unless it impacts system availability in some way.

System programmers have an array of tools available to them that perhaps are not available to the developer. The developer will have access to the same Linux tools as the system programmer, but will likely be more interested in different features. The system programmer will often have tools and facilities beyond what is available on Linux. Under VM, there are very useful diagnostic features, and even under LPAR, there is some extra capability.

Following is a summary of many of the debugging and diagnosis tools.

### 19.1.1 objdump

For OS/390 system programmers, **objdump** has similar capabilities to the OS/390 service aid AMBLIST. A binary object (load module) can be mapped for symbols, and even disassembled (a note of caution is needed, however: while this is acceptable and intended under the GPL, it may breach the conditions of other licences).

The following lists all the options for objdump.

```
$ objdump --help
Usage: objdump [-ahifCdDprRtTxS1w] [-b bfdname] [-m machine] [-j
section-name]
    [--archive-headers] [--target=bfdname] [--debugging] [--disassemble]
    [--disassemble-all] [--disassemble-zeroes] [--file-headers]
    [--section-headers] [--headers]
    [--info] [--section=section-name] [--line-numbers] [--source]
    [--architecture=machine] [--reloc] [--full-contents] [--stabs]
    [--syms] [--all-headers] [--dynamic-syms] [--dynamic-reloc]
    [--wide] [--version] [--help] [--private-headers]
    [--start-address=addr] [--stop-address=addr]
    [--prefix-addresses] [--[no-]show-raw-insn] [--demangle]
    [--adjust-vma=offset] [-EB|-EL] [--endian={big|little}] objfile...
at least one option besides -l (--line-numbers) must be given
objdump: supported targets: elf32-s390 srec symbolsrec tekhex binary ihex
Report bugs to bug-gnu-utils@gnu.org
```

The following example shows how to get a report of **bash** (the **B**ourne **a**gain **s**hell), where it will be disassembled, and symbol tables produced (although symbols may not be produced unless the `gcc -g` option was used when compiled).

Note that the **-D** option will disassemble all sections, whereas the **-d** option only disassembles code sections. The following example shows that data can be misinterpreted as instructions; the sections `.interp` and `.note.ABI-tag` are both data sections, and disassembling them may lead to misleading results.

---

```
$ objdump -D /bin/bash > bash.objlst
objdump: /bin/bash: No symbols
$ cat bash.objlst|more

/bin/bash:      file format elf32-s390

Disassembly of section .interp:

004000f4 <.interp>:
  4000f4:      2f 6c                swr    %f6,%f12
  4000f6:      69 62 2f 6c         cd     %f6,3948(%r2,%r2)
  4000fa:      64 2e 73 6f         .long 0x642e736f
  4000fe:      2e 31                awr    %f3,%f1
  ...

Disassembly of section .note.ABI-tag:

00400104 <.note.ABI-tag>:
  400104:      00 00 00 04         .long 0x00000004
  400108:      00 00 00 10         .long 0x00000010
  40010c:      00 00 00 01         .long 0x00000001
  400110:      47 4e 55 00         bm     1280(%r14,%r5)
  400114:      00 00 00 00         .long 0x00000000
  400118:      00 00 00 02         .long 0x00000002
  ...
```

---

The report produced above is directed into the file `bash.objlst`, which can be viewed using any browser/editor of choice. In this instance it was printed on the screen by using the `cat` command.

A visual inspection of the section using the **-s** option can often confirm that a section contains character data only. However, the **-h** option will explicitly confirm the attributes, but not allow the data to be seen.

To print out a map of the sections, use the **--section-headers**, **--headers**, or **-h** options, which are the same. This will include the attributes of the section, and in particular, you should note whether it has the data or code attribute. See the following example.

---

```
$ objdump -h /bin/date|more
```

```
/bin/date:      file format elf32-s390
```

```
Sections:
```

Idx	Name	Size	VMA	LMA	File off	Algn
0	.interp	0000000d	004000f4	004000f4	000000f4	2**0
			CONTENTS, ALLOC, LOAD, READONLY, DATA			
1	.note.ABI-tag	00000020	00400104	00400104	00000104	2**2
			CONTENTS, ALLOC, LOAD, READONLY, DATA			
2	.hash	0000016c	00400124	00400124	00000124	2**2
			CONTENTS, ALLOC, LOAD, READONLY, DATA			
3	.dynsym	00000340	00400290	00400290	00000290	2**2
			CONTENTS, ALLOC, LOAD, READONLY, DATA			
4	.dynstr	000001fe	004005d0	004005d0	000005d0	2**0
			CONTENTS, ALLOC, LOAD, READONLY, DATA			
5	.gnu.version	00000068	004007ce	004007ce	000007ce	2**1
			CONTENTS, ALLOC, LOAD, READONLY, DATA			
6	.gnu.version_r	00000030	00400838	00400838	00000838	2**2
			CONTENTS, ALLOC, LOAD, READONLY, DATA			
7	.rela.got	00000024	00400868	00400868	00000868	2**2
			CONTENTS, ALLOC, LOAD, READONLY, DATA			
8	.rela.bss	00000060	0040088c	0040088c	0000088c	2**2
			.CONTENTS, ALLOC, LOAD, READONLY, DATA			
9	.rela.plt	000001ec	004008ec	004008ec	000008ec	2**2
			CONTENTS, ALLOC, LOAD, READONLY, DATA			
10	.init	00000074	00400ad8	00400ad8	00000ad8	2**2
			CONTENTS, ALLOC, LOAD, READONLY, CODE			
11	.plt	00000540	00400b4c	00400b4c	00000b4c	2**2
			CONTENTS, ALLOC, LOAD, READONLY, CODE			
12	.text	00005e48	00401090	00401090	00001090	2**3
			CONTENTS, ALLOC, LOAD, READONLY, CODE			
13	.fini	00000044	00406ed8	00406ed8	00006ed8	2**2
			CONTENTS, ALLOC, LOAD, READONLY, CODE			
14	.rodata	00000004	00406f1c	00406f1c	00006f1c	2**0
			CONTENTS, ALLOC, LOAD, READONLY, DATA			
15	.data	0000004c	00407f20	00407f20	00006f20	2**2
			CONTENTS, ALLOC, LOAD, DATA			
16	.eh_frame	00000004	00407f6c	00407f6c	00006f6c	2**2
			CONTENTS, ALLOC, LOAD, DATA			
17	.ctors	00000008	00407f70	00407f70	00006f70	2**2
			CONTENTS, ALLOC, LOAD, DATA			
18	.dtors	00000008	00407f78	00407f78	00006f78	2**2
			CONTENTS, ALLOC, LOAD, DATA			
19	.got	000000c4	00407f80	00407f80	00006f80	2**2
			CONTENTS, ALLOC, LOAD, DATA			
20	.dynamic	000000a0	00408044	00408044	00007044	2**2

		CONTENTS, ALLOC, LOAD, DATA				
21	.bss	000001b8	004080e4	004080e4	000070e4	2**2
		ALLOC				
22	.comment	00000214	00000000	00000000	000070e4	2**0
		CONTENTS, READONLY				
23	.note	00000118	00000214	00000214	000072f8	2**0
		CONTENTS, READONLY				

The next example shows a dump of sections in hex, followed by disassembled machine code back to assembly instructions of all sections:

```
$ objdump -Ds /bin/date|more
objdump: /bin/date: No symbols

/bin/date:      file format elf32-s390

Contents of section .interp:
 400f4 2f6c6962 2f6c642e 736f2e31 00          /lib/ld.so.1.
Contents of section .note.ABI-tag:
400104 00000004 00000010 00000001 474e5500  ....GNU.
400114 00000000 00000002 00000000 00000000  ....
Contents of section .hash:
400124 00000025 00000034 0000001a 00000026  ...%.4.....&
400134 00000005 00000000 00000018 00000000  ....
400144 00000009 00000000 00000001 0000002d  ....-
400154 00000028 00000031 00000032 00000000  ...(.1...2...
400164 0000002a 00000020 00000011 00000012  ...*...
400174 0000000c 00000016 00000000 00000004  ....
400184 00000010 0000000e 00000000 00000021  ....!
400194 00000014 00000008 00000017 0000002b  ....+
...
Contents of section .dynsym:
400290 00000000 00000000 00000000 00000000  ....
4002a0 00000001 00000000 00000000 20000000  ....
4002b0 00000010 00400d2c 0000010e 12000000  ....@.,.....
4002c0 00000028 00400bac 000000a0 12000000  ...(.@.....
4002d0 00000048 0040106c 0000000e 12000000  ...H.@.1.....
4002e0 0000004f 00400eac 000000aa 22000000  ...O.@....."....
...
Contents of section .dynstr:
4005d0 005f5f67 6d6f6e5f 73746172 745f5f00  .__gmon_start__.
4005e0 5f5f6465 72656769 73746572 5f667261  __deregister_fra
4005f0 6d655f69 6e666f00 5f5f7265 67697374  me_info.__regist
400600 65725f66 72616d65 5f696e66 6f006c69  er_frame_info.li
400610 62632e73 6f2e3600 73747263 70790074  bc.so.6.strcpy.t
400620 65787464 6f6d6169 6e007072 696e7466  extdomain.printf
...
Contents of section .gnu.version:
```

```

4007ce 00000000 00020002 00020002 00020002 .....
4007de 00020002 00020002 00020002 00020002 .....
4007ee 00020002 00020002 00020002 00020002 .....
4007fe 00020002 00020002 00020002 00020002 .....
40080e 00030002 00020002 00020002 00020002 .....
40081e 00020002 00030002 00010002 00020002 .....
40082e 00020002 00020002 .....
Contents of section .gnu.version_r:
400838 00010002 0000003e 00000010 00000000 .....>.....
400848 0d696911 00000003 000001ea 00000010 .ii.....
400858 0d696910 00000002 000001f4 00000000 .ii.....
Contents of section .rela.got:
400868 0040803c 0000030a 00000000 00408034 .@.<.....@.4
400878 0000020a 00000000 00408030 0000010a .....@.0....
400888 00000000 ....
Contents of section .rela.bss:
40088c 004080e4 00003209 00000000 004080e8 .@....2.....@..
40089c 00000709 00000000 004080ec 00002209 .....@.....".
4008ac 00000000 004080f0 00000d09 00000000 .....@.....
4008bc 004080f4 00000909 00000000 004080f8 .@.....@..
...
Contents of section .rela.plt:
4008ec 00407f8c 0000190b 00000000 00407f90 .@.....@..
4008fc 0000300b 00000000 00407f94 0000030b ..0.....@.....
40090c 00000000 00407f98 00001c0b 00000000 .....@.....
40091c 00407f9c 00000b0b 00000000 00407fa0 .@.....@..
40092c 00001f0b 00000000 00407fa4 0000150b .....@.....
...
Contents of section .init:
400ad8 90bff02c a7d50008 000000b0 ffbff520 .....,.....
400ae8 000074a0 181fa7fa ffa05010 f00058c0 ..t.....P...X.
400af8 d0081acd 5810d000 5811c000 1211a784 ...X...X.....
400b08 00075810 d0044111 d0000de1 a7d50004 ..X...A.....
400b18 000006ac 5810d000 4111d000 0de10707 ...X...A.....
400b28 a7d50004 00006334 5810d000 4111d000 .....c4X...A...
400b38 0de10707 a7d50002 5840f098 98bff08c .....X@.....
400b48 07f40707 ....
Contents of section .plt:
400b4c 5010f01c 0d105810 1012d203 f0181004 P.....X.....
400b5c 58101008 07f10000 00407f80 00000000 X.....@.....
400b6c 0d105810 10165810 100007f1 0d105810 ..X...X.....X.
400b7c 100ea7f4 ffe70000 00407f8c 00000000 .....@.....
400b8c 0d105810 10165810 100007f1 0d105810 ..X...X.....X.
...
Contents of section .text:
401090 0dd0a7da 00384140 f0045830 f000a708 .....8A@..X0....
4010a0 fff814f0 a7fa99f8 d703f000 f00090ef .....
4010b0 f0604170 f0605860 d0045850 d0005820 .`Ap.`X`..XP..X
4010c0 d0085810 d00c0de1 00000040 0ad80040 ..X.....@...@

```

```

4010d0 6ed80040 25dc0040 0e6c0000 00020001 n..@%..@.l.....
4010e0 90bff02c a7d5000e 00006e40 00006e3c ....,.....n@..n<
4010f0 000000b4 000000b8 fffffc44 00006e98 .....D..n.
...
.401e50 73706163 65730a00 0a526570 6f727420 spaces...Report
401e60 62756773 20746f20 3c627567 2d73682d bugs to <bug-sh-
401e70 7574696c 7340676e 752e6f72 673e2e00 utils@gnu.org>..
...

```

Disassembly of section .interp:

```

004000f4 <.interp>:
 4000f4:      2f 6c                swr    %f6,%f12
 4000f6:      69 62 2f 6c         cd     %f6,3948(%r2,%r2)
 4000fa:      64 2e 73 6f         .long 0x642e736f
 4000fe:      2e 31                awr    %f3,%f1
...

```

Disassembly of section .init:

```

00400ad8 <.init>:
 400ad8:      90 bf f0 2c         stm   %r11,%r15,44(%r15)
 400adc:      a7 d5 00 08         bras  %r13,0x400aec
 400ae0:      00 00 00 b0         .long 0x000000b0
 400ae4:      ff bf f5 20         .long 0xffbff520
 400ae8:      00 00 74 a0         .long 0x000074a0
 400aec:      18 1f                lr    %r1,%r15
 400aee:      a7 fa ff a0         ahi  %r15,-96
 400af2:      50 10 f0 00         st   %r1,0(%r15)
 400af6:      58 c0 d0 08         l    %r12,8(%r13)
 400afa:      1a cd                ar   %r12,%r13
 400afc:      58 10 d0 00         l    %r1,0(%r13)
 400b00:      58 11 c0 00         l    %r1,0(%r1,%r12)
 400b04:      12 11                ltr  %r1,%r1
 400b06:      a7 84 00 07         jz   0x400b14
 400b0a:      58 10 d0 04         l    %r1,4(%r13)
...

```

Disassembly of section .plt:

```

00400b4c <.plt>:
 400b4c:      50 10 f0 1c         st   %r1,28(%r15)
 400b50:      0d 10                basr  %r1,%r0
 400b52:      58 10 10 12         l    %r1,18(%r1)
...

```

Disassembly of section .text:

```

00401090 <.text>:
 401090:      0d d0                basr  %r13,%r0
 401092:      a7 da 00 38         ahi  %r13,56
 401096:      41 40 f0 04         la   %r4,4(%r15)
 40109a:      58 30 f0 00         l    %r3,0(%r15)

```

```

40109e:    a7 08 ff f8          lhi    %r0,-8
4010a2:    14 f0                nr     %r15,%r0
4010a4:    a7 fa ff 98          ahi    %r15,-104
4010a8:    d7 03 f0 00 f0 00   xc     0(4,%r15),0(%r15)
4010ae:    90 ef f0 60          stm    %r14,%r15,96(%r15)
4010b2:    41 70 f0 60          la     %r7,96(%r15)
4010b6:    58 60 d0 04          l      %r6,4(%r13)
4010ba:    58 50 d0 00          l      %r5,0(%r13)
4010be:    58 20 d0 08          l      %r2,8(%r13)
4010c2:    58 10 d0 0c          l      %r1,12(%r13)
4010c6:    0d e1                basr   %r14,%r1
4010c8:    00 00 00 40          .long 0x00000040
4010cc:    0a d8                svc    216
4010ce:    00 40 6e d8          .long 0x00406ed8
...

```

---

Use the `--section=<section name>` option if you are only interested in a specific section:

```

$ objdump -s --section=.init /bin/date|more

/bin/date:    file format elf32-s390

Contents of section .init:
400ad8 90bff02c a7d50008 000000b0 ffbff520  ....
400ae8 000074a0 181fa7fa ffa05010 f00058c0  ..t.....P...X.
400af8 d0081acd 5810d000 5811c000 1211a784  ...X...X.....
400b08 00075810 d0044111 d0000de1 a7d50004  ..X...A.....
400b18 000006ac 5810d000 4111d000 0de10707  ...X...A.....
400b28 a7d50004 00006334 5810d000 4111d000  .....c4X...A...
400b38 0de10707 a7d50002 5840f098 98bff08c  .....X@.....
400b48 07f40707                ....

```

Notice that, by using `--adjust-vma=nnn`, the offset is advanced by nnn bytes (nnn modulo 4096; that is, for nnn > 4 K, it will be divided by 4 K and the remainder used. This is because a page is 4 K, and therefore no displacement can be larger):

```

$ objdump -s --section=.init --adjust-vma=256 /bin/date|more

/bin/date:    file format elf32-s390

Contents of section .init:
400bd8 90bff02c a7d50008 000000b0 ffbff520  ....
400be8 000074a0 181fa7fa ffa05010 f00058c0  ..t.....P...X.
400bf8 d0081acd 5810d000 5811c000 1211a784  ...X...X.....
400c08 00075810 d0044111 d0000de1 a7d50004  ..X...A.....
400c18 000006ac 5810d000 4111d000 0de10707  ...X...A.....
400c28 a7d50004 00006334 5810d000 4111d000  .....c4X...A...

```



```

400c38 0de10707 a7d50002 5840f098 98bff08c .....X@.....
400c48 07f40707                ....

```

Following is the same section, disassembled:

---

```

$ objdump -d --section=.init /bin/date | more
objdump: /bin/date: No symbols

```

```

/bin/date:      file format elf32-s390

```

```

Disassembly of section .init:

```

```

00400ad8 <.init>:

```

```

400ad8:      90 bf f0 2c          stm    %r11,%r15,44(%r
400adc:      a7 d5 00 08          bras  %r13,0x400aec
400ae0:      00 00 00 b0          .long 0x000000b0
400ae4:      ff bf f5 20          .long 0xffbff520
400ae8:      00 00 74 a0          .long 0x000074a0
400aec:      18 1f                lr     %r1,%r15
400aee:      a7 fa ff a0          ahi   %r15,-96
400af2:      50 10 f0 00          st    %r1,0(%r15)
400af6:      58 c0 d0 08          l     %r12,8(%r13)
400afa:      1a cd                ar     %r12,%r13
400afc:      58 10 d0 00          l     %r1,0(%r13)
400b00:      58 11 c0 00          l     %r1,0(%r1,%r12)
..400b04:      12 11                ltr   %r1,%r1
400b06:      a7 84 00 07          jz    0x400b14
400b0a:      58 10 d0 04          l     %r1,4(%r13)
400b0e:      41 11 d0 00          la    %r1,0(%r1,%r13)
400b12:      0d e1                basr  %r14,%r1
400b14:      a7 d5 00 04          bras  %r13,0x400b1c
400b18:      00 00 06 ac          .long 0x000006ac
400b1c:      58 10 d0 00          l     %r1,0(%r13)
400b20:      41 11 d0 00          la    %r1,0(%r1,%r13)
400b24:      0d e1                basr  %r14,%r1
400b26:      07 07                bcr   0,%r7
400b28:      a7 d5 00 04          bras  %r13,0x400b30
400b2c:      00 00 63 34          .long 0x00006334
400b30:      58 10 d0 00          l     %r1,0(%r13)
400b34:      41 11 d0 00          la    %r1,0(%r1,%r13)
400b38:      0d e1                basr  %r14,%r1
400b3a:      07 07                bcr   0,%r7
400b3c:      a7 d5 00 02          bras  %r13,0x400b40
400b40:      58 40 f0 98          l     %r4,152(%r15)
400b44:      98 bf f0 8c          lm    %r11,%r15,140(%r15)
400b48:      07 f4                br    %r4
400b4a:      07 07                bcr   0,%r7

```

---

## 19.1.2 strace

The **strace** facility is quite similar to the Generalized Trace Facility (GTF) on OS/390. It allows tracing of system calls. At first this may seem limiting, but most events will involve a system call. For a list of Linux system calls, see:

[http://penguinvm.princeton.edu/notes.html#\\_Hlk471721277](http://penguinvm.princeton.edu/notes.html#_Hlk471721277)

These can be very useful to trace. All system calls will be traced, as the following example shows:

---

```
# strace date
execve("/bin/date", ["date"], [/* 25 vars */]) = 0
brk(0) = 0x40829c
open("/etc/ld.so.preload", O_RDONLY) = -1 ENOENT (No such file or
directory)
open("/etc/ld.so.cache", O_RDONLY) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=18019, ...}) = 0
old_mmap(NULL, 18019, PROT_READ, MAP_PRIVATE, 3, 0) = 0x40017000
close(3) = 0
open("/lib/libc.so.6", O_RDONLY) = 3
fstat(3, {st_mode=S_IFREG|0755, st_size=1276967, ...}) = 0
read(3, "\177ELF\1\2\1\0\0\0\0\0\0\0\0\0\3\243\220\0\0\0\1\0\1"... , 4096)
= 46
old_mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0
old_mmap(NULL, 1013884, PROT_READ|PROT_EXEC, MAP_PRIVATE, 3, 0) =
0x4001d000
mprotect(0x4010d000, 30844, PROT_NONE) = 0
old_mmap(0x4010d000, 20480, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED, 3,
0xe0
old_mmap(0x40112000, 10364, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANOO
close(3) = 0
munmap(0x40017000, 18019) = 0
personality(PER_LINUX) = 0
getpid() = 2113
brk(0) = 0x40829c
brk(0x4082d4) = 0x4082d4
brk(0x409000) = 0x409000
time([992616771]) = 992616771
open("/etc/localtime", O_RDONLY) = 3
read(3, "TZif\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\3\0\0\0\3\0"... , 44) =
44
read(3, "\236\246\36p\237\272\353`\240\206\0p\241\232\315`\242e"... , 1170)
= 110
fstat(3, {st_mode=S_IFREG|0644, st_size=1250, ...}) = 0
old_mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0
```

```

read(3, "\377\377\307\300\1\0\377\377\271\260\0\4\377\377\307\300"...,
4096) = 6
close(3) = 0
munmap(0x40017000, 4096) = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 0), ...}) = 0
old_mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0
ioctl(1, TCGETS, {B9600 opost isig icanon echo ...}) = 0
write(1, "Fri Jun 15 10:52:51 EDT 2001\n", 29Fri Jun 15 10:52:51 EDT 2001
) = 29
close(1) = 0
munmap(0x40017000, 4096) = 0
_exit(0) = ?

```

---

The following simple example demonstrates the power and use of the **strace** facility when combined with **grep**:

```

# strace /etc/rc.d/init.d/syslog status 2>&1|grep open
open("/etc/ld.so.preload", O_RDONLY) = -1 ENOENT (No such file or
directory)
open("/etc/ld.so.cache", O_RDONLY) = 3
open("/lib/libdl.so.2", O_RDONLY) = 3
open("/lib/libc.so.6", O_RDONLY) = 3
open("/dev/tty", O_RDWR|O_NONBLOCK|O_LARGEFILE) = 3
open("/etc/rc.d/init.d/syslog", O_RDONLY|O_LARGEFILE) = 3
open("/etc/rc.d/init.d/functions", O_RDONLY|O_LARGEFILE) = 3
# strace date 2>&1|grep open
open("/etc/ld.so.preload", O_RDONLY) = -1 ENOENT (No such file or
directory)
open("/etc/ld.so.cache", O_RDONLY) = 3
open("/lib/libc.so.6", O_RDONLY) = 3
open("/etc/localtime", O_RDONLY) = 3

```

---

There other options that are documented in the man pages, but the **-f** option will cause **strace** to trace forked processes too, as shown in the following example:

```

# strace -f /etc/rc.d/init.d/syslog status 2>&1|grep open
open("/etc/ld.so.preload", O_RDONLY) = -1 ENOENT (No such file or
directory)
open("/etc/ld.so.cache", O_RDONLY) = 3
open("/lib/libdl.so.2", O_RDONLY) = 3
open("/lib/libc.so.6", O_RDONLY) = 3
open("/dev/tty", O_RDWR|O_NONBLOCK|O_LARGEFILE) = 3
open("/etc/rc.d/init.d/syslog", O_RDONLY|O_LARGEFILE) = 3
open("/etc/rc.d/init.d/functions", O_RDONLY|O_LARGEFILE) = 3
[pid 2110] open("/etc/ld.so.preload", O_RDONLY) = -1 ENOENT (No such file
or d)

```

```

[pid 2110] open("/etc/ld.so.cache", O_RDONLY) = 3
[pid 2110] open("/lib/libc.so.6", O_RDONLY) = 3
[pid 2110] open("/dev/null", O_RDONLY|O_NONBLOCK|O_DIRECTORY) = -1 ENOTDIR
(No)
[pid 2110] open("/proc", O_RDONLY|O_NONBLOCK|O_DIRECTORY) = 3
[pid 2110] open("/proc/1/stat", O_RDONLY) = 4
[pid 2110] open("/proc/1/cmdline", O_RDONLY) = 4
[pid 2110] open("/proc/2/stat", O_RDONLY) = 4
[pid 2110] open("/proc/2/cmdline", O_RDONLY) = 4
[pid 2110] open("/proc/3/stat", O_RDONLY) = 4
[pid 2110] open("/proc/3/cmdline", O_RDONLY) = 4
[pid 2110] open("/proc/4/stat", O_RDONLY) = 4
[pid 2110] open("/proc/4/cmdline", O_RDONLY) = 4
[pid 2110] open("/proc/5/stat", O_RDONLY) = 4
[pid 2110] open("/proc/5/cmdline", O_RDONLY) = 4
[pid 2110] open("/proc/6/stat", O_RDONLY) = 4
...
[pid 2111] open("/proc/2107/stat", O_RDONLY) = 4
[pid 2111] open("/proc/2107/cmdline", O_RDONLY) = 4
[pid 2111] open("/proc/2108/stat", O_RDONLY) = 4
[pid 2111] open("/proc/2108/cmdline", O_RDONLY) = 4
[pid 2111] read(4, "grep\0open", 1024) = 9
[pid 2111] open("/proc/2109/stat", O_RDONLY) = 4
[pid 2111] open("/proc/2109/cmdline", O_RDONLY) = 4
[pid 2111] open("/proc/2111/stat", O_RDONLY) = 4
[pid 2111] open("/proc/2111/cmdline", O_RDONLY) = 4

```

---

### 19.1.3 ulimit

To allow core dumps to be taken, the `ulimit` command must be used. It can also be used to check to see that core dumps have been enabled. The `ulimit` command allows user limits to be displayed and set.

The `ulimit -a` command will display what the user limits are, as the following example shows:

```

$ ulimit -a
core file size (blocks)      0
data seg size (kbytes)      unlimited
file size (blocks)          unlimited
max locked memory (kbytes)  unlimited
max memory size (kbytes)    unlimited
open files                  1024
pipe size (512 bytes)       8
stack size (kbytes)         8192
cpu time (seconds)          unlimited
max user processes          256
virtual memory (kbytes)     unlimited

```

In that example, the core file size is currently set to 0 blocks, which means no core dumps can be taken. To change this and set the blocks to an unlimited amount, use the following:

```
$ ulimit -c unlimited
```

Superuser authority is required to set the core file size, although it appears that Turbolinux will allow a normal user to reduce or zero the size, but not increase it. This only becomes apparent when an actual number is used, and then the following message is received in response:

```
$ ulimit -c 1000
bash: ulimit: cannot modify limit: Operation not permitted
```

To reset to the default core file size, if it was non-zero, login again.

## 19.1.4 gdb

For OS/390 system programmers, **gdb** has similar capabilities to IPCS insofar as it has features that allow reading and analyzing dumps. It also contains features that allow interactive debugging, but these are of more interest to developers.

For system programmers, the interesting thing is that **gdb** can be used to read core dumps. A core dump is taken when a program crashes or fails (hanging is another matter, but a dump can be generated when crashed using **kill**). The dump *and* the binary object file is required for the version of the program that failed.

Use the **ulimit** command to ensure that core dumps can be taken. When a program crashes, the core dump generated will be left in the file called **core** in the home directory for that process. The command format when reading a core dump is:

```
gdb <program binary file> <core dump file>
```

When you use **gdb**, it's like entering a "gdb session" in which there are subcommands that will display different data. The following example shows its invocation and messages. Notice the **(gdb)** prompt.

---

```
$ gdb /usr/bin/gdb /home/cav/core
GNU gdb 5.0
Copyright 2000 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain
conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB. Type "show warranty" for details.
```

```
This GDB was configured as "s390-ibm-linux"...(no debugging symbols
found)...
Core was generated by `gdb /usr/bin/gdb /home/linuser/core'.
Program terminated with signal 11, Segmentation fault.
Reading symbols from /usr/lib/libncurses.so.5...(no debugging symbols
found)...
done.
Loaded symbols for /usr/lib/libncurses.so.5
Reading symbols from /lib/libm.so.6...done.
Loaded symbols for /lib/libm.so.6
Reading symbols from /lib/libdl.so.2...done.
Loaded symbols for /lib/libdl.so.2
Reading symbols from /lib/libc.so.6...done.
Loaded symbols for /lib/libc.so.6
Reading symbols from /lib/ld.so.1...done.
Loaded symbols for /lib/ld.so.1
#0 0x4014c67a in poll () from /lib/libc.so.6
(gdb)
```

---

The **info** subcommands are very useful. The various options are shown in the following examples.

The **info stack** option displays stack entries, as follows:

```
(gdb) info stack
#0 0x4014c67a in poll () from /lib/libc.so.6
#1 0x4c3af6 in delete_file_handler ()
#2 0x4c3286 in _initialize_thread ()
```

The **info registers** option displays the values of the registers at the time of the failure, as shown in the following example:

---

```
(gdb) info registers
pswm          0x709d000      118083584
pswa          0xc014c67a    -1072380294
gpr0          0x5d3938 6109496
gpr1          0x400becec    1074523372
gpr2          0xffffffffc   -4
gpr3          0x1          1
gpr4          0xffffffff    -1
gpr5          0x7ffff5e0    2147481056
gpr6          0x0          0
gpr7          0x1          1
gpr8          0x599a58 5872216
gpr9          0x0          0
gpr10         0x400bdbac    1074518956
gpr11         0x1          1
gpr12         0xc0196558    -1072077480
```

```

gpr13      0xc014c610      -1072380400
gpr14      0xc014c658      -1072380328
gpr15      0x7ffff448      2147480648
acr0       0x0          0
acr1       0x0          0
acr2       0x0          0
acr3       0x0          0
acr4       0x1          1
---Type <return> to continue, or q <return> to quit---
acr5       0x0          0
acr6       0x0          0
acr7       0x0          0
acr8       0x0          0
acr9       0x0          0
acr10      0x0          0
acr11      0x0          0
acr12      0x0          0
acr13      0x0          0
acr14      0x0          0
acr15      0x0          0
cr0        0x0          0
cr1        0x0          0
cr2        0x0          0
cr3        0x0          0
cr4        0x0          0
cr5        0x0          0
cr6        0x0          0
cr7        0x412e8480      1093567616
cr8        0x0          0
cr9        0x0          0
cr10       0x0          0
cr11       0x41306ce2      1093692642
---Type <return> to continue, or q <return> to quit---
cr12       0xaaaaaaaaab      -1431655765
cr13       0x0          0
cr14       0x0          0
cr15       0x0          0
fpc        0x0          0

```

---

The following example displays the complete stack frame entry using the **info frame** option. By default, the current frame (stack level 0) is used, but others may be specified. This is useful for tracing back through the call flow.

```
(gdb) info frame
```

```

Stack level 0, frame at 0x7ffff448:
  pswa = 0x4014c67a in poll; saved pswa 0x4c3af6
  called by frame at 0x7ffff4c0
  Arglist at 0x7ffff448, args:
  Locals at 0x7ffff448, Previous frame's sp is 0x7ffff4fc

```

Saved registers:  
 gpr6 at 0x7ffff4d8, gpr7 at 0x7ffff4dc, gpr8 at 0x7ffff4e0,  
 gpr9 at 0x7ffff4e4, gpr10 at 0x7ffff4e8, gpr11 at 0x7ffff4ec,  
 gpr12 at 0x7ffff4f0, gpr13 at 0x7ffff4f4, gpr14 at 0x7ffff4f8

## 19.1.5 uptime, top commands

The **uptime** command displays the time; the length of time the system has been up; how many users are logged on; the load average for the past 1, 5, and 15 minutes:

```
$ uptime
9:01am up 23:43, 1 user, load average: 0.00, 0.00, 0.00
```

The **top** command gives a display similar to those found in OS/390 SDSF DA and RMFMON ASD, and it also includes the uptime information at the very beginning of its output. This display will dynamically update; entering q will terminate the dynamic display and return you to the shell prompt:

---

```
$ top
9:04am up 23:46, 1 user, load average: 0.00, 0.00, 0.00
30 processes: 29 sleeping, 1 running, 0 zombie, 0 stopped
CPU states: 0.1% user, 0.1% system, 0.0% nice, 99.6% idle
Mem: 516352K av, 87252K used, 429100K free, 45984K shrd, 53156K
buff
Swap: 143980K av, 0K used, 143980K free 11280K
cached
```

PID	USER	PRI	NI	SIZE	RSS	SHARE	LC	STAT	%CPU	%MEM	TIME	COMMAND
1960	cav	19	0	936	936	744	0	R	0.3	0.1	0:00	top
1	root	0	0	492	492	432	0	S	0.0	0.0	0:01	init
2	root	0	0	0	0	0	0	SW	0.0	0.0	0:00	kmcheck
3	root	0	0	0	0	0	0	SW	0.0	0.0	0:00	kflushd
4	root	0	0	0	0	0	0	SW	0.0	0.0	0:00	kupdate
5	root	0	0	0	0	0	0	SW	0.0	0.0	0:00	kpiod
6	root	0	0	0	0	0	0	SW	0.0	0.0	0:00	kswapd
155	root	0	0	568	568	464	0	S	0.0	0.1	0:00	syslogd
163	root	0	0	728	728	428	0	S	0.0	0.1	0:00	klogd
173	daemon	0	0	508	508	440	0	S	0.0	0.0	0:00	atd
183	root	0	0	636	636	544	0	S	0.0	0.1	0:00	crond
193	root	0	0	508	508	440	0	S	0.0	0.0	0:00	inetd
197	root	0	0	860	860	720	0	S	0.0	0.1	0:00	sshd
201	lp	0	0	736	736	632	0	S	0.0	0.1	0:00	lpd
212	root	0	0	3500	3500	3368	0	S	0.0	0.6	0:00	httpd
220	root	0	0	3188	3188	1312	0	S	0.0	0.6	0:00	miniserv.pl
222	root	0	0	1156	1156	868	0	S	0.0	0.2	0:00	bash

---



## 19.1.6 System.map

This file contains a storage map of the kernel modules, similar to an OS/390 MVS nucleus map (IPCS NUCMAP). This will help determine which kernel module failed, and the offset of the failing instruction.

If there is a system crash, get the PSWA and map it to a kernel module using the System.map file.

For PSW=000A0000 800355F8, which is the disabled wait code received if the boot files are not in the correct order, do **grep -i 00035 /boot/System.map**, as follows:

```
# grep -i 00035 /boot/System.map
000352f8 T sys_delete_module
000355c0 t qm_modules
00035764 t qm_deps
0003598c t qm_refs
00035b8c t qm_symbols
00035e00 t qm_info
00035f10 T sys_query_module
```

From here it is easy to work out that the PSW above is pointing to qm\_modules+38 (hex offset).

For PSW=000A0000 80057BE4, which is the disabled wait code received if the boot files are not blocked correctly, do **grep -i 00057 /boot/System.map**, as follows:

```
# grep -i 00057 /boot/System.map
000570b8 T sys_newlstat
0005718c T sys_fstat
00057280 T sys_newfstat
00057374 T sys_readlink
000574e0 T binfmt_setup
00057510 T register_binfmt
00057570 T unregister_binfmt
000575b0 T open_dentry
000576b4 T sys_uselib
000577d8 t count
0005785c T copy_strings
00057ae8 T setup_arg_pages
00057c04 T read_exec
00057d70 t exec_mmap
00057efc T flush_old_exec
```

The PSW above in this instance is pointing to module setup\_arg\_pages+EC (again, hex offset).

## 19.1.7 ksymoops

The **ksymoops** utility decodes Linux kernel *Oops* reports. The object of this utility is to provide information about the modules that crashed by decoding PSW and stack information, and mapping it to the `/boot/System.map`.

For it to be really effective, the messages need to be abend-type messages—but for a *kernel* crash; otherwise, the utility cannot decode it. Unfortunately, in our testing there were no kernel crashes and accompanying messages to use as input. However, the following example shows both how to use **ksymoops**, and some of the options you need to use:

```
# ksymoops -m /boot/System.map /home/oops.file2
ksymoops 0.7c on s390 2.2.16. Options used
  -V (default)
  -k /proc/ksyms (default)
  -l /proc/modules (default)
  -o /lib/modules/2.2.16/ (default)
  -m /boot/System.map (specified)

Jun 12 21:07:43 vmlinux7 kernel: CPU:    0
Jun 12 21:07:43 vmlinux7 kernel: Process petri (pid: 6037,
stackpage=1F67F000)
```

The most important thing is to identify the file that contains the system map. From the above example, you can see that it is `/boot/System.map`. The last argument is the file name that contains the *Oops* messages (in this case, `/home/oops.file2`), which is equally important.

## 19.1.8 Log files

You should not underestimate the diagnostic value of logs when doing problem determination. While core dumps are obviously also useful for diagnosis, the core dump is a snapshot of memory at the time of failure, and therefore shows only an instant in time. Depending on what residual data exists, there may be no trace of something that happened, for example, 7 minutes or even 7 seconds before the dump was generated. In contrast, log files can help you reconstruct a *sequence* of events leading up to a failure. Log files are kept in the `/var/log` directory.

Logs may be kept in different places for different distributions or different applications, and the following command will help find them:

```
find / -iname *log -maxdepth 5|more
```

Use **find** as the user `root`; otherwise, you will abound in permission denied messages.

For example, files and directories with “log” as part of their names can be listed by using the following command:

```
# find / -iname *log -maxdepth 3 |more
/etc/logrotate.d/syslog
/etc/logrotate.d/mars-nwe.log
/var/log
/var/log/lastlog
/var/log/maillog
/var/log/radius.log
/var/log/boot.log
/var/log/xdm-error.log
/var/log/radwatch.log
/dev/log
...
```

## syslogd, klogd daemons

These daemons are important because they are needed to provide Linux system logging and kernel logging. They should be started automatically after boot.

The startup script is in `/etc/rc.d/init.d/syslog`; it can be used to check the status of these daemons and, if necessary, start them. To check them use `/etc/rc.d/init.d/syslog status`, as shown in the following example

```
# /etc/rc.d/init.d/syslog status
syslogd (pid 155) is running...
klogd (pid 163) is running...
```

If these messages are not received, then start the daemons with `/etc/rc.d/init.d/syslog start` and investigate why they were not started.

## The system log

The system log contains all messages produced by the running system, and it is the place to start looking for many errors. This log is kept in `/var/log/messages`.

The following example shows extracts where a segment fault occurred. (The program interrupt code = 10 means a virtual address did not translate into a real one; this generally means the address is bad and is out of bounds.)

---

```
# cat /var/log/messages |more
Jun  3 04:02:37 vmlinux7 syslogd 1.3-3: restart.
Jun  3 05:01:01 vmlinux7 PAM_pwdb[7052]: (su) session opened for user news
Jun  3 05:01:01 vmlinux7 PAM_pwdb[7052]: (su) session closed for user news
Jun  3 06:01:00 vmlinux7 PAM_pwdb[7113]: (su) session opened for user news
Jun  3 06:01:00 vmlinux7 PAM_pwdb[7113]: (su) session closed for user news
Jun  3 07:01:00 vmlinux7 PAM_pwdb[7174]: (su) session opened for user news
Jun  3 07:01:00 vmlinux7 PAM_pwdb[7174]: (su) session closed for user news
Jun  3 08:01:00 vmlinux7 PAM_pwdb[7235]: (su) session opened for user news
```

```
...
Jun  7 12:25:41 vmlinux7 kernel: User process fault: interruption code 0x10
Jun  7 12:25:41 vmlinux7 kernel: failing address: 0
Jun  7 12:25:41 vmlinux7 kernel: CPU:    0
Jun  7 12:25:41 vmlinux7 kernel: Process xedit (pid: 439, stackpage=1A74B00
Jun  7 12:25:41 vmlinux7 kernel:
Jun  7 12:25:41 vmlinux7 kernel: User PSW:   0709d000 c023ad30
Jun  7 12:25:41 vmlinux7 kernel: task: 1a74a000 ksp: 1a74bd00 pt_regs: 1a74
Jun  7 12:25:41 vmlinux7 kernel: User GPRS:
Jun  7 12:25:41 vmlinux7 kernel: 00000003 402ce1c8 402ce1c8 00000018
Jun  7 12:25:41 vmlinux7 kernel: 00000000 00417350 fffffffe 402ce1a8
Jun  7 12:25:41 vmlinux7 kernel: 4023a49c 402ce1a8 00000000 00000010
Jun  7 12:25:41 vmlinux7 kernel: c02cf558 c023a4a4 c023b8e0 7fffb3a8
Jun  7 12:25:41 vmlinux7 kernel: User ACRS:
...
Jun  7 14:01:00 vmlinux7 PAM_pwdb[725]: (su) session closed for user news
Jun  7 14:17:49 vmlinux7 PAM_pwdb[808]:(login) session opened for user cav1
Jun  7 14:19:12 vmlinux7 PAM_pwdb[822]:(login) session opened for user cav2
```

---

Additionally, log messages related to mail are often kept in the file `/var/log/maillog`, and messages related to the X display manager are often kept in the file `/var/log/xdm-error.log`.

## 19.1.9 dmesg

The **dmesg** command displays messages that have been sent to the kernel ring buffer. Initially, the kernel ring buffer will contain the messages seen at bootup, but it will also contain messages about recent program checks and summary abend information as errors occur. In a way it is like an extract of the system log of serious recent events.

There is a suggestion implied in the man pages, and from the empty file `/var/log/boot.log` found on Turbolinux, that the boot messages should be saved by automating a command like **dmesg > /var/log/boot.log** at each startup. (For example, this could be put in the file `/etc/rc.d/init.d/boot` on a SuSE distribution).

This file can be used later for debugging, if needed, because older messages will be lost from the kernel ring buffer as they become overwritten by new messages.

The following example shows captured boot messages:

---

```
# dmesg
Linux version 2.2.16 (root@s390.dev.us.tlan) (gcc version 2.95.2 19991024
(rele0
Command line is: root=/dev/dasdb1 ro noinitrd dasd=191,201-206
We are running under VM
This machine has an IEEE fpu
```

```

Detected device 292C on subchannel 0000 - PIM = 80, PAM = 80, POM = FF
Detected device 292D on subchannel 0001 - PIM = 80, PAM = 80, POM = FF
Detected device 0191 on subchannel 0002 - PIM = F0, PAM = F0, POM = FF
Detected device 0201 on subchannel 0003 - PIM = F0, PAM = F0, POM = FF
...
Highest subchannel number detected (hex) : 0013
SenseID : device 292C reports: Dev Type/Mod = 3088/60
SenseID : device 292D reports: Dev Type/Mod = 3088/60
SenseID : device 0191 reports: CU Type/Mod = 3990/E2, Dev Type/Mod = 3390/
SenseID : device 0201 reports: CU Type/Mod = 3990/EC, Dev Type/Mod = 3390/
SenseID : device 0202 reports: CU Type/Mod = 3990/EC, Dev Type/Mod = 3390/
...
Calibrating delay loop... 1055.13 BogoMIPS
Memory: 516352k/524288k available (1160k kernel code, 4k reserved, 6772k da
Dentry hash table entries: 65536 (order 7, 512k)
Buffer cache hash table entries: 524288 (order 9, 2048k)
Page cache hash table entries: 131072 (order 7, 512k)
debug: 16 areas reserved for debugging information
debug: reserved 4 areas of 4 pages for debugging ccwcache
ccwcachedebug area is 0x 1760a8
VFS: Diskquotas version dquot_6.4.0 initialized
POSIX conformance testing by UNIFIX
Detected 1 CPU's
Boot cpu address 0
cpu 0 phys_idx=0 vers=FF ident=040ECB machine=2064 unused=0000
Linux NET4.0 for Linux 2.2
Based upon Swansea University Computer Society NET3.039
NET4: Unix domain sockets 1.0 for Linux NET4.0.
NET4: Linux TCP/IP 1.0 for NET4.0
IP Protocols: ICMP, UDP, TCP, IGMP
TCP: Hash tables configured (ehash 524288 bhash 65536)
Initializing RT netlink socket
Starting kswapd v 1.5
pty: 256 Unix98 ptys configured
RAM disk driver initialized: 16 RAM disks of 8192K size
md driver 0.36.6 MAX_MD_DEV=4, MAX_REAL=8
...

```

---

In the following example, **dmesg** is used to recapture information about a segment fault which occurred when using XEDIT:

---

```

# dmesg
0000 - PIM = 80, PAM = 80, POM = FF
Detected device 292D on subchannel 0001 - PIM = 80, PAM = 80, POM = FF
Detected device 0191 on subchannel 0002 - PIM = F0, PAM = F0, POM = FF
Detected device 0201 on subchannel 0003 - PIM = F0, PAM = F0, POM = FF
Detected device 0202 on subchannel 0004 - PIM = F0, PAM = F0, POM = FF
...

```

```

User process fault: interruption code 0x10
failing address: 0
CPU: 0
Process xedit (pid: 439, stackpage=1A74B000)

User PSW: 0709d000 c023ad30
task: 1a74a000 ksp: 1a74bd00 pt_regs: 1a74bf68
User GPRS:
00000003 402ce1c8 402ce1c8 00000018
00000000 00417350 ffffffff 402ce1a8
4023a49c 402ce1a8 00000000 00000010
c02cf558 c023a4a4 c023b8e0 7fffb3a8
User ACRS:
00000000 00000000 00000000 00000000
00000001 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
Kernel BackChain CallChain
1a74bd00 [<0011b040>]
1a74bd80 [<00000001>]
1a74bde8 [<000189ca>]
1a74be98 [<00012aa0>]
1a74bf08 [<00016732>]

```

---

### 19.1.10 Standalone dump

Much like OS/390, Linux now has a standalone dump facility. Its use and installation instructions are documented in *Using the Dump Tools*, LNUX-1208. This publication specifies using the 2.4 Linux kernel; using this facility with earlier kernels may or may not work.

These tools are available in an s390-tools rpm package called **zip1**. Even though these tools have not been tested, they are mentioned here for completeness.

## 19.2 VM diagnostic tools

VM provides many features that are useful for debugging errors in a guest operating system. Refer to the VM library (specifically *VM/ESA CMS User's Guide*, SC24-5775 and *VM/ESA Quick Reference*, SX24-5290 to start with, and *Debugging on Linux for 390*) for more complete information. In this section, we simply provide a basic overview of VM diagnostic tools.

The most useful thing about VM CP commands is that they can be executed concurrently while the Linux system is up and running. The following simple example shows a trace of SVC 5, which is Linux's open system call.

Using the previous **date** example:

```
# strace date 2>&1|grep open
open("/etc/ld.so.preload", O_RDONLY) = -1 ENOENT (No such file or direct
open("/etc/ld.so.cache", O_RDONLY) = 3
open("/lib/libc.so.6", O_RDONLY) = 3
open("/etc/localtime", O_RDONLY) = 3
```

```
#CP TRACE SVC 5
# date
date
-> 40013AA0" SVC 0A05 -> 00016060' CC 0
-> 40013AA0" SVC 0A05 -> 00016060' CC 0
-> 40013AA0" SVC 0A05 -> 00016060' CC 0
-> 400C0588" SVC 0A05 -> 00016060' CC 0
```

Using the **trace** feature of VM, you can trace events beyond system calls (such as instruction execution, storage alteration, register alteration, and I/O activity). In addition, a combination of up to 255 of these event traps can be set.

Storage can also be displayed and altered (at your own risk). The **#CP D** command displays storage, registers, and instructions. The **#CP ST** command will alter storage and registers.

## 19.3 HMC diagnostic facilities

There is not much diagnostic capability built into the HMC, from a system software point of view. The only thing of value is the ability to alter and display *real* and *virtual* storage. In most cases you'd want to display virtual storage. The alter capability needs to be used wisely, as it has the ability to crash a running system.

From a hardware and I/O configuration point of view, many other features are also available which are not discussed in this publication.

## 19.4 Debugging for developers

All Linux tools are available for use by application developers, including features that have interactive source-level debugging capabilities. The man pages document these options and features; in particular, **gdb** and **strace** are useful.

Note that the following sections contain information only, and do not provide examples or specific instructions.

### 19.4.1 strace

**strace** is very useful when trying to understand how unexpected output or results have come about. Refer to 19.1.2, “strace” on page 356 for working examples. Its use will allow you to understand the sequence of events from a system call perspective.

### 19.4.2 gdb

**gdb** can be used with source code, for C, C++ and Modula2 programs. For a complete guide to using **gdb** as a source-level debugger, the man page says the following

For full details on GDB, see *Using GDB: A Guide to the GNU Source-Level Debugger*, by Richard M. Stallman and Roland H. Pesch. The same text is available online as the `gdb` entry in the `info` program.

## 19.5 Support

Because of the GPL and the GPL philosophy, there is no warranty support for Linux per se by its developers. It is a collective and collaborative effort in much the same vein of any scientific community, the spirit of which is captured in a quote of Sir Isaac Newton, “I stand on the shoulders of others.” Linus Torvalds started it, but many others have contributed and carried it further.

This does not mean that support is not available, far from it—there are many sources for technical support. What it does mean is that there is generally no entitlement for “free” (or warranty) support. The area of free support was not investigated for this redbook.

The more general case will be that a support agreement will be entered into with, perhaps, the distributor or the hardware vendor. The price of this may be included in the price of the machine or in the price of the distribution, or it may be entirely separate.

In this section we do not list types of support contracts and support providers; if you need detailed information about Linux on zSeries and S/390 support, you can contact a marketing representative or the IBM Support Center, or go to the following site and select **Service Offerings**, and then **Support Line**:

<http://www.ibm.com/linux>



Instead, we discuss the diagnostic data that will be useful to a support provider, and how to collect it. In addition, we describe how to develop a problem description, and we provide useful search arguments for searching IBM's problem databases. Finally, we list sites for problem databases.

### 19.5.1 Support Line

Support Line is IBM's generic software support service offering. Perhaps the only thing “generic” about it is the *type* of service offered, however, because the scope of software products covered can vary widely. Some agreements are platform-wide (for example, all IBM software running on zSeries), while others are specifically for a product that might span several hardware platforms.

The difference with Support Line for Linux is that this is the first time IBM has a software support service offering for another vendor's software on this scale. While there may have been special “one offs” before, this offering uses the central support structure of IBM and is delivered through IBM Support Centers.

The following descriptions and procedures are based on what an IBM Support Center would use and request. However, they are likely to be applicable to any support provider.

### 19.5.2 Getting support

IBM generally makes the assumption that system programmers or personnel with equivalent technical skills will be the customer contact. Developers should consult with their system programmers (unless they themselves are the system programmers for their system).

A hierarchical support structure works effectively and efficiently. System programmers or those deemed equivalent should be responsible for reporting problems, that is, where a software defect is suspected.

For Q & A support, the system programmer may not be appropriate, particularly when the query involves knowledge in the skilled use of an end-use product or application.

### 19.5.3 Problem description

The problem reporting process begins with a *problem description*. A good problem description can go a long way toward speeding problem resolution, by reducing the number of interactions required between the user and support.

A problem description should include the following items:

- ▶ Date and time of occurrence.
- ▶ Environment (hardware will be S/390 or zSeries) Linux Kernel level, maintenance level.
- ▶ Problem symptom - wait code, messages, message codes, wait or hang.
- ▶ Failing or crashing program or component name.
- ▶ Sequence of events immediately leading up to failure. How many times has the failure occurred?
- ▶ Recent change history.
- ▶ Diagnostic data available.

### 19.5.4 Diagnostic data

The record most likely to be available if a program crashes is a *core dump*. The core dump should be saved, along with the binary of the program that actually crashed. That point is important—the binary for the program must actually *match* the one that crashed, so it is best that both the dump and the binary be saved together. This is just in case there is change or upgrade to the program in question between the time of the crash, and the time when the problem is reported and the data finally sent to the support provider.

The system log around the time of the crash will also be useful, since it will contain messages generated around the time of, and leading up to, the failure. Having this data will help you understand and confirm the sequence of events leading to the error.

### 19.5.5 Data collection

Linux, with its use of pipes and file redirection, makes it easy to collect diagnostic data; in most cases all that is needed is to save the data to a file.

#### Getting a dump

Generally if a program crashes, a core dump will be generated automatically—if dumps have been enabled. Use the following command to enable them:

```
# ulimit -c unlimited
```

Dumps will need to be enabled for processes before one can be obtained, which means that if a process is hung, it is too late to enable a dump for that process. Likewise, for hung processes, dumps will not automatically be generated because the processes have not crashed. However, even though they have not crashed, a dump can still be obtained by using the following command from another terminal or login session:

```
# kill -SIGSEVG <pid>
```

This is the equivalent of a **cancel** with a **dump** on OS/390.

## 19.5.6 Searching

IBMLink provides an interface to IBM's problem databases. Searches are best conducted initially using the program name and symptoms that include any message codes or failure codes. There are various formats that are used by convention for these symptoms to form "keywords". There are informational APARs that describe these keywords. A search on "keyword" should list one of these.





## LDAP

Today there are many Lightweight Directory Access Protocol (LDAP) implementations available, such as those provided by Netscape, Sun, Novell and IBM. OpenLDAP is an LDAP implementation created at the University of Michigan. It is part of the SuSE Linux 7.0 distribution. In this chapter, we give an overview of LDAP and describe how to install, configure and run OpenLDAP for a sample application on this Linux distribution.

## 20.1 What is LDAP

The first attempt to provide an open directory service was the OSI service directory called X.500, the Directory Access Protocol (DAP). It includes a rich set of functions, which also makes it very complex. Due to this complexity, X.500 applications are difficult to implement and use.

The Lightweight Directory Access Protocol (LDAP) was designed to act as a gateway to X.500 directory servers. The use of this gateway makes it easy to access the data, but it still requires a full X.500 implementation. LDAP was then further enhanced to act as a directory service itself, eliminating the need for X.500.

LDAP is an open Internet standard for providing directory services over the TCP/IP communication protocol. It allows information to be managed and queried through a set of easy-to-use utilities and API.

### What can be stored in the directory

The LDAP directory service model is based on entries. An *entry* is a collection of attributes that has a name, called a distinguished name (DN). The DN is used to refer to the entry unambiguously.

Each of the entry's attributes has a type and one or more values. The *types* are often character strings, like *cn* for common name, or *mail* for e-mail address. A *mail* attribute might contain an e-mail address with an attribute value of *evayan@us.ibm.com*. A *jpegPhoto* attribute would contain a photograph in binary JPEG format.

### How the information is arranged

The data in an LDAP server is arranged in a hierarchical format that reflects political, geographic or organizational boundaries. For example, entries representing countries appear at the top of the tree. Below them are entries representing states or national organizations. Below them might be entries representing people, organizational units, printers, documents, or just about anything else you can think of. Figure 20-1 shows an LDAP directory tree.

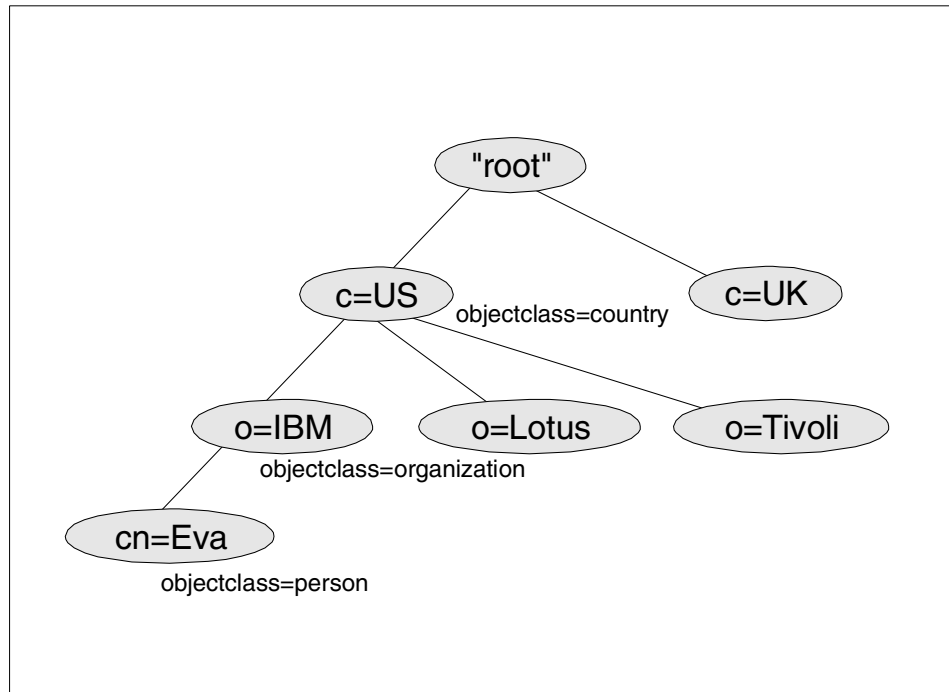


Figure 20-1 LDAP directory tree

LDAP allows you to control which attributes are required and allowed in an entry through the use of a special attribute called *objectClass*. The values of the *objectClass* attribute determine the attributes that can be specified in the entry.

### How the information is referenced

An entry is referenced by its distinguished name, which is constructed by taking the name of the entry itself (called the relative distinguished name, or RDN) and concatenating the names of its ancestor entries. For example, the entry for Eva Yan in the preceding example has an RDN of `cn=Eva Yan` and a DN of `cn=Eva Yan, o=IBM, c=US`.

### How the information is accessed

LDAP defines operations for managing information in the directory through a set of simple-to-use utilities and APIs. These operations are provided for adding and deleting an entry from the directory and modifying an existing entry. The LDAP search operation allows some portion of the directory to be searched for entries that match some criteria specified by a search filter. Information can be requested from each entry that matches the criteria.

For example, you might want to search the entire directory subtree below IBM for people with the name Eva Yan, retrieving the e-mail address of each entry found. Or you might want to search the entries directly below the c=US entry for organizations with the string Lotus in their name, and that have a fax number.

### **How information is protected**

An Access Control List (ACL) provides a means to protect information stored in an LDAP directory. Administrators use ACLs to restrict access to different portions of the directory, specific directory entries, or information within an entry. Access control can be specified for individual users or groups.

## **20.2 How does LDAP work**

LDAP directory service is based on a client/server model. An LDAP server contains the data making up the directory. An LDAP client application connects to this server using LDAP APIs and requests a function. The server responds with a reply, or with a pointer to where the application can get more information (typically, another LDAP server).

The OpenLDAP server consists of two daemons: slapd and slurpd. The slapd (or standalone LDAP) daemon provides a directory service of your own. Your directory can contain anything you want to put in it. You can connect it to a global LDAP directory service, or run a service all by yourself.

The slurpd (or standalone replication) daemon helps slapd provide replicated service. It is responsible for distributing changes made to the master slapd database out to the various slapd replicas.

We do not discuss slurpd in this chapter. For additional information about slapd and slurpd, refer to The OpenLDAP Administration Guide available at:

<http://www.openldap.org/doc/admin>

## **20.3 Using OpenLDAP**

The remainder of this chapter describes using the OpenLDAP package.

### **20.3.1 Centralized user account management**

A common problem facing customers today is how to manage large numbers of user accounts on many different hosts. Wouldn't it be nice to have a centralized authentication mechanism? LDAP enables this and much more.



LDAP allows all user information to be kept in one place, and yet be accessed over the network. As with a centralized `/etc/passwd` file, LDAP allows network administrators to store passwords, user names, preferences, telephone numbers and anything else they choose, in one place. Having a single instance of users on the network allows administrators to maintain users on many hosts from a single management point (i.e., you can create and delete accounts in the LDAP server, and the changes are available immediately to LDAP clients).

The remainder of this section describes how to set up the necessary components to authenticate and authorize users on resources in a standard Linux fashion, using the architecture shown in Figure 20-2 on page 379.

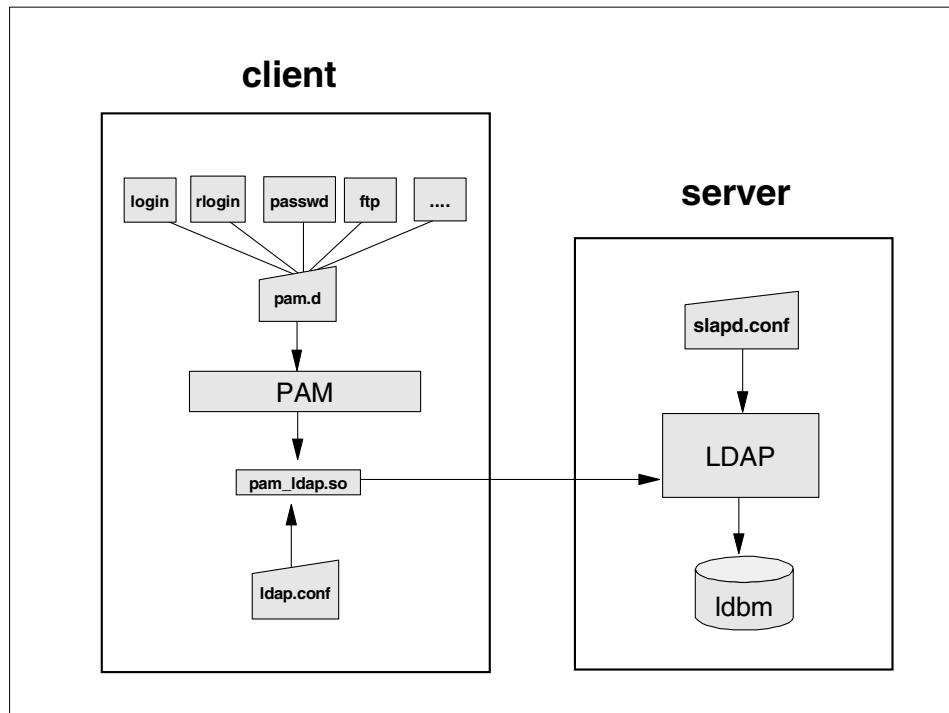


Figure 20-2 Necessary components for authenticating and authorizing users

On the server side, LDAP is used to provide clients with information about user accounts and user groups. OpenLDAP, an LDAP server, must be installed and configured. Information in the `/etc/passwd` and `/etc/shadow` files is migrated to the LDAP database with a migration tool available via the Internet.

## The Pluggable Authentication Module (PAM)

On the client side, a `pam_ldap` module is used to provide the authentication mechanism. The Pluggable Authentication Module (PAM) is the standard authentication framework of the SuSE Linux distribution. It provides an API through which authentication requests for services such as `login`, `passwd`, `ftp`, etc. are mapped into technology-specific actions (implemented in `pam` modules). This mapping is done by PAM configuration files (found in `/etc/pam.d`) in which each service is given the authentication mechanisms to use.

In our case, the `pam_ldap` module, implemented in the shared library `pam_ldap.so`, allows user and group authentication using an LDAP service. Each service that needs an authentication facility can be configured through the PAM configuration files to use this authentication method.

## The Name Service Switch (NSS)

Also on the client side, the Name Service Switch (NSS) needs to be configured.

Once a user is authenticated, many applications still need access to user information. This information is traditionally contained in text files (`/etc/passwd`, `/etc/shadow`, and `/etc/group`), but can also be provided by other name services. As a new name service (such as LDAP) is introduced, it can be implemented either in the C library (as it was for NIS and DNS) or in the application that wants to use the new name service.

This can be avoided by using a common, general purpose, name service API and by using a set of libraries to perform the task of retrieving this information by performing technology-based operations. This solution was adopted in the GNU C Library that implements the Name Service Switch. NSS uses a common API and a configuration file (`/etc/nsswitch.conf`) in which the name service providers for every supported database are specified.

### 20.3.2 Setting up the OpenLDAP server

In this section we describe installing and configuring the OpenLDAP server, and migrating data to the database.

#### Installing the OpenLDAP server

OpenLDAP is packaged with the SuSE distribution. You can install it by using `yast`. The package can be found in the Network Support Series. Simply select `openldap` and proceed with the installation. You can make sure that the installation was successful by checking the following:

- ▶ Information about the installed package now exists; verify this by using the `rpm` command:

```
# rpm -qa | grep ^ldap
```

This command should return the LDAP package name installed, such as `ldaplib-1.2.11-50`.

- ▶ A new directory named `/etc/openldap` was created.

### **Configure the LDAP server**

The standalone LDAP server, `slapd`, is highly configurable through a configuration file known as `slapd.conf`, which is located in `/etc/openldap`. You need to edit this file to make it specific to your domain and server environment.

Following is simple example of the configuration file:

```
#####  
# slapd.conf example  
# global options  
#####  
include          /etc/openldap/slapd.at.conf 1  
include          /etc/openldap/slapd.oc.conf 2  
schemacheck      off 3  
pidfile          /var/run/slapd.pid  
argsfile         /var/run/slapd.args  
loglevel         256 4  
#####  
# ldbm database definitions  
#####  
database         ldbm 5  
suffix           "o=Demo" 6  
rootdn           "cn=admin, o=Demo" 7  
rootpw           secret 8  
directory        /var/lib/ldap 9  
#####  
# ldbm database access definitions  
#####  
defaultaccess    read 10  
access to attr=userpassword 11  
by self write 12  
by dn="cn=admin,o=Demo" write 13
```

The first section in this configuration file contains global options which apply to the LDAP server as a whole, including all back-end databases:

- 1, 2                    Specifies configuration files containing attribute and object class definitions.
- 3                      Turns off schema checking.
- 4                      Specifies the level of message logging.

The next section of this configuration file defines a back-end database that will handle queries for a specific domain:

- 5                      Specifies the start of the database definition.
- 6                      Specifies the DN suffix for queries to pass to this database.
- 7 8                    Identifies the “super user” entry and associated password.
- 9                      Specifies the directory in which the database files reside.

The last section in this configuration file contains the back-end database access control definitions.

- 10 Specifies that all attributes allow read access by default.
- 11 The userPassword attribute is writable by the entry.
- 12 The userPassword attribute is writable by the “admin” entry.
- 13 The userPassword attribute is comparable by everyone else.

## Starting the LDAP server

You are now ready to start the standalone LDAP server, slapd, by executing the command:

```
# /etc/rc.d/ldap start
```

Slapd supports a monitoring interface you can use to find out status information about the LDAP server. You can access this information by doing a base search of the SLAPD\_MONITOR\_DN from the include/ldapconfig.h file (cn=monitor, by default). To verify that the server is running and configured correctly, issue the following search against the LDAP database:

```
# ldapsearch -s base -b cn=monitor,'objectclass=*'
```

An entry similar to the following should be returned:

```
CN=MONITOR
version=slapd 1.2.11-Release (Sat Oct 28 01:15:31 GMT 2000)
threads=1
connection=7 : 20010615114627Z : 2 : 1 : NULLDN :
currentconnections=1
totalconnections=6
dtablesize=1024
writewaiters=0
readwaiters=0
opsinitiated=17
opscompleted=16
entriessent=54
bytessent=17280
currenttime=20010615114627Z
starttime=20010615113849Z
nbackends=1
```

### 20.3.3 Migrating data to the LDAP database

Once slapd is properly configured and started, you need to migrate old user account information to LDAP.

Useful tools to convert existing databases can be found on the Web. For example, PADL provides a tool which can be downloaded at:

<ftp://ftp.padl.com/pub/MigrationTools.tar.gz>

The MigrationTools are a set of Perl scripts for migrating users, groups, aliases, hosts, netgroups, networks, protocols, RPCs, and services from existing nameservices (flat files, NIS, and NetInfo) to LDAP. The tools require the **ldapadd** and **ldif2dbm** commands, which are distributed with OpenLDAP.

Here is a summary of the Perl scripts provided by this tool:

migrate_base.pl	Creates naming context entries, including subordinate contexts such as ou=people and ou=devices
migrate_aliases.pl	Migrates aliases in /etc/aliases
migrate_group.pl	Migrates groups in /etc/group
migrate_hosts.pl	Migrates hosts in /etc/hosts
migrate_networks.pl	Migrates networks in /etc/networks
migrate_passwd.pl	Migrates users in /etc/passwd
migrate_protocols.pl	Migrates protocols in /etc/protocols
migrate_services.pl	Migrates services in /etc/services
migrate_netgroup*.pl	Migrates netgroups in /etc/netgroups
migrate_rpc.pl	Migrates RPCs in /etc/rpc

The configuration for these Perl scripts is contained in the migrate\_common.ph file. You must edit this file to reflect your configurations, such as the DEFAULT\_BASE.

Next, you need to decide what data to migrate. If you are an expert at LDAP, then you may want to only migrate the data you want by selecting the appropriate Perl scripts listed above. The easiest and simplest way is to use the migrate\_all\_offline.pl perl script. This script will migrate all the data, including the data in the /etc/passwd and /etc/shadow files. Refer to the README file for detailed information on how to use the MigrationTools.

You can verify that the user account data is migrated successfully by performing a search similar to the following on the LDAP database for a valid user:

```
# ldapsearch -b "uid=eva,ou=People,ou=Demo" "objectclass=*"
```

The search should be successful and information about this user should be returned, for example:

```
uid=eva,ou=People,o=Demo
uid=eva
cn=eva
objectclass=account
objectclass=posixAccount
objectclass=top
objectclass=shadowAccount
shadowlastchange=11477
shadowmax=99999
shadowwarning=7
loginshell=/bin/bash
uidnumber=100
gidnumber=100
homedirectory=/home/eva
```

### 20.3.4 Setting up the PAM-LDAP module

The `pam_ldap` module is packaged with the SuSE distribution. You can install it by using `yast`. The package can be found in the Network Support Series. Simply select `pam_ldap` and proceed with the installation.

To authenticate users, the PAM configuration files need to be changed. All PAM configuration files reside in `/etc/pam.d/`. Each service, such as FTP and login, has a configuration file in this directory. Generally the modifications that need to be made to the files involve adding lines of the following form to the configuration files:

```
type      sufficient      pam_ldap.so
```

Where `type` is one of `account`, `auth`, `password`, or `session`. A sample login configuration file, `etc/pam.d/login`, is shown here:

```
##PAM-1.0
auth      required      /lib/security/pam_securetty.so
auth      required      /lib/security/pam_nologin.so
auth      sufficient    /lib/security/pam_ldap.so
auth      required      /lib/security/pam_unix_auth.so use_first_pass
account   sufficient    /lib/security/pam_ldap.so
account   required      /lib/security/pam_unix_acct.so
password  required      /lib/security/pam_cracklib.so
password  sufficient    /lib/security/pam_ldap.so
password  required      /lib/security/pam_unix_passwd.so use_first_pass
session   required      /lib/security/pam_unix_session.so
```

## Setting up Name Switch Service

The `nss_ldap` module is packaged with the SuSE distribution. You can install it by using `yast`. The package can be found in the Network Support Series. Simply select `nss_ldap` and proceed with the installation. This installs `/lib/libnss_ldap.so`, which is the `nss_ldap` library, and a set of example configuration files, `/etc/nsswitch.ldap` and `/etc/ldap.conf`, in case they do not exist already.

Next, you must edit the NSS configuration file `/etc/nsswitch.conf`. With the following configuration, entries are first looked for in the system files and, if no value is returned, the LDAP server is queried.

```
passwd: files ldap
group:  files ldap
shadow: files ldap
```

## Setting up the client environment

The LDAP client configuration file, `/etc/openldap/ldap.conf`, is read by `pam_ldap` and `nss_ldap`, as well as by other LDAP clients. The following example directs all LDAP client requests such as `pam_ldap`, `nss_ldap`, `ldapadd`, `ldapsearch`, and so on to the local LDAP server. It is possible to use a remote LDAP server by altering the host entry in this configuration file.

```
# Your LDAP server
host    localhost
# The distinguished name of the search base
base    o=Demo
```

## Putting it all together

At this point you should be able to test your setup. Start the LDAP server, `slapd`, if it has not been started yet. If logging is enabled in the `slapd` configuration file, and `syslog` is configured to receive messages, then you should see messages indicating that LDAP is waiting for requests.

Now log in with a valid user ID to the client machine. The login should be successful and `syslog` should have messages indicating the authentication was successful.

You can set `slapd` to start automatically with the `chkconfig` command (however, on SuSE, this is a disabled script).

## 20.3.5 Possible deployment scenarios

As you can see, this sample LDAP application uses a client/server model. Since all the components involved are based on open source software, it is possible to distribute the client and server on different platforms.



We tested three different scenarios:

- ▶ Both client and server reside on the same Linux machine.
- ▶ The client and server reside on different Linux machines.
- ▶ The client resides on a Linux machine and the server resides on a z/OS machine.

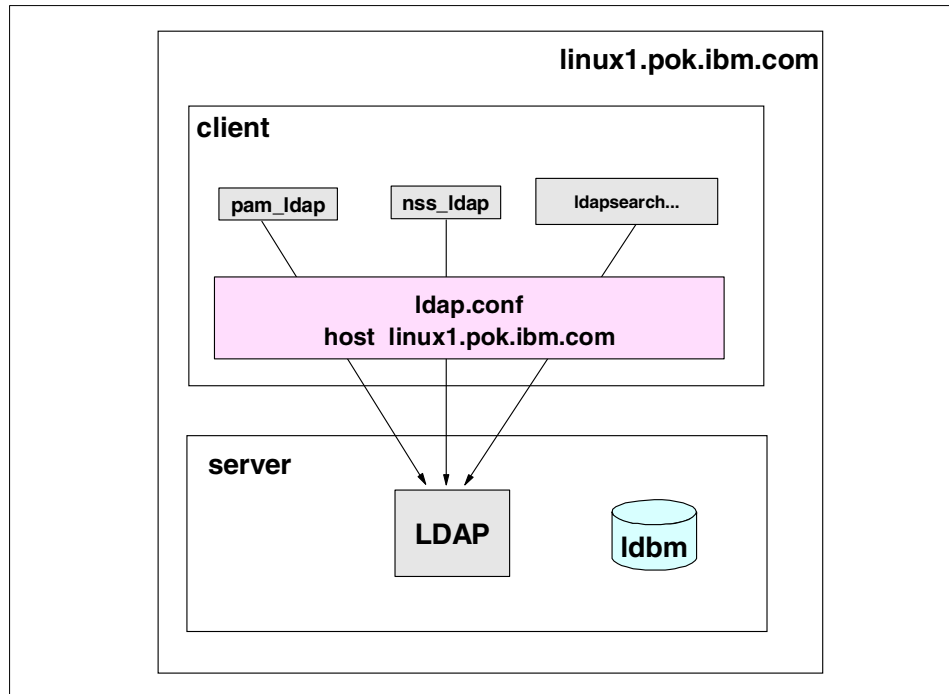


Figure 20-3 Client and server on the same Linux machine

In the configuration shown in Figure 20-3, both the client and server reside on the same machine. The client configuration file `ldap.conf` simply points to its own IP address, well-known name or `localhost`. All client LDAP queries will be directed to the local LDAP server.

In our sample application, the password file, `/etc/passwd`, was migrated and then imported into the same system. A user logging in to the `linux1.pok.ibm.com` system is authenticated with the LDAP database.

This configuration is suitable for:

- ▶ Testing applications
- ▶ Deploying simple and self-contained LDAP applications

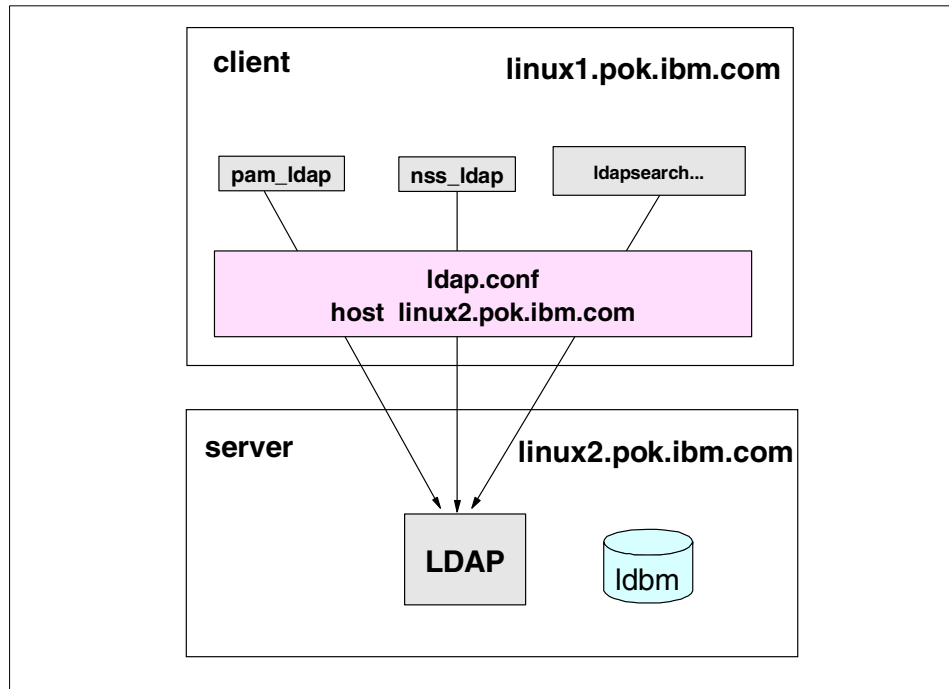


Figure 20-4 Client and server on different Linux machines

In the configuration shown in Figure 20-4, the client and the server reside on different Linux machines. The client configuration file `ldap.conf` simply points to the IP address or well-known name of the remote LDAP server. All client LDAP queries will be directed to this remote LDAP server.

In our example, the password file, `/etc/passwd`, was migrated and then transferred to the remote server, where it is imported into the LDAP database. A user logging in to the `linux1.pok.ibm.com` system is authenticated with the LDAP database on `linux2.pok.ibm.com`.

With this configuration, it is possible for the server LDAP database to act as a centralized repository for user information. Each client would have a unique search base defined for it; refer to Figure 20-5 on page 389.

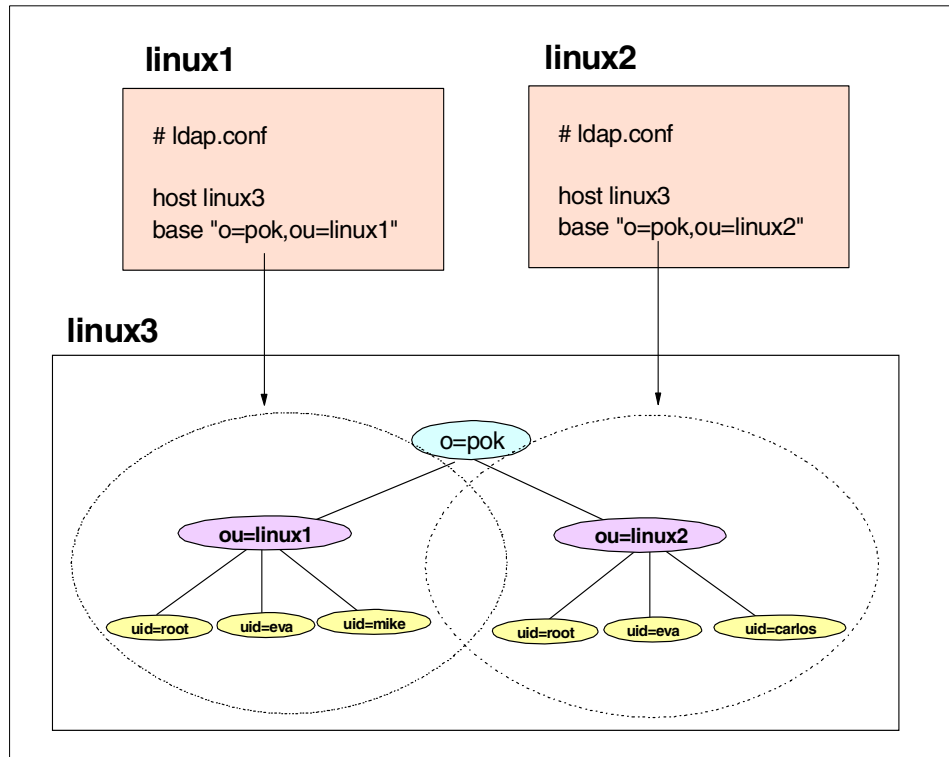


Figure 20-5 Client on Linux machine and server on z/OS

For example, linux1 can be defined as `ou=linux1,o=pok`, while linux2 can be defined as `ou=linux2,o=pok`. This scheme allows non-unique user names to exist on the same server. The client LDAP configuration file can point to the remote server with its search base.

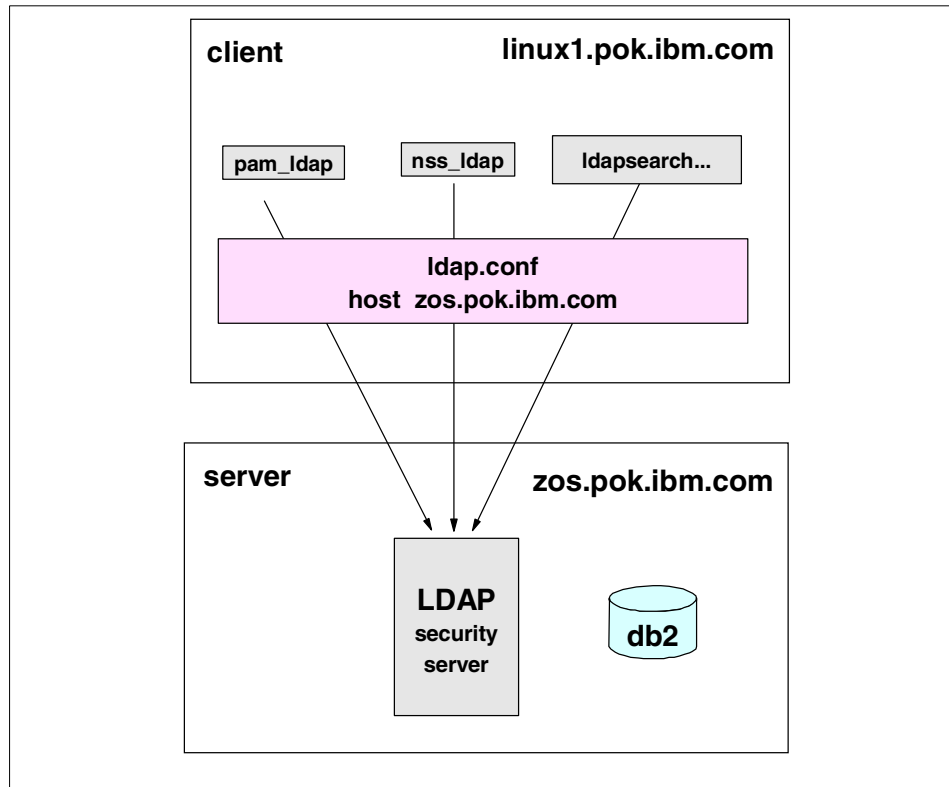


Figure 20-6 Client on Linux and server on z/OS

The configuration shown in Figure 20-6 is similar to the previous one. The difference is that the LDAP server resides on a z/OS (or OS/390) instead of Linux. The z/OS LDAP server is part of the SecureWay Security Server (commonly known as RACF). It provides the same features and capabilities as OpenLDAP.

It is outside the scope of this redbook to explain how to set up the LDAP server in the z/OS environment. You can refer to the following documentation for detailed information regarding the installation and usage of the z/OS LDAP server:

- ▶ *SecureWay Security Server LDAP Server Administration and Usage Guide*, SC24-5961
- ▶ *Understanding LDAP*, SC24-4986

Depending on the version of LDAP client and the type of data you require, it might be necessary to update your schemas with object and attribute definitions that are not part of the z/OS LDAP server base. In our testing of this sample application, we needed to define additional attributes such as shadow account-related information.

For crypt encryption of password, which is used by most systems including our Linux system, note that the values returned by the crypt algorithm are not portable to other X/Open-conformant systems. This means the crypt() algorithm cannot be used for replication between z/OS and non-z/OS environment.

Since the migrated password was crypted by the client system, it cannot be used as is. We had to reset all users' passwords manually on the server side with the ldapmodify tool. Alternatively, this task can be accomplished with a simple shell script if a large volume of data needs to be reset. Note that the user of the ldapmodify tool must have sufficient access to update the userPassword attribute.

Following are some of the added benefits of using a z/OS LDAP server:

### ***A robust database***

The z/OS LDAP server comes with a TDBM back-end database based on DB2. The TDBM database is a more highly scalable database implementation.

### ***Access to RACF data***

The z/OS LDAP server can be configured to provide access to RACF user and group profiles using the LDAP protocol. (RACF is a component of the SecureWay Security Server for z/OS.) If the RACF data is shared across the sysplex, then users and groups in the sysplex can be managed using LDAP. The LDAP server's access to RACF is managed by an additional configurable back-end called SDBM.

### ***Secure communications***

The z/OS LDAP server can be configured to encrypt data to and from LDAP clients using the z/OS Cryptographic Services System SSL. It has a variety of ciphers for encryption to choose from, all of which provide server and optional client authentication through the use of X.509 certificates.

### ***Multiple concurrent servers***

The z/OS LDAP server can be configured to permit multiple instances to serve the same DB2-based backing store at the same time. The multiple server instances may run on the same z/OS image, and they may run on multiple z/OS images in a Parallel Sysplex. This support is available for either TDBM or RDBM back-ends. This improves availability and may offer improved performance in certain configurations.

### ***Dynamic workload management***

The z/OS LDAP server can be configured to participate in dynamic workload management in a Parallel Sysplex by exploiting TCP/IP connection optimization. With multiple concurrent server instances configured in this way, availability is improved, as is resource utilization. In addition, performance improvements may be experienced because sysplex resource utilization is more evenly balanced across z/OS systems in the sysplex.

## **20.4 Other considerations**

Other considerations include security, referrals and replication.

### **20.4.1 Security**

The data in the directory tree will have to be protected in different ways: certain information must be searchable by everybody, and some of it must be readable, and most of it will be write-protected. In LDAP Version 3, there are no defined attributes to handle this. As a result, vendors support their own implementations of authorization. This is done by different implementations of access control lists (ACLs).

ACLs are used to define access rules to the different entries in the directory tree. The following is an example of an ACL implementation. The pertinent control attributes used here are `acldsource`, `aclpropagate`, and `aclentry`, where `aclentry`, for example, is the attribute that specifies who has access to the entry and what level of access he or she has. In this example, `cn=Eva Yan,ou=Pok,o=IBM,c=US` has read, write, search and compare (`rwsc`) rights for normal, sensitive and critical data (the entry is highlighted and spilled into two lines in the following example).

When setting up access control lists, it is important to do it with the goal of minimizing later administration. It is useful to delegate the access control hierarchically, if possible.

```
dn: ou=Pok, o=IBM, c=US
objectclass: top
objectclass: organizationalUnit
ou: Pok
description: Poughkeepsie Office
entryowner: access-id:cn=admin,o=IBM,c=US
inheritoncreate: TRUE
ownerpropagate: TRUE
aclpropagate: TRUE
ownersource: default
acldsource: OU=Pok,O=IBM,C=US
```

```
ac1entry: access-id: CN=EvaYan,OU=Pok,O=IBM,C=US:
object:a:normal:rwc:sensitive:rwc:critical:rwc
ac1entry: group:CN=ANYBODY:normal:rsc
```

## 20.4.2 Referrals

Using referrals provides a way for servers to refer clients to additional directory servers. With referrals you can:

- ▶ Distribute namespace information among multiple servers
- ▶ Provide knowledge of where data resides within a set of interrelated servers
- ▶ Route client requests to the appropriate server

Following are some of the advantages of using referrals:

- ▶ Distribute processing overhead, providing primitive load balancing
- ▶ Distribute administration of data along organizational boundaries
- ▶ Provide potential for widespread interconnection, beyond an organization's own boundaries

## 20.4.3 Replication

Once the LDAP server is installed and configured, users can access the directory, add objects, delete objects, or perform search operations to retrieve particular sets of information.

Replication is a process which keeps multiple databases in sync. Through replication, a change made to one database is propagated to one or more additional databases. In effect, a change to one database shows up on multiple different databases. This means there are two types of databases, masters and replicas, which are defined as follows:

Master	All changes to the database are made to the master server. The master server is then responsible for propagating the changes to all other databases. It is important to note that, while there can be multiple databases representing the same information, only one of those databases can be the master.
Replica	Each of the additional servers which contain a database replica. These replica databases are identical to the master database.







## System management tools

With Linux, in many cases, you can manage a *single* server, or even a small set of servers, by using built-in Linux tools. A production environment, however, often has *multiple* Linux servers, assigned to perform different functions (Web serving, FTP serving, mail serving, file serving, and more).

As the number of servers grows, system management becomes more difficult and the need for appropriate tools for remote monitoring, automating and control increases. In this chapter we describe tools and products you can use to administer, monitor, troubleshoot and manage Linux systems.

For our examples, we used a SuSE Linux system; for other distributions, however, the paths and settings needed may differ.

## 21.1 Common Linux tools

We first provide a brief overview of some of the commonly used Linux tools that allow you to check system status, configure the network, and perform other administrative tasks. All of these tools have associated *man pages* containing detailed descriptions of their usage. You invoke man pages by entering the `man` command, followed by the name of the command you're interested in.

### 21.1.1 The /proc file system

The /proc file system is a virtual file system, mounted at boot time, which provides access to current system status information. The /proc file system is used by many tools to gather information, which is then presented by the tools in a human-readable form.

The directory structure of the /proc file system on a S/390 system appears as follows:

```
# ls /proc/
.      20405 2637 493  630  8484      interrupts meminfo  stat
..     20406 2645 5   645  bus      ioports  misc    swaps
1      20407 2898 513  678  cmdline  kcore    modules sys
1540   20409 2910 517  710  config.gz kcore_elf mounts  tty
2      20600 2911 5184 8478  cpuinfo  kmsg     net      uptime
20028  20942 2912 525  8479  dasd     ksyms    partitions version
20029  2431  2913 579  8480  devices  loadavg  s390dbf
20032  2432  3    582  8481  dma      locks    scsi
20033  2433  3313 6    8482  filesystems lvm      self
20401  2632  4    623  8483  fs       mdstat   slabinfo
#
```

The numbered directories contain information for the currently running processes under the respective process id. You can read information out of the /proc file system by using the `cat` command:

```
# cat /proc/dasd/devices
0201(ECKD) at (94:0) is dasda:active at blocksize: 4096, 468000 blocks,
1828 MB
# cat /proc/meminfo
      total:      used:      free:  shared: buffers:  cached:
Mem:  528703488 124760064 403943424 42131456 39223296 17481728
Swap:      0      0      0
MemTotal:  516312 kB
MemFree:   394476 kB
MemShared: 41144 kB
Buffers:   38304 kB
Cached:    17072 kB
```

```
SwapTotal:      0 kB
SwapFree:       0 kB
#
```

The man page called `man proc` contains a detailed description of the `/proc` file system.

## 21.1.2 The `ps` command

You can use the `ps` command to snapshot the current running processes on the system and the resources used by each process. It provides many different options you can use to set the detail and appearance of the output, as shown:

```
# ps
  PID TTY          TIME CMD
 3313 console 00:00:00 mingetty
 2432 pts/1    00:00:00 login
 2433 pts/1    00:00:00 bash
 5153 pts/1    00:00:00 ps

# ps aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0   396   208 ?        S    May24   0:00 init
root         2  0.0  0.0     0     0 ?        SW   May24   0:00 [kmcheck]
root         3  0.0  0.0     0     0 ?        SW   May24   0:00 [kflushd]
root         4  0.0  0.0     0     0 ?        SW   May24   0:22 [kupdate]
root         5  0.0  0.0     0     0 ?        SW   May24   0:00 [kpiod]
root         6  0.0  0.0     0     0 ?        SW   May24   0:18 [kswapd]
bin         493  0.0  0.0  1164   420 ?        S    May24   0:00
/sbin/portmap
...
```

Refer to the man page for `ps` for more information and available parameters .

## 21.1.3 The `top` command

The `top` command, which provides information similar to the `ps` command, has a continuously self-updating interface with additional summary information. (Note that the `top` command does not work in a 3270 session, which you have when working from a VM virtual console.)

```
9:00am up 4 days, 16:19, 1 user, load average: 0.19, 0.07, 0.02
45 processes: 43 sleeping, 1 running, 1 zombie, 0 stopped
CPU states: 0.1% user, 0.3% system, 0.0% nice, 99.4% idle
Mem:  516312K av, 121904K used, 394408K free, 33276K shrd, 38304K buff
Swap:   OK av,      OK used,      OK free      17132K cached

  PID USER      PRI  NI  SIZE  RSS SHARE STAT  LIB %CPU %MEM  TIME COMMAND
 5807 root         8   0 1032 1032  864 R      0  0.5  0.1  0:00 top
```

```

1 root      0  0  208 208  172 S      0  0.0  0.0  0:00 init
2 root      0  0    0  0    0 SW     0  0.0  0.0  0:00 kmcheck
...

```

Refer to the man page for **top** for more information and available parameters.

## 21.1.4 The **vmstat** command

The **vmstat** command gives you information about memory, swap space, I/O activity as well as CPU usage, as shown:

```

# vmstat
procs          memory    swap          io    system    cpu
r  b  w  swpd  free  buff  cache  si  so  bi  bo  in  cs  us  sy  id
0  0  0    0 394360 38304 17116  0  0   1   1   0  18  1   1  99

```

An easy way to gather information continuously about resource usage on a system is to invoke **vmstat** in a cron job (cron is a daemon used to execute scheduled commands), as follows:

```
0 0 * * * /usr/bin/vmstat -n 300 288 > logfile.`date +%d%m%y` &
```

When this line is added to the file `/etc/crontab` (cron's configuration file), it produces output in 5-minute (300 seconds) intervals for 24 hours with one header at the top, piped into a log file, with the date as the file extension. You can use these generated log files for further analyses; for example, they could be used in spreadsheet applications or to generate graphs in Web pages using CGI scripts.

Refer to the man page for **vmstat** for more information.

## 21.1.5 The **df** command

With the **df** command, you can display the usage of complete disks/file systems. The **-h** flag makes the output easier to read, as shown:

```

# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/dasdb1     787M  214M  533M  29% /
/dev/dasdc1     1.4G  387M  926M  29% /home
/dev/dasdd1     6.8G  5.4G  1007M  85% /home/redbooks

```

As described in the man page for **df**, a set of options is available to manipulate the output.

## 21.1.6 The du command

The **du** command reports the amount of space used by files or complete directory structures, and provides options to manipulate the output:

```
# du
4    ./news/OLD
8    ./news
360  ./httpd
4    ./ircd
4    ./uucp
4    ./radacct
836  .
vmlinux1:/var/log # du -h
4.0k  ./news/OLD
8.0k  ./news
360k  ./httpd
4.0k  ./ircd
4.0k  ./uucp
4.0k  ./radacct
836k  .
# du -hs
836k  .
```

For example, with **du** you can create a list of the top 10 users who are consuming the most DASD space on your system. By writing a small cron job, you can easily create a weekly overview:

```
0 0 * * 6 du -s /usr/bin/du /home/* | sort -rg | head
```

Refer to the man page for **du** for more information.

## 21.1.7 The netstat command

The **netstat** command is used for network analysis; it provides a huge set of options for gathering information on network interfaces, socket connections and protocols. A sample is shown in Example 21-1:

*Example 21-1 netstat output*

---

```
# netstat -a
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 vmlinux1.itso.:www-http 9.12.2.169:wag-service  ESTABLISHED
tcp        0      0 vmlinux1.itso.:www-http 9.12.2.169:connection  ESTABLISHED
tcp        0      138 vmlinux1.itso.ib:telnet  9.12.2.169:bmc-ar       ESTABLISHED
tcp        0      0 *:bb                    *:.*                     LISTEN
```

```

tcp      0      0 *:www-http          *.*          LISTEN
tcp      0      0 *:smtp              *.*          LISTEN
tcp      0      0 *:printer           *.*          LISTEN
tcp      0      0 *:swat              *.*          LISTEN
tcp      0      0 *:http-rman         *.*          LISTEN
tcp      0      0 *:finger            *.*          LISTEN
tcp      0      0 *:pop3              *.*          LISTEN
tcp      0      0 *:login             *.*          LISTEN
tcp      0      0 *:shell             *.*          LISTEN
tcp      0      0 *:telnet            *.*          LISTEN
tcp      0      0 *:ftp               *.*          LISTEN
tcp      0      0 *:time              *.*          LISTEN
tcp      0      0 *:nfs                *.*          LISTEN
tcp      0      0 *:con                *.*          LISTEN
tcp      0      0 *:sunrpc            *.*          LISTEN
udp      0      0 *:snmp              *.*          LISTEN
udp      0      0 *:ntalk             *.*          LISTEN
udp      0      0 *:talk              *.*          LISTEN
udp      0      0 *:time              *.*          LISTEN
udp      0      0 *:nfs                *.*          LISTEN
udp      0      0 *:756                *.*          LISTEN
udp      0      0 *:703                *.*          LISTEN
udp      0      0 *:sunrpc            *.*          LISTEN
raw      0      0 *:icmp              *.*          7
raw      0      0 *:tcp                *.*          7

```

Active UNIX domain sockets (servers and established)

Proto	RefCnt	Flags	Type	State	I-Node	Path
unix	5	[ ]	DGRAM		378	/dev/log
unix	0	[ ACC ]	STREAM	LISTENING	394	/var/run/keyservsock
unix	0	[ ACC ]	STREAM	LISTENING	533	/var/run/printer
unix	0	[ ACC ]	STREAM	LISTENING	578911	/var/run/.nscd_socket
unix	0	[ ACC ]	STREAM	LISTENING	563	/var/run/sendmail/control
unix	0	[ ]	DGRAM		588	
unix	0	[ ]	DGRAM		561	
unix	0	[ ]	DGRAM		480	
unix	0	[ ]	DGRAM		449	
unix	0	[ ]	DGRAM		433	
#						

There are many situations where you will need to use **netstat**, so refer to the man pages as necessary.

## 21.1.8 Quotas

Using quotas, you can limit the amount of space that can be used on file systems by users and groups. To do this, you have to enable quotas for the file system, and set the quota policies for the users and groups.

For example, to enable quotas for your /home file system, you have to modify the entry for /home in /etc/fstab as follows:

```
/dev/dasdd1 /home ext2 defaults,usrquota,grpquota 1 2
```

Then remount the /home file system:

```
# mount -o remount /home
# mount
/dev/dasda1 on / type ext2 (rw)
none on /proc type proc (rw)
/dev/dasdc1 on /opt type ext2 (rw)
/dev/dasdd1 on /home type ext2 (rw,usrquota,grpquota)
none on /dev/pts type devpts (rw,gid=5,mode=620)
```

Next, run **quotacheck** to create the files on /home where the quotas are stored. Then activate the quotas for the file system with the **quotaon** command:

```
# quotacheck /home
Cannot get exact used space... Results might be inaccurate.
# quotaon /home
```

You can create and modify the quotas for different users with the **edquota** command:

```
# edquota user1
...
Edit block and inode quota for user user1:
Device /dev/dasdd1 (/home):
Used 56KB, limits: soft=10000 hard=15000
Used 14 inodes, limits: soft=100 hard=120
...
#
```

In this example, we set the soft space limit for user1 to 10 MB, and the hard limit to 15 MB. This means that if user1 exceeds the 10 MB border, he will receive warning messages; if he reaches 15 MB, he will not be able to allocate more DASD space. We did the same with the inodes, allowing user1 to create up to 100 files as a soft limit and 120 files as a hard limit.

You can also copy these quotas for other users with the **edquota** command, so that you don't have to edit the quotas for each user by hand. Therefore, in our example, where all normal users have user IDs above 500, we can apply the quotas from user1 to all of them with the following line, utilizing the **awk** command to process the output of **edquota**:

```
# edquota -p user1 `awk -F: '$3 > 499 {print $1}' /etc/passwd`
```

You can obtain a report on the system quotas by using the **repquota** command as follows:

```

# repquota -a
*** Report for user quotas on device /dev/dasdd1 (/home)
Block grace time: 7 days; Inode grace time: 7 days
      Block limits                File limits
User      used  soft  hard  grace  used  soft  hard  grace
-----
root      --    20    0    0              6    0    0
user1     --    56 10000 15000          14   100  120
user2     --    56 10000 15000          14   100  120
user3     --    56 10000 15000          14   100  120

```

For more information on how to implement and use quotas, refer to the man pages.

## 21.2 Administration tools

In this section, we discuss tools you can use to manage a Linux system. These tools are supplementary to the common Linux command line tools, and generally provide a graphical interface. They can be used to manage users and network services, add software packages, and more.

### 21.2.1 YaST

Yet another Setup Tool, or YaST, is the administration tool that comes with the SuSE distribution. YaST is used for installation, administration and software package management on SuSE Linux systems. YaST is developed and maintained by SuSE, and has a special license and is not GPL. For more information on how to use YaST, refer to 6.1, “Using YaST” on page 92.

### 21.2.2 Linuxconf

The package linuxconf is an Open Source administration tool for Linux systems. It is the default administration tool on Red Hat Linux systems, but can be used for other distributions as well.

With linuxconf, you can manage users, system services and devices for your Linux system. It has a dialog-based command line interface and can also be configured for Web access. For more information on how to use linuxconf, refer to 12.2, “Installing linuxconf” on page 178.



## 21.2.3 Webmin

Webmin is an open source, Web-based system administration tool. It consists of a huge set of Perl scripts and comes with its own http server. You can manage users, configure daemons, and more from your Web browser. Webmin is available at:

<http://www.webmin.com>

### Installation

Download the Webmin source package to a local directory and extract it into the directory `/usr/local` as follows:

```
# ls
. .. webmin-0.85.tar.gz
# tar -xzf webmin-0.85.tar.gz -C /usr/local/
# cd /usr/local/webmin-0.85/
```

Run the setup script and answer the questions as shown in Example 21-2.

#### *Example 21-2 Webmin setup*

---

```
vmlinux1:/usr/local/webmin-0.85 # ./setup.sh
*****
*           Welcome to the Webmin setup script, version 0.85           *
*****
Webmin is a web-based interface that allows Unix-like operating
systems and common Unix services to be easily administered.

Installing Webmin in /usr/local/webmin-0.85 ...

*****
Webmin uses separate directories for configuration files and log files.
Unless you want to run multiple versions of Webmin at the same time
you can just accept the defaults.

Config file directory [/etc/webmin]:
Log file directory [/var/webmin]:

*****
Webmin is written entirely in Perl. Please enter the full path to the
Perl 5 interpreter on your system.

Full path to perl (default /usr/bin/perl):

Testing Perl ...
Perl seems to be installed ok

*****
For Webmin to work properly, it needs to know which operating system
```

type and version you are running. Please select your system type by entering the number next to it from the list below

- 
- |                      |                              |
|----------------------|------------------------------|
| 1) Sun Solaris       | 2) Caldera OpenLinux eServer |
| 3) Caldera OpenLinux | 4) Redhat Linux              |
| 5) Slackware Linux   | 6) Debian Linux              |
| 7) SuSE Linux        | 8) Corel Linux               |
| 9) TurboLinux        | 10) Cobalt Linux             |
| 11) Mandrake Linux   | 12) Delix DLD Linux          |
| 13) Conectiva Linux  | 14) MkLinux                  |
| 15) LinuxPPC         | 16) XLinux                   |
| 17) LinuxPL          | 18) Linux From Scratch       |
| 19) Trustix          | 20) Cendio LBS Linux         |
| 21) Ute Linux        | 22) FreeBSD                  |
| 23) OpenBSD          | 24) BSDI                     |
| 25) HP/UX            | 26) SGI Irix                 |
| 27) DEC/Compaq OSF/1 | 28) IBM AIX                  |
| 29) SCO UnixWare     | 30) SCO OpenServer           |
| 31) MacOS Server X   |                              |
- 

Operating system: 7

Please choose which version of SuSE Linux you are running, by entering the number next to it from the list below

- 
- |                   |                    |
|-------------------|--------------------|
| 1) SuSE Linux 5.1 | 2) SuSE Linux 5.2  |
| 3) SuSE Linux 5.3 | 4) SuSE Linux 6.0  |
| 5) SuSE Linux 6.1 | 6) SuSE Linux 6.2  |
| 7) SuSE Linux 6.3 | 8) SuSE Linux 6.4  |
| 9) SuSE Linux 7.0 | 10) SuSE Linux 7.1 |
- 

Version: 9

\*\*\*\*\*

Webmin uses its own password protected web server to provide access to the administration programs. The setup script needs to know :

- What port to run the web server on. There must not be another web server already using this port.
- The login name required to access the web server.
- The password required to access the web server.
- The hostname of this system that the web server should use.
- If the webserver should use SSL (if your system supports it).
- Whether to start webmin at boot time.

Web server port (default 10000):

Login name (default admin):

Login password:

Password again:

Web server hostname (default vmlinux1): vmlinux1.itso.ibm.com

```
The Perl SSLay library is not installed. SSL not available.
Start Webmin at boot time (y/n): y
*****
Creating web server config files..
..done

Creating access control file..
..done

Inserting path to perl into scripts..
..done

Creating start and stop scripts..
..done

Copying config files..
..done

Configuring Webmin to start at boot time..
Created init script /sbin/init.d/webmin
..done

Creating uninstall script /etc/webmin/uninstall.sh ..
..done

Changing ownership and permissions ..
..done

Attempting to start Webmin mini web server..
Starting Webmin server in /usr/local/webmin-0.85
..done

*****
Webmin has been installed and started successfully. Use your web
browser to go to

    http://vmlinux1.itso.ibm.com:10000/

and login with the name and password you entered previously.

vmlinux1:/usr/local/webmin-0.85 #
```

---

And that's it; you can now login with your Web browser (see Figure 21-1 on page 406).

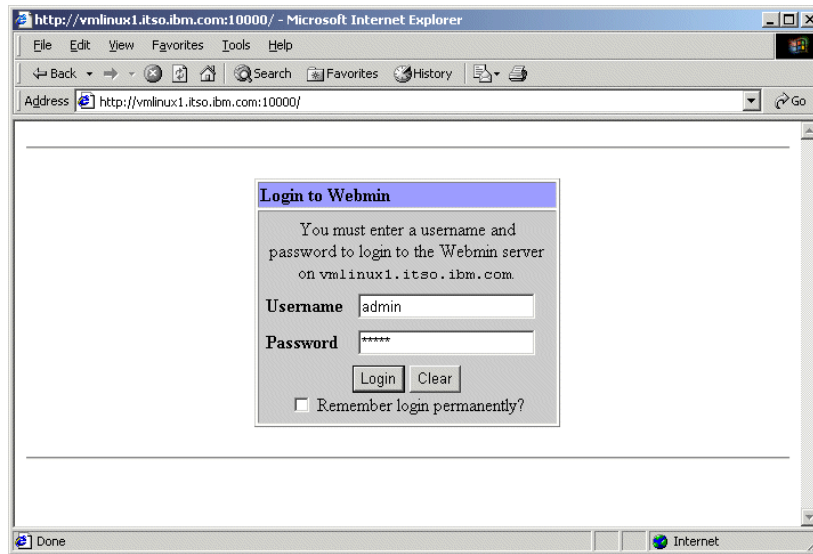


Figure 21-1 Webmin login

By default, you have five groups for handling different administrative tasks: the Webmin menu, the System section, the Servers section, the Hardware section, and the Others section.

The Webmin menu (not shown) offers choices for configuring Webmin itself.

The System section (see Figure 21-2 on page 407) offers options for configuring the boot process, administering user authentication, and more.

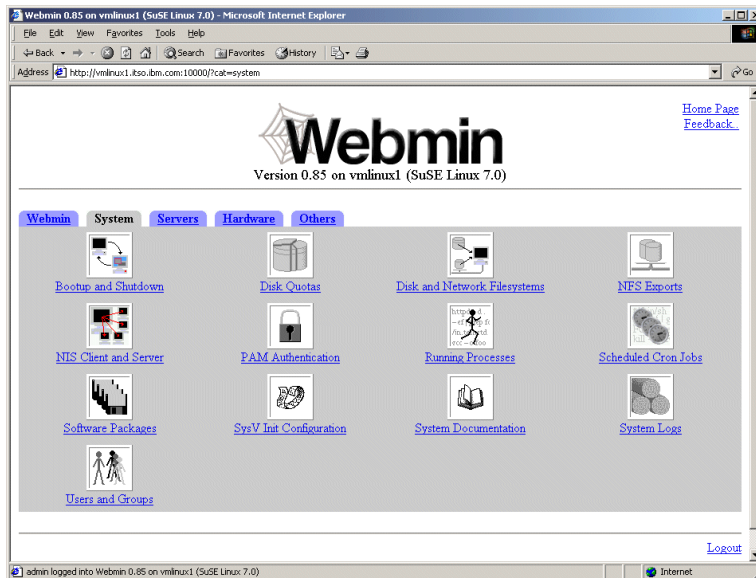


Figure 21-2 Webmin System section

From the Servers section (see Figure 21-3), you can configure the daemons.

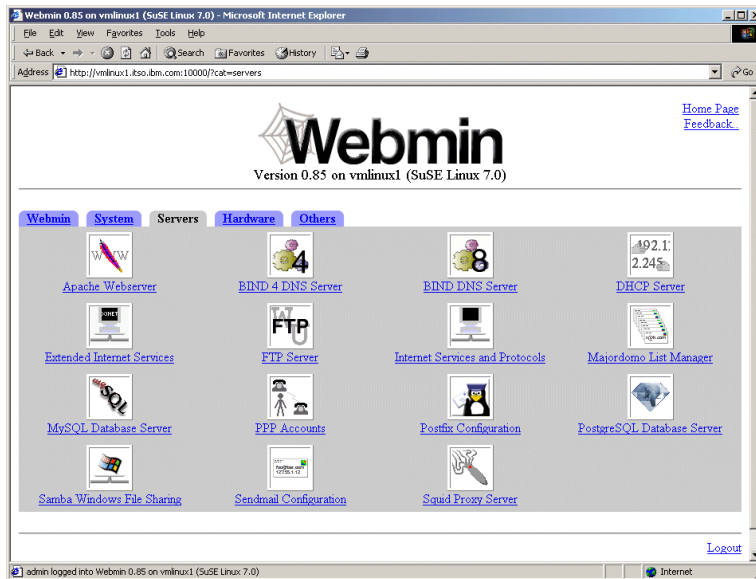


Figure 21-3 Webmin Servers section

The Hardware section (not shown) allows you to integrate hardware into your system. The Others section (not shown) deals with telnet, ssh, and so on.

You can also set up Webmin for secure Web access using SSL connections; refer to the appropriate documentation.

## 21.2.4 AdminUX

AdminUX is an “all-in-one” UNIX administration tool developed by Green Light Advantage. It is not open source. It provides a single menu-driven interface for system administration. It can be used for user administration, supervising daemons, monitoring system resources and more. It provides a large, preconfigured set of monitors and alarms that work pretty much right “out of the box”. In addition, it also allows you to reconfigure existing monitors—or write your own.

For more information, visit:

<http://www.gladvantage.com>

## 21.3 Remote network management tools

With established network connections and secure tools such as `ssh`, an administrator can always connect to a system and work on it as if the computer were local. However, this effort becomes increasingly difficult and inefficient as more and more systems need to be monitored and maintained.

This leads to a need for tools and solutions that provide a single focal point, where data from all systems can be collected and then presented in different views.

Automation can provide a further solution; some programmable agents may react automatically to certain events without invoking manual action from the system administrator, thus automating some system programmer tasks.

In this section, we describe some of the tools and products that are available which make remote network management easier and more efficient.

## 21.3.1 snmp tools

The Simple Network Management Protocol (SNMP) was developed by Carnegie-Mellon University and allows the management of large networks with the help of the Management Information Base (MIB). The MIB contains information about servers, routers, workstations, printers and other details about the network.

In order to query information from remote systems, there has to be a snmp daemon (**snmpd**) running on the remote system. This daemon is included in most of the distributions and the configuration file is `/etc/snmpd.conf`. In the configuration file, you specify the type of information that will be available for different communities.

The snmp daemon is started by the runlevel start/stop scripts. With the SuSE distribution, you would start **snmpd** with:

```
# /sbin/init.d/snmpd start
Starting snmpd:                               done
#
```

You can set snmpd to start automatically at reboot time with the **chkconfig** command (however, on SuSE, this is a disabled script).

A set of command line tools that come with the snmp package can be used to query and set MIB variables, for example:

snmpget	query database variables
snmpnetstat	similar to netstat, with usage of the SNMP-protocol
snmpset	set variables
snmptrap	send event signals
snmpwalk	browse a MIB tree

Example 21-3 illustrates the use of **snmpwalk**:

*Example 21-3 snmpwalk sample*

---

```
# snmpwalk vmlinux2 public system
system.sysDescr.0 = Linux vmlinux2.itso.ibm.com 2.2.19-0.07vrdr #1 SMP Tue
Apr 24 15:17:30 EDT 2001 s390
system.sysObjectID.0 = OID: enterprises.ucdavis.ucdSnmpAgent.linux
system.sysUpTime.0 = Timeticks: (421) 0:00:04.21
system.sysContact.0 = Root <root@localhost> (configure
/etc/snmp/snmp.local.conf)
system.sysName.0 = vmlinux2.itso.ibm.com
system.sysLocation.0 = Unknown (edit /etc/snmp/snmpd.conf)
system.sysORLastChange.0 = Timeticks: (4) 0:00:00.04
system.sysORTable.sysOREntry.sysORID.1 = OID: ifMIB
```

```

system.sysORTable.sysOREntry.sysORID.2 = OID:
.iso.org.dod.internet.snmpV2.snmpModules.snmpMIB
system.sysORTable.sysOREntry.sysORID.3 = OID: tcpMIB
system.sysORTable.sysOREntry.sysORID.4 = OID: ip
system.sysORTable.sysOREntry.sysORID.5 = OID: udpMIB
system.sysORTable.sysOREntry.sysORID.6 = OID:
.iso.org.dod.internet.snmpV2.snmpModules.snmpVacmMIB.vacmMIBConformance.vac
mMIBGroups.vacmBasicGroup
system.sysORTable.sysOREntry.sysORID.7 = OID:
.iso.org.dod.internet.snmpV2.snmpModules.snmpFrameworkMIB.snmpFrameworkMIBC
onformance.snmpFrameworkMIBCompliances.snmpFrameworkMIBCompliance
system.sysORTable.sysOREntry.sysORID.8 = OID:
.iso.org.dod.internet.snmpV2.snmpModules.snmpMPDMIB.snmpMPDMIBConformance.s
nmpMPDMIBCompliances.snmpMPDCompliance
system.sysORTable.sysOREntry.sysORID.9 = OID:
.iso.org.dod.internet.snmpV2.snmpModules.snmpUsmMIB.usmMIBConformance.usmMI
BCompliances.usmMIBCompliance
system.sysORTable.sysOREntry.sysORDescr.1 = The MIB module to describe
generic objects for network interface sub-layers
system.sysORTable.sysOREntry.sysORDescr.2 = The MIB module for SNMPv2
entities
system.sysORTable.sysOREntry.sysORDescr.3 = The MIB module for managing TCP
implementations
system.sysORTable.sysOREntry.sysORDescr.4 = The MIB module for managing IP
and ICMP implementations
system.sysORTable.sysOREntry.sysORDescr.5 = The MIB module for managing UDP
implementations
system.sysORTable.sysOREntry.sysORDescr.6 = View-based Access Control Model
for SNMP.
system.sysORTable.sysOREntry.sysORDescr.7 = The SNMP Management
Architecture MIB.
system.sysORTable.sysOREntry.sysORDescr.8 = The MIB for Message Processing
and Dispatching.
system.sysORTable.sysOREntry.sysORDescr.9 = The management information
definitions for the SNMP User-based Security Model.
system.sysORTable.sysOREntry.sysORUpTime.1 = Timeticks: (2) 0:00:00.02
system.sysORTable.sysOREntry.sysORUpTime.2 = Timeticks: (2) 0:00:00.02
system.sysORTable.sysOREntry.sysORUpTime.3 = Timeticks: (2) 0:00:00.02
system.sysORTable.sysOREntry.sysORUpTime.4 = Timeticks: (2) 0:00:00.02
system.sysORTable.sysOREntry.sysORUpTime.5 = Timeticks: (2) 0:00:00.02
system.sysORTable.sysOREntry.sysORUpTime.6 = Timeticks: (2) 0:00:00.02
system.sysORTable.sysOREntry.sysORUpTime.7 = Timeticks: (4) 0:00:00.04
system.sysORTable.sysOREntry.sysORUpTime.8 = Timeticks: (4) 0:00:00.04
system.sysORTable.sysOREntry.sysORUpTime.9 = Timeticks: (4) 0:00:00.04
End of MIB
#

```

---



Especially for large environments, SNMP-based systems management is almost a standard: most of the tools described in the following sections have SNMP support, and most of the commercial system management products are based upon SNMP. For more information on how to configure `snmpd` and how to use `snmp` tools, refer to the man pages.

### 21.3.2 Scotty/Tkined

The software packets Scotty and Tkined were developed at the Technical University of Braunschweig (Germany) by Juergen Schoenwaelder. These packages are included in many distributions today. If not in your distribution, you can obtain the source code from:

<http://wwwhome.cs.utwente.nl/~schoenw/scotty/>

#### Features

Scotty is based on the Tool Command Language (Tcl) and provides various special network management extensions (Tnm Tcl extension) for use in other scripts or programs. According to the home page, the Tnm extensions supports the following protocols:

1. SNMP (SNMPv1, SNMPv2c, SNMPv2u including access to MIB definitions)
2. ICMP (echo, mask, timestamp and udp/icmp traceroute requests)
3. DNS (a, ptr, hinfo, mx and soa record lookups)
4. HTTP (server and client side)
5. SUN RPC (portmapper, mount, rstat, etherstat, pcnfs services)
6. NTP (version 3 mode 6 request)
7. UDP (send and receive UDP datagrams - no channels yet)

Tkined is a graphical network editor for the X-Windows system, based on Tcl/Tk (see Figure 21-4 on page 412). It provides a framework for an extensible network management platform, so you can run `tkined` either on Linux for S/390 and redirect the output to your local X-Server, or just run it on your local Linux workstation.

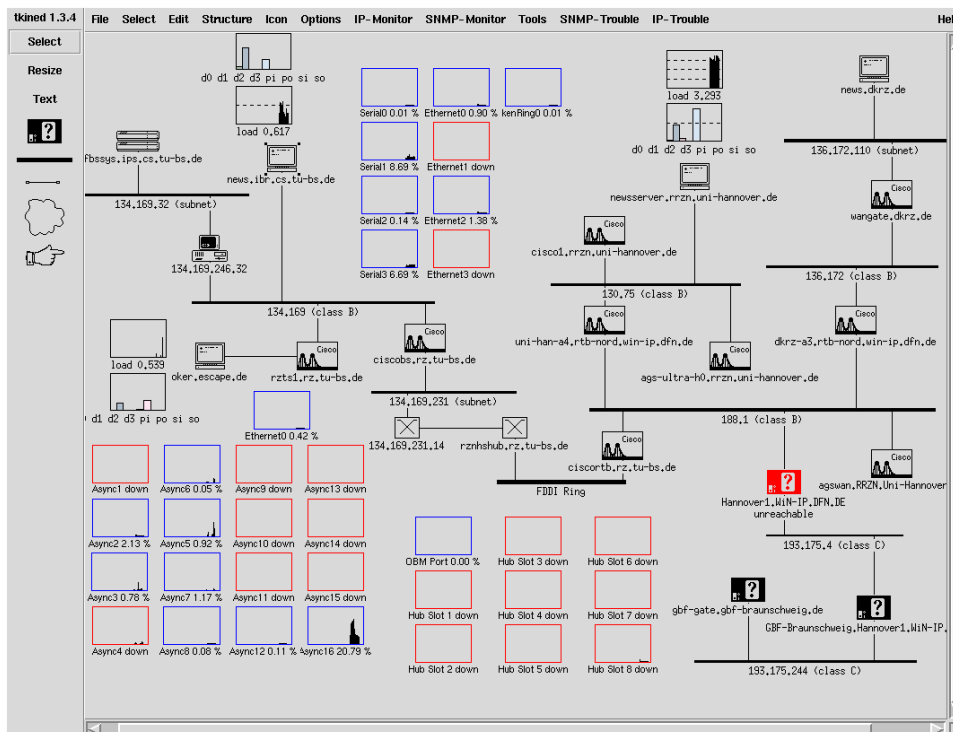


Figure 21-4 Scotty/Tkined example from home page

## Installation and configuration

To install Scotty/Tkined you usually only have to install the scotty rpm package. If you don't have the binary rpm, get the source from the Web page and do a install from source.

With **tkined**, you can easily draw your network maps using predefined objects (see Figure 21-5 on page 413); it's almost as easy to use as any other vector-oriented drawing program. You would then change the properties of the different objects, like IP addresses and names, to correspond to your network nodes.

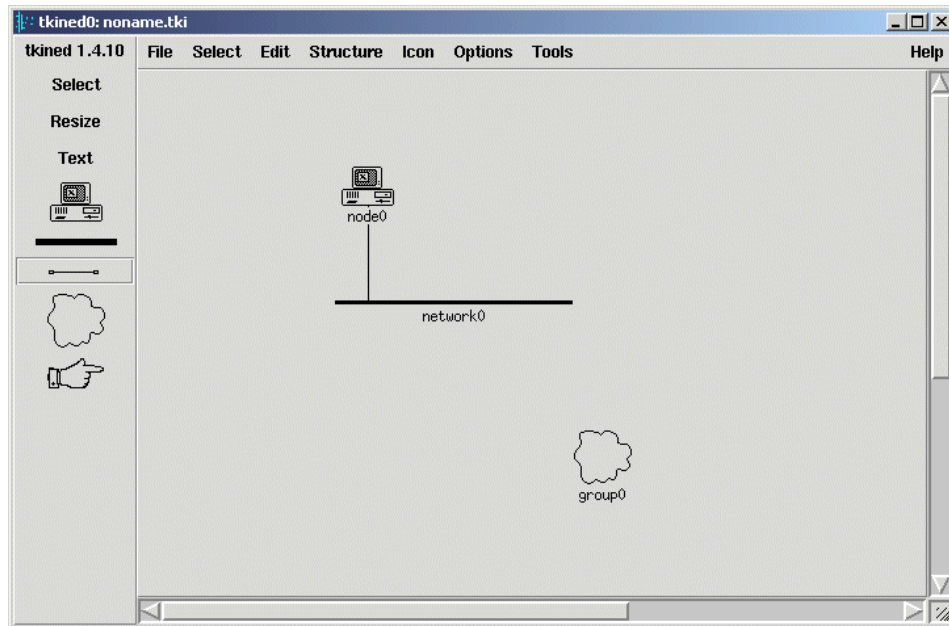


Figure 21-5 Scotty/Tkined creating network maps

A set of tools is available under the Tools menu, including some which you can use to automatically discover your network, so that you don't have to draw it all by yourself. These auto-generated maps then usually need some editing and customizing.

You have several IP and SNMP monitoring and troubleshooting tools available under the Tools menu, as follows:

IP Trouble	Tools like ping, telnet, finger
IP Monitor	IP-based monitors like cpu and disk activity
IP Layout	Tools for designing network maps
IP Discover	Port scanners and traceroutes to discover the network
SNMP Trouble	Query SNMP variables from other hosts
SNMP Host	Query SNMP system information from other hosts
SNMP Monitor	SNMP-based monitors like ethernet load, tcp services
SNMP Browser	Browse the MIB of other hosts
SNMP Tree	Graphical presentation of the MIB tree

You use these tools by selecting them and applying them to your network object.

Figure 21-6 shows a network map from the temporary network we used during our testing, as an example of how it looks when you use several tools and monitors:

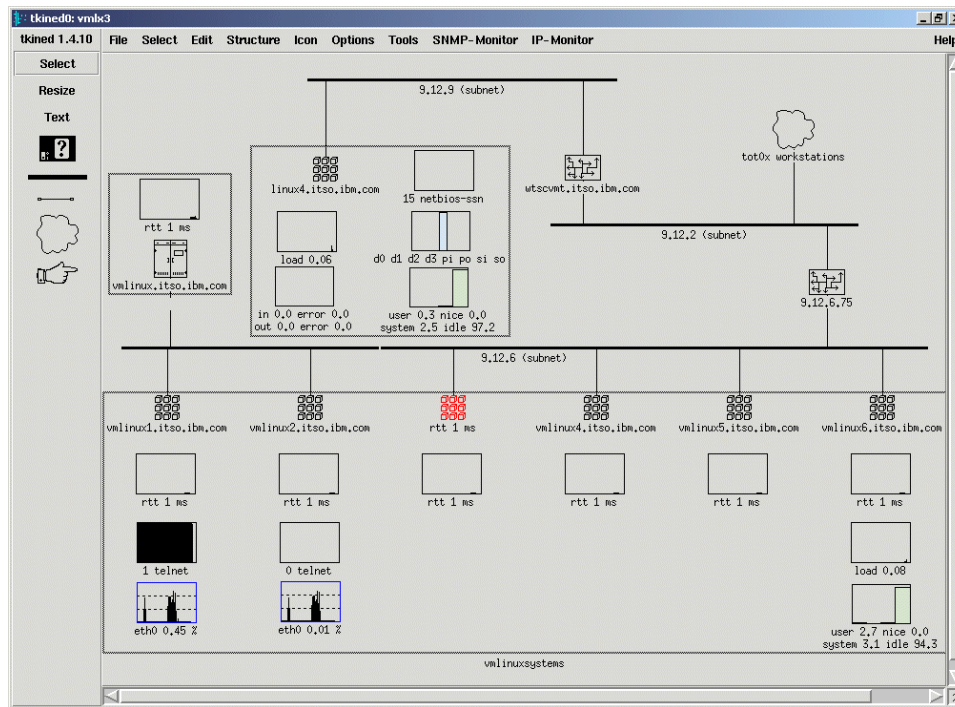


Figure 21-6 Scotty/Tkined sample network map

All graphs in this figure depict one currently-running monitor process for the various hosts. You can set the refreshing intervals and alarms, and also create your own monitors. For basic setups, **tkined** is very easy to use—it took us only about 30 minutes to create this map. When creating larger network maps, it is useful to group the systems.

Refer to the Scotty/Tkined documentation for more detailed information about installation, customization and use.

### 21.3.3 Big Brother

Big Brother is a Web-based client-server monitoring tool which runs on many different platforms. You can obtain Big Brother as source code from:

<http://www.bb4.com>

Big Brother is free for personal use only; be sure to read the license information on the home page.

## Features

Big Brother provides a consolidated view for multiple monitors for many different hosts on Web pages. The status of each system and service is presented in a matrix form, with colored and animated dots for each item (see Figure 21-7). You can view historical data online, create reports, and write additional plug-ins. Big Brother provides notification functions, such as automatic e-mail or SMS messages and more.

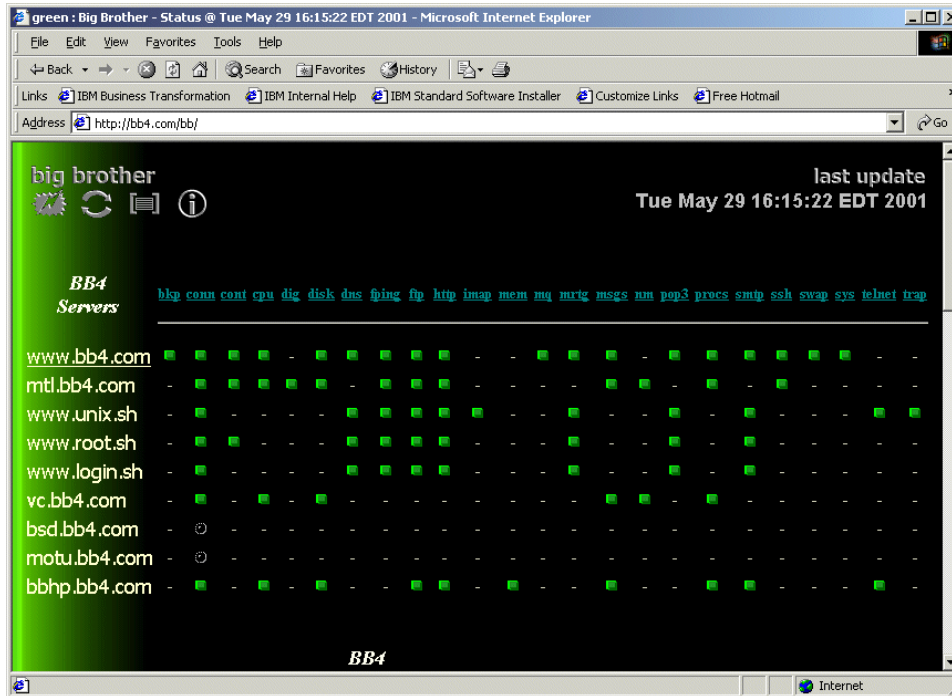


Figure 21-7 Big Brother example from home page

## Installation and configuration

First you have to download the source from the Big Brother home page. We recommend that you run Big Brother under a separate user. In our example, therefore, we created a user `bb` with its own group `bb` and `/home/bb` as its home directory.

After that, logon as that user and extract the source files from the archive. The first extract step leaves you with a readme file (which you should read for the first install) and another tar archive, which you extract afterwards:

```
$ ls
bb-1.8a.tar.gz
$ tar -xzf bb-1.8a.tar.gz
$ ls
README.FIRST bb-1.8a.tar.gz bb18a.tar
$ tar -xf bb18a.tar
$ ls
README.FIRST bb-1.8a.tar.gz bb18a bb18a.tar bbvar
$
```

Change into the newly created directory bb18a. Here you will find several readme files for installation and configuration, which you should read (at least the installation manual) before starting the installation. Change into the directory install, **su** to root, and start the configuration script as shown in Example 21-4.

*Example 21-4 Big Brother installation*

---

```
$ cd bb18a
# cd install/
# su
Password:
# ./bbconfig
...      <- license information will be displayed
Do you agree to the terms of this license (y/n): y
...
Please enter your OS: [suse]:
---> OK, we'll try suse...

*** WARNING: Don't run BB as root ! ***
           Executing BB as root is not recommended

Prevent the execution of BB as user 'root' (y/n) [y]:
---> OK... BB is NOT ALLOWED to run as root

BB will only start under a designated user id. The
startup script will verify that the current user ID
and the designated user ID are identical.
Note: This check is only performed during the startup script.
     It does not prevent the execution of other BB
binaries/scripts
     while working using another user ID. It only prevents
     you from starting BB while working another user ID.

What will be the user ID of BB [bb]:
---> BB will only run from user 'bb'
```

Making sure BBHOME </home/bb/bb18a> is writable...

----> OK, /home/bb/bb18a is fine...

Do you want to preserve the old style directory structure ?  
You may want to do so if you use BB extensions or  
externals that do not understand the new directory structure.  
This option is \*NOT\* recommend as keeping the old directory  
around represents a security risk.

Old-style directory structure (y/n): [n]

When you set up your machines, you should use Fully Qualified  
Domain names, this means you use the whole name, like www.bb4.com,  
instead of just 'www'. This is recommended.

Use FQDN (y/n): [y]

----> Good, we'll use FQDN

Big Brother creates HTML pages with the status of your network.  
You'll need a web server to publish this information.

What machine will be the BBDISPLAY [vmlinux1]: **vmlinux1.itso.ibm.com**

----> OK... vmlinux1.itso.ibm.com will be a BBDISPLAY

Big Brother sends important messages to a pager server. This  
machine will at a minimum to be able to send mail.

What machine will be the BBPAGER [vmlinux1.itso.ibm.com]:

----> OK... vmlinux1.itso.ibm.com will be a BBPAGER

Some questions regarding the current host  
(vmlinux1) will be asked.

Is this host a BBDISPLAY host (y/n): [y]

Is this host a BBPAGER host (y/n): [y]

Enter the default recipient: [root@localhost]

Since Big Brother produces results to be displayed on web  
pages, we need to know where to view these results.

Enter the base URL for BB [/bb]: **/bb/www**

----> OK... Big Brother will live under http://vmlinux1.itso.ibm.com/bb/www

Big Brother also uses CGI scripts to create dynamic output.  
What directory do these scripts live in?

```
Enter CGI directory [/home/www/httpd/cgi-bin]: /usr/local/httpd/cgi-bin
---> OK... CGI scripts will live at /usr/local/httpd/cgi-bin
Enter the base URL of the CGI scripts [/cgi-bin]:
---> OK... The base URL location of CGI scripts is in /cgi-bin
```

```
-----
--> UPDATING Makefile
--> UPDATING runbb.sh
--> UPDATING bbsys.local
--> CHECKING COMMAND PATHNAMES
*** Verifying pathnames to necessary commands...
*** All pathnames OK.
--> UPDATING bbdef.sh
--> UPDATING URL location
--> INSTALLING CGI scripts
```

BB needs to set the group name of the www/rep directory  
to the group name of the web server by using its user name

```
Enter web server user id [nobody]: wwwrun
```

You may override the group name determined by the previous step.

```
Enter group name [nogroup]:
```

```
--> SETTING WRITE PERMISSION FOR OWNER AND GROUP FOR www/rep
--> CHANGING THE GROUP ID OF www/rep
--> UPDATING pager scripts
```

```
-----
--> Done.
```

---

After the configuration, you need to compile and install the program:

```
# cd ../src
# make
...
# make install
```

You need to customize the Big Brother host configuration file in /etc:

```
# cd ../etc/
# vi bb-hosts
```



Comment out all the predefined hosts and insert the hostnames and services you want to monitor. In our example configuration, we are monitoring Linux systems under VM and their HTTP and FTP servers, so we set the system `vmlinux1` as the Big Brother display and network monitor server, as follows:

```
# cat bb-hosts
...
9.12.6.98 vmlinux1.itso.ibm.com # BBDISPLAY BBNET ftp smtp
http://vmlinux1.itso.ibm.com/
9.12.6.99 vmlinux2.itso.ibm.com # ftp smtp http://vmlinux2.itso.ibm.com/
9.12.6.100 vmlinux3.itso.ibm.com # ftp smtp http://vmlinux3.itso.ibm.com/
9.12.6.80 vmlinux4.itso.ibm.com # ftp smtp http://vmlinux4.itso.ibm.com/
9.12.6.76 vmlinux5.itso.ibm.com # ftp smtp http://vmlinux5.itso.ibm.com/
9.12.6.74 vmlinux6.itso.ibm.com # ftp smtp http://vmlinux6.itso.ibm.com/
```

You can check the config files with checkscrips that are located in the `etc` directory:

```
# ./bbchkcfg.sh
```

If any comments are displayed, please fix the entries in your configuration

```
*** Verifying pathnames to necessary commands...
```

```
*** All pathnames OK.
```

```
bbwarnsetup.cfg: /dev/cuaa0 is an invalid entry in the ttyline token
```

```
This is a valid error only if this host is a BBPAGER host
```

```
# ./bbchkhosts.sh
```

If any comments are displayed, please fix the entries in your configuration

Note that some error messages for tags of external scripts

After that, go back to the `bb` home directory and change the ownership of the Big Brother directories to the `bb` user.

```
# cd ../..
# chown -R bb bb18a
# chown -R bb bbvar
```

Then you have to create a symbolic link in your Web server's html document root, pointing to the Big Brother Web pages. In our case, we used Apache and placed the symbolic link into a separate directory to protect the access, as shown:

```
# cd /                                     <- only because the commands are so long
...
# mkdir /usr/local/httpd/htdocs/bb
# ln -s /home/bb/bb18a/www /usr/local/httpd/htdocs/bb/www
```

You need to configure the Web server to follow symbolic links in the Big Brother directory. Since this is a security concern, we protected this directory by user ID and password. So in our case, we added the following directory statement in the Apache configuration file `/etc/httpd/httpd.conf`:

```
# section for Big Brother
<Directory /usr/local/httpd/htdocs/bb>
    AllowOverride AuthConfig
    Options FollowSymLinks
    AuthName "Big Brother"
    AuthType Basic
    AuthUserFile /etc/httpd/passwd
    require user bb
</Directory>
```

Then you need to create the password file (or add a entry) for the bb user with the `htpasswd` command:

```
# htpasswd -c /etc/httpd/passwd bb
New password:
Re-type new password:
Adding password for user bb
#
```

Now you can go back to the bb user and start Big Brother:

```
# exit
exit
$ cd ..
$ ./runbb.sh start
Starting the Big Brother System & Network monitor
    Starting Big Brother Daemon (bbd)...
    Starting Local tests (bb-local)...
    Starting Network tests (bb-network)...
    Starting Display process (bb-display)...
Big Brother 1.8a started
```

At this point, you should be able to login via your Web browser (see Figure 21-8 on page 421) as the bb user when you go to:

```
http://<yourhost>/bb/www
```



Figure 21-8 Big Brother Web login

After login, you should see a screen something like that shown in Figure 21-9, depending on your configuration:

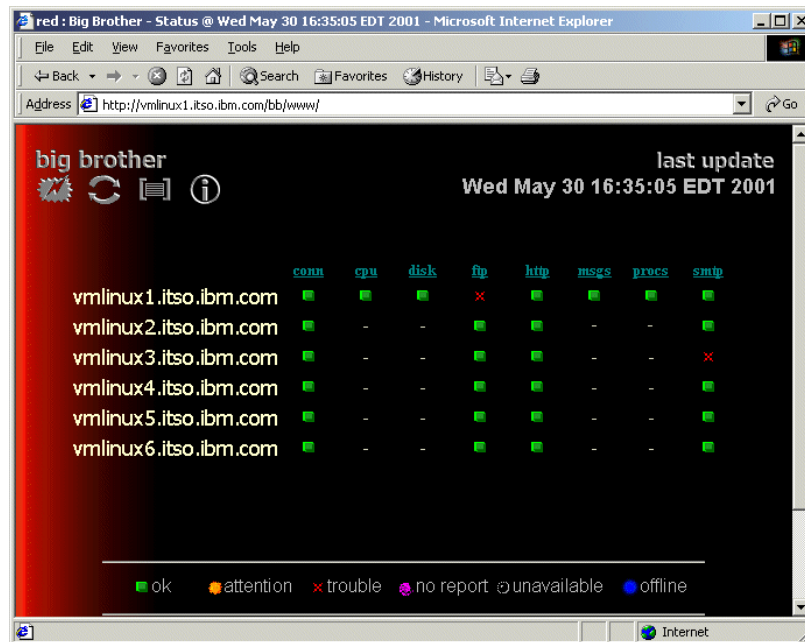


Figure 21-9 Big Brother first page

As you can see, network services like ftp, http and smtp can be monitored without having a Big Brother client installed on the remote hosts. For the other data, we needed to deploy clients on the systems we want to monitor.

In our case, we created a client package for one of our hosts specified in `bb-hosts`, by using the `bbclient` command:

```
$ ./bbclient vmlinux2.itso.ibm.com
*** Adding standard things...
*** Creating tar file for Big Brother on vmlinux2.itso.ibm.com
...
$ cd ../../
$ ls
README.FIRST  bb-vmlinux2.itso.ibm.com.tar  bb18a.tar
bb-1.8a.tar.gz  bb18a                          bbvar
```

You should add a `bb` user, in the same way as on the server, on the remote host where you want to install the client. After that, transfer the newly created archive to the `bb` user's home directory on the remote host as follows:

```
$ ftp vmlinux2
Connected to vmlinux2.itso.ibm.com.
...
ftp> put bb-vmlinux2.itso.ibm.com.tar
```

Login to the remote host as the `bb` user and untar the client archive:

```
$ ls
bb-vmlinux2.itso.ibm.com.tar
$ tar -xf bb-vmlinux2.itso.ibm.com.tar
```

Start the client the same way as the server, by running the `runbb.sh` script:

```
$ cd bb18a/
$ ./runbb.sh start
Starting the Big Brother System & Network monitor
    Starting Local tests (bb-local)...
Big Brother 1.8a started
```

After a while you should see an updated display in the Big Brother browser window, as shown in Figure 21-10 on page 423.

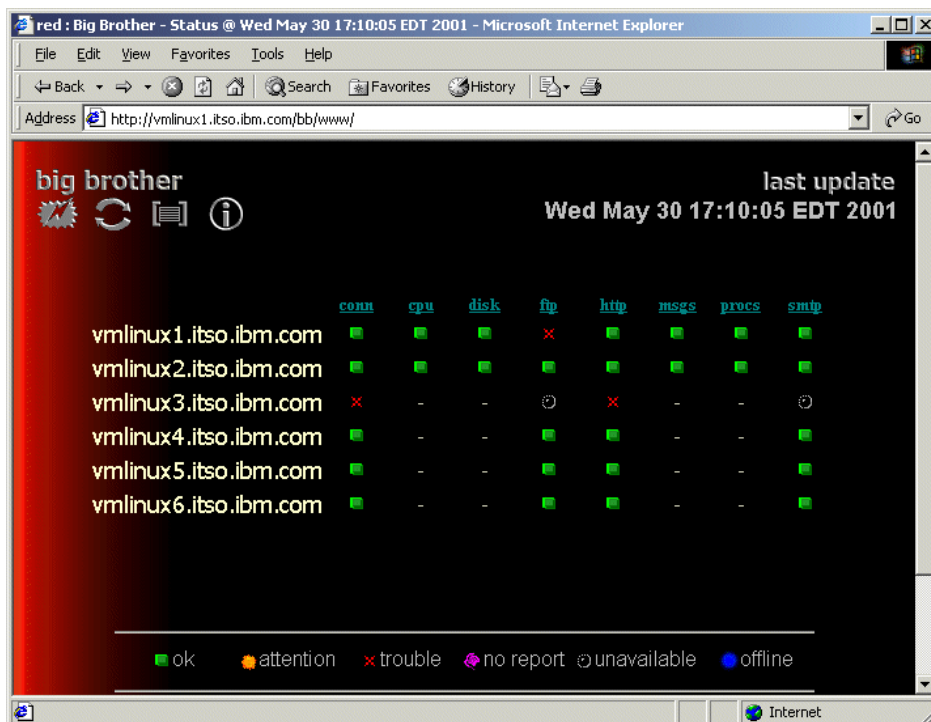


Figure 21-10 Big Brother display with 1 client

Repeat these steps for every Big Brother client you want to deploy.

To automatically start Big Brother at system boot, you can put the following line in your start-up script:

```
su - bb -c "cd /home/bb/bb18a;./runbb.sh start"
```

Big Brother provides history views for the different monitors when you click on the symbols. You can also generate reports that show the average availability for your systems and services for a defined period of time. To get help on specific monitors, click the monitor name in the column header. For more information how to configure and set up monitors, refer to the Big Brother documentation.

### 21.3.4 Remstats

Remstats is a Web-based system monitoring program written by Thomas Erskine from the Communication Research Center in Canada. It gathers data from remote systems, stores it by using the RRDtool, and presents it in charts. With Remstats, you can monitor your systems and automatically issue alerts.

Remstats can be obtained from:

<http://silverlock.dgim.crc.ca/remstats/release>

You also need the rrdtool and the Time::HiRes Perl module, which are available from the same site.

The RRDtool is a system that you can use to store and display time-series data. It stores the data in a very compact way that will not expand over time, and it presents useful graphs by processing the data to enforce a certain data density. It can be used either via simple wrapper scripts (from shell or Perl), or via front-ends that poll network devices and put friendly-user interfaces on them. For more information, refer to:

<http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/>

## Features

Remstats has many different monitoring tools and alert functions. All together, Remstats is a system of programs to:

- ▶ Gather data from servers and routers
- ▶ Store and maintain the data for long periods
- ▶ Produce graphs and Web pages tying them together
- ▶ Monitor the data for anomalous behaviors and issue alerts

For an impression of what can be done with the Remstats tool, visit the home page where live data from many systems is provided; see Figure 21-11 on page 425.

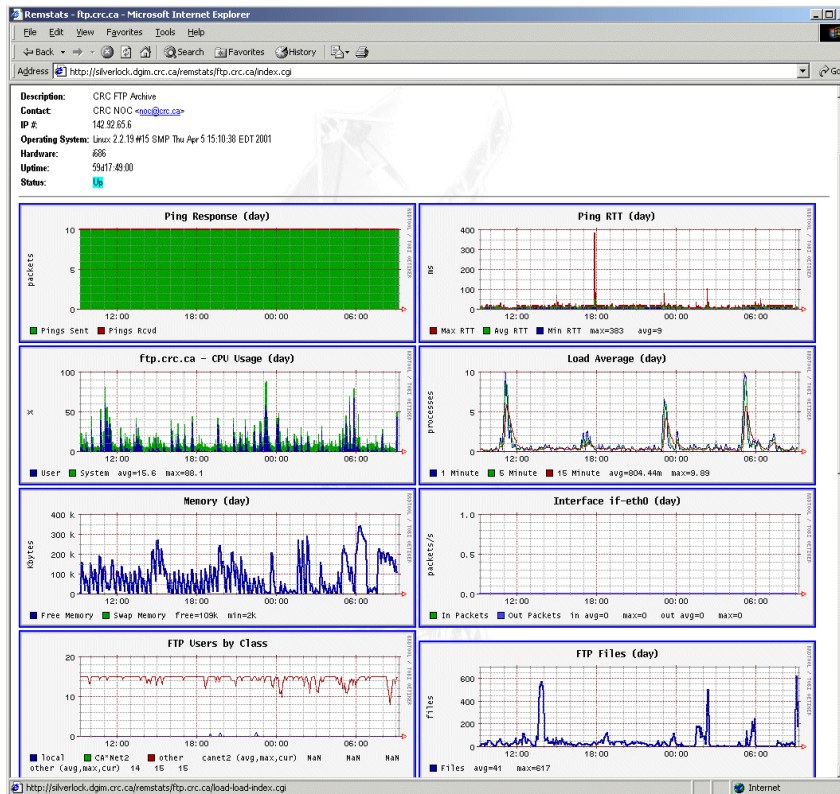


Figure 21-11 Remstats home page with live data

## Installation and configuration

After downloading Remstats and the RRDtool sources, you need to compile, install and configure the programs.

```
# ls
. .. Time-HiRes-01.19.tar.gz remstats-0.13.1.tar.gz rrdtool-1.0.33.tar.gz
```

First you install the rrdtool and the Time::HiRes module, which is shown here in a small example:

```
# tar -xzf Time-HiRes-01.19.tar.gz
# cd Time-HiRes-01.19
# perl Makefile.PL
...
# make
...
# make install
...
```

```

# cd ..
# tar -xzf rrdtool-1.0.33.tar.gz
# cd rrdtool-1.0.33
# vi config/ltconfig
...
set the host variable to host=s390
...
# ./configure --prefix=/usr/local/rrdtool
...
# make
...
# make install

```

Now you can build Remstats itself. First create a user remstats with its own group remstats. Then extract the sources, and compile and install it:

```

# tar -xzf remstats-0.13.1.tar.gz
# cd remstats-0.13.1
# less INSTALL
...
You should read the installation manual for the first install.
...
# ./configure
...
# make all
...
# make install

```

Configure the setting:

```

> cd /var/remstats/etc/config-base/
vmlinux1:/var/remstats/etc/config-base >

```

Edit the file named general (see Figure 21-5) to specify which monitors and collectors you want to run:

*Example 21-5 Remstats configuration file general*

---

```

vmlinux1:/var/remstats/etc/config-base > cat general
# The data for a given host is stored under "DATADIR/$hostname/xxx.rrd"
datadir /var/remstats/data
# where to put the datapage scripts
datapagedir /var/remstats/etc/config/datapages
# How long before we count status as stale
staletime      1800
# How long a machine must be up before it stops being flagged as recently
up
minuptime      86400
# Do we want uptime alerts? Trigger it if uptime is less than this (in
seconds)
uptimealert    3600

```



```

# Where are log-files stored (long-term records of alerts)
logdir /var/remstats/data/LOGS
# Some SNMP traps are boring (most of them). Ignore the ones which
# match this pattern
#trapignore FIXME
# How long to let a collector run before assuming that it's hung.
watchdogtimer 500
# Pre-collector ping pass
# pinger ping-collector
# Collectors to run
collectors ping port snmp-route log unix-status snmp
# Monitors to run
monitors ping alert
# Pagemakers to run
pagemakers graph-writer snmpif-setspeed datapage-interfaces
datapage-inventory view-writer

```

---

Edit the alerts file to insert the e-mail address of the recipient of alerts:

```

vmlinux1:/var/remstats/etc/config-base > cat alerts
# alerts - what to do for various levels of alert

# N.B. Only the first matched alert is triggered, so put the wild-cards
LAST
# for each level.

# level host realrrd var mintime interval prog addresses
# If interval is zero, you get only one alert, no repeats

warn * MISC UPTIME 0 0 /var/remstats/bin/alert-email remstats@localhost
error * * * 600 0 /var/remstats/bin/alert-email remstats@localhost
critical * * * 600 0 /var/remstats/bin/alert-email remstats@localhost

```

Set the variables in the file named html for Web access:

```

htmlurl /remstats/www
# where is the logo for this site (a URL)
logourl /remstats/www/IMAGES/logo.gif
# Where is the "home" for this site
homeurl /remstats/www/quick-index.cgi
# Where should pages go back to for top
topurl /remstats/www/quick-index.cgi
...
# Where is rrdcgi (part of rrdtool, but it could be installed anywhere)
rrdcgi /usr/local/rrdtool/bin/rrdcgi

```

Change back to the /etc directory and create a file with the hosts you want to monitor. Then use the Remstats tools to create a configuration:

```
# cd ..
# su remstats
# vi vmlxhosts
...
vmlinux1.itso.ibm.com
vmlinux2.itso.ibm.com
vmlinux3.itso.ibm.com
vmlinux4.itso.ibm.com
vmlinux5.itso.ibm.com
vmlinux6.itso.ibm.com
...
remstats@vmlinux1:/var/remstats/etc > ../bin/new-config config
remstats@vmlinux1:/var/remstats/etc > ../bin/new-ping-hosts vmlxgroup
vmlxhosts
remstats@vmlinux1:/var/remstats/etc > crontab -e
...
setup a cron job for Remstats
0,5,10,15,20,25,30,35,40,45,50,55 * * * * /var/remstats/bin/run-remstats
...
crontab: installing new crontab
remstats@vmlinux1:/var/remstats/etc > exit
```

Create a directory with a symbolic link in your Web server's document root directory to point to the Remstats html directory:

```
# mkdir /usr/local/httpd/htdocs/remstats
# ln -s /var/remstats/html /usr/local/httpd/htdocs/remstats/www
```

**Note:** Remstats requires that cgi scripts be executed as the remstats user. You can accomplish that by configuring suEXEC or CGIWrapper for Apache, or by simply running a Web server under the remstats user ID (which is the method we chose).

Configure the Web server to run under the user remstats, then password-protect the Remstats page, and allow symbolic links and scripts, as follows:

```
# vi /etc/httpd/httpd.conf
...
User remstats
...
# section for Remstats
<Directory /usr/local/httpd/htdocs/remstats>
    AllowOverride AuthConfig
    Options FollowSymLinks ExecCGI
    AuthName "Remstats"
    AuthType Basic
```

```
AuthUserFile /etc/httpd/passwd
require user remstats
</Directory>
```

Add a entry for the remstats user in the Web server password file:

```
# htpasswd /etc/httpd/passwd remstats
New password:
Re-type new password:
Adding password for user remstats
```

Restart the Web server:

```
# /sbin/init.d/apache restart
Shutting down service httpd           done
Starting service httpd                 done
```

To gather system information like load, cpu and memory usage, you have to install additional servers (daemons) on these systems; for example, you can use the snmp daemon and the corresponding Remstats collectors. Also, the **unix-status-server** that comes with Remstats can be installed on the remote systems to gather information.

Create a remstats user on the remote system where you want to install the **unix-status-server**. Then you have to add a port entry in /etc/services:

```
unix-status 1957/tcp # remstats unix-status server
```

Next, add a entry (as a single line!) in /etc/inetd.conf to get it invoked by **inetd**:

```
unix-status stream tcp nowait remstats /var/remstats/bin/unix-status-server
unix-status-server
```

On Red Hat systems where you use **xinetd**, create a new file called unix-status in /etc/xinetd.d/:

```
# cat /etc/xinetd.d/unix-status
service unix-status
{
    disable = no
    socket_type = stream
    wait = no
    protocol = tcp
    user = remstats
    server = /var/remstats/bin/unix-status-server
}
```

After that, you need to have **inetd/xinetd** to reread the configuration:

```
kill -HUP <pid-of-inetd/xinetd>
```

At this point, the **unix-status-server** should be running and listening on port 1957. You can verify that by checking with the **netstat -a** command.

Add entries to the configuration on the Remstats server for the remote hosts to collect cpu and load data:

```
# cd /var/remstats/etc/config/hosts/
# cat vmlinux1.itso.ibm.com
# hosts/vmlinux1.itso.ibm.com
desc    vmlxgroup host
group   vmlxgroup
ip      9.12.6.98
tools   ping traceroute
rrd     ping
rrd     cpu
rrd     load
rrd     port-ssh
rrd     port-http
community public
```

You can now logon to the Remstats page with your browser and provide user ID and password to view the statistical data.

In our case, we were unable to get everything working correctly on the SuSE where we had to compile the **rrdtool** ourselves, because of problems with CGI scripts generated by **rrdcgi** (requiring more time for investigation than we had at this point).

We later tried Remstats on the ThinkBlue 64-bit distribution (where the **rrdtool** is included as a binary rpm package), and there the CGI scripts worked. However, the collectors worked and the data was correctly stored in RRD databases. Figure 21-12 on page 431 shows the quick index view:

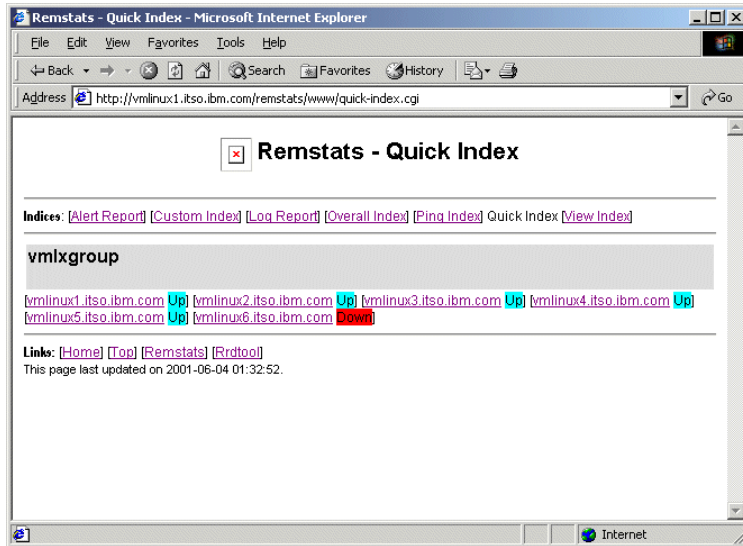


Figure 21-12 Remstats quick index

This view was generated on the Think Blue system, where the CGI scripts worked correctly. Figure 21-13 on page 432 shows data for ping time, load, CPU, users and the response time of the http and ssh port.

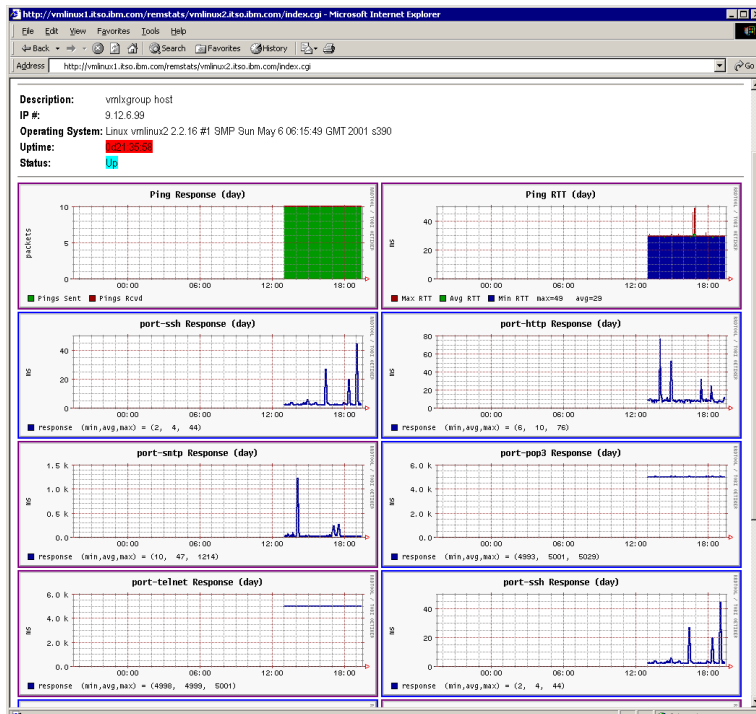


Figure 21-13 Remstats data page

## 21.3.5 Other remote network management tools

### BMC Patrol

BMC's Patrol is an enterprise UNIX systems management product with many features. Patrol is now also available for Linux for S/390. For more information, visit:

<http://www.bmc.com/ega/preview/8.3.01A>

### NetSaint

NetSaint is a powerful network monitoring application with a Web-based interface. It provides many different functions:

- Monitoring of network services (SMTP, POP3, HTTP, NNTP, ping, etc.)
- Monitoring of host resources (processor load, disk usage, etc.)
- Simple plug-in interface to allow for user-developed service checks
- Contact notifications when problems occur and get resolved (via e-mail, pager, or user-defined method)

- Ability to define event handlers to be run during service or host events for proactive problem resolution, logging events to a database, etc.
- Web interface for viewing current status, log file, notifications, event history, and network layout
- Parallelized service checks
- Host dependencies
- Automatic log rotation
- Helpful documentation

Unfortunately, because of time constraints, we were unable to install NetSaint during this project. If you want to try NetSaint, you have to download, compile and install it.

NetSaint is available on the Web at:

<http://www.netsaint.org>

## Big Sister

Big Sister is a Web-based system monitoring program, very similar and even compatible to Big Brother (see 21.3.3, “Big Brother” on page 414). Big Sister is open source software and licensed under the GPL. To get more information on how to get and use it, visit the project home page:

<http://bigsisiter.graeff.com>

## Mon

Mon is a multipurpose Perl-based system monitoring tool. It provides a CG-based Web interface that allows you to configure different views for system status and supervise system services. To learn more about how to get and configure Mon, visit the project home page:

<http://www.kernel.org/pub/software/admin/mon/html>

See 18.5.3, “The mon tool and test” on page 333 for a more extensive description of this tool.

## MRTG

The Multi Router Traffic Grapher (MRTG) is a network interface monitor. It is based on SNMP and RRDtool, and provides a Web-based interface with graphs on the network interface usage of the monitored systems. Get more information from the project home page:

<http://people.ee.ethz.ch/~oetiker/webtools/mrtg>

## **OpenNMS**

OpenNMS is an open source project which implements an enterprise SNMP-based system management tool. It is written completely in Java, and provides lots of features that are comparable to commercial products.

<http://www.opennms.org>





## Overview of security on Linux

Although an entire book could easily be devoted to the subject of security, in this chapter we provide only an overview of the basics of security on Linux.

And while you can obtain very powerful tools to increase your Linux system's security, it's important to keep in mind that you will not automatically arrive at a security solution that is 100% effective, regardless of the operating system and the software. Why? Because effective security is not achieved simply through the use of tools. Rather, it has to be carefully planned in advance by means of a thoughtful *security policy*. In this chapter we discuss tools and hints you can utilize to increase the security level of your system after you install Linux for zSeries and S/390.

## 22.1 Security basics

The security basics can be summarized as follows:

- ▶ Disable unneeded services
- ▶ Use Secure Shell for remote access
- ▶ Use shadow password utilities
- ▶ Use the Pluggable Authentication Module (PAM)
- ▶ Monitor security news and alerts
- ▶ Use hardening tools
- ▶ Integrate z/VM security

In the following sections, we describe these points in more detail.

### 22.1.1 Disable unneeded services

Many network requests are handled by either the Internet Daemon (inetd), or the Extended Internet Daemon (xinetd). The Red Hat system uses xinetd as its default Internet daemon. All other distributions use inetd.

The inetd daemon listens on several ports on the network and starts the program which handles the connection. It has its own configuration file, where all services served by inetd are listed. However, some services are not handled by inetd; instead, they run their own daemon and listen on their agreed-upon port (an example is a Web server).

We recommend that you check all services running on the system *before* you connect Linux to a insecure network. All services that are not used should be disabled.

The services activated by default will differ, depending on which distribution is used. Table 22-1 lists which services are activated, by default, by each distribution.

*Table 22-1 Active/Inactive services after default installation*

Service	SuSE	Turbo	Red Hat	Caiman	Served by
telnet	active	- *	-	active	inetd/xinetd
ftp	active	- *	-	active	inetd/xinetd
smtp	active *	-	active	-	daemon **
time	active	-	-	-	inetd/xinetd
finger	active	-	-	-	inetd/xinetd
http	active	- *	-	-	daemon (**)

Service	SuSE	Turbo	Red Hat	Caiman	Served by
pop-3	active	-	-	-	inetd/xinetd
login	active	-	-	-	inetd/xinetd
shell	active	-	-	-	inetd/xinetd
printer	active	-	-	-	daemon **
nfs	active *	-	-	-	daemon **
ssh	-	active	active	-	daemon **

**Notes:**

\* Activation setting asked during installation  
\*\* Runs as a daemon, starts/stops via a separate script

## Inetd

Services can be easily deactivated by editing the file `/etc/inetd.conf`. To disable services in `inetd`, place a hash character (`#`) before each service you want to disable.

If you want to disable telnet, for example, change this line:

```
telnet stream tcp    nowait root    /usr/sbin/tcpd in.telnetd
```

to this:

```
#telnet stream tcp    nowait root    /usr/sbin/tcpd in.telnetd
```

After making any changes to the file `/etc/inetd.conf`, you then have to force `inetd` to reload this information. The most basic way to accomplish this is to determine the process ID (pid) of `inetd` via the `ps` command, and send it the hangup (HUP) signal via the `kill` command, as shown:

```
# ps aux | grep inetd
root      233  0.0  0.0  1188  516 ?        S    Jun05   0:00 /usr/sbin/inetd
root      8861 0.0  0.1  1364  528 pts/3   S    15:39   0:00 grep inetd
# kill -HUP 233
```

Several scripts in the distributions will do this for you, as shown in Table 22-2:

Table 22-2 Reloading `inetd` configuration file after any changes

Distribution	Command
SuSE	<code>rcinetd reload</code>
Turbo	<code>/etc/rc.d/init.d/inetd reload</code>

Distribution	Command
Caiman	/etc/init.d/inetd reload

## xinetd

You can use a similar procedure to manipulate **xinetd**. To disable a service in `/etc/xinetd.conf`, add a line with the `disabled=<service>` value for disabling the respective service.

However, on Red Hat-based systems, besides the file `/etc/xinetd.conf`, there is a directory named `/etc/xinetd.d/`. Under that directory, there is a configuration file for each service. In the appropriate file (`/etc/xinetd.d/<service>`), you have to add a line such as:

```
disable=yes
```

Example 22-1 provides an example of the `xinetd.conf` configuration files :

*Example 22-1 xinetd.conf - Extended Internet daemon*

---

```
#
# xinetd.conf
#
# Copyright (c) 1998-99 SuSE GmbH Nuernberg, Germany.
#

defaults
{
    log_type          = FILE /var/log/xinetd.log
        log_on_success  = HOST EXIT DURATION
    log_on_failure   = HOST ATTEMPT RECORD
#    only_from         = localhost
        instances      = 2

#
# The specification of an interface is interesting, if we are on a firewall.
# For example, if you only want to provide services from an internal
# network interface, you may specify your internal interfaces IP-Address.
#
# interface = 127.0.0.1
#

#
# If you want to enable one of the following services, you only have to
# comment it out. After that, send SIGUSR1 to xinetd to force a
# reload of it's configuration
#

# disabled= ftp
```

```

        disabled      = rstatd
# disabled= telnet
disabled= shell
# disabled= login
disabled= finger
disabled= pop3
disabled= comsat
disabled= ntalk
disabled= talk
disabled= discard
disabled= chargen
disabled= daytime
disabled= time
disabled= echo
disabled= daytime
disabled= time
disabled= smtp
disabled= ident
}

##
## Now the definitions of the different services
##
##

service telnet
{
    socket_type      = stream
    protocol         = tcp
    wait             = no
    user             = root
    server           = /usr/sbin/in.telnetd
    server_args= -n
# only_from= localhost
no_access=
}
...

```

---

Send **xinetd** the hangup signal after any changes in the file `/etc/xinetd.conf`:

```

# ps aux | grep inetd
root      233  0.0  0.0  1188  516 ?        S    Jun05   0:00 /usr/sbin/xinetd
root      8861 0.0  0.1  1364  528 pts/3    S    15:39   0:00 grep xinetd
# kill -SIGUSR1 233

```

## ***The insecurity of Telnet***

Most TCP/IP network traffic is sent without any encryption. Every computer on a local area network (LAN) segment is able to read all TCP/IP packets coming through. Sensitive information like passwords can be easily *sniffed* and stored by a program.

The common way to connect to a remote machine is via telnet. Although the typed-in password is hidden on the screen during the login process, it is still transferred over the network in clear text. Example 22-2 shows the very insecure password of *password* being sniffed (some characters are doubled because of console echo):

*Example 22-2 A sniffed telnet session*

---

```
ÿÿÿÿÿÿÿÿ ÿÿ!ÿÿ"ÿÿ'ÿÿÿÿ#ÿÿÿÿ ÿÿ#ÿÿ'ÿÿÿÿÿÿ!ÿÿ"ÿÿÿÿ ÿÿÿÿ#ÿÿÿÿ'ÿÿÿÿÿÿÿÿ P ÿÿÿÿ
38400,38400ÿÿÿÿ# linux.local:0ÿÿÿÿ' PRINTERlp DISPLAYlinux.local:0ÿÿÿÿ
KVTÿÿÿÿÿÿÿÿÿWelcome to SuSE Linux 7.0 (s390) - Kernel 2.2.16 (0).
ÿÿ
vmlinux2 login: rroooott
```

**Password: password**

```
You have new mail in /var/spool/mail/root.
Last login: Mon May 21 14:37:58 from 9.12.2.171
Have a lot of fun...
# llaa
```

```
[0mtotal 28
drwx--x--x  3 root    root      4096 May 21 14:36 [01;34m.[0m
drwxr-xr-x  19 root    root      4096 May 18 16:01 [01;34m..[0m
-rw-----  1 root    root        476 May 21 15:59 [0m.bash_history[0m
-rw-r--r--  1 root    root      1124 Jun 20  2000 [0m.exrc[0m
-rw-r--r--  1 root    root      1301 Jun 20  2000 [0m.xinitrc[0m
drwxr-xr-x  2 root    root      4096 May 18 16:01 [01;34mbin[0m
# eexxiitt
logout
```

---

To avoid sending passwords over the network in clear text, several encryption methods can be used (TCP/IP version 6 (IPv6) is able to do encryption at the network layer, but this goes beyond the scope of this chapter).

## **22.1.2 Use Secure Shell for remote access**

With Secure Shell (SSH), you can connect as telnet does, but the connection is encrypted. An SSH daemon must be running on the target host in order to be able to establish the connection.

SSH protects you from:

- ▶ IP spoofing
- ▶ DNS spoofing
- ▶ Interception of cleartext passwords

Authentication can be done in several ways:

- ▶ By password
- ▶ By key authentication
- ▶ Via host authentication
- ▶ Via Kerberos

## Password

Authentication by password is the common way to login via SSH.

## Key authentication

Using key authentication, you can connect to a remote host without using a password. But using no password is a security risk, because if someone obtains the private key, he will have access to the remote machine.

To avoid this scenario, you can protect your key with a *passphrase*. Users will then need both the key and the passphrase to connect to the remote machine.

## Generating and using SSH keys

Before you can use the authentication method with keys, you have to generate the keys. The command **ssh-keygen** can be used by users to generate their SSH keys, or it can be used by the administrator to generate keys for the host.

```
$ ssh-keygen
Generating RSA keys: Key generation complete.
Enter file in which to save the key (/home/tot12/.ssh/identity):
Created directory '/home/tot12/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/tot12/.ssh/identity.
Your public key has been saved in /home/tot12/.ssh/identity.pub.
The key fingerprint is:
a1:44:ec:33:7e:c5:89:aa:e8:e1:1a:dd:8e:e0:4a:7d tot12@vmlinux2
$
```

The key generation routine generates two files in the `.ssh/` directory of the user:

**identity**                    The private key

**identity.pub**                The public key

You should ensure that the file `identity` is read/writeable only by the user.

The directory structure would then look as follows:

```
$ ls -la .ssh/
drw----- 2 tot12 users 4096 Mai 21 09:14 .
drw----- 30 tot12 users 16384 Mai 29 13:50 ..
-rw----- 1 tot12 users 524 Mai 21 09:14 identity
-rw-r--r-- 1 tot12 users 328 Mai 21 09:14 identity.pub
```

To connect to another host without the usual password authentication, the file `identity.pub` has to be attached to the file `.ssh/authorized_keys` in the home directory of the remote user.

The directory structure of the remote `.ssh/` directory will be:

```
$ ls -la .ssh/
drw----- 2 user13 users 4096 Mai 21 09:14 .
drw----- 30 user13 users 16384 Mai 21 09:50 ..
-rw-r--r-- 1 user13 users 328 Mai 22 10:00 authorized_keys
-rw----- 1 user13 users 524 Mai 15 19:04 identity
-rw-r--r-- 1 user13 users 328 Mai 15 19:04 identity.pub
```

You can attach your own public key file. The following command will attach a public key file to your own `authorized_keys` file:

```
$ cat identity.pub >> .ssh/authorized_keys
```

**Attention:** Give only your *public* key to other users!

The first time you establish a connection to a remote host, you will be prompted to accept the fingerprint of the host key of the remote machine. This ensures that the remote host identifies itself to you, so you can be sure that the host isn't replaced by someone (known as a "man in the middle attack").

The remote host fingerprint is stored in the file `.ssh/known_hosts`. If the host changes because a new system install was performed, you have to delete the line which includes the fingerprint of the host. Otherwise, `ssh` will refuse to connect to the host, as shown:

```
$ ssh vmlinux1
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@   WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!   @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that the RSA host key has just been changed.
Please contact your system administrator.
Add correct host key in /home/tot12/.ssh/known_hosts to get rid of this
message.
```



RSA host key for vmlinux1 has changed and you have requested strict checking.

The first time you connect to a remote host, the host's fingerprint will be saved for future connections and verification:

```
$ ssh vmlinux1
The authenticity of host 'vmlinux1' can't be established.
RSA key fingerprint is 7e:53:7d:54:99:47:34:47:8d:8d:85:23:0a:f5:d2:f1.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'vmlinux1,9.12.6.98' (RSA) to the list of known
hosts.
tot12@vmlinux1's password:
Have a lot of fun...
```

You can also connect to a different user on the remote host by using the `-l` parameter:

```
$ ssh -l poke vmlinux2
poke@vmlinux2's password:
Last login: Thu Jun  7 17:08:58 2001 from vmlinux1.itso.ibm.com
Have a lot of fun...
```

Example 22-3 shows the many options that `ssh` can take.

*Example 22-3 ssh options*

---

```
# ssh -h
Usage: ssh [options] host [command]
Options:
  -l user      Log in using this user name.
  -n          Redirect input from /dev/null.
  -A          Enable authentication agent forwarding.
  -a          Disable authentication agent forwarding.
  -X          Enable X11 connection forwarding.
  -x          Disable X11 connection forwarding.
  -i file     Identity for RSA authentication (default: ~/.ssh/identity).
  -t          Tty; allocate a tty even if command is given.
  -T          Do not allocate a tty.
  -v          Verbose; display verbose debugging messages.
              Multiple -v increases verbosity.
  -V          Display version number only.
  -P          Don't allocate a privileged port.
  -q          Quiet; don't display any warning messages.
  -f          Fork into background after authentication.
  -e char     Set escape character; ``none'' = disable (default: ~).
  -c cipher   Select encryption algorithm: ``3des'', ``blowfish''
  -p port     Connect to this port. Server must be on the same port.
  -L listen-port:host:port Forward local port to remote address
  -R listen-port:host:port Forward remote port to local address
              These cause ssh to listen for connections on a port, and
```

```
forward them to the other side by connecting to host:port.
-C      Enable compression.
-N      Do not execute a shell or command.
-g      Allow remote hosts to connect to forwarded ports.
-4      Use IPv4 only.
-6      Use IPv6 only.
-2      Force protocol version 2.
-o 'option' Process the option as if it was read from a configuration file.
```

---

## Use secure copy for copying files

It is common to use FTP to transfer files over the network. But with FTP, as with telnet, the password is transferred in clear text and can be sniffed by an attacker. Therefore we recommend that rather than using FTP, you use the secure copy command **scp** for file transfers, because the connection will be encrypted and you can take advantage of the other authentication methods.

The following example shows how to copy the file `openssh.rpm` from `vmlinux2` to `vmlinux1`:

```
$ scp openssh.rpm vmlinux1:
tot12@vmlinux1's password:
openssh.rpm          100% |*****| 1264 KB    00:01
```

This is an example of copying in the reverse direction, from the remote host `vmlinux1` to the local machine:

```
$ scp vmlinux1:/tmp/openssh.rpm ./
tot12@vmlinux1's password:
openssh.rpm          100% |*****| 1264 KB    00:01
```

To copy a whole directory recursively, use the **-r** flag:

```
$ scp -r * vmlinux1:/tmp/
tot12@vmlinux1's password:
openssh.rpm          100% |*****| 1264 KB    00:01
archiv.tar.gz        100% |*****| 3874 KB    00:05
...
```

The connection is closed after the transfer has finished. Transfer with **scp** is somewhat slower than normal FTP, because of the encryption.

## Secure login (slogin)

To execute a command or program on a remote machine, use the `slogin` command. The syntax is:

```
slogin <host> <command>
```

Following is an example of running the command `uptime` on the remote host `vmlinux1`:

```
$ slogin vmlinux1 uptime
tot12@vmlinux1's password:
 8:28pm up 1 day, 23 min,  2 users,  load average: 0.00, 0.07, 0.10
```

## SSH daemon configuration

Now let's look at the major settings of the `sshd` daemon. For proper operation of the ssh connection, several settings can be changed in the ssh configuration file, as shown in Example 22-4.

*Example 22-4 /etc/ssh/sshd\_config - ssh daemon configuration file*

---

```
# This is ssh server systemwide configuration file.

Port 22
#Protocol 2,1
ListenAddress 0.0.0.0
#ListenAddress ::
HostKey /etc/ssh/ssh_host_key
ServerKeyBits 768
LoginGraceTime 600
KeyRegenerationInterval 3600

This enables root login if set to yes: (should be no)
PermitRootLogin no
#
# Don't read ~/.rhosts and ~/.shosts files
IgnoreRhosts yes
# Uncomment if you don't trust ~/.ssh/known_hosts for RhostsRSAAuthentication
#IgnoreUserKnownHosts yes
StrictModes yes
This enables you to run X-sessions over the encrypted connection
X11Forwarding yes
X11DisplayOffset 10
PrintMotd yes
KeepAlive yes

# Logging
SyslogFacility AUTH
LogLevel INFO
#obsoletes QuietMode and FascistLogging
```

**This enables remote host authentication: (should be no)**

```
RhostsAuthentication no
#
# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
RhostsRSAAuthentication no
#
```

**Enables the key authentication when set to yes:**

```
RSAAuthentication yes
```

```
# To disable tunneled clear text passwords, change to no here!
```

**Enables password authentication if set to yes (should be no if you only want key authentication, but turn RSAAuthentication!!):**

```
PasswordAuthentication yes
```

**Allows empty passwords if set to yes: (should be no)**

```
PermitEmptyPasswords no
```

```
# Uncomment to disable s/key passwords
```

```
#SkeyAuthentication no
```

```
#KbdInteractiveAuthentication yes
```

```
# To change Kerberos options
```

```
#KerberosAuthentication no
```

```
#KerberosOrLocalPasswd yes
```

```
#AFSTokenPassing no
```

```
#KerberosTicketCleanup no
```

```
# Kerberos TGT Passing does only work with the AFS kaserver
```

```
#KerberosTgtPassing yes
```

```
CheckMail no
```

```
#UseLogin no
```

---

**Note:** We recommend you turn off root login.

## Differences between SSH protocol version 1 and version 2

There are two protocol versions of SSH, which are SSH1 and SSH2.

SSH1 has the following attributes:

- ▶ It uses the RSA algorithm, which is patented until Sept 2000.
- ▶ It is well-tested, and free.

SSH2 has the following attributes

- ▶ It has been totally rewritten.
- ▶ It has improved privacy.
- ▶ It has a more restrictive license than SSH1; it is free for non-commercial use only.

### SSH client/server sources

There are quite a few sources of SSH. All of the distributions include OpenSSH, which is on the Web at:

<http://www.openssh.org>

If you are using a Windows client, there are also quite a few SSH clients:

TeraTerm Pro is on the Web at:

<http://hp.vector.co.jp/authors/VA002416/teraterm.html>

Putty is on the Web at:

<http://www.chiark.greenend.org.uk/~sgtatham/putty>

SSH-win32 is on the Web at:

<http://guardian.htu.tuwien.ac.at/therapy/ssh>

## 22.1.3 Use shadow password utilities

Passwords and information about users are usually stored in the file `/etc/passwd`. Although the password is stored in an encrypted format, with some tools it's possible to *crack* the password if you have read access to this file.

The shadow password utility stores the encrypted password in the file `/etc/shadow`, separated from `/etc/passwd`, because some applications have to read the user/group id of users, so `/etc/passwd` must be readable to all.

The access privileges to `/etc/passwd` are:

```
$ ls -l /etc/passwd
-rw-r--r-- 1 root root 3534 Jun 6 09:52 /etc/passwd
```

The access privileges to `/etc/shadow` do not allow world read, and only allow group read to members of the shadow group:

```
$ ls -l /etc/shadow
-rw-r----- 1 root shadow 2083 Jun 7 10:02 /etc/shadow
```

Therefore, a normal user cannot read the shadow password file, as shown:

```
$ head -1 /etc/passwd
root:x:0:0:root:/root:/bin/bash
$ head -1 /etc/shadow
head: /etc/shadow: Permission denied
```

To avoid this, you should use shadow password utilities instead of normal password files. With this utility, the file `/etc/passwd` does not contain the encrypted password from the user, but it does contain all other necessary information, such as the user's user/group id, shell, and so on. The encrypted passwords are stored in the file `/etc/shadow`, which is only readable to the user root. The `/etc/passwd` file has an entry with the following format:

```
username:password:UID:GID:name:home_directory:shell
```

Table 22-3 Description of `/etc/passwd`

Attribute	Login
<b>username</b>	The character login name of the user
<b>Password</b>	Password ('x', if shadow password is used)
<b>UID</b>	The integer user ID
<b>GID</b>	The integer group ID
<b>Home directory</b>	Full path to home directory
<b>Default shell</b>	The executable shell

Now we log in as root and observe the root entry in the shadow password file, `/etc/shadow`. The letter x in the second field of the `/etc/passwd` file informs the system that the root password information is stored in the shadow password file:

```
# head -1 /etc/passwd
root:x:0:0:root:/root:/bin/bash
# head -1 /etc/shadow
root:U2TpkwoRQ7d1o:11446:0:10000:::
```

The `/etc/shadow` file has many more attributes than `/etc/passwd`:

```
username:password:last:may:must:warn:expire:disable:reserved
```

Table 22-4 lists and describes these attributes:

Table 22-4 Description of /etc/shadow

<b>username</b>	User name (login)
<b>password</b>	Encoded password
<b>last</b>	Days since Jan 1, 1970 password was last changed
<b>may</b>	Days before password may be changed
<b>must</b>	Days after which password must be changed
<b>warn</b>	Days before password is to expire that user is warned
<b>expire</b>	Days after password expires that account is disabled
<b>disable</b>	Days since Jan 1, 1970 that account is disabled
<b>reserved</b>	Reserved for future use

The first two characters of the encrypted password are the *SALT* value. Following are shadow utility commands:

<b>useradd</b>	Adds a user to the system; can only be run by root.
<b>userdel</b>	Deletes a user from system; can only be run by root.
<b>usermod</b>	Shows/modifies user related data like name, UID and so on; can only be run by root.
<b>passwd</b>	Changes the password of a user; if run by root, it can change the password of any user.
<b>chage</b>	Changes the user password expiry information; can only be run by root.

Following are examples of tasks that can be accomplished via the shadow password utilities.

- ▶ To set the validity of a user ID for a short period of time, add the **-e** flag to the **useradd** command:

```
# useradd -m -e 2001-06-15 user23
```

The account will become invalid on June 15, 2001.

- ▶ To force the user to change his password regularly, add the **-M** flag to the **chage** command:

```
# chage -M 7 user23
```

The user will have to change his password every 7 days (of course, he will only be able to change it once, because after that it will be expired).

- ▶ Use a password generator to avoid weak passwords:

```
# head --bytes=16 /dev/urandom | md5sum
230ca11cfe3acd5cc6645121e3db52c5 -
```

Note, however, that overly complex passwords are often written down, which adds a new security exposure.

- ▶ Force the user to change his password on the first login by adding the `-m` flag to the `useradd` command:

```
# useradd -m user23
# passwd user23
...
# chage -m 0 -M 99999 -d 0 user23
```

In the preceding example, we added `user23` to the system and changed set the password. We then changed the age at which he has to change the password on his first login. The initial password can be used for the first login, but then it must be reset; see the `change` man page for details.

When the user logs into the system, he will be prompted to use a new password, as follows:

```
$ ssh vmlinux2 -l user23
The authenticity of host 'vmlinux2 (9.12.6.99)' can't be established.
RSA1 key fingerprint is 52:ad:c2:36:4c:08:78:0b:34:a7:54:ae:64:9f:a8:69.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'vmlinux2,9.12.6.99' to list of known hosts.
user23@vmlinux2's password: <-- assigned password
Password changing requested. Choose a new password.
Warning: Password has expired, please change it now <- user must change
...
```

## 22.1.4 Use the Pluggable Authentication Module (PAM)

The traditional way to authenticate users is by password; see Figure 22-1. Many programs write their own authentication code. Sometimes the code is well-written, and sometimes it is not. But whenever authentication code exists in many programs, it is duplicated. If you want to use another authentication method, you have to recompile *each* program.

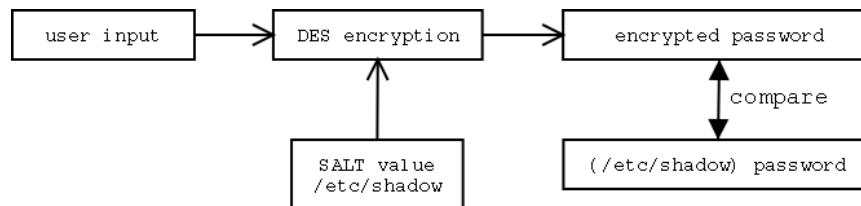


Figure 22-1 Traditional password authentication



In contrast, however, if the program is compiled with the Pluggable Authentication Module (PAM) support, you can switch between several authentication methods without having to recompile each program. This uses the power of Linux's share libraries.

The advantages of using PAM lie in its *transparent authentication* method. In a heterogeneous network, for example, the system administrator wants to authenticate the users against a NT box, or a NIS service. If there is a PAM module for such authentication, the administrator can perform the authentication easily, without having to maintain several different databases where users are listed for different systems—one authentication method can be used for different systems. And the authentication can be done with a magnetic card, retina scan, or whatever.

Figure 22-2 illustrates a PAM authentication block; PAM is installed by default on all distributions.

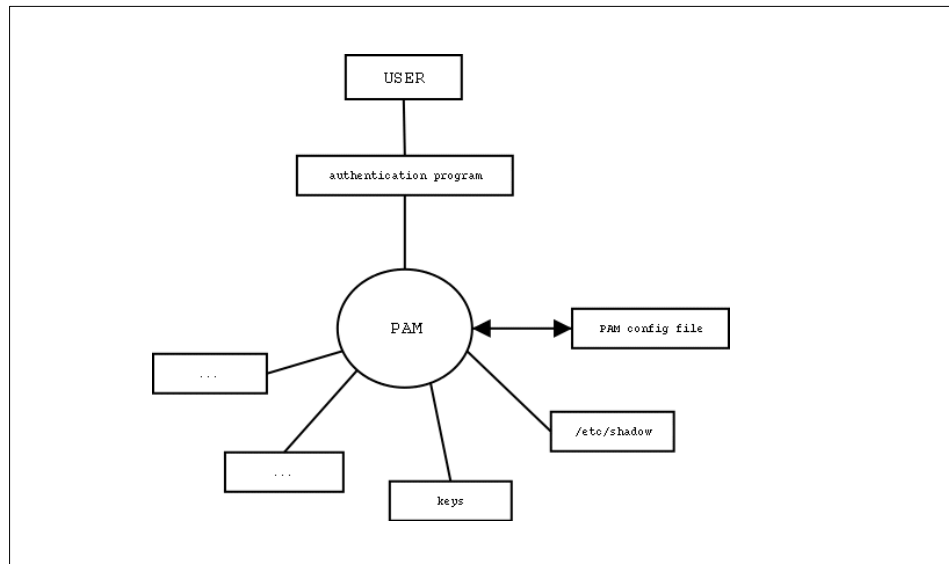


Figure 22-2 PAM authentication block diagram

## Configuring PAM

The services are listed in the directory `/etc/pam.d/`. Each service has its own configuration file, which means that every service listed in the directory can have different security policies; see Example 22-5 on page 452.

Example 22-5 /etc/pam.d/login

---

```
#%PAM-1.0
#type control module-path module-arguments
auth requisite /lib/security/pam_unix.so nullok #set_secrcp
#auth required /lib/security/pam_securetty.so
auth required /lib/security/pam_nologin.so
#auth required /lib/security/pam_homecheck.so
auth required /lib/security/pam_env.so
auth required /lib/security/pam_mail.so
account required /lib/security/pam_unix.so
password required /lib/security/pam_pwcheck.so nullok
password required /lib/security/pam_unix.so nullok use_first_pass use_authok
session required /lib/security/pam_unix.so none # debug or trace
session required /lib/security/pam_limits.so
```

---

The configuration file contains four columns, which are type, control, module-path, and module-arguments:

<b>type</b>	Tells PAM what type of authentication is used for the module; can be one of the following values:
<b>auth</b>	Checks user ID and password.
<b>account</b>	Determines if the user has access to the service.
<b>password</b>	What to do if the user changes his authentication (password).
<b>session</b>	What to do after/before successful authentication.
<b>control</b>	Tells PAM what to do if the authentication by the module fails; can be one of the following values:
<b>requisite</b>	If authentication with this module fails, the authentication will be denied.
<b>required</b>	If authentication fails, PAM still calls the other modules before denying authentication.
<b>sufficient</b>	If authentication is successful, PAM grants authentication even if a previous module failed.
<b>optional</b>	Whether the module succeeds or fails is only significant if it is the only module of its type for this service.
<b>module-path</b>	The path to the PAM module.
<b>module-arguments</b>	The arguments for the PAM module.

Following are examples using the PAM-module pam\_unix. This module can be used in the following management groups (with examples):

► **auth**

```
auth requisite /lib/security/pam_unix.sonullok
```

The module gets the user ID and password. You should disable the NULL password by removing the nullok argument.

► **account**

```
account required /lib/security/pam_unix.so
```

The module gets information such as password expire, last\_change, and so on from /etc/shadow .

► **password**

```
password required /lib/security/pam_unix.sonullok use_first_pass
```

This is used for updating/changing passwords. You should disable the nullok argument so it will not allow NULL passwords.

► **session**

```
session required /lib/security/pam_unix.so
```

This logs the username to the syslog.

Table 22-5 lists which module comes with each distribution, and also provides a brief description of each module.

*Table 22-5 Available modules - /lib/security/*

Module	Description	SuSE	Turbo	Red Hat	Caiman	Think Blue
pam_access	Provides logdaemon-style login access control	yes	yes	yes	yes	yes
pam_chroot	Puts user into chroot environment	-	-	yes	-	yes
pam_console	Gives users access at the physical console	-	yes	yes	-	yes
pam_cracklib	Checks strength for passwords	yes	yes	yes	-	yes
pam_deny	Denies everything	yes	yes	yes	yes	yes
pam_env	Allows you to set/unset environment variables	yes	yes	yes	yes	yes

Module	Description	SuSE	Turbo	Red Hat	Caiman	Think Blue
pam_filter	Applies filter rules to input/output stream	yes	yes	yes	yes	yes
pam_ftp	Provides anonymous ftp	yes	yes	yes	yes	yes
pam_group	Provides group settings	yes	yes	yes	yes	yes
pam_issue	Displays the issue file /etc/issue	yes	yes	yes	yes	yes
pam_krb5	Performs Kerberos verification	-	-	yes	-	yes
pam_krb5afs	Kerberos	-	-	yes	-	yes
pam_lastlog	Writes last login into /var/log/lastlog	yes	yes	yes	yes	yes
pam_ldap	Authentication with LDAP	*	-	yes	-	yes
pam_limits	Sets some system limits to the user	yes	yes	yes	yes	yes
pam_listfile	Deny or allow services based on an arbitrary file	yes	yes	yes	yes	yes
pam_localuser	User must be listed in /etc/passwd	-	-	yes	-	yes
pam_mail	Indicates if user has mail	yes	yes	yes	yes	yes
pam_mkhome	Creates home directory on the fly	-	yes	yes	yes	yes
pam_motd	Displays the motd file /etc/motd	-	yes	yes	-	yes
pam_nologin	Prevent from login if file /etc/nologin exists	yes	yes	yes	yes	yes
pam_permit	Permits without confirmation	yes	yes	yes	yes	yes
pam_pwddb	Give status of user account	yes	yes	yes	-	yes
pam_rhosts_auth	Authenticate via rhosts	yes	yes	yes	yes	yes

Module	Description	SuSE	Turbo	Red Hat	Caiman	Think Blue
pam_rootok	Rootaccess to a service without a password	yes	yes	yes	yes	yes
pam_securetty	Checks in file /etc/securetty if the TTY is valid for login	yes	yes	yes	yes	yes
pam_shells	Checks in file /etc/shells if shell is valid	yes	yes	yes	yes	yes
pam_stack	Propagates login and password to underlying module	-	-	yes	-	yes
pam_stress	Stress test applications	yes	yes	yes	yes	yes
pam_tally	Counts login attempts and can deny access	yes	yes	yes	yes	yes
pam_time	Gives users restricted permissions depending on time and date to login	yes	yes	yes	yes	yes
pam_userdb	Checks username/password against Berkeley DB	yes	yes	yes	yes	yes
pam_unix	Checks passwords and more	yes	yes	yes	yes	yes
pam_warn	Logs information about a proposed authentication to update a password	yes	yes	yes	yes	yes
pam_wheel	Root access if user is member of the group wheel	yes	yes	yes	yes	yes
pam_xauth	Forwards xauth keys	-	yes	yes	-	yes
pam_homecheck	Checks if homedirectory of user exists	yes	-	-	yes	-
pam_radius	Authenticate via radius	yes	-	-	-	-

Module	Description	SuSE	Turbo	Red Hat	Caiman	Think Blue
pam_pwcheck	Additional checks upon password changes	yes	-	-	-	-
pam_smb_auth	Authenticate against NT Server	*				
* Not in default installation - extra package						

The file `/etc/pam.d/others` is for all unknown services that are not handled by a pam configuration file. You can edit the file to make the default policy stronger, but be careful with the module `pam_deny`, because it denies every attempt to login, even the user ID and the password suits.

For further information about the modules and their arguments, consult the PAM HOWTO file.

## 22.1.5 Monitor security news and alerts

Software on your system might have bugs or vulnerabilities that are not known at a certain point in time. When such vulnerabilities are discovered, it is important that they be fixed as soon as possible, because they can quickly be exploited by crackers.

Many Web sites exist where you can check for the latest news on vulnerabilities in software you plan to use, or are currently using; following are useful security-related sites:

<http://securitytracker.com>  
<http://www.securityportal.com>  
<http://www.securitysearch.net>  
<http://www.securityfocus.com>  
<http://www.lwn.net>

### Monitor your own log files

It's important for you to know where your log files are being written, so you can monitor them. For example, nearly all applications write to log files in the directory `/var/log/`. By default, warning messages are sent to the file `/var/log/warn`, and all other messages are written to the file `/var/log/messages`.

However, many applications write to other directories and this may vary by distribution. For example, Samba typically writes log files to the directory `/var/log/samba`.

## 22.1.6 Use hardening tools

The kernel itself is able to handle incoming and outgoing network traffic in several ways. For each TCP or UDP packet, you can specify what to do with it—and although a discussion of how to set up a complete firewall is outside the scope of this redbook—we want to give an overview of how the utilities can be used.

### *IPchains*

IPchains is used to set up, maintain, and inspect the IP firewall rules in the Linux kernel. Following are some of the basic IPchains commands:

<b>ipchains -L</b>	List all chains and the applied rules
<b>ipchains -A &lt;chain&gt; &lt;rule&gt;</b>	Add <rule> to chain named <chain>
<b>ipchains -D &lt;chain&gt; &lt;rule-number&gt;</b>	Delete <rule-number> in chain named <chain>
<b>ipchains -P &lt;chain&gt; &lt;policy&gt;</b>	Set default policy for chain named <chain> to <policy>

IPchains is used not only to act as a firewall between several networks, it can also be used to restrict network access to your machine. For example, if you want to give only dedicated hosts access to services such as http, use the following command:

```
# ipchains -A input -p tcp -s vmlinux1.itso.ibm.com/32 -d vmlinux2.itso.ibm.com/32 80
# ipchains -L
Chain input (policy DENY):
target    prot opt    source                destination            ports
-         tcp  ----- vmlinux1.itso.ibm.com vmlinux2.itso.ibm.com any -> http
Chain forward (policy DENY):
Chain output (policy ACCEPT):
```

**Note:** Take care with your configuration, because it's possible to cut yourself off the host with an ipchains rule! If this happens, networking will effectively be down and you'll only be able to get to the host to fix the problem via the local console under VM or the HMC.

If you want to use ip-forwarding in order to use ipchains with several networking devices, you have to enable it in the kernel. (We recommend you only enable ip-forwarding if you use ipchains to control network traffic and access, especially if you have more than one networking device.)

```
$ echo 1 > /proc/sys/net/ipv4/ip_forward
```

IPtables is the replacement for IPchains in the 2.4 kernel, but is not discussed further in this redbook.

## Tripwire

If attackers manage to infiltrate a system, it is possible that they could install *trapdoors* or other utilities for sniffing passwords. This may not only be difficult to detect in your system, but it may also be difficult to determine *how* your system has been compromised.

One safe approach is to totally reinstall the operating system, but again, you may not even know your system is compromised. Because of this exposure, tools like **tripwire** have been developed to maintain an overview of all files and their modification in the system.

Tripwire is able to detect the following file/directory modifications:

- ▶ Added files/directories
- ▶ Deleted files/directories
- ▶ Changed files/directories

The latest version of Tripwire is available on the Web at:

<http://www.tripwire.org>

In the SuSE distribution that comes with Tripwire version 1.2, the files are installed in directory `/var/adm/tripwire/`. However, it is not installed with all installations. If you don't have tripwire installed, look for the rpm on CD1 under the `sec` directory, as shown:

```
# file /suse/cd1/suse/sec1/tripwire.rpm
/suse/cd1/suse/sec1/tripwire.rpm: RPM v3 bin tripwire-1.2-262
```

The `tw.config` file is an important configuration file. In it, you add all directories and files that you want tripwire to monitor. You can also exclude files from a directory; as follows:

```
#
# Tripwire config-file
#
/ R
!/proc
!/var
!/root
/root/bin
!/dev
!/tmp
!/etc/mtab
!/etc/ntp.drift
!/etc/ld.so.cache
!/etc/snmpd.agentinfo
!/etc/ssh_random_seed
!/etc/mail/sendmail.st
```



## Port scanning detection

An attacker can figure out what services are running on the target host by scanning a server's ports. That's why we recommend that you disable unnecessary services, because the fewer "doors" on your system, the harder it is for a cracker to get in.

### *nmap*

Most port scanners can guess the operating system, and the attacker could have information about vulnerable services on the host. Some tools are able to recognize if a host is scanned by port scanners. Example 22-6 shows a host being scanned with the **nmap** tool.

*Example 22-6 Scanning a host with nmap*

---

```
# nmap -0 vmlinux2
```

```
Starting nmap V. 2.53 by fyodor@insecure.org ( www.insecure.org/nmap/ )
```

```
Interesting ports on vmlinux2.itso.ibm.com (9.12.6.99):
```

```
(The 1506 ports scanned but not shown below are in state: closed)
```

Port	State	Service
21/tcp	open	ftp
22/tcp	open	ssh
23/tcp	open	telnet
25/tcp	open	smtp
37/tcp	open	time
79/tcp	open	finger
80/tcp	open	http
110/tcp	open	pop-3
111/tcp	open	sunrpc
513/tcp	open	login
514/tcp	open	shell
515/tcp	open	printer
789/tcp	open	unknown
901/tcp	open	samba-swat
1507/tcp	open	symplex
1533/tcp	open	virtual-places
2049/tcp	open	nfs

```
TCP Sequence Prediction: Class=random positive increments
```

```
Difficulty=5525585 (Good luck!)
```

```
Remote operating system guess: Linux 2.1.122 - 2.2.14
```

```
Nmap run completed -- 1 IP address (1 host up) scanned in 1 second
```

```
vmlinux1:~ #
```

---

### **scanlog**

The **scanlog** daemon can be used to recognize a scan if hosts request more than a specific amount of ports per second. In this example, we have scanlog running and the scan is detected and written to the file `/var/log/messages`:

```
# grep scanlogd /var/log/messages
```

```
Jun  6 16:35:24 vmlinux2 scanlogd: 9.12.6.98 to 9.12.6.99 ports 36886, 19, 258, 348, 424, 205, 406, ..., f??pauxy, TOS 00 @16:35:23
```

## **22.1.7 Integrate VM and z/VM security**

In this section we describe the security available in VM and z/VM. VM guest security can be used in addition to Linux security.

Operating system failures that occur in virtual machines do not normally affect the VM operating system running on the real processor. If the error is isolated to a virtual machine, only that virtual machine fails, and the user can re-IPL without affecting the testing and production work running in other virtual machines.

The Control Program (CP) common to VM and z/VM has integrity such that programs running in a virtual machine are unable to do the following:

- ▶ Circumvent or disable the CP real or auxiliary storage protection.
- ▶ Access a resource protected by RACF (resources protected by RACF include virtual machines, minidisks, and terminals)
- ▶ Access a CP password-protected resource
- ▶ Obtain control in a real supervisor state, or with a privilege class authority or directory capabilities greater than those it was assigned
- ▶ Circumvent the system integrity of any guest operating system that itself has system integrity as the result of an operation by any VM CP facility

Following are more detailed explanations of these terms:

**Real storage protection** Refers to the isolation of one virtual machine from another. CP accomplishes this by hardware dynamic address translation, start interpretive-execution guest storage extent limitation, and the Set Address Limit facility.

### **Auxiliary storage protection**

Refers to the disk extent isolation implemented for minidisks/virtual disks through channel program translation.

**Password-protected resource**

Refers to a resource protected by CP logon passwords and minidisk passwords.

**Guest operating system** Refers to a CP (such as Linux for S/390 and zSeries) that operates under the CP.

**Directory capabilities** Refers to those directory options that control functions intended to be restricted by specific assignment, such as those that permit system integrity controls to be bypassed, or those not intended to be generally granted to users.

VM includes several facilities to enhance or improve the security and integrity of the system:

- ▶ Each guest and CMS user runs in a unique virtual machine definition which, in combination with hardware features, prohibits one user from accessing another's data in storage (unless specifically allowed through shared segments, communication vehicles such as IUCV and APPC/VM, or ESA/XC data sharing services).
- ▶ VM, in combination with hardware features, provides protection against channel programs accessing another user's virtual addresses.
- ▶ A password facility provides minidisk security to control both read-only and read-write access.
- ▶ Both user ID and password checking are provided to minimize unauthorized system access.
- ▶ User class restructure provides customers with the ability to control access to commands and DIAGNOSE codes more precisely through customer-defined classes.
- ▶ Journaling is supported on VM. In addition, RACF provides customers with many of these facilities, as well as other security capabilities.

In addition to these features for VM, z/VM offers the following function:

Directory control statements and system configuration file statements provide controls for certain POSIX-related functions, such as the ability to change another virtual machine's POSIX security values.

The TCP/IP Feature for z/VM offers:

- Kerberos authentication service
- Secure Socket Layer (SSL) support

## **RACF under z/VM**

The Resource Access Control Facility (RACF) licensed program is a strategic security facility that provides comprehensive security capabilities. RACF controls user access to the system, checks authorization for use of system resources, and audits the use of system resources.

In the z/VM environment, RACF verifies logon passwords and checks access to minidisks, data in spool files, and RSCS nodes. You can use RACF commands to audit security-relevant events and prevent users from entering the CP DIAL and MSG commands before they log on.

The events you can audit include:

- ▶ Any CP command or DIAGNOSE code (including privileged commands and DIAGNOSE codes)
- ▶ Creating, opening, and deleting spool files
- ▶ Dumping and loading spool files through the SPXTAPE and SPTAPE commands
- ▶ IUCV CONNECT and SEVER operations, and certain VMCF functions
- ▶ APPC/VM CONNECT and SEVER operations
- ▶ Creating and deleting logical devices



## Backup and restore

In this chapter we discuss the following primary topics related to backing up and restoring data under Linux for zSeries and S/390:

- ▶ Backup of entire Linux images under VM using DDR

This allows you to back up all of the VM minidisks that comprise a Linux system (we recommend that you shut the system down during the backup process).
- ▶ Tape support under Linux for zSeries and S/390

This allows you to perform backups from the Linux guest directly to tape. However, because Linux for zSeries is “young”, it has taken some time for a tape driver to be made available. For this reason, the generally available distributions do not currently include tape support (although it’s anticipated they will soon). Because tape support is not standard, we describe how to rebuild the Linux kernel and include tape support.
- ▶ Disaster backup and restore

This allows you to create contingencies to handle a disaster where an entire system is wiped out due to fire, flood, a malicious cracker, or other unexpected causes.
- ▶ Incremental backup

This allows you to handle situations where individual files or directories are lost and need to be restored quickly (which is often due to users accidentally deleting data, or to poorly-written applications deleting data).

## 23.1 Linux image backup and restore under VM

VM and z/VM provide a useful facility called DASD Dump Restore (DDR) which, although having other capabilities, can also be used to back up a Linux guest. In the following section, we examine this capability of DDR as it relates to disaster recovery.

### 23.1.1 Backing up a Linux guest using DDR

A restorable image of an entire Linux for zSeries and S/390 system can be made using DDR by following these basic steps:

1. Create a simple REXX script to run DDR.
2. Run your DDR script with the appropriate responses to back up your Linux guest.
3. Run your DDR script with the appropriate responses to restore your Linux guest.

Use of DDR requires read access to your source device, write access to your destination device, and an attached tape drive. You may need to coordinate with your system administration staff for such capability.

#### Bring down your Linux guest

To use DDR, your Linux system should be offline. You can bring it offline by using the command `shutdown -h now`, while logged in as root on Linux. Then log on to VM from a 3270 session, and use the command `CP IPL CMS` to bring up CMS.

#### Creating the REXX EXEC

You can create a file on VM, which in our example we named `RUNDDR EXEC`, similar to the following:

```
/* */
'CLOSE RDR'
'PURGE RDR ALL'
'SPOOL PUN *'
'PUNCH IPL DDRXA S (NOHEADER'
'SPOOL RDR HOLD CLASS *'
'IPL C'
```

This exec puts the standalone DDRXA program in your VM reader, and then IPLs that system from the reader.

## Running DDR to back up a Linux guest

You execute the RUNDDR EXEC by entering the command **RUNDDR**. Respond to each of the DDR ENTER: prompts with the responses:

1. **SYSPRINT CONS** (sends system messages to the console)
2. **IN 201 3390** (201 is the source DASD, which is the entire root file system)
3. **OUT B38 TAPE** (B38 is the physical address of the tape drive)
4. **DUMP ALL** (source cylinders could also be specified, such as **COPY 0 3338**)

To get out of DDR, just press **ENTER** at the ENTER: prompt. The output of the DDR process should be similar to that shown in Example 23-1:

### *Example 23-1 Running DDR to back up a Linux guest*

---

```
Ready; T=0.01/0.01 15:09:47
runddr
0000001 FILE PURGED
RDR FILE 0016 SENT FROM VMLINUX3 PUN WAS 0016 RECS 1298 CPY 001 A NOHOLD
NOKEEP
z/VM DASD DUMP/RESTORE PROGRAM
ENTER CARD READER ADDRESS OR CONTROL STATEMENTS
ENTER: SYSPRINT CONS
ENTER: IN 201 3390
ENTER: OUT b38 TAPE
ENTER: DUMP ALL
HCPDDR711D VOLID READ IS LNX201
DO YOU WISH TO CONTINUE? RESPOND YES, NO OR REREAD: YES
DUMPING LNX201
DUMPING DATA 06/06/01 AT 19.11.14 GMT FROM LNX201
INPUT CYLINDER EXTENTS OUTPUT CYLINDER EXTENTS
START STOP START STOP
END OF VOLUME CYL 00000803 HD 11, MOUNT NEXT TAPE
00000000 00000803 00000000 00000803
15:14:44 HCPERP2233I TAPE 0B38 NOT READY, CP-OPERATOR NOTIFIED
END OF VOLUME CYL 00001579 HD 14, MOUNT NEXT TAPE
00000803 00001579 00000803 00001579
15:21:03 HCPERP2233I TAPE 0B38 NOT READY, CP-OPERATOR NOTIFIED
END OF VOLUME CYL 00002353 HD 13, MOUNT NEXT TAPE
00001579 00002353 00001579 00002353
15:24:41 HCPERP2233I TAPE 0B38 NOT READY, CP-OPERATOR NOTIFIED
00002353 00002599 00002353 00002599
END OF DUMP
ENTER:
```

---

## Running DDR to restore a Linux guest

Again, execute **RUNDDR**. Respond to each of the four DDR ENTER: prompts by using the responses from the following list:

1. **SYSPRINT CONS** (send system messages to the console)
2. **IN B38 TAPE** (B38 is the source tape drive)
3. **OUT 202 3390** (202 is the destination minidisk to contain the root file system).
4. **RESTORE ALL** (a destination start and stop cylinder could also be specified, such as RESTORE 0 3338)

The entire DDR restore process should resemble the screen shown in Example 23-2:

*Example 23-2 Running DDR to restore a Linux guest*

---

```

runddr
Ready; T=0.01/0.01 15:53:43
0000001 FILE PURGED
RDR FILE 0019 SENT FROM VMLINUX3 PUN WAS 0019 RECS 1298 CPY 001 A NOHOLD
NOKEEP
z/VM DASD DUMP/RESTORE PROGRAM
ENTER CARD READER ADDRESS OR CONTROL STATEMENTS
ENTER: SYSPRINT CONS
ENTER: IN B38 TAPE
ENTER: OUT 202 3390
ENTER: RESTORE ALL
HCPDDR717D DATA DUMPED FROM LNX201 TO BE RESTORED
DO YOU WISH TO CONTINUE? RESPOND YES, NO OR REREAD: YES
HCPDDR711D VOLID READ IS LNX201
DO YOU WISH TO CONTINUE? RESPOND YES, NO OR REREAD: YES
RESTORING LNX201
DATA DUMPED 06/06/01 AT 19.11.14 GMT FROM LNX201 RESTORED TO LNX201
INPUT CYLINDER EXTENTS      OUTPUT CYLINDER EXTENTS
      START      STOP      START      STOP
END OF VOLUME CYL 00000803 HD 11, MOUNT NEXT TAPE
 15:58:56 HCPERP2233I TAPE 0B38 NOT READY, CP-OPERATOR NOTIFIED
END OF VOLUME CYL 00001579 HD 14, MOUNT NEXT TAPE
 16:04:14 HCPERP2233I TAPE 0B38 NOT READY, CP-OPERATOR NOTIFIED
END OF VOLUME CYL 00002353 HD 13, MOUNT NEXT TAPE
 16:09:17 HCPERP2233I TAPE 0B38 NOT READY, CP-OPERATOR NOTIFIED
      00000000      00002599      00000000      00002599
END OF RESTORE
ENTER:

```

---

We chose to restore the Linux guest to the 202 minidisk so as to preserve the system on 201. We could then IPL the restored Linux system via the command **CP IPL 202**. If your Linux system exists on multiple minidisks, you would simply back up and restore them in sets.



## 23.2 Tape support under Linux for zSeries and S/390

The use of S/390 and zSeries tape hardware is still in the early stages of development and therefore is not currently integrated into any GA Linux distributions. The Version 6.5 “beta” of Turbolinux does include the tape driver, but the /dev files are not included and must be created as documented in this section.

Documentation for the tape driver was found in the 2.2 kernel document *LINUX for S/390 Device Drivers and Installation*. It is on the Web at:

<http://oss.software.ibm.com/developerworks/opensource/linux390/docu/l390dd03.pdf>

At the start of this document is the warning:

“This driver is considered to be BETA. Do NOT use it in production environments. Feel free to test it and report problems back to us.”

Despite the warning, we proceeded; we recompiled the kernel to include tape support, and observed no instabilities with the current development beta driver off the IBM developerWorks Web site. In the following section we document the steps taken to add tape support into Linux for zSeries and S/390.

**Note:** Tape hardware is not currently supported under VIF because there is no means to define a tape device to any of the Linux guests.

### 23.2.1 Download kernel source, kernel patch, and lcs module

In order to rebuild your kernel, you must get the *vanilla* Linux kernel, the kernel patches, and the network device driver to the corresponding kernel. The vanilla kernel can be obtained from kernel.org. We downloaded the 2.2.19 kernel from the following URL:

<http://www.kernel.org/pub/linux/kernel/v2.2/linux-2.2.19.tar.gz>

To add tape support, you must get the driver from developerworks. We downloaded the tape driver from the following URL:

[http://oss.software.ibm.com/developerworks/opensource/linux390/exp\\_src.html](http://oss.software.ibm.com/developerworks/opensource/linux390/exp_src.html)

There we found the *experimental patch for linux kernel version 2.2.19* with the tarball named `linux-2.2.19-s390.tar.gz`, dated 2001-04-18. It contained the following release notes:

This patch provides an updated S/390 backend for linux kernel version 2.2.19. It contains the following:

- all bug fixes and updates dropped for kernel version 2.2.16 plus some recent ones
- a backport of the tape device driver:

The tape device driver has support for multiple volumes.  
Note that you must use the `mt` tool for the rewind/unload operation (it must be software driven). The tape device driver cannot control rewind/unload operations performed directly on the device.  
This restriction is permanent.

- support for VM pseudo page fault handling

When we clicked the file link, we were asked to accept with the GPL before we could download the patch.

As mentioned, in addition to the kernel patch, a network device driver is needed. The IBM zSeries network drivers are object code only (OCO). As such, the driver must correspond to the version of the kernel you will be building. We found the modules at the following URL:

[http://oss.software.ibm.com/developerworks/opensource/linux390/240/240\\_obj.html](http://oss.software.ibm.com/developerworks/opensource/linux390/240/240_obj.html)

We needed the LCS Driver (e.g. ethernet, fast ethernet, token ring) because we were using fast ethernet via an OSA card. We found it in the file `lcs-2.2.19-s390.tar.gz`, dated 2001-04-26. It contained the following release notes:

```
The following device drivers are compatible with the experimental Linux
kernel version 2.2.19 and are object code only.
Documentation on their use can be found in the LINUX for S/390
device drivers manual in the Documentation section.
```

When we clicked the file link, we were asked to accept an IBM license. This license gives us the right to use, copy and distribute the object code, but not the right to the source code, nor to modify the program.

## 23.2.2 Extract kernel source and patch

After downloading all files, we copied the kernel source and kernel patch to the `/usr/src` directory. By convention, the directory `/usr/src/linux` is a symbolic link to the current source tree. We therefore removed the link and pointed the new one to `/usr/src/linux-2.2.19`. We executed the following commands:

```
# cd /usr/src
# rm -fv linux
# mkdir linux-2.2.19
# ln -s linux-2.2.19 linux
# tar -xvzf linux-2.2.19.tar.gz
# tar -xvzf linux-2.2.19-s390.tar.gz
# mkdir linux-2.2.19/configs
# cp linux-2.2.16/configs/kernel-2.2.16-127-vm-s390.config
  linux-2.2.19/configs
```

### 23.2.3 Apply the patch

When we untarred the kernel patch, we got the release notes and the IBM license, but the important file is the diff file. Let's look at the head of that file:

```
# cd /usr/src
# head -4 linux-2.2.19-s390.diff
diff -urN linux-2.2.19/Documentation/Debugging390.txt
    linux-2.2.19-s390/Documentation/Debugging390.txt
--- linux-2.2.19/Documentation/Debugging390.txt Sun Mar 25 18:31:59 2001
+++ linux-2.2.19-s390/Documentation/Debugging390.txt Mon Jan 29 16:49 2001
@@ -1,19 +1,21 @@
```

As you can see, the IBM patch file appends a kernel version number (-2.2.19), while the vanilla kernel just goes to the /linux directory. Thus the symbolic link from linux/ to linux-2.2.19/ was critical:

```
# ls -ld linux-2.2.19 linux
lrwxrwxrwx  1 root  root           12 Jul 3 14:19 linux -> linux-2.2.19
drwxrwxr-x  2 root  root          4096 Jul 3 14:19 linux-2.2.19
```

Now we applied the patch from /usr/src:

```
# patch -p0 < linux-2.2.19-s390.diff
```

Ensure the patch is applied cleanly (no error messages are issued).

### 23.2.4 Recompile kernel

We recompiled the Linux kernel by first cleaning up, then copying the .config file and issuing **make menuconfig** to bring up the kernel configuration menu:

```
# cd /usr/src/linux
# make clean
# make mrproper
# cp configs/kernel-2.2.16-127-vm-s390.config .config
# make oldconfig
# make menuconfig
```

We chose the **S/390 character device drivers** ---> menu option, and changed settings on the next page to match those shown in Example 23-3:

*Example 23-3 Linux kernel S/390 character device drivers panel*

```
Linux Kernel v2.2.19 Configuration
-----
+----- S/390 character device drivers -----+
| Arrow keys navigate the menu. <Enter> selects submenus --->. |
| Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, |
| <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help. |
| Legend: [*] built-in [ ] excluded <M> module < > module capable |
| +-----+ |
| | <M> S/390 tape device support | |
| | --- S/390 tape interface support | |
| | [*] Support for tape character devices | |
| | [*] Support for tape block devices | |
| | --- S/390 tape hardware support | |
| | [*] Support for 3490 tape hardware | |
| | [*] Support for 3480 tape hardware | |
| | | |
| | +-----+ |
| +-----+ |
| | <Select> < Exit > < Help > | |
| +-----+ |
+-----+
```

We used the **Escape** key to exit, selecting **Yes** when prompted to save the new configuration file.

```
# cp .config configs/kernel-2.2.19-vm-s390tape.config
# make dep modules modules_install image
```

If you would like to see the output of the compile, but would prefer not to watch the entire process, you can redirect the output as shown:

```
make dep modules modules_install image > /var/log/mycompile.log
```

## 23.2.5 Copy the new kernel files

We now made copies of the important files to the boot directory:

```
# cp /usr/src/linux/Makefile /boot/Makefile-2.2.19-s390t
# cp /usr/src/linux/System.map /boot/System.map-2.2.19-s390t
# cp /usr/src/linux/arch/s390/boot/image /boot/image-2.2.19-s390t
# cp -f /usr/src/linux/arch/s390/boot/ip1* /boot/
```

## 23.2.6 Replace the System.map

We replaced the system map for debugging purposes:

```
# rm -fv /boot/System.map
# ln -s /boot/System.map-2.2.19-s390t /boot/System.map
```

## 23.2.7 Extract the new LCS driver

We downloaded the new LCS driver to the directory `/lib/modules/2.2.19/net` and executed the following commands:

```
# cd /lib/modules/2.2.19/net
# tar -xvzf lcs-2.2.19-s390.tar.gz
```

Because we had a copy of this source tarball, we removed the extraneous files from the `/lib/modules` directory tree, as shown:

```
# rm -fv lcs-2.2.19-s390.tar.gz
# rm -fv README
# rm -fv LICENSE
```

## 23.2.8 Enable the new LCS driver

In case you need to go back to your previous kernel, it might be a good idea to make a copy of your `/etc/modules.conf` file at this time, as follows:

```
# cp /etc/modules.conf /etc/modules.conf.2.2.16
```

Edit `/etc/modules.conf` and change the two LCS lines to resemble the following example:

```
alias eth0 lcs-2.2.19-s390
options lcs-2.2.19-s390 auto
```

## 23.2.9 Update boot record and reboot to new kernel

The command `sil0` is used to prepare a DASD to become an IPL device. In order to use our new kernel, we had to rerun `sil0` to update the boot record. Our root file system is on the first DASD, `/dev/dasda`. After updating the IPL device, we rebooted:

```
# /sbin/sil0 -f /boot/image-2.2.19-s390t -d /dev/dasda -p /boot/parm -b
/boot/ipleckd.boot -t2
# /sbin/shutdown -r now
```

## 23.2.10 Recovering from a bad compile

The previous process was tested and found to work. However, should something go wrong and you find that you're unable to re-IPL Linux, you'll need to reboot your system from your original installation media (such as your installation tape, VM reader, or so forth) in order to "rescue" your system back to the previous kernel version.

Based upon the previous steps, the following "rescue procedure" was tested with our VM guest-based Turbolinux Version 6.0 system, to return our system from the 2.2.19 kernel to the original 2.2.16 kernel:

1. Run `lin` (this is the LIN EXEC created during the Turbolinux install, executed from a CMS VM Read prompt).
2. Answer `n` to the `eth0`, `tr0`, `ctc0`, and `escon0` configuration prompts.
3. Press **<Enter>** *twice* for each of the Configure Device Independent Network Settings prompts.
4. Answer `y` to accept the default settings, which should lead you to a Linux prompt. Then try to back out your change, as follows:

```
# insmod dasd_mod dasd=201
# mkdir /mnt/save
# mount /dev/dasda1 /mnt/save
# chroot /mnt/save
# cp /etc/modules.conf /etc/modules.conf.2.2.19
# cp /etc/modules.conf.2.2.16 /etc/modules.conf
# rm -fv /boot/System.map
# ln -s System.map-2.2.16-127-tape System.map
# /sbin/silo -f /boot/image-2.2.16-127-tape -d /dev/dasda -p /boot/parm
  -b /boot/ipleckd.boot -t2
# halt
```

5. Now IPL from VM or, if you're running in an LPAR, from the HMC. Here is our example from VM:

```
IPL 201 CLEAR
```

## 23.2.11 Create tape devices

Since Linux for S/390 distributions may not include any tape devices, the following commands may be used to create a Linux device for each available tape device:

```
# mknod /dev/rtibm0 c 254 0 (first rewinding character device)
# mknod /dev/ntibm0 c 254 0 (first non-rewinding character device)
# mknod /dev/btibm0 b 254 0 (first block device)
```

```
# mknod /dev/rtibm1 c 254 1 (second rewinding character device)
# mknod /dev/ntibm1 c 254 1 (second non-rewinding character device)
# mknod /dev/btibm1 b 254 1 (second block device)
...
```

**Note:** In the previous example, the owner of each new tape device would be root. If an alternative owner is required, the `mknod --mode=` switch can be used to specify permissions when each device is created, or `chmod` may be used to modify permissions at a later date.

### 23.2.12 Load the tape driver

We loaded the tape driver via the command:

```
# /sbin/modprobe tape390
```

Because we were running Linux under z/VM and there is only one tape drive assigned, the tape driver associates `/dev/ntibm0` with the physical tape drive at address B38.

### 23.2.13 Test the tape driver

At this point, we had rebuilt the kernel with tape support, created devices in the `/dev` directory, and added the `tape390` module.

The Magnetic Tape tool, or `mt`, is used for general tape control operations. It can also be used to test whether the new device is working properly: mount a tape in the tape drive and execute the following commands to test the newly added tape support (assuming the first tape device on the system):

```
# mt -f /dev/ntibm0 rewind
# mt -f /dev/ntibm0 retension
# mt -f /dev/ntibm0 erase
```

If these commands are successful, it means you have tape support on your Linux system.

## 23.3 Disaster backup and restore

Disaster backup and recovery addresses the possibility of your entire system being wiped out as a result, for example, of fire or flood. Comprehensive disaster recovery planning can be a complex process, requiring the consideration of a variety of factors such as tape rotation schemes, offsite tape storage, tape vaults, testing and documentation of procedures, and so on. In fact, entire books, such as *Unix Backup & Recovery* by W. Curtis Preston, are devoted to this subject.

For the purposes of this redbook, we limit the scope of our discussion to the following topics:

- ▶ A brief introduction to a complete file system-based backup and restore
- ▶ Back up and restore of the entire Linux file system using standard Linux utilities such as **tar** and **cpio** on the local Linux system
- ▶ Back up and restore of the entire Linux file system using the Tivoli Storage Manager (TSM) client to an enterprise's existing TSM server

### 23.3.1 Complete file system-based backup and restore

While it might seem like an obvious point, a complete, or “full” file system-based backup and restore requires access to the file system being either backed up or restored. This means that the following conditions must exist:

- ▶ The Linux user that is backing up the file system must have sufficient rights to *read* from all of the directories and files on the Linux system being backed up.
- ▶ The Linux user that is restoring the file system must have sufficient rights to *write* to all of the directories and files on the Linux system being restored.
- ▶ If backing up to, or restoring from, a local tape device, tape support must be working on the system being backed up or restored to.
- ▶ If backing up to, or restoring from, a device and/or other system across a network connection, network connectivity must be working.
- ▶ A complete restore of a missing or destroyed system first requires that the Linux file system be accessible, generally requiring that Linux be reinstalled and function *prior* to restoration. Furthermore, if restoring from a local tape device, tape support might need to be reinstalled as well.

### 23.3.2 Complete file system backup and restore via Linux utilities

Linux and UNIX have a variety of tools that allow for the copy and/or manipulation of files and file system. Because system administrators often prefer to use the particular tools that they are most familiar with, the examples in this section are based upon three of the most common tools: **tar**, **cpio**, and **afio**.

The command **tar** (Tape ARchive), is the most commonly used.

**Note:** A discussion of the **afio** command was included in this section because of its enhanced ability to deal with corrupted archives, as compared with **cpio**. However, **afio** may not be installed and available on all systems.



### Back up and restore using tar:

- ▶ To back up: `tar -cpzvf /dev/ntibm0 / --exclude /proc`
- ▶ To restore: `tar -xpvf /dev/ntibm0`
- ▶ To list tape contents: `tar -tpzvf /dev/ntibm0`
- ▶ The `-M` switch can also be used to support multiple tapes, *but is incompatible with the `-z` compression switch*, and was not tested here

### Back up and restore using cpio:

- ▶ To back up: `find / -print|cpio -oacvB > /dev/ntibm0`
- ▶ To restore: `cpio -icvdBum * < /dev/ntibm0`
- ▶ To list tape contents: `cpio -itcvB < /dev/ntibm0`

### Back up and restore using afio:

- ▶ To back up: `find / -print|afio -oZv /dev/ntibm0`
- ▶ To restore: `afio -iZv * /dev/ntibm0`
- ▶ To list tape contents: `afio -tZv /dev/ntibm0`

## 23.3.3 Complete file system backup and restore via TSM Client

Up to this point, all of our discussion about backup and restore under Linux for zSeries and S/390 has centered around localized schemes, wherein tape hardware that is directly attached to the system is used. However, another common backup and restore method is *enterprise backup*, wherein all of the backup and restore operations for an entire organization are managed centrally on an *enterprise backup server*.

Two examples of enterprise backup servers are the Advanced Maryland Automatic Network Disk Archiver (AMANDA), an open-source offering, and the Tivoli Storage Manager (TSM), a commercial software package.

Additional information on AMANDA can be found on the AMANDA Web site:

<http://www.amanda.org>

We focus on using the 4.2 Version of the TSM Client to allow for the backup and restoration of Linux for zSeries and S/390 file systems and files.

## Installation of the TSM Client

The minimum system requirements for installing the Version 4.2 TSM backup client and APIs are as follows:

- Linux kernel 2.2.16 or higher
- glibc 2.1.2 or higher
- libstdc++ 2.9.0 or higher
- RPM 3.0.4 or higher
- 50 MB of available file system space under /opt

Version 4.2 of the TSM Client for S/390 is distributed in Redhat Package Manager (RPM) format. After confirming that your Linux for zSeries and S/390 installation fulfills the minimum system requirements, and downloading the TSM client, you can install the software as follows:

1. `rpm -ivh TIVsm-BA.s390.rpm` (this will install the TSM backup client)
2. `rpm -ivh TIVsm-API.s390.rpm` (this will install the TSM backup APIs)

All of the TSM files will be installed under /opt/tivoli/tsm/client.

**Note:** The Version 4.2 TSM Client that we tested was a beta version that should soon be Generally Available (GA). However, because the client was still in beta, we were unable to include a URL for downloading. Tivoli's TSM Web site can be accessed at the following URL:

[http://www.tivoli.com/products/documents/updates/storage\\_mgr\\_features.html](http://www.tivoli.com/products/documents/updates/storage_mgr_features.html)

## Configuration of the TSM Client

The basic configuration of the TSM Client is contained within two files:

- `dsm.sys`, the main configuration file, contains the actual TSM server information.
- `dsm.opt` is a secondary configuration file that can be used to specify a specific TSM server name if more than one TSM server is defined in the `dsm.sys` file.

Each of these files resides in the /opt/tivoli/tsm/client/ba/bin directory.  
Example 23-4 shows a sample dsm.sys file:

*Example 23-4 dsm.sys*

---

```
*****
* Tivoli Storage Manager                                     *
*                                                         *
* Sample Client System Options file for UNIX (dsm.sys.smp) *
*****

* This file contains the minimum options required to get started
* using TSM. Copy dsm.sys.smp to dsm.sys. In the dsm.sys file,
* enter the appropriate values for each option listed below and
* remove the leading asterisk (*) for each one.

* If your client node communicates with multiple TSM servers, be
* sure to add a stanza, beginning with the SERVERNAME option, for
* each additional server.

*****

SErvername  server_a
COMMmethod  TCPip
TCPport    1500
TCPserveraddress  node.domain.company.COM
```

---

Example 23-5 shows a sample dsm.opt file:

*Example 23-5 dsm.opt*

---

```
*****
* Tivoli Storage Manager                                     *
*                                                         *
* Sample Client User Options file for UNIX (dsm.opt.smp) *
*****

* This file contains an option you can use to specify the TS
* server to contact if more than one is defined in your clie
* system options file (dsm.sys). Copy dsm.opt.smp to dsm.op
* If you enter a server name for the option below, remove th
* leading asterisk (*).

*****

* SErvername      A server name defined in the dsm.sys file
```

---

Configuration of the TSM Client is accomplished by completing the following steps:

```
# cd /opt/tivoli/tsm/client/ba/bin
# cp dsm.sys.smp dsm.sys
# cp dsm.opt.smp dsm.opt
# vi dsm.sys
... change the "TCPServeraddress" to match that of your TSM server
# vi dsm.opt
... change the "Servername" to match that of your TSM server
# vi /home/yourbackupusername/.profile
... add the following lines:
export DSM_CONFIG=/opt/tivoli/tsm/client/ba/bin/dsm.opt
export DSM_DIR=/opt/tivoli/tsm/client/ba/bin
```

## Using the TSM Client

A complete discussion of the capabilities and options of the TSM Client is beyond the scope of this redbook. However, there are many ways you can get additional documentation on the TSM Client including the following:

- ▶ The TSM client help facility (**dsmc**--->?)
- ▶ The TSM documentation that is supplied with the client software in the file `/opt/tivoli/tsm/client/books/pdf/tsmref/ansbcr41.pdf`
- ▶ The Web-based downloadable documentation:  
[http://www.tivoli.com/support/public/Prodman/public\\_manuals/storage\\_mgr/admanual.htm#tarcom](http://www.tivoli.com/support/public/Prodman/public_manuals/storage_mgr/admanual.htm#tarcom)

The following common commands will provide a brief introduction:

- ▶ To back up, use the command:  

```
# dsmc incremental -password=<your_password>
```
- ▶ To restore, use the command:  

```
# dsmc restore -password=<your_password> <files_to_be_restored>
```
- ▶ To query backed-up files:  

```
# dsmc query backup -password=<your_password> <files_to_be_queried>
```
- ▶ To enable the TSM scheduler (allowing backups to be scheduled from the TSM server), use the command:  

```
# dsmc schedule -password=your_password
```

**Note:** If you'd like the TSM scheduler to start automatically at system startup, and you're running under a SystemV-compatible Linux distribution such as Red Hat or Turbolinux, add this line to the `/etc/rc.d/rc.local` file:

```
/opt/tivoli/tsm/client/ba/bin/dsmc schedule -password=yournodespassword
```

## The TSM Web client

Although not used in our testing, Version 4.2 of the TSM Client for Linux for zSeries also includes an optional Web client, which can be used instead of the **dsmc** command line for client-side TSM tasks. Refer to the README.WEBCLI.txt file included with the TSM Client for further information on the configuration and usage of this feature.

## Troubleshooting the TSM Client

The basic architecture of TSM is such that storage management, policy, and the management of scheduled tasks takes place on the server, and the parameters related to a specific activity (such as which files should be backed up, which directories should be excluded, and so on), are managed by the client.

In situations where a specific task did not complete as expected, the following two client-side log files may be assistance:

- ▶ `/currentdirectory/dsmerror.log` - records any errors received in response to issued **dsmc** commands
- ▶ `/opt/tivoli/tsm/client/ba/bin/dsmsched.log` - records any logged client-side errors related to the TSM scheduler

## 23.4 Incremental backup and restore

Incremental backup addresses the case of accidentally deleting files. In this section we discuss the following topics:

- ▶ A brief introduction to incremental file system-based backup and restore
- ▶ Incremental backup and restore using standard Linux utilities such as **tar** and **cpio** on the local Linux system
- ▶ Incremental backup and restore using the Tivoli Storage Manager (TSM) client to an enterprise's existing TSM server

### 23.4.1 Incremental file system-based backup and restore

The similarities between a complete (or “full”) file system-based backup and restore, and an incremental file system-based backup and restore are probably obvious: access to the actual file system to be backed up or restored is required, (either by local access or via a network), as are sufficient user privileges to either read from or write to the file system.

In addition to sufficient file system access, however, an incremental backup or restore requires some method of differentiating between specific versions (“generations”) of files. This differentiation is usually accomplished through an examination of the date and time a file was *last modified*.

Knowing when a file was last changed allows a backup operator (or an end user with access to backup media and the appropriate software tool) to make comparisons between current and archived/backed-up versions of files, so that the appropriate files can then be either backed up or restored/replaced.

## 23.4.2 Incremental file system backup and restore via Linux utilities

In this section, we once again examine two of the most common Linux tools, **tar** and **cpio**, as their use relates to incremental backup and restore.

### Incremental backup and restore using tar

- To back up:  

```
# tar -cpzvf -g /var/log/tarbu.log -f /dev/ntibm0 / --exclude /proc
```
- To restore:  

```
# tar -xpvzf /dev/ntibm0
```
- To list tape contents:  

```
# tar -tpzvf /dev/ntibm0
```
- The **-M** switch can also be used to support multiple tapes, *but is incompatible with the -z compression switch*, and was not tested here

**Note:** The **-g filename** option merely records filenames and modification times/dates in the filename file (which is `/var/log/tarbu.log`, in our example). The tarballs created with this option are standard, so there is no difference in the “incremental” restore or list commands.

Removal of the filename file, or use of an alternative empty file, will then allow for a new complete or “full” backup to occur.

Tar also has a ‘G’ incremental switch, but it creates non-standard tar files.

### Incremental backup and restore using cpio

- ▶ First day’s backup:  

```
# find / -print > /var/log/MMDDbu.log  
# cat /var/log/MMDDbu.log|cpio -oacvB > /dev/ntibm0  
# find / -print > /var/log/0530bu.log
```

```
# cat /var/log/0530bu.log|cpio -oacvB > /dev/ntibm0
```

► Next day's backup:

```
# find / -newer /var/log/prevMMDDbu.log -print > /var/log/MMDDbu.log  
# cat /var/log/MMDDbu.log|cpio -oacvB > /dev/ntibm0  
# find / -newer /var/log/0530bu.log -print > /var/log/0531bu.log  
# cat /var/log/0531bu.log|cpio -oacvB > /dev/ntibm0
```

**Note:** In the second **cpio** incremental backup example, the previous day's log file, `/var/log/0530bu.log`, is left on the system and will probably need to be deleted at some point.

Alternatively, you could choose to append the following line to the end of the "Next day's backup" command line:

```
a ;rm -fv /var/log/0530bu.log
```

Or retain the file, and use another file maintenance method at a later date.

► To restore:

```
# cpio -icvdBum * < /dev/ntibm0
```

► To list tape contents:

```
# cpio -itcvB < /dev/ntibm0
```

## Incremental backup and restore using **afio**

► First day's backup:

```
# find / -print > /var/log/MMDDbu.log  
# cat /var/log/MMDDbu.log|afio -oZv /dev/ntibm0  
# find / -print > /var/log/0530bu.log  
# cat /var/log/0530bu.log|afio -oZv /dev/ntibm0
```

► Next day's backup:

```
find / -newer /var/log/prevMMDDbu.log -print > /var/log/MMDDbu.log  
# cat /var/log/MMDDbu.log|afio -oZv /dev/ntibm0  
# find / -newer /var/log/0530bu.log -print > /var/log/0531bu.log  
# cat /var/log/0531bu.log|afio -oZv /dev/ntibm0
```

**Note:** In the second **afio** incremental backup example, the previous day's log file, `/var/log/0530bu.log`, is left on the system and will probably need to be deleted at some point.

Alternatively, you could choose to append the following line to the end of the "Next day's backup" command line:

```
d a ;rm -fv /var/log/0530bu.log
```

Or retain the file and use another file maintenance method at a later date.

- ▶ To restore:

```
# afio -iZv * /dev/ntibm0
```

- ▶ To list tape contents:

```
# afio -iZv /dev/ntibm0
```

### 23.4.3 Incremental file system backup and restore via TSM Client

In a sense, this topic is covered in "Using the TSM Client" on page 478, as all TSM backups are inherently incremental in nature. A full backup is only done the first time, and all future backups reference the original backup, as well as intermediary backups, and back up incrementally.

That said, however, there are specific TSM options that allow for a more granular selection of files to be backed up and restored, as follows:

- ▶ Backup and/or restore using **include** and **exclude**
- ▶ Restore based upon file dates and times

#### TSM backup using include and exclude

Specific **include** and **exclude** statements can be added to the `dsm.sys` file, allowing for a more granular selection of which files you wish to back up or restore. Example 23-6 shows the `dsm.sys` file we used to test this functionality:

*Example 23-6 dsm.sys with include and exclude statements*

```
*****
* Tivoli Storage Manager                                     *
*                                                           *
* Sample Client System Options file for UNIX (dsm.sys.smp) *
*****
* This file contains the minimum options required to get started
* using TSM. Copy dsm.sys.smp to dsm.sys. In the dsm.sys file,
* enter the appropriate values for each option listed below and
```



\* remove the leading asterisk (\*) for each one.

\* If your client node communicates with multiple TSM servers, be  
\* sure to add a stanza, beginning with the SERVERNAME option, for  
\* each additional server.

\*\*\*\*\*

```
SErvername server_a
  COMMmethod TCPip
  TCPPort 1500
  TCPServeraddress wtscmx.itso.ibm.com
```

```
NODename vmlinux3
```

```
EXCLUDE.DIR /proc
EXCLUDE.DIR /tmp
EXCLUDE.DIR /home/vmlinux3/amanda
EXCLUDE.DIR /home/vmlinux3/backup
EXCLUDE.DIR /home/vmlinux3/incoming
EXCLUDE.DIR /home/vmlinux3/taper
EXCLUDE.DIR /home/vmlinux3/tsmrestdir
EXCLUDE /home/vmlinux3/*
INCLUDE /home/vmlinux3/TSMClient4.2/*
```

**Note:** In our testing of the 4.2 beta version of the TSM Client for Linux for zSeries and S/390, we were unable to get includes to work for files and/or directories located *underneath* directories specified with EXCLUDE.DIR statements, seemingly due a higher precedence being granted to EXCLUDE.DIR. (INCLUDE and EXCLUDE statements are normally read from the bottom up, though that did not appear to be the case with EXCLUDE.DIR.)

Our workaround, as shown in Example 23-6 on page 482, was to use EXCLUDE instead of EXCLUDE.DIR for directories that might also have INCLUDEed content.

Once you've defined your includes and excludes in the dsm.sys file, you can then use standard TSM dsmsc commands to either back up or restore files, as described in "Using the TSM Client" on page 478.

## TSM restore based upon file dates and times

You can also perform a selective TSM restore that is based upon the state of files at a particular date and time, as follows:

```
dsmc restore -pitd=dd/mm/yyyy -pitt=hh:mm:ss -password=<your_password>  
<files_to_be_restored>
```

For example, we used the command:

```
# dsmc restore -pitd=06/01/2001 -pitt=13:00:00 -password=vmlinux3  
/home/vmlinux3
```



## DB2

Every operating system needs a professional database, to either access the data structures on another platform, or to store the data on the system itself—especially in an e-business environment.

The Open Software Community offers a product called MySQL, which enables basic database functionality, but most likely is not sufficient to run enterprise-based transaction systems on Linux on S/390 or zSeries.

To satisfy the requirement for reliability, high availability and performance, IBM offers the database management software DB2 Universal Database (UDB). In this chapter, we discuss DB2 UDB and Linux.

## 24.1 DB2 UDB overview

Because UDB is basically the same software as DB2 on OS/390 (now z/OS), it is especially useful for customers who keep their main data on OS/390 systems. It allows them perfect cross-system interoperability and reuse of the database know-how in the company. The application can decide whether the data should be stored on a central OS/390 system, or whether it should be kept on the more decentralized Linux on S/390 environment. You can run your application using open software technology on Linux, and use the superior systems management tools on OS/390.

With the DB2 Connect software available with UDB on Linux, you can access a remote DB2 on OS/390 as though it were local. If you consider the future availability of Hipersockets between the Linux on S/390 and OS/390 on the same box, this will allow you a huge bandwidth transfer rate between application and database server.

Customers who want to use data stored in the IBM Information Management System (IMS) database system can use the new IMS Connect feature to access the remote system via the TCP/IP network.

For more information on DB2 Universal Database and the DB2 Connect product, go to the following sites:

<http://www.ibm.com/software/data/db2/linux>

<http://www.ibm.com/software/data/db2/linux/s390>

## 24.2 Downloading DB2 UDB for Linux on S/390

IBM offers a 90-day trial copy of DB2 UDB EE (Enterprise Edition) for Linux for S/390 and zSeries on the Internet, available at:

<http://www6.software.ibm.com/d1/db2udbd1/db2udbd1-p>

Download DB2 UDB EE on Linux for S/390 and zSeries V7.1 for Linux. You can also select the required language for the software product. After registering yourself and obtaining a user ID/password, you can sign in and start the actual download.

IBM will ask you to answer a few questions about yourself and your installation before you can accept the license. Be aware that the complete product download will be between 57 MB and 80MB, and may take some time to complete.

## 24.3 Installing DB2 UDB on Linux

In our project, we used a Red Hat 7.0 beta version for our DB2 UDB installation. Unfortunately, this caused some problems which we expect will be solved by the time Red Hat for Linux on S/390 goes GA.

To complete the installation we had to install an additional component from the SuSe distribution, which is not really a good solution. But the fact that it worked afterwards made us confident that you will not have these kind of problems if you use the SuSe distribution instead of Red Hat.

Following is a list of the steps and problems we encountered during the installation:

1. Copy the `linuxee.tar` file to your Linux on S/390 system

If you did not do the download directly to the target system, you could use FTP, SMB, or NFS to binary-transfer the file. In our case, we just used one of our home directories to store the package.

2. Untar the file using the `-xvf` option to expand the directory structure:

```
# cd /home/eterwedo
# tar -xvf linux390.english.tar
```

3. Remove existing user IDs.

In some distributions, the user IDs for UDB/DB2 are already set up but they do not always match the specifications required for a successful installation.

Remove all user IDs starting with `db2*` using `yast` or any other setup tool.

4. We next followed the installation instructions and ran the **db2setup** script:

```
# ./db2setup
```

We got the following error only during our first test with a Red Hat distribution. (If you do not encounter this error in your distribution, just skip to the next step):

```
./db2inst: error while loading shared libraries:
libstdc++libc6.1-2.so.3: cannot load shared object file: No such file or
directory
```

We found out that this error was caused by Red Hat including a newer `libstdc` package into the distribution, as required by DB2 UDB:

```
# rpm -qa | grep libstdc
libstdc++-2.95.3-0.07
# rpm -q1 libstdc++-2.95.3-0.07
/usr/lib/libstdc++-3-libc6.2-2-2.10.0.a
/usr/lib/libstdc++-3-libc6.2-2-2.10.0.so
```

The version included by Red Hat is libc6.2, while DB2 UDB required libc6.1. To solve this, we loaded the gppshare-2.95.2-165 RPM from the SuSe distribution. Unfortunately, this required us to completely download the cd1 Suse ISO image and mount it on a Linux file system:

```
# rpm -i /usr/src/gppshare-2.95.2-165
```

Afterwards we had the required libraries on our system:

```
# rpm -ql gppshare-2.95.2-165
/usr/lib/libg++-3-libc6.1-2-2.8.1.3.a
/usr/lib/libg++-3-libc6.1-2-2.8.1.3.so
/usr/lib/libg++-libc6.1-2.a.3
/usr/lib/libg++-libc6.1-2.so.3
/usr/lib/libstdc++-3-libc6.1-2-2.10.0.a
/usr/lib/libstdc++-3-libc6.1-2-2.10.0.so
/usr/lib/libstdc++-libc6.1-2.a.3
/usr/lib/libstdc++-libc6.1-2.so.3
```

Therefore, we could run **db2setup**.

#### 5. Select setup options.

Follow the directions in the setup panel. You can use the Tab key to skip from one selection field to the next, the space key to select/deselect, and the Enter key to execute a selection.

We selected all components offered for the trial version:

```
[*] DB2 Run-Time Client           [ Customize... ]
[*] DB2 UDB Enterprise Edition   [ Customize... ]
[*] DB2 Connect Enterprise Edition [ Customize... ]
[*] DB2 Application Development Client [ Customize... ]
```

#### 6. Select options to create the DB2 Instance and Administration Server.

By default, the home directory of the instance will be located in the /home directory. Modify the entry if you want to place it somewhere else.

#### 7. Run the installation.

#### 8. Check the installation log file.

This log file can be found in /tmp/db2setup.log. Everything should install successfully.

The product will be in the /usr/IBMDB2/V7.1/ Linux directory.

## 24.4 Verifying the UDB DB2 installation

The database instance is automatically started as part of the installation process.

1. Logon to the Linux system using the db2inst1 user ID.

To verify the successful installation, use a telnet session and login with the user ID `db2inst1`. (Note that this is the default; if you changed this user ID during the installation process, then you must use the user ID you specified.)

The default password for `db2inst1` is `ibmdb2`. (It's either this, or the one you specified during installation.) In case of a problem, you can always reset the password using your root account and the `passwd db2inst1` command.

2. Verify that DB2 is running.

You can check if DB2 is already running by entering the `db2start` command. The following message indicates that the database is already started:

```
SQL11026N The database manager is already active.
```

3. Create the sample database.

Enter the `db2samp1` command to create the sample database.

4. Connect to the sample database.

You can connect to the sample database by using the DB2 command line interface. Enter the following:

```
$ db2 connect to sample
Database Connection Information
Database server      = DB2/LINUX 7.1.0
SQL authorization ID = DB2INST1
Local database alias = SAMPLE
```

5. Issue a sample query to verify the installation:

```
$ db2 "select * from department"
```

DEPTNO	DEPTNAME	MGRNO	ADMRDEPT	LOCATION
A00	SPIFFY COMPUTER SERVICE DIV.	000010	A00	-
B01	PLANNING	000020	A00	-
C01	INFORMATION CENTER	000030	A00	-
D01	DEVELOPMENT CENTER	-	A00	-
D11	MANUFACTURING SYSTEMS	000060	D01	-
D21	ADMINISTRATION SYSTEMS	000070	D01	-
E01	SUPPORT SERVICES	000050	A00	-
E11	OPERATIONS	000090	E01	-
E21	SOFTWARE SUPPORT	000100	E01	-

9 record(s) selected.





# WebSphere Application Server

The WebSphere Application Server (WAS) is a Java 2 Enterprise Edition-based e-business application runtime environment. It's available on almost all Java-capable platforms.

The Standard Edition allows you to run Java servlets according to JavaSoft V.2.11 and JavaServer Pages written to JavaSoft V.10 or .91. The Advanced Edition, which is available for Linux for zSeries, adds the full Enterprise Java Beans (EJB) standards. The Enterprise Edition contains all features of the standard and advanced edition, and includes also the object management features of the IBM Component Broker technology.

The current version on most platforms is WebSphere V.4.0, which combines the WebSphere 3.5 Standard Edition and Enterprise Edition into a single product. The only available WebSphere product for Linux for zSeries is Version 3.5 Advanced Edition. The installation of this product is limited to the SuSe distribution. We tried to run the installation under Red Hat, but this failed due to incompatibility between Java and the Red Hat kernel. We recommend to use the SuSe distribution until IBM changes or enhances their recommendation to Red Hat or Turbolinux.

**Note:** Based on our experience, WebSphere needs at least 512 MB of physical memory; otherwise, it will not be usable at all.

## 25.1 Downloading the evaluation version of WebSphere

IBM offers an evaluation version of WebSphere 3.5 for Linux for S/390 on the Web at:

[http://www.ibm.com/software/webservers/appserv/download\\_linux\\_s390.html](http://www.ibm.com/software/webservers/appserv/download_linux_s390.html)

It's currently limited to the installation in a SuSe distribution, and includes following components:

- ▶ IBM Developer Kit, Java 2, Technology Edition, 1.2.2
- ▶ IBM HTTP Server 1.3.12
- ▶ InstantDB

InstantDB will allow you to use WebSphere in your environment even if you don't have a database installed. We recommend that you use the DB2/UDB product for Linux for zSeries instead. This will give you a full-function database and it will be a more productive environment. The installation and use of this product is described in Chapter 24, "DB2" on page 485.

For the download you have to register for the IBM Download Zone. (Remember this user ID/password because you'll need it later on for further downloads and you are repeatedly asked for this information.)

## 25.2 Installing WebSphere

You install WebSphere by following these steps:

### ***Unload the tar file***

After you've transferred the tar file in binary to your Linux for zSeries system, you have to untar the file to run the installation script. The tar file contains its own directory structure, and it will create a new directory by the name of /advtrial during the reload process:

```
# tar -xvf ../download/adv_trial_was3.5_s390.tar
```

You will use this unload directory only for installation. The actual product will be installed under the /opt directory.

### ***Stop the Web server***

During the installation, you'll be offered the option of using either the IBM HTTP Server, or customizing your already-installed Apache Web server.

We tried both options, and make the following recommendations:

- ▶ If you do not need to keep your WebSphere application work separate from the normal Web server environment, you can use the standard Apache server.
- ▶ If you decide to install and use the IBM HTTP Server for WebSphere, and still keep your existing environment running, you should move your distribution Web server to another port number.
  - a. To do this, open up your Web server configuration file (for SuSe this is `/etc/httpd/httpd.conf`) and change the directive **Port** to another number, e.g. **Port 8080**.
  - b. Stop and restart your Web server and verify that the standard port number 80 is now free to use for the IBM HTTP server. Running the installation script with an active Web server will cause problems.

### ***Modify the install.sh script (req. only for IBM HTTP Server)***

We noticed that the latest version of the WebSphere evaluation copy had a conflict with a file and package that is a basic part of the SuSE distribution. Although this does not really cause problems (WebSphere is trying to create a directory which is already allocated as a symbolic link), you have to add the `--force` option to the `rpm` command for the IBM HTTP Server.

Use your favorite editor to change the rpm line in `install.sh` as follows:

```
rpm -Uvh --force ./IBM_HTTP_Server/*/IBM_HTTP_Server-1.3.12-*.rpm
```

If you run the install script without this modification and you get the rpm error message, we strongly recommend that you uninstall all rpms using YaSt and start from scratch. The unsuccessful installation of the IBM HTTP Server causes the script to skip several important steps in the installation of other products, and fixing these problems individually requires an in-depth understanding of Linux shell scripts and rpm processing.

### ***Run the install script***

In this example, we install the IBM HTTP Server. (Note that answering 2.) No to the question `Install and use IBM HTTP Server?` will modify the installation process slightly.)

```
# ./install.sh
#####
Installing pre-requisites: Java: Version 1.2.2
#####
IBMJava2-SDK #####

Install and use IBM HTTP Server? [default is yes]
1.) Yes
2.) No
1
```

Installing IBM HTTP Server Version 1.3.12.2

```
#####
Select the database you wish to use and press <ENTER>.
1.) DB2 Quick Install [default]
2.) Custom
2
Custom Selected
#####
Select the database you wish to use and press <ENTER>.
1.) DB2
2.) Oracle
1
DB2 Selected
#####
```

WebSphere Custom Repository configuration initiated.

Please answer the following questions. Your responses will be used to configure WebSphere Application Server for your database installation.

What is your database user id?  
database user id:**db2inst1**

Database user: db2inst1  
What is your database user password?  
database user password: **start**

Database user password: start  
Where is the base database directory/home so the JDBC driver can be located?

Default is: /home/db2inst1/  
Database Base Directory:  
**/home/db2inst1**

What is your JDBC database url?  
The default is: jdbc:db2:was  
Note: DB2 URL format: jdbc:db2:<dbname> Oracle format:  
jdbc:oracle:thin:@<hostname>:<port>:<sid>  
**jdbc:db2:was**

Database JDBC URL:

JDBC URL: jdbc:db2:was  
#####

Installing IBM WebSphere Application Server for Linux

```

IBMWebAS
#####
A copy of the configuration file was made and is located at
/opt/IBMWebAS//properties/bootstrap.properties.bak
A copy of the configuration file was made and is located at
/opt/IBMWebAS/bin/admin.config.bak
A copy of the configuration file was made and is located at
/opt/IBMWebAS/bin/startupServer.sh.bak
A copy of the configuration file was made and is located at
/opt/IBMWebAS/properties/initial_setup.config.bak
IBMWebAS-ADV-doc-en
#####

```

IBM WebSphere Application Server for Linux installation is complete. Please run `startupServer.sh` in `/opt/IBMWebAS/bin` to invoke the server. Also, please restart your web server if you selected IHS at this time to ensure the plug-in configuration is updated. If you installed with Apache, please refer to the install documentation for configuration information.

```
#####
```

## 25.3 Upgrading WebSphere to Fixpack Level 3.5.4

You can find the latest support information about WebSphere on all platforms, as well as a link to download fixpacks, on the Web at:

<http://www.ibm.com/software/webservers/appserv/support.html>

IBM provides several fixpacks for WebSphere on all supported systems. At the time of writing this rebook, the latest fixpack available is level 3.5.4. You always have to use the Linux for S/390-specific version of the fixpack. Non-S/390 Linux versions will not work on this platform.

Reply to all authorization requests with your download user ID and password, which you registered during the download of the WebSphere for Linux for S/390 evaluation copy.

1. Binary-transfer the file to Linux.
2. Unpack the fixpack file.

Create an empty directory and untar the file with:

```
tar -xvf ../download/was35_adv_ptf_4_LINUX.s390.tar
```

Read the `was35_adv_ptf_3.readme` file for the latest information.

3. Stop the WebSphere Server and the Web Server (if active).

The WAS server can be stopped by using the Administration Client. Select your host name and click the STOP icon. This will also automatically terminate your Administration Client.

The Web Server can be stopped by issuing:

```
Apache Server = /etc/rc.d/apache stop
IBM HTTP Server = /opt/IBMHTTPServer/bin/apachectl stop
```

4. Add the JVM to your PATH.

To allow the script to use the java JVM from the command line, you must add the java bin directory to your PATH:

```
PATH=/opt/IBMWebAS/IBMJava2-122/bin:$PATH
```

The following command will confirm that you can now use the JVM:

```
java -fullversion
java full version "J2RE 1.2.2 IBM build c390_x122-20001209"
```

5. Run the installation script:

```
./install.sh
```

During the installation, you will be asked to enter the WebSphere installation directory /opt/IBMWebAS and the Web server root directory:

```
Apache Server = /usr/local/httpd/htdocs
```

```
IBM HTTP Server = /opt/IBMHTTPServer/htdocs/en_US.
```

For information how to uninstall the package, refer to the readme file in step 2.

## 25.4 Starting the Web server

Some of the customization is only required for the Apache Web server, because the IBM HTTP server is automatically configured for the use in a WebSphere environment. In the following procedure, ensure that you always use the correct file names and directories for your specific configuration.

1. Copy the WebSphere sample pages.

Before we can start the Web server we have to copy some WebSphere-related example pages to the Web server document root directory.

**For the Apache server:**

```
cp -r /opt/IBMWebAS/WSsamples /usr/local/httpd/htdocs
cp -r /opt/IBMWebAS/WSsamplesIDB /usr/local/httpd/htdocs
```

### For the IBM HTTP Server:

```
cp -r /opt/IBMWebAS/WSsamples /opt/IBMHTTPServer/htdocs/en_US
cp -r /opt/IBMWebAS/WSsamplesIDB /opt/IBMHTTPServer/htdocs/en_US
```

#### 2. Update the Web server configuration file.

To configure the Apache Web server, and for an error-free presentation of the example Web pages, we have to update the Web server configuration file.

For Apache, update `/etc/httpd/httpd.conf` with the following:

```
LoadModule app_server_module /opt/IBMWebAS/bin/mod_app_server.so
AddModule mod_app_server.c
Alias /IBMWebAS/ /opt/IBMWebAS/web
NcfAppServerConfig BootFile
/opt/IBMWebAS/properties/bootstrap.properties

Alias /theme/
"/opt/IBMWebAS/hosts/default_host/WSsamples_app/web/theme/"

<Directory "/opt/IBMWebAS/hosts/default_host/WSsamples_app/web/theme/">
    Options Indexes MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
```

For the IBM HTTP server, update `/opt/IBMHTTPServer/conf/httpd.conf` with the following:

```
Alias /theme/
"/opt/IBMWebAS/hosts/default_host/WSsamples_app/web/theme/"

<Directory "/opt/IBMWebAS/hosts/default_host/WSsamples_app/web/theme/">
    Options Indexes MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
```

#### 3. Start the Web server.

The following command will check whether the IBM HTTP Server is already started and stop/restart if necessary. If it is currently down, it will simply issue a start command.

```
Apache server = /etc/rc.d/apache restart
IBM HTTP server = /opt/IBMHTTPServer/bin/apachectl restart
```

## 25.5 Starting the WebSphere Application Server

Starting the WebSphere Application Server will take some time, so be aware of this and resist the temptation to interrupt the process. To begin, enter the following start command:

```
# /opt/IBMWebAS/bin/startupserver.sh &
```

The ampersand (&) at the end of the command forces the process to start in the background, and it will free up your normal shell session. If you omit this, you can still free up your session with <Ctrl>c. This will not interrupt the processing of your WebSphere server.

During the startup process, WebSphere writes a number of messages to the `/opt/IBMWebAS/logs/tracefile`. You can check this file for errors. When you receive the message `The WebSphere Administration Server is open for e-business`, it indicates the server started successfully. You can now use the Advanced Administration Client to control your WebSphere environment.

## 25.6 Test the WebSphere Application Server

Note that the Advanced Administration Console is only required to control the WebSphere runtime environment. It can be used to perform following functions:

- ▶ Stop the WebSphere Application Server
- ▶ Stop and start the servlet engine
- ▶ Enable, disable and configure individual servlets
- ▶ Add/delete servlets and Web applications
- ▶ Control JDBC Connection and Data Sources
- ▶ Control the interface to the Web server

To test the WebSphere Application Server, perform the following steps.

1. Start the WebSphere Administration Console.

Note that you must use the console the first time to start the default servlet engine and to do some customization, but it's not required for a normal runtime environment.

You need an XWindow-based workstation to run this Administration Client, so we recommend you run it from a KDE workstation—or at least have an XServer (like Exceed) active.

To allow your telnet session to communicate with your workstation, you must set the `DISPLAY` environment variable, e.g. `export DISPLAY=9.12.2.116:0`, to the IP address of the workstation.



You start the WebSphere Administration Console with the following command:

```
# /opt/IBMWebAS/bin/adminclient.sh&
```

This will give you a graphical interface on your XWindow session. (Again, this may take some time.) If you use a KDE session, you can run the `xosview` command to monitor the system activity, to verify that your system is busy processing your request.

The only alternative to using the WebSphere Administration Console from the XWindow system is—install it on another platform and connect to your application server remotely. We tried this with a Windows 2000 system and it worked fine. The only issues are:

- You may need an additional license.
- You must have an exact match of WebSphere versions, and both platforms must run on the same fixpack level.

Both types of administrative consoles will give you exactly the same kind of functionality.

## 2. Start the Default Server.

- Expand the topology view by clicking the plus (+) sign next to the Administrative Domain.
- Expand the entry with your machine name.
- Expand the entry Default Server to display the default container and servlet engine.
- Select the Default Server.
- If the current status is stopped, then click the START icon on the tool bar, or right-click the server for the menu START option. Wait until you get a confirmation message and all icons show a little blue wheel, representing an active server.
- You can now expand the entries Default Server, Default Servlet Engine and default\_app or examples to see the installed servlets.
- By clicking each of the examples, you can see how to call them from a Web browser. The URL information is in the Servlet Web Path List.

## 3. Run a few examples from a browser.

```
http://vmlinux6.itso.ibm.com/servlet/snoop
http://vmlinux6.itso.ibm.com/servlet/hello
http://vmlinux6.itso.ibm.com/webapp/examples/simpleJSP
http://vmlinux6.itso.ibm.com/webapp/examples/ping
http://vmlinux6.itso.ibm.com/webapp/examples/showCfg
http://vmlinux6.itso.ibm.com/webapp/examples/HitCount
```

If these examples only work without the full-qualified domain name (FQDN), for example `http://vmlinux6/servlet/snoop`, then follow the instructions in step 4 to modify the Virtual Host alias entry.

4. Modify the Virtual Host alias (if required).

Add a virtual host alias for the fully-qualified domain name (FQDN) of your machine running the Web server (the short name is automatically defined, but some systems do not automatically define the FQDN):

- Go to the Topology view of the Administrative Console.
- Select **default\_host**.
- Edit the Advanced properties panel and add a fully-qualified virtual host alias, including domain name.
- Restart the Default Server Application Server.

## 25.7 Setting up the WebSphere Samples Gallery

IBM delivers a complete set of examples to run database accesses and enterprise java beans in your new WebSphere environment. However, to do this you have to install at least the InstantDB on your server. We ran all the tests with a full UDB DB2 installation.

You will find the Samples Gallery on your Web server on URL:

```
http://<your_server_name>/WSsamples
```

Figure 25-1 on page 501 shows the Samples Gallery page.

Although the explanations on the Web pages are quite useful, in this section we provide additional detail which will allow you to work on the WebSphere Administration Console without having to switch back to your Web browser session.



Figure 25-1 WebSphere Samples Gallery page

## 25.7.1 Configuring the database

First you have to configure the database. To do this, click the **Database configuration** link and choose **Start DB2 Database configuration**.

You'll see the screen shown in Figure 25-2 on page 502.

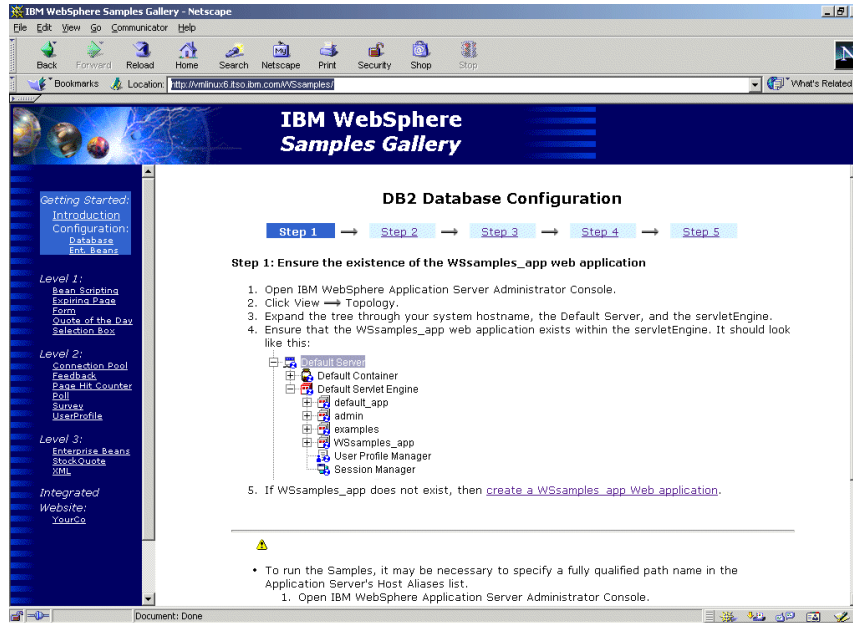


Figure 25-2 DB2 Database Configuration Step 1

At this point, you should have already added the fully qualified host name into the Application Server Host Alias list, as described in “Modify the Virtual Host alias (if required).” on page 500.

After you follow the instructions listed in Figure 25-2, click **Step 2**. You’ll see the screen shown in Figure 25-3 on page 503, from which you can create the SAMPLE Database.

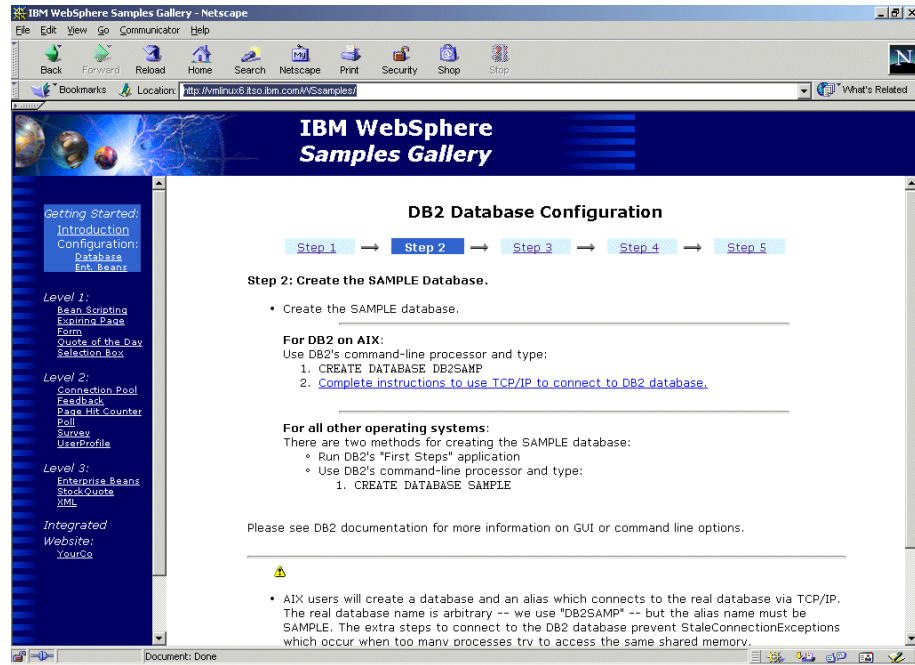


Figure 25-3 DB2 Database Configuration Step 2

If you've followed the instructions in "Create the sample database", everything should already be in place and you can continue by clicking **Step 3**.

You'll see the screen shown in Figure 25-4 on page 504, from which you can create a DB2 database user for SAMPLE.

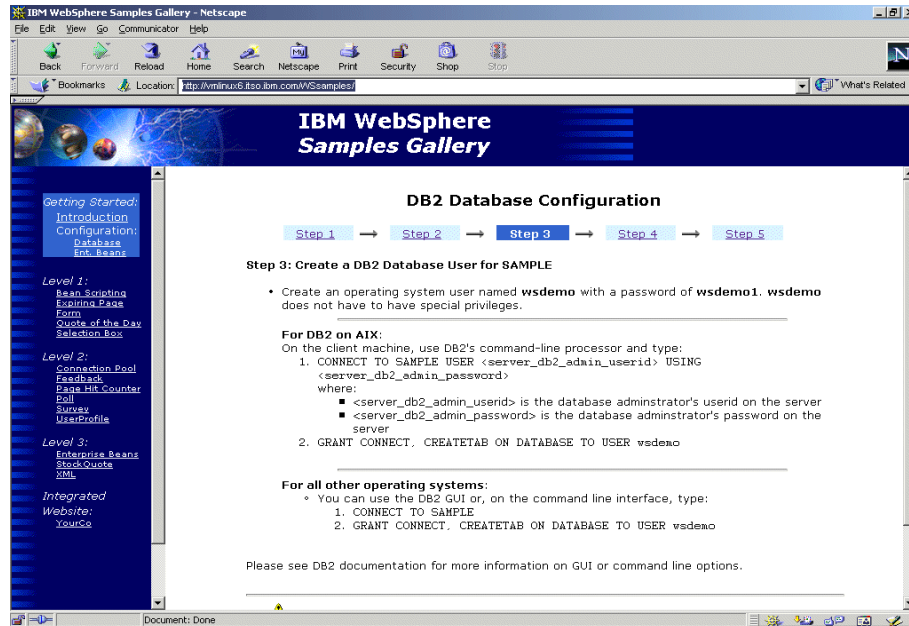


Figure 25-4 DB2 Database Configuration Step 3

To allocate a new user ID, you have to logon as root to your Linux for zSeries system and use **Yast -> System Administration -> User Administration -> PF4 Create User**.

To issue DB2 commands, you should logon as db2inst1 and use the **db2** command. For more information about using db2, see "Verifying the UDB DB2 installation" on page 488.

After you complete the instructions, click **Step 4**. You'll see the screen shown in Figure 25-5 on page 505, from which you can create a DataSource Object.

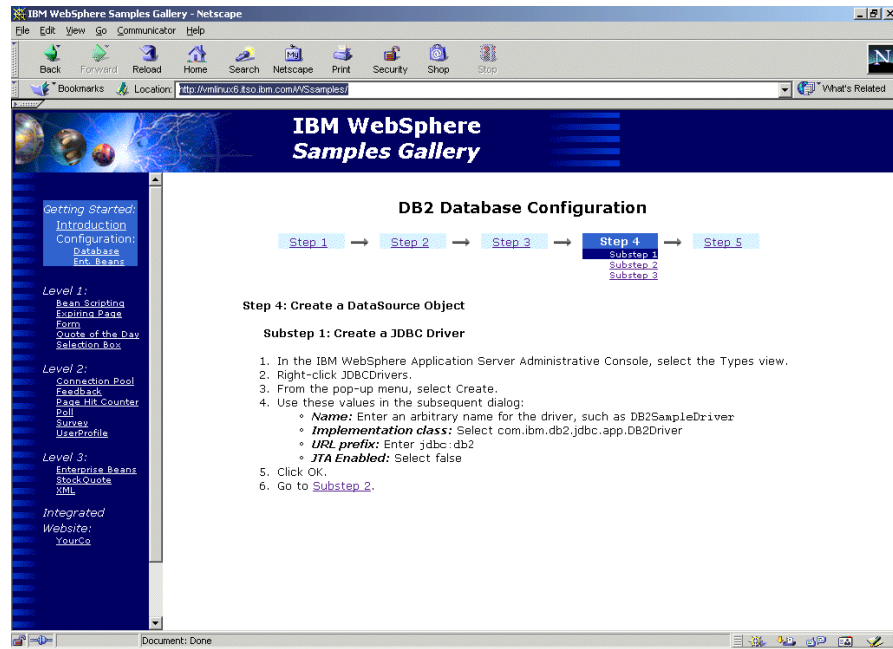


Figure 25-5 DB2 Database Configuration Step 4 - Substep 1

Follow the instructions on the page and then click **Substep 2**. You'll see the screen shown in Figure 25-6 on page 506, from which you define the DataSource.

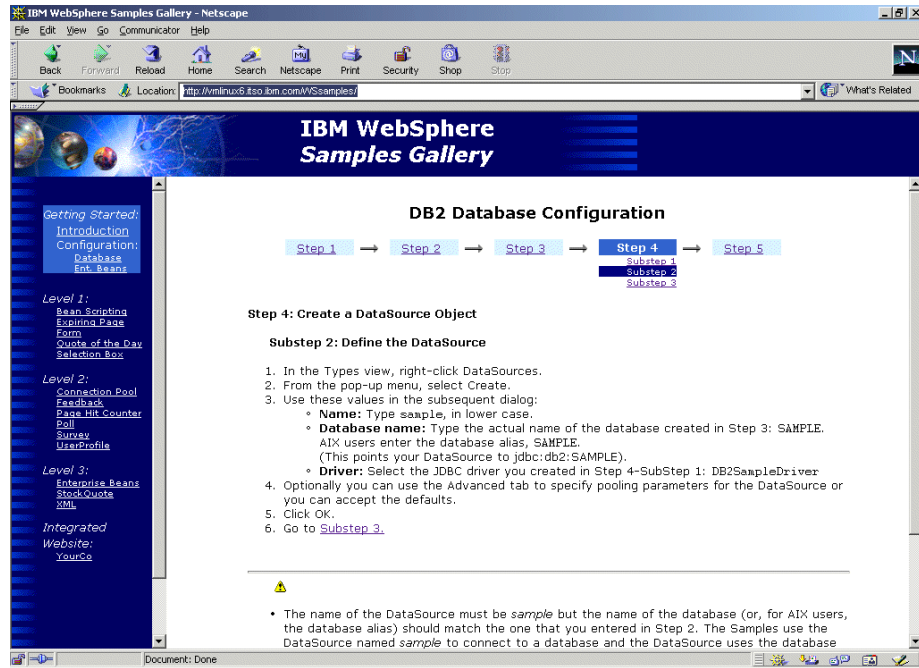


Figure 25-6 DB2 Database Configuration Step 4 - Substep 2

Follow the instructions and enter the Database Name `SAMPLE` in upper case letters. Then click **Substep 3**.

You'll see the screen shown in Figure 25-7 on page 507, from which you can install the driver.



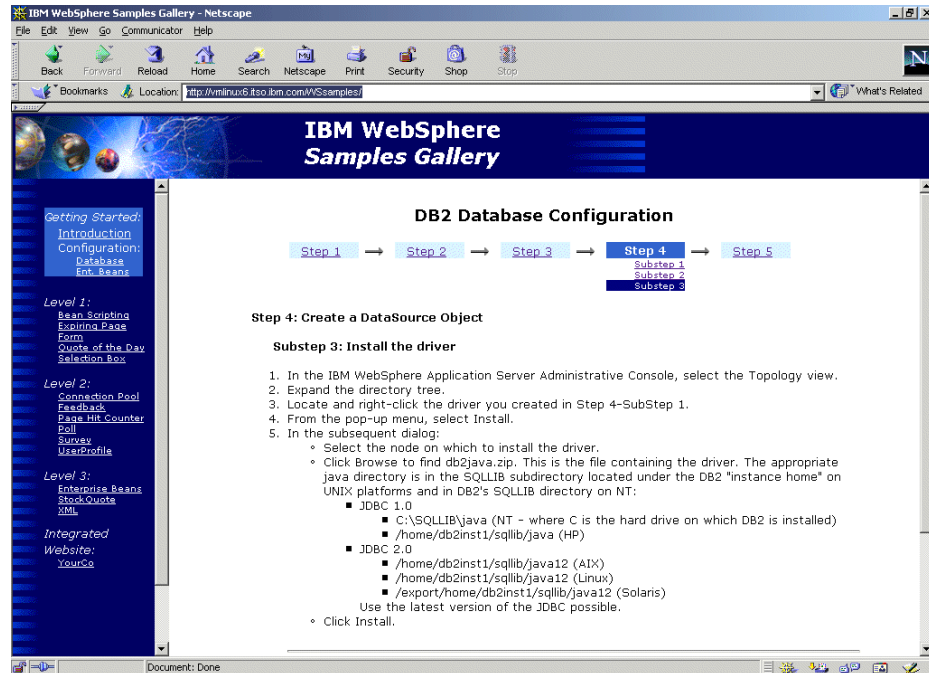


Figure 25-7 DB2 Database Configuration Step 4 - Substep 3

To start the connection, you may need to stop and start the Default Server in the Application Server Administrator Console. To do this:

1. Select the Topology view.
2. Expand the tree until the Default Server is displayed.
3. Right-click the Default Server and choose **Stop**; the wheel icons indicating the server's status should be red and white.
4. Once the server is stopped, right-click again and choose **Start**; the wheel icons indicating the server's status should be blue and white.

Afterwards, click **Step 5**.

You see the screen shown in Figure 25-8 on page 508, from which you can run the database servlet.

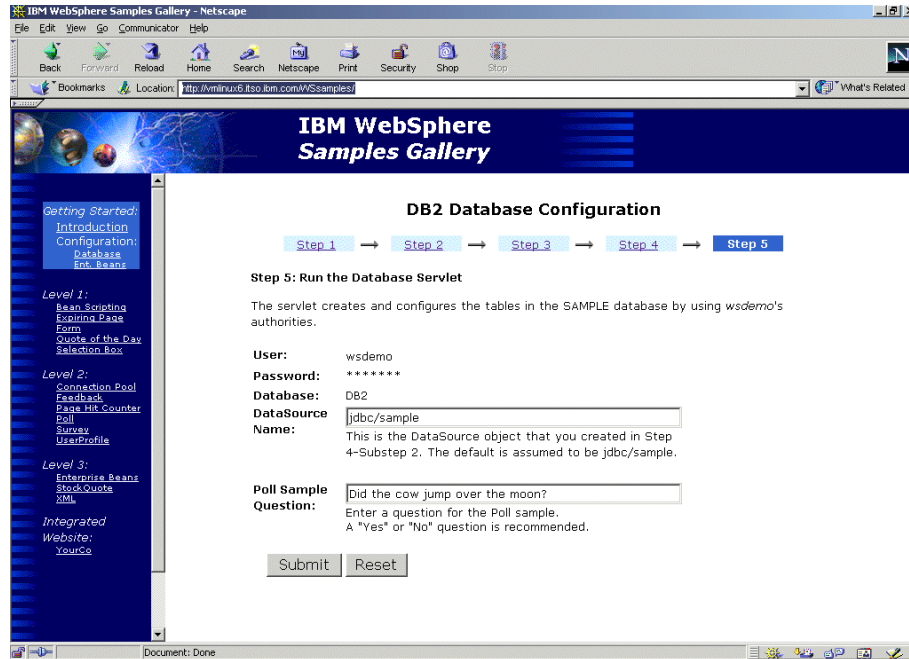


Figure 25-8 DB2 Database Configuration Step 5

Click **Submit** to run the example. Your WebSphere Application should now be able to connect successfully to the UDB DB2 database on your Linux system.

## 25.7.2 Configuring Enterprise Java Beans

If you want to run the complete Samples Gallery, you also have to configure your system for Enterprise Java Beans. To do this click, the **Enterprise Beans** link on the left side of the Getting Started - Configuration frame. Select **Start DB2 Enterprise Beans Configuration**.

You'll see the screen shown in Figure 25-9 on page 509, from which you can create a database user.

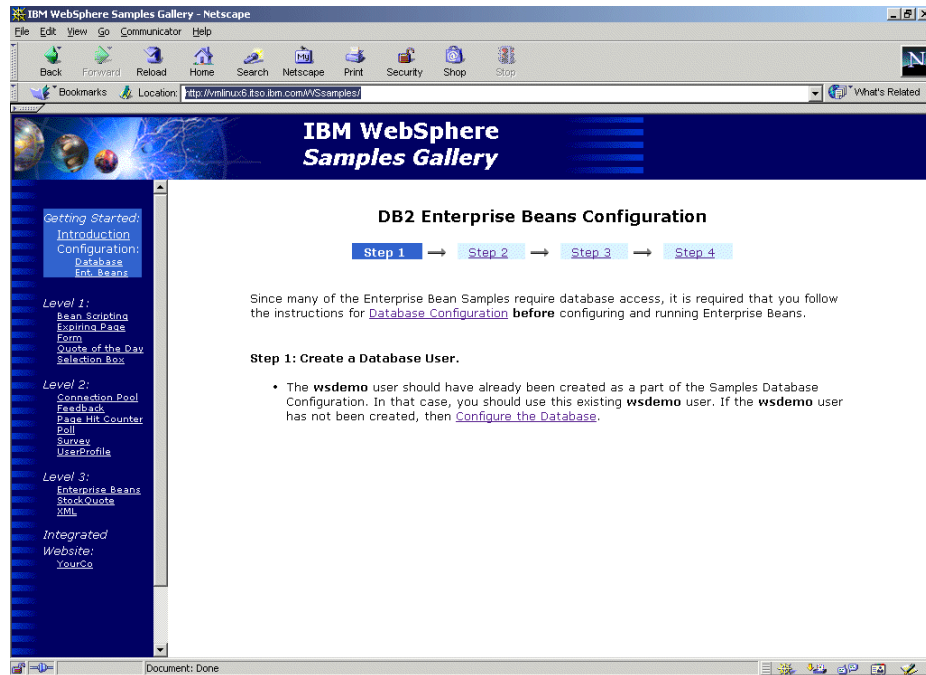


Figure 25-9 DB2 Enterprise Beans Configuration Step 1

This is a reminder that you have to perform the Configure Database step first, before you can start DB2 Enterprise Beans Configuration. Just click **Step 2** for confirmation.

You'll see the screen shown in Figure 25-10 on page 510, from which you can specify a default data source for the EJB Container.

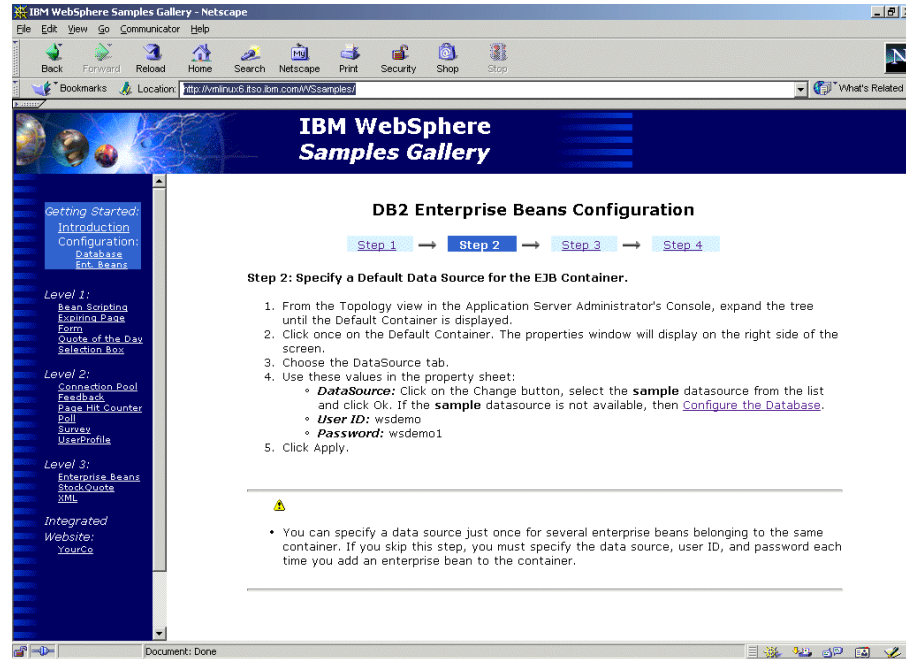


Figure 25-10 DB2 Enterprise Beans Configuration Step 2

Follow the instructions on this page and click **Step 3**.

You'll see the screen shown in Figure 25-11 on page 511, from which you can install and deploy the Enterprise Beans.

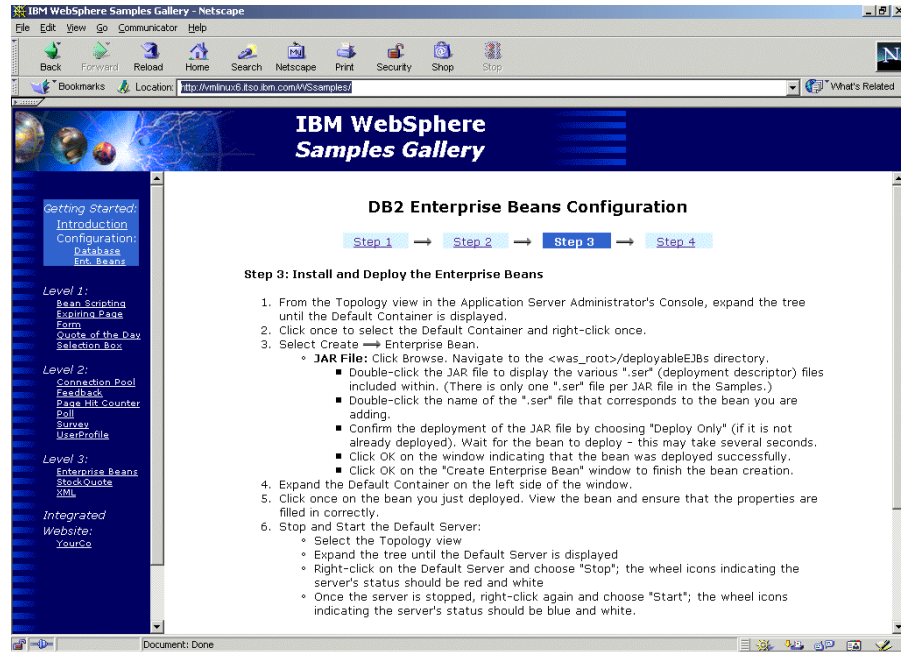


Figure 25-11 DB2 Enterprise Beans Configuration Step 3

**Note:** When you browse the Enterprise Bean directory for the first time, you will be positioned to the directory **deployedEJBs**. Do *not* try to deploy any of these beans again, because it will seriously damage your environment.

Instead, use the **Up-on-Level** icon and open the **deployableEJBs** directory. This step is documented on the page, but it can be easily overlooked.

To run all samples, you have to repeat the process for all beans in the **deployableEJBs** directory. Afterwards, click **Step 4**.

You'll see the screen shown in Figure 25-12 on page 512, from which you can configure the YourCo SAMPLE database.

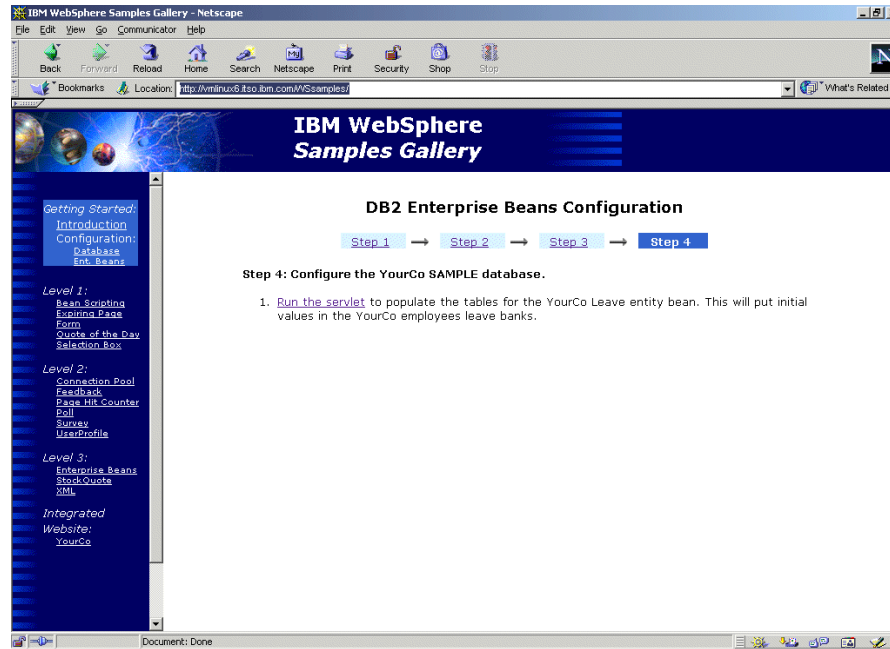


Figure 25-12 DB2 Enterprise Beans Configuration Step 4

Now you've almost completed the configuration procedure. After clicking **Run the servlet**, you'll receive a confirmation and you can run all the example servlets from the left side of the frame.



## Conclusion

In this chapter we briefly compare the Linux for zSeries distributions that we worked with. As stated in the overview, we do not specifically recommend a single distribution. It is difficult to draw conclusions from so many aspects of the many distributions we tested.

Table 26-1 compares the three strategic Linux distributions.

*Table 26-1 Linux for zSeries and S/390 distribution comparison*

Attribute	SuSE	Turbolinux	Red Hat
Linux kernel	2.2.16	2.2.16 (v6.0)	2.2.19-0.07 2.4.3-2.14.3 (optional)
Distribution media	3 CDs	1 CD	Web
CDs free on Web?	Yes <b>1</b>	Yes <b>2</b>	No ISO CD images, RPMs only <b>3</b>
Number of RPMs	1,068 Binary 689 Source	518 Binary	945 Binary 45 Source (as of June 12, 2001)
LVM support?	Yes	No in v6.0, Yes in v6.5	No
Journalled file system support?	No	No in v6.0, Yes in v6.5	No

Attribute	SuSE	Turbolinux	Red Hat
Primary system management tool	YaST	None, Webmin is a good candidate	linuxconf
Notes: <b>1:</b> URL is ftp://ftp.suse.com/pub/suse/s390/suse-us-s390/ <b>2:</b> URL is ftp://ftp.turbolinux.com/pub/product/s390/ <b>3:</b> URL is ftp://ftp.redhat.com/pub/redhat/linux/rawhide/s390			

## 26.1 Observations

In this section we list our high-level observations that resulted from this project:

- ▶ All of the six distributions we worked with installed and ran well on the z900. Those based on the Linux 2.2 kernel (the majority) were more stable than those that could run with the Linux 2.4 kernel.
- ▶ Tape support was still not built in to Linux for zSeries at the time of writing, though Turbolinux v6.5 will probably have it built in. The tape driver proved to be stable for us, but we did not exercise it heavily.
- ▶ While there are many journaled file systems that are candidates to become the de facto standard on Linux for zSeries, none were enterprise-ready as of July 2001.
- ▶ Webmin is a dependable and usable general purpose system administration tool.
- ▶ It is advisable to keep the memory size of Linux systems as small as possible (64 MB or even less). However, some Linux guests need large memory sizes (up to 512 M for WebSphere). Running under z/VM, this seemed to cause hangs when the guests were being paged out on system. Modification to the z/VM storage buffers resolved the problem. We modified the following:
 

```
STORBUF: Q1=125% Q2=105% Q3=95%
```

 to:
 

```
STORBUF: Q1=375% Q2=315% Q3=285%
```
- ▶ Though it is generally agreed that zSeries hardware should not be used to drive graphical applications, KDE2 is useful for displaying a graphical desktop from Linux for zSeries to your desktop.
- ▶ VIF is useful as a platform for creating and maintaining multiple Linux images. However, it has limitations, especially because of its limited function (tuning tools, tape support, etc.) and IBM's statement that it will not enhance it.



As such, z/VM is recommended for customers who need capabilities beyond those provided by VIF, and have (or can develop) the skills necessary to manage the more full-function environment it provides.





# Commands and other references

In this appendix, we concisely list the syntax of commonly used Linux commands and the layout of commonly used files

## Command syntax

Following are simple examples of some commonly used Linux commands

```
# insmod dasd dasd=201-203
# dasdfmt -b 4096 -y -f /dev/dasdc
# mke2fs -b 4096 /dev/dasdc1

# ifconfig iucv0 10.1.1.1 pointopoint 10.1.2.1 mtu 1492 up
# route add default gw 10.10.2.1
# inted &

# mkdir /suse/ /suse/cd1 /suse/cd2 /suse/cd3
# mount -o loop /linuxSystems/suse/suse-us-s390-cd1.iso /suse/cd1
# mount
# umount /suse/cd1

# cd /boot
# silo -b ./ipleckd.boot -d /dev/dasda -f ./image -p ./parmfile -t2
```

```
# df -h
# rpm -q package
# cat /proc/dasd/devices
# cat /proc/meminfo
# cat /proc/cpuinfo
```

## VIF commands

Following is a quick reference to VIF commands (and we assume you have written a `v` script as described in 16.6.1, “Writing a simple `v` script” on page 288). This is not the complete list of VIF commands, but rather is a large subset. For a complete description of these commands, refer to *S/390 Virtual Image Facility for LINUX Guide and Reference*, GC24-5930.

## Query commands

```
v q all           // query everything
v q net          // IUCV network
v q boot        // query uptime
v q conf        // answers to install questions
v q vol         // query volumes
v q par         // query partitions
v q act         // active images
v q level       // VIF release level ('about VIF')
v q image 'name' // query a linux image by name
```

## HYPervisor commands

```
v hyp errors                // write ./ereprept.yyyymmdd
v hyp install 'IP@' 'user' 'pw' 'path' // replace kernel, RAMdisk, parmfile
v hyp net add 'name' 'IP@' 'mask' // add an IUCV connection
v hyp restart              // restart hypervisor and all images
v hyp shutdown            // shutdown VIF (careful!)
v hyp ver full            // verify hypervisor environment
v hyp vol add image 'addr' 'label' // add volume for partitions
v hyp vol add paging 'addr' 'label' // add volume for paging
v hyp vol del image 'addr' // delete a partition volume
v hyp vol map free 'addr' // query free space on volume
v hyp vol map used 'addr' // query used space on volume
```

## IMAGE commands

```
v image create 'name' // create an image
v image del 'name' // delete an image
v image start 'name' // start an image
```

```

v image stop 'name'           // stop an image
v image stopall              // stop all images
v image set 'name' boot 'addr' // set image's boot partition
v image set 'name' cpu 'num'  // set image's number of CPUs
v image set 'name' sto 'addr' // set image's storage size in MB

```

## PARTition commands:

```

v part create 'name' 'addr' 'size' // create a partition for an image
v part copy 'name' 'addr' to 'name' 'addr' // copy a partition
v part del 'name' 'addr' // delete a partition
v part share 'name' 'addr' with 'name' 'addr' // share a partition

```

## Run levels

Linux run levels are defined in the Linux Standards Base at:

<http://www.linuxbase.org/spec/gLSB/gLSB/runlevels.html>

Originally SuSE had a different set of run levels. Distributions other than SuSE for zSeries and S/390 comply with this standard. On the PC platform, SuSE has complied with these run levels since V7.1. However, the current SuSE distribution on zSeries and S/390 does not. See Table 26-2:

Table 26-2 *Run levels*

Run level	LSB	SuSE	Red Hat
0	Halt	Halt	Halt
S	Undefined	Single user mode	Single user mode
1	Single user mode	Multiusers without network	Single user mode
2	Multiusers with no network services exported	Multi-user with network	Multiusers, without NFS (The same as 3, if you do not have networking)
3	Normal/full multiusers	Multiusers with xdm or equivalent	Full multiusers mode
4	Reserved for local use, default is normal/full multiusers	Unused	Unused

Run level	LSB	SuSE	Red Hat
5	Multiuser with xdm or equivalent	Unused	X11
6	Reboot	Reboot	Reboot
7, 8, 9	Undefined	Unused	Unused

## Parameter file values

In this appendix, we document the valid parameters of the Linux for S/390 parameter file. This is sometimes called the *parmfile* or the *parmline file*. Where appropriate, we also discuss how parameters are passed to the drivers when they are compiled as modules. Following is a summary of the valid parameters.

- ▶ `condev` - `condev=cuu`
- ▶ `ctc` - `ctc=prid,0xrxxx,0xwww,dddd` -or- `ctc=noauto`

For multiple CTC pairs, and `ctc` support compiled into the kernel:

```
ctc=0,0x3000,0x3001,ctc0 ctc=0,0x2000,0x2001,ctc1
```

For multiple CTC pairs, and `ctc` support compiled as a module, the entry in the `/etc/modules.conf` file would be specified as:

```
options ctc setup="ctc=0,0x3000,0x3001,ctc0 ctc=0,0x2000,0x2001,ctc1"
```

- ▶ `dasd` (2.2 kernel DASD device driver) - `dasd=[devno | devlist] | [autodetect] | [probeonly] dasd_force_diag=range[,...]`
- ▶ `dasd_mod` (2.4 kernel modularizable DASD device driver)
- ▶ `lcs` - `lcs=hwstats,ip_checksumming`
- ▶ `mdisk` - `mdisk=vdev [,vdev[,...]]`
- ▶ `netiucv` - `iucv=USERIDx[,USERIDy[,...]]`
- ▶ `root` - `root=path [ro] [noinitr]`
- ▶ `xpram` - `xpram_parts=<number_of_partition>[,<partition_size>[,...]]`

In regard to the parameter file, note the following:

- ▶ Linux for zSeries and S/390 has a general read routine that reads in chunks of 1024 bytes. This routine handles short blocks. When reading the parameter file, this will be normally the case.
- ▶ The Linux for S/390 kernel handles parameter data up to only 896 bytes. This is a *hard limit*. You cannot exceed it.
- ▶ In case you have a setting that ends exactly in the last column of a “card”, insert a blank in column one on the following card if you have to continue with further parameters. This is necessary because all card images are concatenated and presented as one line of text to the kernel.

For drivers that are compiled as modules, an important file to configure properly is `/etc/modules.conf`. (You may also see an `/etc/conf.modules` file on your system but that file has been made obsolete (deprecated) and will be ignored in some future release of Linux.)

During the actual installation process, distributions such as SuSE will put the necessary entries into `/etc/modules.conf`. If you make any changes after installation that affect the modules, you will need to update `/etc/modules.conf` yourself.

## Condev - 3215 line mode terminal

The 3215 terminal device driver makes it possible to use a 3270 terminal in 3215 emulation mode as a line mode terminal with Linux for S/390. This is typically the case only when running Linux for S/390 under VM or z/VM. The intended use of the 3215 terminal device driver is solely to launch Linux. When Linux is running, the user should access Linux for S/390 via telnet, because the terminal is a line mode terminal and is unable to support *curses* applications such as vi.

**Note:** This parameter is only necessary if your Linux for S/390 virtual machine's console is defined on an address other than the default of 0x0009.

Since the 3215 driver is needed to be able to IPL, it must be compiled into the kernel, and cannot be a module. Therefore, there should be no entry for it in `/etc/modules.conf`.

## Syntax

The syntax for this parameter is as follows:

```
condev=cuu
```



where:

**cuu** - is the unit address/device number in hexadecimal.

## Example

```
condev=0x001f
```

This forces the 3215 device driver to use the device number 0x1f for its 3215 device (the prefix 0x denotes a hexadecimal number).

## ctc - CTC/ESCON

Normally the CTC driver selects the channels in order (automatic channel selection), and alternately assigns them as read channels and write channels. If you need to use the channels in a different order, or only need to use a specific set of addresses, or do not want to use automatic channel selection with your installation, you can make these choices using the **ctc=** parameter in the parameter file, or in `/etc/modules.conf`.

## Syntax

The syntax for this parameter is as follows:

```
ctc=prid,0xrxxx,0xwww,dddd
```

where:

**prid** - is the protocol id (must always be 0)

**rrrr** - is the read channel address (hexadecimal)

**www** - is the write channel address (hexadecimal)

**dddd** - is the network device (ctc0 to ctc7 for a parallel channel, escon0 to escon7 for ESCON channels).

To switch automatic channel selection off, use the **ctc=noauto** parameter. This might be necessary if your installation uses 3172 devices or other devices that use the CTC device type and model, but operate with a different protocol.

## Examples

For one network device, using the CTCs at addresses 600/601:

```
ctc=0,0x600,0x601,ctc0
```

This will assign CTC 600 as the read channel, and 601 as the write channel for device ctc0.

For two network devices at addresses 600/601 and 605/608:

```
ctc=0,0x601,0x600,ctc0 ctc=0,0x605,0x608,escon3
```

This will assign CTC 601 as the read channel, and 600 as the write channel for ctc0. ESCON channel 605 will be assigned the read channel, and 608 as the write channel for escon3.

If the defaults had been taken, 600 would have been the read channel for ctc0, and 601 would have been the write channel. ESCON channels 605 & 608 would still have been a read/write pair, but they would have been assigned to escon0, not escon3.

When the CTC driver is compiled as a module, you must add some entries to the `/etc/modules.conf` file. To duplicate the previous example for one CTC connection, you would add the following. (Note the use of *both* single and double quotes):

```
alias ctc0 ctc
options ctc setup="ctc=0,0x0600,0x0601,ctc0"
```

For multiple CTC connections, the entries would be these:

```
alias ctc0 ctc
alias escon3 ctc
options ctc setup="ctc=0,0x0601,0x0600,ctc0 ctc=0,0x0605,0x0608,escon3"
```

If for any reason you need to issue the `insmod` command for the CTC driver from the VM 3215 console, and specify parameters other than what is in `/etc/modules.conf`, there is another consideration. By default, the double quote character (") is the VM escape character, similar to the Linux back slash (\). Unless you change that default using the `CP set term` command, you will need to double up on the double quotes as follows:

```
insmod ctc setup='""ctc=0,0x0601,0x0600,ctc0 ctc=0,0x0605,0x0608,escon3"""'
```

## dasd or dasd\_mod - DASD

The DASD driver is configured by the `dasd=` or `dasd_mod=` kernel parameter in the parameter file. Depending on what version of the kernel you are running, the driver may be named `dasd.o`, or `dasd_mod.o` (if you're not sure, try the commands `locate dasd.o` and `locate dasd_mod.o`). Unless you are running from a ramdisk during installation, this will not be important to you, because the driver must be compiled into the kernel, and not as a module if your root file system is on DASD.

It is conceivable that your root file system is on a volume specified by the `mdisk=` parameter, so the entries needed in `/etc/modules.conf` are shown here.

### Syntax

The syntax for this parameter is as follows:

```
dasd=address[,...] | autodetect | probeonly dasd_force_diag=range[,...]
```

or

```
dasd_mod=address[,...] | autodetect | probeonly dasd_force_diag=range[,...]
```

where:

**Address** - is in the form `from(device number) - to(device number)`, or explicitly specifies each device number, separated by commas. From - to and explicit device number can be specified multiple times and must be specified in hex, without a leading 0x. If multiple instances are specified, then Linux for S/390 will reserve a minor number for each DASD device specified, up to 64 devices. These devices are reserved in the order that they appear, and all extraneous device specifications (past 64) are ignored.

Multiple `dasd=` statements are allowed. Device names are assigned in the same order as the devices are specified.

**Autodetect** - determines DASD that is actually registered for use by Linux for S/390. Autodetect is the default for kernel version 2.2.14. If autodetect is specified without specifically calling out a DASD range or DASD device number, then the DASD will be ordered by subchannel path id (chpid) in ascending order.

**Probeonly** - is the default mode if there is no `dasd=` keyword in the parameter file. The device driver will then only report all the DASDs found, but not actually register them for use by Linux for S/390.

**dasd\_force\_diag** - tells the DASD driver to use the DIAG 250 instruction to access devices (minidisks) instead of channel programs.

## Examples

```
dasd=9AC,996-998 dasd_force_diag=100
```

This reserves minor numbers 0,4,8,12 for devices 9AC, 996, 997, and 998 respectively. Device 100 is accessed by means of VM's DIAG 250 instruction. This would be equivalent to specifying:

```
dasd=9AC,996,997,998 dasd_force_diag=100
```

When the DASD driver is compiled as a module, you must put the following entries in `/etc/modules.conf`. You will need to check whether your system is using the `dasd.o` or `dasd_mod.o` module names, and use the corresponding entry.

**Note:** Remember that your root file system cannot be on a DASD volume if you make the DASD driver a module. It must either be on a minidisk (with the minidisk driver compiled into the kernel), or on a ramdisk.

```
alias block-major-94 dasd
options dasd dasd=9AC,996-998
```

## lcs - LAN channel station (OSA cards)

The `lcs` driver is an Object Code Only (OCO) module provided by IBM. It can only be used as a module. It cannot be compiled directly into the kernel. You must specify the parameters for the module in the `parmfile`, or in `/etc/modules.conf`.

## Syntax

The syntax for this parameter is really three separate parameters, as follows:

```
lcs=hw-stats,ip-checksumming[,additional_cu_model,
    max_rel_adapter_no.[,additional cu model,max rel adapter no[,...]]]
lcs_devno=devno,rel_adapter_no pairs
lcs_noauto
```

where:

**hw-stats** - gets network statistics from LANSTAT LCS primitive, as opposed to doing it in software. This is not used by MVS, it is not recommended, it does not appear to work, and it usually does not work.

**ip-checksumming** - does IP checksumming on inbound packets. It is normally not required because Ethernet CRC32 is usually more than adequate (except perhaps for financial applications).

**additional\_cu\_model, max\_rel\_adapter\_no** - comprised of sets of model/maximum relative adapter number pairs. It is important that the parameters are entered in pairs (2, 4, 6 or 8 parameters), because the cu model and max rel adapter no are a pair.

**lcs\_devno=devno,rel\_adapter\_no** - takes devno,rel\_adapter\_no(port) pairs. Relative adapter number of -1 indicates that you should not use this adapter. This can be used to force certain CHPIDs to use a particular port number if the LCS protocol provides an incorrect one (hopefully this will fix a few problems happening in the field). This is identical to the devno\_portno\_pairs parameter described for modules. The name lcs\_devno is quite short because of limitations in the allowable length of kernel parameter names.

**lcs\_noauto** - tells the kernel to set auto-detection to off. You then have to explicitly configure LCS devices with the lcs\_devno parameter.

## Examples

```
lcs=1,1,97,8
```

will collect network statistics from the hardware; IP checksumming will be on; the model number is 0x61 (97 decimal); and an attempt is made to detect 8 ports on that model.

```
lcs_devno=0x2928,0 lcs_noauto
```

will not try to autodetect any LCS devices, and will use device number 2928 (hexidecimal), port 0 for the LCS device.

When the LCS driver is not specified in the kernel parameters, a different set of parameters that must be specified in /etc/modules.conf:

```
options lcs [use_hw_stats] [do_sw_ip_checksumming]
           [additional_model_info=model,port[,model,port[...]]
           [devno_portno_pairs=baseaddress,portnum] [noauto=1]
```

**use\_hw\_stats** - gets network statistics from LANSTAT LCS primitive, as opposed to doing it in software. This is not used by MVS, it is not recommended, it does not appear to work, and it usually does not work.

**do\_sw\_ip\_checksumming** - does IP checksumming on inbound packets. Normally not required because Ethernet CRC32 is usually more than adequate (except perhaps for financial institutions).

additional\_model\_info - made up of sets of model/maximum relative adapter number pairs. It is important that the parameters are entered in pairs (2, 4, 6 or 8 parameters), because the cu model and max rel adapter no are a pair. For example:

```
insmod lcs additional_model_info=0x70,3,0x71,5
```

will look for 3 ports on a 3088 model 0x70 and will also look for 5 ports on a 3088 model 0x71

devno\_portno\_pairs - takes devno,rel\_adapter\_no(port) pairs. Relative adapter number of -1 indicates that you should not use this adapter. This can be used to force certain chipids to use a particular port number if the LCS protocol provides an incorrect one (hopefully this will fix a few problems happening in the field). For example:

```
insmod lcs devno_portno_pairs=0x1c00,0,0x1c02,1,0x1d00,-1
```

tells the LCS device driver to:

- ▶ Only use port 0 if available for the device numbers 0x1c00 and 0x1c01
- ▶ Only use port 1 if available for the device numbers 0x1c02 and 0x1c03
- ▶ Do not under any circumstances use the device at 0x1d00 and 0x1d01 as an LCS device.

noauto=1 - tells the kernel to set auto-detection to off. You then have to explicitly configure LCS devices with the devno\_portno\_pairs parameter.

## mdisk - VM/CMS minidisks

VM minidisks can be reserved for Linux for S/390 use with the mdisk= kernel parameter in the parameter file. A reserved minidisk must be formatted with a blocksize of 512 bytes, 1 KB, 2 KB, or 4 KB. They can be of any size. It is possible to reserve a minidisk and create a file the size of the entire disk. This file can then be written to using the DIAG 250 instruction.

### Syntax

The syntax for this parameter is as follows:

```
mdisk=<vdev>[:<size>:<offset>:<blksize>][,<vdev>[:<size>:<offset>...]]
```

Since most people only specify the virtual device number, this is typically shortened to:

```
mdisk=vdev[,vdev[,...]]
```

where:

**vdev** is the virtual device number specified in hex without the leading 0x.

**size** - the size of the device in kilobytes.

**offset** - the offset after which the minidisk is available.

**blksize** - the blocksize with which the minidisk was formatted.

Multiple `mdisk=` statements are allowed. The minor numbers of the device will be assigned in the order that they appear in the parmline file.

## Example

```
mdisk=193,194
```

This reserves minor numbers 0 and 4 for minidisks at addresses 193 and 194.

When the minidisk driver is compiled as a module, you must put the following entries in `/etc/modules.conf`.

Note: Remember that your root file system cannot be on a minidisk if you make the minidisk driver a module. It must either be on a DASD volume (with the DASD driver compiled into the kernel), or on a ramdisk.

```
alias block-major-95 mdisk
options mdisk mdisk=193,194
```

## netiucv - IUCV communications

The Inter-User Communications Vehicle is a type of communications, used on z/VM and VIF, between guest virtual machines running on the same machine. This is similar to using virtual CTCs in that the connections are point-to-point. The IUCV driver needs to know the VM user IDs of the other virtual machines with which it can communicate.

## Syntax

The syntax for this parameter is as follows:

```
iucv=useridx[,userid[,...]]
```

where:

**useridx** - is the userid of the other virtual machine with which to communicate

## Example

```
iucv=tcPIP,linux3
```

In this example the Linux virtual machine can establish IUCV communication with user ID TCPIP (the VM TCP/IP service machine) on device iucv0, and with user ID LINUX3 (Linux for S/390 running in another virtual machine) on device iucv1.

When the iucv driver is compiled as a module, you must put the following entries in `/etc/modules.conf`.

```
alias iucv netiucv
options iucv iucv=tcPIP,linux3
```

## root

This parameter tells Linux where to find the root file system.

## Syntax

The syntax for this parameter is as follows:

```
root=path [ro] [noinitrd]
```

where:

**path** - points to the root filesystem.

**ro** - initially sets the filesystem to read-only; later it is accessed read/write

**noinitrd** - does not use the initial RAMdisk.

## Examples

```
root=/dev/ram0 ro
```

This tells Linux that the root file system is to be found on a ramdisk. This is a temporary RAMdisk (ram0) used to get a mini-Linux system running, so that you can perform the rest of the IPL or repair tasks.

```
root=/dev/dasda1 noinitrd
```

This tells Linux that the root file system is to be found on the first DASD device specified by the `dasd=` kernel parameter. Additionally, the initial ramdisk is not to be used.



# xpram

Although Linux addresses only about 2 GB (1919 MB) of memory, you can also access expanded storage. The xpram driver maps a file system or a swap space onto expanded storage.

An xpram device has major number 35 and can be partitioned, starting with minor number 0. You can have up to 32 partitions. The associated device node is `/dev/xpram<letter>`.

## Syntax

The syntax for this parameter is as follows:

```
xpram_parts=<number_of_partitions>[,<partition_size> [,...]]
```

where:

**number\_of\_partitions** - can be a number from 1 to 32; the default is 1.

**partition\_size** - is divided into three parts: `<hex><number><unit>`. For the first part, hex has a value 0x and specifies that the size is in hexadecimal. If this part is omitted, the size is treated as decimal. For the third part, the unit can be k or K for kilobytes, m or M for megabytes, and g or G for gigabytes. A size of 0 requests the driver to allocate the rest of expanded storage that is available.

## Examples

```
xpram_parts=1,0x200m
```

This reserves one partition (`/dev/xpram0`) of hex 200 megabytes (decimal 512 MB).

```
xpram_parts=2,200m,1g
```

In contrast, this reserves two partitions: one with a size of 200 MB (`/dev/xpram0`), and a second with a size of 1 GB (`/dev/xpram1`).

When the xpram driver is compiled as a module, you must add some entries to the `/etc/modules.conf` file.

```
alias block-major-35 xpram
options xpram devs=<number_of_devices> [sizes=<size>[,<size>,...]]
```





# Customizing Linux checklist

This appendix provides a checklist of tasks that you may want to do to customize Linux:

- ▶ Add a non-root user ID for yourself and anyone else that will be using the system. This is an important (and standard) systems administration practice. Logging on as root all the time can lead to disaster.

Enable Secure Shell (SSH). Unless you are using the one secure telnet that is available (and you almost certainly are not), this is a security *must*. It also provides a secure copy (**scp**) program that replaces/supplements the typical non-secure ftp client.

In the case of the SuSE distribution, you will have to download the RPMs for `openssh` and `openssl` from `ftp://ftp.suse.de/pub/suse/s390/7.0/sec1/` since they were not included in the main distribution.

- ▶ Install some kind of system administration tool. For examples of these tools, refer to 21.2, “Administration tools” on page 402, and 21.3, “Remote network management tools” on page 408.
- ▶ Enable and customize a firewall using `ipchains` or `iptables`.
- ▶ Enable and customize an FTP server.
  - Decide if you want to allow anonymous access.
  - Enable or disable anonymous access per your decision.
- ▶ Enable and customize the **sudo** command.

- ▶ Customize the `/etc/suauth` file.
- ▶ Enable and customize an HTTP server.
- ▶ Enable and customize Samba.
- ▶ Enable and customize (x)ntpd to keep your system time accurate.
- ▶ Enable and customize a mail server.
  - Alternatively, set a pointer to an existing “smart” relay host.
- ▶ Enable and customize a DNS server.
- ▶ Enable and customize an INN (news) server.
- ▶ Enable and customize xdm/kdm/gdm/whatever for graphical X logins.
- ▶ Create a second, third, fourth, etc., IPL volume.
- ▶ Add additional DNS servers to `/etc/resolv.conf`.
- ▶ Add additional domain search values to `/etc/resolv.conf`.
- ▶ Add entries to `/etc/hosts` for systems that do not have DNS records.
- ▶ Add alias `xxxx off` entries to `/etc/modules.conf` for things that are generating unnecessary error messages.
- ▶ Install a package such as `logrotate` to manage your system logs, or write your own scripts to do something similar.
- ▶ Enable quotas to limit the amount of disk space your users can consume.
- ▶ Make your root file system as small as possible (containing only `/bin`, `/boot`, `/lib`, and `/root`) and put everything else on a journaling file system such as GFS, AFS, `ext3`, JFS, etc. (This may seem to contradict the information in Chapter 26, “Conclusion” on page 513, but eventually journaling file systems on Linux will mature, and this is something you will want to do then.)
- ▶ Enable and customize `lpd` to print to any network printers you might have.



## 64-bit Linux

To run 64-bit Linux, you will need zSeries hardware. It will not run on S/390 hardware. It is not a simple process to create a 64-bit Linux system and we did not try it during the residency to produce this redbook. We recommend you get a system that is already built. See Chapter 15, “Think Blue 64 Linux for zSeries” on page 239 for a description of the *Think Blue* 64-bit distribution.

For reference, to see how a 64-bit system is built, there is Neale Ferguson’s 64-bit starter system Web site at:

<http://penguinvm4.princeton.edu/>

That Web site contains the components necessary for creating a Linux 2.4 system for z/Architecture. The kernel operates in 64-bit mode and the executables are all 64 bit objects. The starter system consists of the following:

- ▶ Kernel for booting from virtual reader
- ▶ Parameter line (EBCDIC)
- ▶ Parameter line (ASCII)
- ▶ Initial RAMdisk - this contains 32 bit objects and is only used so you can build the starter system disk

Following is an overview.

### Building your first kernel

1. Create a cross-compile environment

- binutils-2.10.0.8 + experimental patches
  - gcc-2.95.2 + experimental patches
  - glibc-2.2.2 + experimental patches
2. Download kernel from ftp.kernel.org
  3. Apply patches
  4. Update Makefile:
    - ARCH := s390x
    - CROSS\_COMPILE=s390x-ibm-linux-
  5. Make kernel
    - make menuconfig
    - make dep
    - make
    - make modules
    - make modules\_install
  6. Build silo:
    - cd arch/s390x/tools/silo
    - make
  7. Copy files to /boot:
    - cp arch/s390x/boot/image /boot
    - cp arch/s390x/boot/\*.boot /boot
    - cp System.map /boot
  8. Run silo:
    - cd /boot
    - <path-to-kernel-tools>/silo -f image -p parmfile -d /dev/dasd

## Building a file system from scratch

1. Use first kernel as a base
2. Get “Building LFS from Scratch” from:  
<http://linuxfromscratch.org>
3. Get second disk and mount as /root64
4. Build ncurses using cross-compiler
5. Follow instructions in LFS document to populate /root64
6. Basically a two-stage process:
  - a. Starter pack:
    - Build statically linked versions of core packages (inc. gcc)

- Create traditional file layouts (i.e. /etc, /lib etc.)
  - Create /dev nodes – need to manually define /dev/dasdx
  - Create /etc contents (e.g. passwd, groups)
  - chroot to /root64
- b. Production pack:
- Build new glibc-2.2.2
  - Rebuild core packages - this time dynamically linked
  - Create and tailor startup scripts
  - Copy /boot contents & /lib/modules
  - Update parameter file and run silo
  - Run depmod
  - Reboot off the /root64 disk

## Relevant linux-390 appends

Following are two appends from the linux-390 list server which may help with the build process.

“I'm trying to build the cross-platform environment for s390x, I've followed the steps below and have successfully performed steps 1-7. When I go to rebuild gcc, this time enabling c++ I get the following:”

```
/root64/home/usanefe/gcc-build/gcc/xgcc
-B/root64/home/usanefe/gcc-build/gcc/ -B/usr/local/s390x-ibm-linux/bin/ -c
-g -O2 -fvtable-thunks -D_GNU_SOURCE -fno-implicit-templates -I.
-I../../gcc-2.95.2/libio -nostdinc++ -D_IO_MTSAFE_IO
../../gcc-2.95.2/libio/pfstream.cc
xgcc: ../../gcc-2.95.2/libio/pfstream.cc: C++ compiler not installed on
this system
```

but g++ had just been built:

```
gcc -DCROSS_COMPILE -DIN_GCC      -g -O2 -DHAVE_CONFIG_H -o g++ \
gcc.o g++spec.o intl.o prefix.o version.o  obstack.o \
../libiberty/libiberty.a
rm -f g++-cross
cp g++ g++-cross
```

(The lack of success comes from it trying to link prior to there being any objects to link against (namely crt1.o). What you get at this stage is a cross-compiler good enough to build glibc. glibc links only stuff that it has created which includes crt1.o. Thus after you do the make install you can go back to gcc and rebuild, this time it will find what it needs to link and can build libiberty successfully. Now if only mine would build libio!

What I've tracked down is that when `xgcc` is invoked it checks the suffix of the filename (in this case `.cc`) and matches it against a table of things. If the language is supported it will find in this table an entry whose "specs" member contains '@' as the first character. In my case it is finding '#' which is the default table implying it has failed to update this table with the correct state of play.)

-----  
The following steps should provide you a cross build environment:

- 1) unpack all source packages

```
tar xzf binutils-2.10.1.tar.gz
tar xzf gcc-2.95.2.tar.gz
tar xzf glibc-2.2.tar.gz
cd glibc-2.2
tar xzf ~/glibc-linuxthreads-2.2.tar.gz
cd ..
tar xzf linux-2.4.2.tar.gz
mv linux linux-2.4.2
```
- 2) apply the patches

```
patch -sp0 < ~/binutils-2.10.1-s390.diff
patch -sp0 < ~/gcc-2.95.2-s390x.diff
patch -sp0 < ~/glibc-2.2-s390x.diff
patch -sp0 < ~/linux-2.4.2-s390x.diff
```
- 3) configure, compile and install the binutils

```
mkdir binutils-build
cd binutils-build
../binutils-2.10.1/configure --prefix=/usr/local \
--target=s390x-ibm-linux --host=s390-ibm-linux
make
su (enter root password)
make install
exit
cd ..
```
- 4) configure, compile and install the C compiler

```
mkdir gcc-build
cd gcc-build
../gcc-2.95.2/configure --prefix=/usr/local \
--target=s390x-ibm-linux --host=s390-ibm-linux \
--enable-languages="c" --with-newlib
make (ignore the compile error in libiberty)
su (enter root password)
make install
exit
cd ..
```



```

5) configure the kernel
cd linux-2.2.13
vi Makefile (set "ARCH := s390x" and
             "CROSS_COMPILE=s390x-ibm-linux-")
make menuconfig
cd ..

6) copy kernel header files
su (enter root password)
cp -ar linux-2.4.2/include/linux
/usr/local/s390x-ibm-linux/include/linux
cp -ar linux-2.4.2/include/asm-s390x
/usr/local/s390x-ibm-linux/include/asm
exit

7) configure, compile and install the glibc
mkdir glibc-build
cd glibc-build
../glibc-2.2/configure --prefix=/usr/local/s390x-ibm-linux \
  --host=s390x-ibm-linux --build=s390-ibm-linux \
  --enable-add-ons --disable-profile --enable-omitfp
make
su (enter root password)
make install
exit
cd ..

8) configure, compile and install all needed compilers
cd gcc-build
rm -fr *
../gcc-2.95.2/configure --prefix=/usr/local \
  --target=s390x-ibm-linux --host=s390-ibm-linux \
  --enable-shared --enable-threads --enable-languages="c,c++"
make
su (enter root password)
make install
exit
cd ..

```





## ICKDSF bootstrap job

This appendix lists a sample job for ICKDSF bootstrap. For a description of this process, see 3.4.1, “Using the ICKDSF bootstrap” on page 34. This sample job was obtained on the Web starting at:

<http://pucc.princeton.edu/~rvdheij/linuxipl.html>

```
// SET LINUX=LINUXZ , (*)                                00001001
// SET UID=NL66157                                       00002002
/* after change in volser for LINUXZ you need to change the 00003000
/* verify in the ICKDSF STEP                               00004000
//ASMIPLD EXEC PGM=ASMA90                                 00005000
//SYSLIB DD DSN=SYS1.MACLIB,DISP=SHR                     00006000
//SYSUT1 DD UNIT=VIO,SPACE=(CYL,(10,5))                  00009000
//SYSLIN DD DSN=IPLDECK,BLKSIZE=3120,RECFM=FB,LRECL=80, 00009100
// SPACE=(TRK,(15,0)),DISP=(,PASS)                       00009200
//SYSPRINT DD SYSOUT=*,DCB=(BLKSIZE=3509)                00009300
//SYSIN DD *                                              00009900
* Bootstrap to boot Linux with ramdisk from an OS formatted volume. LIN00010
                                                                    LIN00020
*      13 Feb 2001 Rob van der Heij <rvdheij@iae.nl>      LIN00030
*      7 Jun 2001 rmh Added retry for I/O                  LIN00040
                                                                    LIN00050
* The volume is prepared for IPL by an ICKDSF with the BOOTSTRAP option LIN00060
* Use the concatenation of SYS.SAMPLIB(IPLRECS) and this object deck as LIN00070
* the input for IPLDD. When the volume is IPLed the bootstrap locates LIN00080
* the 3 datasets in the VTOC, loads them into memory and transfer LIN00090
* control to the Linux kernel.                             LIN00100
                                                                    LIN00110
LINUXIPL CSECT                                           LIN00120
```

				LIN00130
IPLPSW	DC	X'00080000',AL4(X'80000000'+ENTRY)		LIN00140
	DC	LINUXIPL+X'200'		LIN00150
	DC	C'+-----+'		LIN00160
	DC	C' Linux for S/390    Bootstrap    '		LIN00170
	DC	C'+-----+'		LIN00180
	ORG	LINUXIPL+X'260'		LIN00190
				LIN00200
ENTRY	DS	OH		LIN00210
	BALR	R12,0		LIN00220
	USING	*,R12		LIN00230
	LA	R13,X'0800'	Use next 2K as work space	LIN00240
	L	R1,X'B8'	Save subchannel address of IPL	LIN00250
	ST	R1,DEVSUBCH	device	LIN00260
	LCTL	R6,R6,=X'FF000000'	Enable for I/O interrupts	LIN00270
				LIN00280
	BAL	R11,READVTOC		LIN00290
	MVI	STATUS,X'30'		LIN00300
				LIN00310
	L	R8,=X'0001000'	Load kernel into memory	LIN00320
	LR	R4,R8		LIN00330
	LA	R6,EXT1		LIN00340
	BAL	R11,READFILE		LIN00350
	MVI	STATUS,X'60'		LIN00360
				LIN00370
	L	R8,=X'00800000'	Load initrd at 8M	LIN00380
	LA	R6,EXT3		LIN00390
	BAL	R11,READFILE		LIN00400
	MVI	STATUS,X'90'		LIN00410
	S	R8,=X'00800000'	Compute size of ramdisk image	LIN00420
	L	R6,=X'00010400'	Parameter area for setup.h	LIN00430
	MVC	14(4,R6),=X'00800000'		LIN00440
	STCM	R8,B'1111',18(R6)		LIN00450
				LIN00460
	L	R8,=X'0008000'	Load parm file at page 8	LIN00470
	LA	R6,EXT2		LIN00480
	BAL	R11,READFILE		LIN00490
	MVI	STATUS,X'B0'		LIN00500
	L	R6,=X'00010480'	Move parm file contents down	LIN00510
	L	R7,=F'896'	into area defined by setup.h	LIN00520
	L	R8,=X'00008000'		LIN00530
	LR	R9,R7		LIN00540
	MVCL	R6,R8		LIN00550
	MVI	STATUS,X'F0'		LIN00560
				LIN00570
	L	R1,=X'0010000'	Run the kernel now	LIN00580
	BR	R1		LIN00590
				LIN00600
				LIN00610
DSN	DC	CL44'SYS1.LINUX.TAPEIPL.IKR'		LIN00620
	DC	CL44'SYS1.LINUX.PARMPFILE'		LIN00630
	DC	CL44'SYS1.LINUX.INITRD'		LIN00640
SCHIB	DS	8D		LIN00650
DROPDEAD	ST	R14,WAITPSW+4	Show caller address instead	LIN00660

	MVC	WAITPSW+3(1),STATUS		LIN00670
	MVI	WAITPSW,X'00'	Make it disabled wait	LIN00680
	LPSW	WAITPSW		LIN00690
	DS	OD		LIN00700
WAITPSW	DC	X'020A0000',X'80DEAD00'		LIN00710
MYNEWPSW	DC	X'00080000',A(X'80000000'+INT)		LIN00720
				LIN00730
IOR6	DS	F		LIN00740
*			Execute a channel program at (R1)	LIN00750
I0	EQU	*		LIN00760
	ST	R6,IOR6	Save R6 to be used as counter	LIN00770
	LA	R6,10	10 retries	LIN00780
	MVC	X'78'(8),MYNEWPSW	Prepare I/O new PSW	LIN00790
	ST	R1,ORB+8		LIN00800
	L	R1,DEVSUBCH	Get subchannel address back	LIN00810
I01	SSCH	ORB		LIN00820
I04	LPSW	WAITPSW		LIN00830
INT	C	R1,X'B8'	Did my I/O complete?	LIN00840
	BNE	I04		LIN00850
	TSCH	IRB		LIN00860
	CLC	=XL2'0C00',IRB+8	Check for I/O completion	LIN00870
	BZ	I05		LIN00880
	BCT	R6,I01		LIN00890
	CLC	=XL2'0C00',IRB+8		LIN00900
I05	L	R6,IOR6	Restore R6	LIN00910
	BR	R5		LIN00920
*			Read a single dataset	LIN00930
*			R6: Points to CCHH of first track and CCHH of last track	LIN00940
*			R8: Address where data must be loaded	LIN00950
*			R4: Number of bytes to skip initially	LIN00960
READFILE	EQU	*		LIN00970
	MVC	CCW1P+8(8),0(R6)	Copy CCHH's to Define Extent	LIN00980
	L	R3,0(R6)	First track	LIN00990
RF01	EQU	*	Repeat for each track in extent	LIN01000
				LIN01010
	ST	R3,CCW2P+4	Put CCHH in Locate Record	LIN01020
	ST	R3,CCW2P+8		LIN01030
				LIN01040
	LA	R1,CCW1	Read a single track	LIN01050
	BAL	R5,IO		LIN01060
				LIN01070
*			Process raw track just read and move the payload to (R8)	LIN01080
*			Skip the first R4 bytes while doing so	LIN01090
	L	R2,CCW3+4	See where data was loaded	LIN01100
	A	R4,=F'8'	Also skip R0	LIN01110
RF02	EQU	*		LIN01120
	SLR	R3,R3		LIN01130
	IC	R3,5(R2)	Load keylength	LIN01140
	AH	R3,6(R2)	Add datalength	LIN01150
	LA	R2,8(R2)	Payload address is beyond count field	LIN01160
	LR	R9,R3		LIN01170
	LTR	R4,R4	Any bytes to skip?	LIN01180
	BZ	RF03		LIN01190
				LIN01200

	CR	R4,R3	Part of payload to be skipped?	LIN01210
	BP	RF04	Less than one record to skip	LIN01220
	AR	R2,R4	Account for bytes skipped	LIN01230
	SLR	R3,R4	So many bytes less to move	LIN01240
	LR	R9,R3	Correct target	LIN01250
	SLR	R4,R4	No more bytes to skip	LIN01260
	B	RF03	Go and move the	LIN01270
RF04	SLR	R4,R3	Skip entire record	LIN01280
	AR	R2,R3	Adjust pointer to input data	LIN01290
	B	RF05		LIN01300
RF03	MVCL	R8,R2		LIN01310
RF05	CLC	0(4,R2),=F'-1'	Next count field -1 ?	LIN01320
	BNZ	RF02		LIN01330
*			Go to next track	LIN01340
	L	R3,CCW2P+4	Get CCHH back again	LIN01350
	C	R3,CCW1P+12	Are we done already?	LIN01360
	BZ	RF09		LIN01370
	LA	R3,1(R3)		LIN01380
	LH	R0,=X'00FF'		LIN01390
	NR	R0,R3		LIN01400
	CH	R0,=H'15'	Beyond last track?	LIN01410
	BNZ	RF01		LIN01420
	A	R3,=AL4(X'10000'-15)		LIN01430
	B	RF01		LIN01440
RF09	BR	R11		LIN01450
				LIN01460
*			Read the VTOC of the volume in a buffer	LIN01470
*			and search for the 3 datasets	LIN01480
READVTOC	EQU	*		LIN01490
	LA	R1,CCW4	Channel program to read R3	LIN01500
	LR	R10,R11	Save my return address	LIN01510
	BAL	R5,I0	Read R3	LIN01520
	BZ	VT01		LIN01530
	LA	R1,SENSEID		LIN01540
	BAL	R5,I0		LIN01550
	BAL	R14,DROPDEAD		LIN01560
VT01	MVC	VOL1BUF+15(4),VOL1BUF+11	Get VOLVTOC pointer	LIN01570
	LA	R6,VOL1BUF+11	And use one track as the extent	LIN01580
	SLR	R4,R4		LIN01590
	L	R8,=X'00800000'		LIN01600
	BAL	R11,READFILE	Read this single track	LIN01610
				LIN01620
	CLC	VOL1BUF+15(4),X'2D'(R8)	Just one track?	LIN01630
	BE	VT02		LIN01640
	L	R8,=X'00800000'		LIN01650
	SLR	R4,R4		LIN01660
	MVC	VOL1BUF+15(4),X'2D'(R8)	Last CCHH used by VTOC	LIN01670
	BAL	R11,READFILE		LIN01680
VT02	L	R3,=F'3'	3 datasets to find	LIN01690
	LA	R2,DSN	Point to first one	LIN01700
	LA	R4,EXT1	Point to first extent	LIN01710
	LR	R9,R8	End of VTOC in memory	LIN01720
VT03	IC	R8,STATUS		LIN01730
	LA	R8,1(R8)		LIN01740

	STC	R8,STATUS		LIN01750
	L	R8,=X'00800000'		LIN01760
VT04	CLC	0(44,R2),0(R8)		LIN01770
	BNZ	VT05		LIN01780
	MVC	0(8,R4),X'06B'(R8)	Copy extent info	LIN01790
	B	VT06		LIN01800
VT05	LA	R8,140(R8)	Advance to next VTOC entry	LIN01810
	CR	R8,R9		LIN01820
	BL	VT04		LIN01830
	BAL	R14,DROPDEAD		LIN01840
VT06	SLR	R1,R1		LIN01850
	LA	R2,44(R2)	Next DSN	LIN01860
	LA	R4,8(R4)	Next extent	LIN01870
	BCT	R3,VT03		LIN01880
	BR	R10		LIN01890
*				LIN01900
	DS	0D		LIN01910
IRB	DS	16F		LIN01920
DEVSUBCH	DS	F		LIN01930
ORB	DC	A(0)		LIN01940
	DC	X'0080FF00'		LIN01950
	DC	A(CCW1)	Channel Program Address	LIN01960
STATUS	DC	AL1(0)		LIN01970
*				LIN01980
	DS	0D		LIN01990
CCW1P	DC	X'00CC0000',X'00000000'	ECKD for sequential read	LIN02000
	DC	X'00000000',X'00000000'		LIN02010
CCW2P	DC	X'4c000001',X'00000000'	Prepare to read full track	LIN02020
	DC	X'00000000',X'01FF0000'		LIN02030
*				LIN02040
SENSEID	DC	X'04000020',A(SENSEID1)		LIN02050
CCW1	DC	X'63400010',A(CCW1P)	Define Extent	LIN02060
CCW2	DC	X'47400010',A(CCW2P)	Locate Record	LIN02070
CCW3	DC	X'DE20F000',X'00700000'	Read full track	LIN02080
				LIN02090
	DS	0D		LIN02100
CCW4P	DC	X'00C00000',X'00000000'	ECKD channel program	LIN02110
	DC	X'00000000',X'00000000'		LIN02120
CCW5P	DC	X'06000001',X'00000000'	Locate R3 for reading	LIN02130
	DC	X'00000000',X'03FF0000'		LIN02140
				LIN02150
CCW4	DC	X'63400010',A(CCW4P)	Define Extent	LIN02160
CCW5	DC	X'47400010',A(CCW5P)	Locate Record	LIN02170
CCW6	DC	X'06200050',A(VOL1BUF)	Read 80 bytes	LIN02180
				LIN02190
SENSEID1	DS	8F		LIN02200
EXT1	DS	2F		LIN02210
EXT2	DS	2F		LIN02220
EXT3	DS	2F		LIN02230
				LIN02240
VOL1BUF	DS	80C		LIN02250
				LIN02260
R0	EQU	0		LIN02270

R1	EQU	1	LIN02280	
R2	EQU	2	LIN02290	
R3	EQU	3	LIN02300	
R4	EQU	4	LIN02310	
R5	EQU	5	LIN02320	
R6	EQU	6	LIN02330	
R7	EQU	7	LIN02340	
R8	EQU	8	LIN02350	
R9	EQU	9	LIN02360	
R10	EQU	10	LIN02370	
R11	EQU	11	LIN02380	
R12	EQU	12	LIN02390	
R13	EQU	13	LIN02400	
R14	EQU	14	LIN02410	
R15	EQU	15	LIN02420	
	END	,	LIN02430	
	//INTLINUX	EXEC PGM=ICKDSF,PARM='NOREPLYU'	00051600	
	//SYSPRINT	DD SYSOUT=*	00051700	
	//IPLTEXT	DD DSN=SYS1.SAMPLIB(IPLRECS),DISP=SHR	00051800	
	//	DD DSN=&&IPLDECK,DISP=(OLD,DELETE)	00051900	
	//LINUX	DD UNIT=SYSALLDA,VOL=SER=&LINUX.,DISP=OLD	00052000	
	//SYSIN	DD *	00060000	
	INIT	DDNAME(LINUX)	-	00190000
		OWNER(SYS1)	-	00200000
		BOOTSTRAP	-	00201000
		IPLDD(IPLTEXT)	-	00202000
		PURGE	-	00210000
		VERIFY(LINUXZ)	-	00220000
		VTOC(1,0,15)	-	00250000
		NIX		00260000
	//TAPEIPL	EXEC PGM=IEBGENER	00290000	
	//SYSPRINT	DD SYSOUT=*	00300000	
	//SYSUT1	DD DSN=&UID..TAPEIPL.IKR,DISP=SHR	00310000	
	//SYSUT2	DD DSN=SYS1.LINUX.TAPEIPL.IKR,DISP=(NEW,KEEP),	00320000	
	//	UNIT=SYSALLDA,RECFM=FB,LRECL=1024,BLKSIZE=0,	00330000	
	//	SPACE=(TRK,(28,0),,CONTIG),VOL=SER=&LINUX	00340002	
	//SYSIN	DD DUMMY	00350000	
	//*		00360000	
	//INITRD	EXEC PGM=IEBGENER	00370000	
	//SYSPRINT	DD SYSOUT=*	00380000	
	//SYSUT1	DD DSN=&UID..INITRD,DISP=SHR	00390000	
	//SYSUT2	DD DSN=SYS1.LINUX.INITRD,DISP=(NEW,KEEP),	00400000	
	//	UNIT=SYSALLDA,RECFM=FB,LRECL=1024,BLKSIZE=0,	00410000	
	//	SPACE=(TRK,(179,0),,CONTIG),VOL=SER=&LINUX	00420002	
	//SYSIN	DD DUMMY	00430000	
	//*		00440000	
	//PARMFILE	EXEC PGM=IEBGENER	00450000	
	//SYSPRINT	DD SYSOUT=*	00460000	
	//SYSUT1	DD DSN=&UID..PARMFILE,DISP=SHR	00470000	
	//SYSUT2	DD DSN=SYS1.LINUX.PARMFILE,DISP=(NEW,KEEP),	00480000	
	//	UNIT=SYSALLDA,RECFM=FB,LRECL=1024,BLKSIZE=0,	00490000	
	//	SPACE=(TRK,(1,0),,CONTIG),VOL=SER=&LINUX	00500002	
	//SYSIN	DD DUMMY	00510000	



# 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 549.

- ▶ The appendix *Linux Software Packages* is associated with this book. It is available separately on the Web at:  
<ftp://www.redbooks.ibm.com/redbooks/SG246264/RPMappendix.pdf>
- ▶ *Linux for S/390*, SG24-4987, on the Web at:  
<http://www.redbooks.ibm.com/abstracts/sg244987.html>
- ▶ *Linux HPC Cluster Installation*, SG24-6041, on the Web at:  
<http://www.redbooks.ibm.com/abstracts/sg246041.html>
- ▶ *Linux for zSeries and S/390: ISP/ASP Solutions*, SG24-6299, currently available as a Redpiece on the Web at:  
<http://www.redbooks.ibm.com/abstracts/sg246299.html>
- ▶ *Understanding LDAP*, SG24-4986

## Other resources

These publications are also relevant as further information sources:

- ▶ SuSE documentation (available only on CD 1, which is the file `suse-us-s390-GA-CD1-with-rb.iso`):
  - `I390gp3.pdf` - *Preparing for Installing SuSE LINUX for S/390*, April 5, 2001, LNUX-1001-01
  - `I390ga3.pdf` - *Installing SuSE LINUX for S/390*, April 5, 2001, LNUX-1002-01
  - `manual.pdf` - *SuSE 7.0 Installation, Networking, Know How*  
<ftp://ftp.suse.com/pub/suse/s390/suse-us-s390/>

- ▶ Turbolinux documentation, available on:
  - <http://www.turbolinux.com/products/s390/index.html>
  - *Preparing for Installing Turbolinux for S/390*, December 2000
  - *Installing Turbolinux for S/390*, December 2000
  - *Turbolinux for zSeries and S/390: User Guide*, December 2000
- ▶ developerWorks documents:
  - *Using the Dump Tools*, July 2001
    - <http://oss.software.ibm.com/developerworks/opensource/linux390/docu/139dmp22.pdf>
  - *LINUX for S/390 Device Drivers and Installation Commands*, July 2000
    - <http://oss.software.ibm.com/developerworks/opensource/linux390/docu/1390dd03.pdf>
  - *LINUX for S/390 ELF Application Binary Interface Supplement*, July 2001
    - <http://oss.software.ibm.com/developerworks/opensource/linux390/docu/1390abi0.pdf>
- ▶ *Caiman Linux for zSeries Installation Guide*, July 2001
  - <ftp://linux390.linuxkorea.co.kr/caiman/doc>
- ▶ *VM/ESA CMS User's Guide*, SC24-5775
- ▶ *VM/ESA Quick Reference*, SX24-5290

## Referenced Web sites

These Web sites are also relevant as further information sources:

- ▶ RPM information associated with this redbook:
  - <ftp://www.redbooks.ibm.com/redbooks/SG246264/RPMappendix.pdf>
- ▶ Downloading SuSE Linux GA:
  - <ftp://ftp.suse.com/pub/suse/s390/suse-us-s390>
- ▶ Downloading Turbolinux GA:
  - <ftp://ftp.turbolinux.com/pub/product/s390>
- ▶ Downloading Turbolinux V6.5 beta
  - <ftp://ftp.turbolinux.com/pub/beta/s390/>
- ▶ Downloading Red Hat Linux beta; start at either:
  - <ftp://ftp.redhat.com/pub/redhat/linux/rawhide/s390>
  - <ftp://ftp.redhat.de/pub/s390>

- ▶ Downloading the Marist file system:  
<ftp://linux390.marist.edu/pub/update>
- ▶ Downloading Caiman Linux from LinuxKorea:  
<ftp://linux390.linuxkorea.co.kr/caiman>
- ▶ Downloading Think Blue 64 Linux from Millenux:  
<ftp://linux.zseries.org/pub/ThinkBlue64-7.1/>
- ▶ Downloading Think Blue Linux from Millenux:  
<ftp://linux.s390.org>
- ▶ Downloading Webmin:  
<http://www.webmin.com/webmin/download.html>
- ▶ LVM information, download, documentation:  
<http://www.sistina.com/lvm>
- ▶ KDE information:  
<http://www.kde.org/>
- ▶ Samba information:  
<http://www.samba.org>
- ▶ Apache information:  
<http://apache.org>

## How to get IBM Redbooks

Search for additional Redbooks or redpieces, view, download, or order hardcopy from the Redbooks Web site:

[ibm.com/redbooks](http://ibm.com/redbooks)

Also download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

Redpieces are Redbooks in progress; not all Redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

## 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.



# Special notices

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter 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 the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

The following terms are trademarks of other companies:

Tivoli, Manage. Anything. Anywhere., The Power To Manage., Anything. Anywhere., TME, NetView, Cross-Site, Tivoli Ready, Tivoli Certified, Planet Tivoli, and Tivoli Enterprise are trademarks or registered trademarks of Tivoli Systems Inc., an IBM company, in the United States, other countries, or both. In Denmark, Tivoli is a trademark licensed from Kjøbenhavns Sommer - Tivoli A/S.

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

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

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

PC Direct is a trademark of Ziff Communications Company in the United States and/or other countries and is used by IBM Corporation under license.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States and/or other countries.

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

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.

# Index

## Symbols

.htaccess file 103  
/boot directory 178  
/dev/ntibm0 480  
/etc/crontab 398  
/etc/fstab 285  
/etc/hosts 229  
/etc/inetd.conf 100, 229  
/etc/ldap.conf 386  
/etc/modules.conf 174, 471  
/etc/nsswitch.conf 380, 386  
/etc/nsswitch.ldap 386  
/etc/pam.d 380  
/etc/passwd 99, 186, 379, 380, 384  
    fields 448  
/etc/proftpd/proftpd.conf 143  
/etc/rc.config 97  
/etc/resolv.conf 146, 184, 209, 252  
/etc/securetty 141  
/etc/sendmail.cf. 256  
/etc/shadow 379, 384  
    fields 448  
/etc/smbpasswd 99  
/etc/snmpd.conf 409  
/etc/zilo.conf 174  
/lib/libnss\_ldap.so 386  
/opt/IBMWebAS/logs/tracefile 498  
/proc file system 261, 396  
/proc/chandev 263  
/proc/dasd/devices 396  
/proc/lvm 305  
/var/log 364  
/var/log/boot.log 366  
/var/log/messages 365

## Numerics

2064 12  
2216 22  
9672 12

## A

Access Control List (ACL) 378

access.conf 103  
aclentry 392  
aclpropagate 392  
aclsource 392  
adduser 232  
administration tools 402  
AdminUX 408  
alien 236  
AllowOverride 104  
Alpha, DEC 5  
AMANDA 475  
Apache 102  
    documentation 106  
    enabling CGI scripts 107  
    protecting a directory 103  
apt-get 214, 235  
awk 401

## B

backup and restore 463  
    incremental 479  
bash 126, 348  
bbclient 422  
bb-hosts 422  
Betzler, Boas 6  
Big Brother 414  
    configuring 420  
    installation and configuration 415  
    using 420  
Big Sister 433  
big-endian architecture 25  
Bigfoot 6  
BLP tape option 45  
BMC Patrol 432  
bonnie 345

## C

Caiman 7  
CAIMAN EXEC 218  
Caiman Linux  
    configuration files 228  
    disk requirements 216

- dselect screen 227
- history 213
- how it differs 214
- installation 216
- installation worksheet 215
- Logical Volume Manager (LVM) 231
- network setup 229
- system configuration 228
  - using Apache 234
  - using FTP 234
  - using Samba 234
- CDs
  - creating from ISO images 57
- CGI scripts 107
- chage 449
- channel device layer 263
- char-major-4 210
- chkconfig 142, 184, 185, 208, 250, 386, 409
- chown 338
- chroot 204
- chroot command 174
- common disk layout (CDL) 26
- common Linux tools 396
- Component Broker 491
- conclusion 513
- cpfmtxa 277
- cpio 474
- ctc.o 16
- Cygwin 153

## D

- DASD
  - partitioning 14
  - planning 13
- dasdfmt 129, 174, 203, 303
- date 369
- DB2 486
  - installing 487
  - verifying installation 488
- DB2 Connect 486
- db2sampl 489
- db2setup 487, 488
- db2start 489
- DDR 463
  - to back up Linux images 465
  - to restore Linux images 465
- Debian 213, 225, 228
- Debugging

- HMC diagnostic facilities 369
- debugging 347
- devfs 26
- df 398
- DIRECTXA command 29
- disabled wait state 278
- disaster recovery 473
- DISKMAP command 29
- DISPLAY environment variable 179
- distinguished name (DN) 376
- Distributions
  - for zSeries and S/390 xv
- distributions
  - Caiman 213
  - Marist 193
  - other 191
  - Red Hat Linux 151
  - SuSE Linux 55
  - Think Blue 64 239
  - Turbolinux 115
- dmesg 366
- DNS 380
  - MX records 187
- double paging 33
- dpkg 235
- dpkg-buildpackage 238
- dpkg-source 237
- dselect 225, 228, 235
- dsmc 478
- du 399

## E

- EBCDIC 7
- ECKD 108
- edquota 401
- Enterprise Java Beans (EJB) 491
- Exceed 113
- exim 233
- exportfs 145
- ext2 24

## F

- fake 339, 341
- fdasd 14, 26
- File system Hierarchy Standard (FHS) 228
- finger 436
- fixpack 495
- Flex-ES 13



Free Software Foundation 7, 153  
fsck 24, 301  
ftp 436  
ftpaccess 185  
ftpconversions 185  
ftpusers 185

## G

G1 12  
G2 - G6 12  
gdb 359, 369, 370  
Generalized Trace Facility (GTF) 356  
GFS 305  
Gigabit Ethernet 16  
Gnome 113  
gnome-linuxconf 179  
gnorpm 261  
GNU 4, 153  
GNU/Linux 7  
gnuplot 262  
grep 357

## H

hardening tools 457  
Hardware Configuration Definition (HCD) 31  
Hardware System Area (HSA) 31  
High availability 315  
HMC  
    booting from 35  
HMC diagnostic facilities 369  
htpasswd 420  
htpasswd command 104  
http 436  
httpd 143  
httpd.conf 103, 184  
hypervisor 269, 274

## I

I/O Configuration Program (IOCP) 32  
IBM JFS 343  
ICKDSF 276  
ICKDSF bootstrap 34  
IEEE Floating Point 12  
IFL 38  
image 269  
inetd 429, 436, 437  
inetd.conf 100

insmod 71, 129  
install.pl 121, 124, 129, 131  
install.sh 493  
installation worksheet  
    Caiman Linux 215  
    Red Hat Linux 162  
    SuSE Linux 65  
    Think Blue Linux 241  
    Turbolinux 122  
Integrated Facility for Linux (IFL) 13  
Integrated Server 12  
IOCDs considerations 17  
IPL  
    VM command 31  
ipvsadm 326, 327, 338  
ISO images  
    creating CDs from 57  
IUCV 270  
iucv.o 16

## J

Java 2 491  
JFS 25, 305, 343  
journalled file systems 24, 343

## K

KDE 113  
KDE2 242  
kdelibs 113  
kernel  
    rebuilding 467  
    recompiling 469  
klogd 365  
kmail 175  
knfsd 145  
kpackage 261  
ksymoos 364

## L

LAN Channel Station (LCS) 16  
lcs.o 16, 204  
LDAP 375  
    security 392  
ldapadd 386  
ldapsearch 383, 384  
ldif2dbm 384  
lib/cpp 204

- Linux kernel
  - 2.2.16 55, 156, 213, 214, 323
  - 2.2.19 151, 156
  - 2.4 261
  - 2.4.3 175, 176
  - recovering 472
- linux kernel
  - upgrading to 2.2.19 469
- Linux Virtual Server (LVS) 317
- linuxconf 151, 178, 179, 182, 251, 402
- LinuxKorea 7
- little-endian 25
- lockd 174
- locsite fix FTP subcommand 30
- log files 364, 456
- Logical Volume Manager (LVM) 231, 301
- login 437
- LPAR
  - preparing for Linux 31
- lvcreate 304, 306
- lvextend 307
- LVM 74, 138
  - basics 302
  - block diagram 302
  - in distributions 301
  - large logical volume 308
  - lvcreate command 304
  - lvextend 307
  - lvremove 307
  - modifying volume groups 111
  - pvmove command 311
  - sample session 303
  - setting up with YaST 108
  - striping 310
  - support 302
  - vgcreate command 304
- lvremove 307

## M

- m4 257
- Mail Transfer Agent (MTA) 107, 186, 256
- mainframe xv, 3
- major numbers 14
- man 357, 396
- man pages 396
- Marist 7
  - anonymous FTP 209
  - customization 206

- disk space requirements 196
- documentation 195
- getting 194
- history 194
- how it differs 195
- installation 196
- installing SSH 206
- root password 200
- using Apache 208
- xdm 209
- Mauelshagen, Heinz 108
- md5sum 56
- MI/X 113
- Millenux 7
  - history 239
- minor numbers 14
- mke2fs 111, 305
- mknod 472
- mk smbpasswd 99
- mk smbpasswd.sh 99
- mkswap 203
- modprobe 473
- mon 333, 338, 341, 343, 433
- Motorola 68000 5
- mount 111
  - loopback 57
- Mozilla 175
- MTU size 23, 69
- Multi Router Traffic Grapher (MRTG) 433
- multiple IPL volumes 207
- Multiprise 3000 12
- MySQL 485

## N

- Name Service Switch (NSS) 380
- Name Switch Service
  - setting up 386
- net use 99
- NetSaint 432
- netstat 399
- network management tools 408
- network setup
  - SuSE Linux 67
- networking
  - planning 15
  - virtual 16
- nfs 437
- NIS 380

nmap 459  
nmbd 95, 96  
nss\_ldap 386  
ntsysv 251

## O

objdump 348, 350  
OpenLDAP 378  
    master server 393  
    migrating data to 383  
    referrals 393  
    replica server 393  
    replication 393  
    starting the server 383  
    using 378  
OpenNMS 434  
OS/390  
    creating boot tape 45  
OSA cards  
    attributes 18  
    planning 16  
    resetting 21  
    types 17  
OSA-Express 174  
OSD 18  
OSE 18  
OSI 376

## P

P/390 12  
PAM 385  
    account argument 453  
    auth argument 453  
    configuring 451  
    password argument 453  
    session argument 453  
pam\_ldap 386  
pam\_ldap module 380, 385  
parameter file  
    size limits 159  
passwd 225  
passwd file 99  
password authentication 450  
password encryption 98  
passwords  
    encrypted 186  
patch 324  
Perl 334, 384

Physical Volume (PV) 302  
pico 236  
Pluggable Authentication Module (PAM) 380  
pop-3 437  
port scanning 459  
potato 214  
preparation  
    for Linux 27  
    of the VM guest 28  
PROFILE EXEC 29  
proftpd 234  
proftpd 143  
proftpd.conf 143  
proxy ARP 270  
ps 397  
pvcreate 303  
pvmove 311

## Q

qdio.o 16  
qeth.o 16  
quotacheck 401  
quotaon 401  
quotas 400

## R

R/390 12  
RACF 391  
    under z/VM 462  
RAMAC Virtual Array (RVA) 9  
rc.config 97  
rcsmb 98  
RDBM 391  
Red Hat  
    founders 5  
Red Hat Linux  
    anaging the network 183  
    common installation steps 165  
    configuring Apache 184  
    customizing and using 177  
    disk space requirements 158  
    documentation 155  
    enabling swat 186  
    getting 152  
    how it differs 156  
    installation in an LPAR 163  
    installation progress panel 172  
    installation under VIF 164

- installation under VM 162
- installing 157
- installing GNOME 188
- installing KDE 187
- managing FTP 184
- managing Samba 185
- network setup 166
- other installation types 156
- Overview 151
- parameter file 156, 159
- parameter values 160
- problems encountered 174
- stallation worksheet 162
- up2date 189
- Redbook
  - team xv
- Redbooks Web site 549
  - Contact us xix
- reiserfs 25, 175
- relative distinguished name 377
- Remstats 423
  - installation and configuration 425
- repquota 401
- resize2fs 113, 307
- resolv.conf file 146, 184
- rhn\_register 189
- rhsetup script 172, 175
- roadmap 26
- rpm 381
- rrdcgi 430
- rrdtool 424, 430
- rsh 297

**S**

- S/360 4
- S/370 4
- S/390 6
- Samba
  - adding a share 100
  - documents online 101
  - password encryption 98
  - security parameter 186
- scanlog 460
- Scotty 411
- scp 444
- SDBM 391
- securetty file 141
- security 435
  - basics 436
  - disabling servies 436
  - monitoring the Web 456
  - VM 460
- sed 296
- sendmail 107, 186, 256
- server consolidation 7
- setup 151
- setup command 178
- sftpserv 142
- shadow password 447
- shell 437
- shutdown 86, 464
- silo 135, 194, 231, 471
- slapd 378
- slapd.conf 381
- slogin 445
- slurpd 378
- smb.conf 96, 228
- smbd 95
- smbpasswd 99, 186
- smtp 436
- SNMP 409
- SNMP Tree 413
- snmpd 411
- snmpget 409
- snmpwalk 409
- socksify 152, 189
- srm.conf 103
- SSH 142, 440
  - daemon configuration 445
  - generating keys 441
  - key authentication 441
  - sources 447
- ssh 437
- sshd 142
- Stallman, Richard 4, 7
- stand-alone dump facility 368
- startkde command 113
- strace 356, 369, 370
- striping 310
- support 370
- Support Line 371
- SuSE
  - founders 5
- SuSE Linux
  - /etc/rc.config file 97
  - Apache files 102
  - Apache Web server 102

- creating IPL EXECs 66
- customizing and using 91
- disk space requirements 63
- documentation with 57
- history 55
- how it differs 60
- installation 61
- installation in an LPAR 88
- installation under VIF 89
- installation under VM 64
- installation worksheet 65
- network setup 67
- overview 55
- RPM series 58
- Samba files 95
- Samba installation 95
- using KDE 113
- SuSEconfig 87
- swap file 33, 292
- swap file systems 25
- swap partition 223
- swap space 33
- swapon 203
- SWAT 96, 144, 234
  - using 100
- syslogd 365
- system log 365
- system management tools 395
- System.map 363
  - replacing 471
- systems used for redbook 8

## T

- tape
  - booting Linux from 44
  - clearing volume label 45
  - creating on OS/390 45
  - creating on VM 50
  - using labelled 46
- tape devices
  - creating 472
  - planning 15
- tape driver
  - Turbolinux support 139
- tape support 467
- tape390 473
- tar 474, 480
- tbsetup 246

- TCP/IP
  - history 4
- TDBM 391
- telnet 126, 179, 221, 436
- thanks to section xvii
- Think Blue 7
- Think Blue Linux
  - disk requirements 242
  - how it differs 240
  - important files 249
  - installation 242
  - installation worksheet 241
  - Logical Volume Manager (LVM) 254
  - managing DASD 254
  - network configuration 252
  - network setup 244
  - parameter file 243
  - system configuration 249
  - tape support 261
  - using Apache 259
  - using FTP 259
  - using Samba 260
- time 436
- Tkined 411
- Tool Command Language (Tcl) 411
- top 397
- Torvalds, Linus xv, 4, 6
- tripwire 458
  - config-file 458
- TSM Client
  - installation 476
- TSM client 475
  - troubleshooting 479
  - using 478
- tsocks 152
- tune2fs command 174
- turbo.ins 129
- Turbolinux
  - customizing and using 141
  - DNS server files 146
  - documentation 118
  - founders 5
  - FTP server 143
  - install method 133
  - installation in an LPAR 126
  - installation under VIF 129
  - installation under VM 123
  - installation worksheet 122
  - installing 121

- LPAR install checklist 126
- LVM 138
- NFS server files 145
- overview 115
- parameter file 124
- planning DASD 123
- purchasing 117
- running install.pl 131
- Samba files 145
- secure shell files 142
- system administration 147
- tape support 139
- telnetting as root 134, 141
- version 6.5 136
- VM checklist 124
- VM IPL EXEC 124
- VM machine size 125
- Web server files 143

## U

- UDB 486
- ulimit 358, 359, 372
- University of Helsinki 4
- University of Michigan 375
- unix-status-server 429
- up2date 189
- upalert 333
- update-modules 230
- USER DIRECT file 29
- user directory 28
- USER DISKMAP file 29
- useradd command 183

## V

- Vepstas, Linas 6
- vgcreate 304
- vgextend 312
- vgscan 303
- VIF
  - applying service 288
  - block diagram 268
  - checklist 272
  - cloning Linux 295
  - cloning scripts 297
  - documentation 268
  - hutting down 299
  - hypervisor 269
  - install file 273

- installation with SuSE Linux 271
- installation with Turbolinux 271
- installation worksheet 279
- loading onto DASD 278
- master Linux 283
- networking options 270
- number of images 13
- overview 268
- paging space 271
- planning 269
- setting file system read-only 285
- template Linux image 289
- terminology 269
- using 287
- v script 288
- vif command 130
- VIP 317, 327
- virtual file systems 25
- Virtual Image Facility for Linux (VIF) 267
- VM
  - booting Linux from reader 31
  - creating IPL tape 50
  - DIRECTXA command 29
  - DISKMAP command 29
  - FTPping installation files to 30
  - getting installation files to 30
  - Installing SuSE Linux under 65
  - installing Turbolinux 123
  - IPL command 31
  - user directory 28
- VM diagnostic tools 368
- vmstat 398
- Volume Group (VG) 108, 302
- VS-NAT 318
- VS-TUN 322

## W

- Webmin 142, 148, 403
  - administrative tasks 406
  - downloading 148
  - installation 148, 403
- Webnin
  - prerequisites 148
- WebSphere
  - administration console 499
  - downloading 492
  - installing 492
  - sample gallery 500

- starting 498
- testing 498
- upgrading 495
- WebSphere Application Server (WAS) 491
- wget 153
- wget command 158
- Windows
  - accessing Samba from 99
  - supplying user ID from 99
- Windows 2000 221
- woody 214
- workload balancing
  - block diagram 317
- workload balancing, 316
- wu-ftpd 234

## **X**

- X.500 376
- XDM 146
- xdm 187
- XEDIT 367
  - setting to mixed case 30
- XEDIT PROFILE 29
- xfs 210
- xinetd 184, 250, 429, 436
- xosview 499

## **Y**

- YAST
  - setting up LVM with 108
- YaST 71, 402
  - completing installation 86
  - CREATING FILESYSTEMS panel 75
  - DASD formatting options 76
  - mask flag 93
  - sending function key values 73
  - specifying FTP server 78
  - SWAP PARTITION panel 74
  - terminal size 92
  - using 92
- yast 283
- YaST--mask argument list 94

## **Z**

- z/OS 387, 391
  - Cryptographic Services System SSL 391
  - LDAP server 391

- z/VM
  - number of images 13
- z900 12
- zilo 172, 178, 248, 254
- zilo.conf file 174
- zipl 368
- zSeries xv, 4







**Redbooks**

# Linux for IBM <sup>®</sup>server zSeries and S/390: Distributions

(1.0" spine)  
0.875" <-> 1.498"  
460 <-> 788 pages







# Linux for IBM server zSeries and S/390: Distributions



## SuSE, Turbolinux and Red Hat Linux on the mainframe

This IBM Redbook describes the Linux distributions available for the mainframe. It will help you to install, customize and maintain the following distributions:

## Installation under LPAR, z/VM and VIF

- SuSE Linux Enterprise Server for S/390
- Turbolinux server for zSeries and S/390
- Red Hat Linux for S/390
- Marist File System
- Caiman Linux
- Think Blue Linux

## Other distributions and topics addressed

We address installation and usage of Linux images on a logical partition (LPAR), under z/VM, and under the Virtual Image Facility (VIF). The following topics are also discussed:

- Managing DASD and file systems
- Logical Volume Manager (LVM)
- High Availability
- Debugging
- Lightweight Directory Access Protocol (LDAP)
- Systems management
- Security
- Backup and restore
- DB2
- Websphere Application Server

## 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](http://ibm.com/redbooks)