

Linux for S/390

How can Linux exploit the strengths of S/390?

What different ways can Linux be installed on S/390?

Which Linux applications can run on S/390?



Erich Amrehn, Joerg Arndt
Dave Bennin, Mark Cathcart
Richard Higson, Cliff Laking
Richard Lewis, Michael Maclsaac
Susan Matuszewski, Eugene Ong
Hans Dieter Mertiens, Eric Schabell

ibm.com/redbooks

Redbooks



International Technical Support Organization

Linux for S/390

September 2000

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix F, "Special notices" on page 501.

First Edition (September 2000)

This edition applies to Linux for S/390 Marist Version, 2.2.15 and SuSE Linux for S/390 pre-release distribution. At the time of writing this redbook there was no Linux for S/390 distribution available from TurboLinux.

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 2000. 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

Figures	xvii
Tables	xxi
Preface	xxiii
The team that wrote this redbook	xxiii
Comments welcome	xxvi
Chapter 1. Technology and business	1
1.1 The open source revolution	2
1.2 Technology directions for business innovation	4
1.3 Edge of Network devices and services	4
1.4 The Utility Service Provision (USP)	7
1.5 Complexity impedance	9
Chapter 2. Open source	11
2.1 The GNU Public License	12
2.2 The IBM Public License	13
2.3 Flourishing through open standards	13
2.4 The Application Framework for e-business	14
2.5 Summarizing open, shared standards	14
Chapter 3. Why Linux	17
3.1 Applications and Linux	19
3.2 IBM strategy for Linux	22
3.2.1 An introduction to Linux for S/390	25
3.3 Running Linux on S/390	26
3.4 Communication and connectivity	28
3.5 Other device support	29
3.6 Linux and S/390 benefits	30
3.7 Application scenarios with Linux on S/390	30
3.8 Tools and technologies	32
3.8.1 DB2 Connect	34
3.8.2 CICS	34
3.8.3 IMS	34
3.8.4 MQ Series	34
3.8.5 DB2 Universal Database (UDB)	34
3.8.6 Tivoli Framework end-point support	35
3.8.7 Tivoli Storage Manager Client	36
3.8.8 Summary	37

Chapter 4. Linux distributions	39
4.1 What a distribution is	41
4.1.1 Announced distributions	41
4.1.2 Distribution media	42
4.1.3 Roll your own	42
4.2 Linux documentation	42
Chapter 5. Native S/390 installation and operation of Linux	45
5.1 Assumptions	45
5.2 Skills and resources required	45
5.3 Hardware preparation	46
5.4 The hardware we used	47
5.5 Activating Linux for S/390	48
5.5.1 Creating an IPLable tape	48
5.5.2 Getting the files to your host system	52
5.5.3 JCL to create the tape	53
5.6 Using the Hardware Management Console (HMC) to IPL	57
5.7 Verifying the IPL from tape	64
5.7.1 IPL messages for Linux	65
5.7.2 Formatting DASD for Linux	66
5.7.3 Upload and customize the new file system	71
5.7.4 Creating and activating swap space	73
5.7.5 Customizing Linux for S/390 configuration files	75
5.7.6 Creating a new kernel	77
5.7.7 Write IPL information to DASD	78
5.7.8 ReIPL with the customized root file system on DASD	80
5.8 Linux for S/390 on a P/390	82
5.8.1 Attempting to install on a P/390	82
Chapter 6. VM installation and operation of Linux for S/390	85
6.1 Linux for S/390 in a virtual machine (as a guest of VM)	85
6.2 Installing Linux for S/390	86
6.2.1 Installation steps overview	87
6.2.2 Decide on the install method	87
6.2.3 Prepare the virtual machine to run Linux for S/390	90
6.2.4 Prepare the networking environment	93
6.2.5 Typical connectivity configuration	93
6.2.6 Obtain the binary files	94
6.2.7 Copy files to VM and reblock	95
6.2.8 Create the initial kernel parameter file	97
6.2.9 Boot initial Linux for S/390 kernel	98
6.2.10 Install the root file system	99
6.2.11 Complete the customization	99

6.3	Installing Marist College binaries	100
6.3.1	Install method	100
6.3.2	Linux for S/390 virtual machine definitions	100
6.3.3	Networking definitions	103
6.3.4	Downloading the binaries	104
6.3.5	Copying Marist files to VM and reblocking	106
6.3.6	Creating the kernel parameter file.	107
6.3.7	Boot the kernel	108
6.3.8	Install the root file system	115
6.3.9	Complete customization	116
6.4	Logging into your Linux for S/390 system	121
6.5	3215 driver considerations	121
6.6	IUCV connections.	122
6.7	Linux for S/390 device files and virtual device numbers	122
6.8	Operational considerations	125
6.8.1	Starting Linux for S/390 virtual machines	125
6.8.2	Stopping Linux for S/390 virtual machines	126
6.8.3	Secondary console interface.	126
6.8.4	Taking backups of Linux for S/390 file systems	127
6.9	Performance considerations	128
6.9.1	Reducing Linux for S/390 swapping	129
6.9.2	Virtual machine priority	131
 Chapter 7. Installing SuSE Linux on S/390		133
7.1	Types of installation	133
7.2	System requirements	133
7.2.1	Required hardware features	134
7.2.2	Required APARs and fixes	134
7.2.3	Software	135
7.3	Connection requirements	135
7.3.1	Console	135
7.3.2	Network connection	136
7.3.3	The telnet client	136
7.3.4	NFS or FTP server	137
7.4	IPLing the install system.	137
7.4.1	IPLing from the VM reader	138
7.4.2	IPLing from tape	138
7.4.3	IPLing from the CD-ROM (emulated tape)	139
7.4.4	The Load from CDROM or server task	139
7.5	Setting the network parameters in Linux.	140
7.6	Loading the DASD device driver.	141
7.7	Installing with YaST	142
7.7.1	Finishing the install when using a CTC network device.	162

7.8 Booting the installed system	163
Chapter 8. Linux for S/390 bootup and shutdown	165
8.1 Linux run levels	165
8.2 Kernel initialization	168
8.3 The init process and run level.	171
8.3.1 System init and inittab	172
8.3.2 Basic system initialization	174
8.4 Shutdown	175
Chapter 9. Linux for S/390 administration	177
9.1 Devices	177
9.1.1 DASD (direct access storage device)	179
9.1.2 VM minidisk	180
9.1.3 XPRAM	181
9.1.4 Creating a device node with mknod	182
9.1.5 Linux for S/390 device node assignment	183
9.2 File system types	185
9.2.1 Block size relation between device and file system.	186
9.2.2 The file system table /etc/fstab	187
9.2.3 Checking and repairing an ext2 file system: e2fsck.	188
9.3 Linux swap space	188
9.3.1 Creating swap spaces	189
9.3.2 Activating and deactivating swap spaces	189
9.3.3 Displaying information on swap spaces	189
9.3.4 Preparing swap space	190
9.4 File systems and devices	192
9.4.1 Formatting a block device: dasdfmt	192
9.4.2 Creating a file system: mke2fs	193
9.4.3 Accessing a file system: mount.	194
9.4.4 Making a device bootable: silo	195
9.5 Users and groups	197
9.5.1 Creating a user account: useradd.	198
9.5.2 Modifying a user account: usermod	199
9.5.3 Deleting a user account: userdel	199
9.5.4 Verifying the integrity of the passwd file: pwck	200
9.5.5 Creating a new group: groupadd.	200
9.5.6 Modifying a group: groupmod	201
9.5.7 Deleting a group: groupdel	201
9.5.8 Verifying the integrity of the group file: grpck	202
9.6 File ownership and access permissions	202
9.7 Changing passwords	203
9.8 Shells	203

9.9	System logs	203
9.10	Cron	206
9.11	Pluggable Authentication Module (PAM)	207
9.12	Interactive administrative utilities	208
9.12.1	Linuxconf	208
9.12.2	YAST	208
9.12.3	YAST2	209
Chapter 10. Backup		
10.1	The general concept	211
10.1.1	Backup strategies	211
10.2	Native backup commands	213
10.2.1	dump/restore	214
10.2.2	cpio	219
10.2.3	tar	222
10.3	Backup programs and tools	224
Chapter 11. System maintenance and upgrade		
11.1	Where to obtain software	225
11.2	Overview of upgrade strategies	226
11.3	Software installation with RPM	226
11.3.1	RPM overview	226
11.3.2	The RPM database	227
11.3.3	Querying package information	228
11.3.4	Checking dependencies	229
11.3.5	Install and update a package	230
11.3.6	Post-installation steps for source RPMs	230
11.3.7	Removing a package	233
11.4	Software installation with tar	233
11.5	Updating libraries	235
11.5.1	Upgrading shared libraries	236
11.5.2	Resolving incompatibilities	237
11.6	Build and customize the kernel	238
11.6.1	Preparing a second bootable device	239
11.6.2	Get the Linux kernel source	241
11.6.3	Recompiling the S/390 tool chain (binutils and gcc)	243
11.6.4	Preparing /usr/src/linux	244
11.6.5	Configure and compile the kernel	246
11.6.6	Install object code only (OCO) modules	249
11.6.7	Activate the new kernel	250
11.6.8	Post-installation steps	250
Chapter 12. Changing your root device		
12.1	Upgrading from Marist-2.2.xx to Marist-2.2.yy	253

12.2	Preparing a new volume	253
12.3	Summation	257
Chapter 13. Hardware connectivity		259
13.1	OSA-2	259
13.1.1	OSA-2 features	260
13.1.2	OSA-2 modes	261
13.2	The 2216 hardware interface	262
13.2.1	2216 ESCON channel adapter features	263
13.2.2	2216 ESCON channel protocols	263
13.3	CTC	264
13.3.1	CTC support	264
Chapter 14. Linux TCP/IP connectivity		267
14.1	Assumptions	267
14.1.1	Skills	267
14.2	TCP/IP protocols	267
14.2.1	Transmission Control Protocol (TCP)	268
14.2.2	User Datagram Protocol (UDP)	268
14.2.3	Internet Control Message Protocol (ICMP)	268
14.3	IP address types	269
14.3.1	Static IP addresses	269
14.3.2	Dynamic IP addresses	269
14.4	Configuration files	269
14.4.1	The hosts file	269
14.4.2	The services file	270
14.4.3	The protocols file	270
14.4.4	The HOSTNAME file	271
14.4.5	The inetd.conf file	271
14.5	The network script	272
14.6	Network daemons	272
14.6.1	Overview of inetd	272
14.6.2	Telnetd	273
14.6.3	Ftpd	274
14.6.4	Syslogd	274
14.7	Troubleshooting	274
14.7.1	The ping command	274
14.7.2	The netstat command	275
14.7.3	The ifconfig command	275
14.7.4	The route command	276
14.8	Access to data and applications	277
14.8.1	The telnet command	277
14.8.2	ftp	279

14.8.3	rlogin, rsh and rcp	279
14.8.4	ssh	279
Chapter 15. Linux for S/390 connectivity to VM, OS/390, VSE		283
15.1	Configuring the network	283
15.2	Logical partition	285
15.2.1	OSA-2 in LPAR	286
15.3	Linux for S/390 running in a virtual machine	286
15.3.1	Networking definitions	287
15.3.2	LAN Channel Station (LCS)	288
15.3.3	IUCV	291
15.3.4	CTC or IUCV	297
15.3.5	Linux for S/390 configuration files	298
15.3.6	VM TCP/IP configuration files	298
15.4	TCP/IP for OS/390 connectivity	300
15.4.1	Where to find daemons or services	302
15.4.2	Troubleshooting OS/390 TCP/IP to Linux for S/390	304
15.4.3	Inetd daemon in OS/390	306
15.5	Access to data and applications	307
15.5.1	OS/390	307
15.5.2	VM/ESA	312
15.5.3	VSE/ESA	314
Chapter 16. Development tools		317
16.1	Archiving and compression tools	317
16.1.1	The gzip command	317
16.1.2	The bzip2 command	317
16.1.3	The compress command	318
16.1.4	The tar command	318
16.1.5	The zip command	319
16.1.6	Other archiving tools	319
16.2	Compilers	319
16.2.1	The gcc and g++ compilers	319
16.2.2	Perl	320
16.2.3	Regina	321
16.3	Editing Linux files	321
16.3.1	The vi editor	321
16.3.2	emacs	322
16.3.3	joe, jove, pico	322
16.3.4	The sed editor	323
16.3.5	The pfe editor	323
16.3.6	The THE editor	323
16.4	Make tools	323

16.4.1	Make	323
16.4.2	automake	324
16.4.3	autoconf	324
16.5	Source code control tools	325
16.5.1	RCS and CVS	325
16.5.2	Kdevelop	326
16.6	Code analyzers	326
16.6.1	LCLint	326
16.6.2	Compiler code analyzer features	326
16.7	Debugging facilities	327
16.7.1	The gdb debugger	327
16.7.2	The Data Display Debugger, ddd	328
16.7.3	The MALLOC_CHECK_ environment variable	328
16.7.4	nana	328
16.8	The strace and ltrace tools	329
16.9	bash's -v and -x option	330
16.10	Performance analysis with gprof	330
16.11	lex and yacc	331
16.12	A simple example	332
Chapter 17. File Transfer Protocol (FTP)		335
17.1	Overview of FTP	335
17.2	TFTP	335
17.3	Anonymous FTP	335
17.4	Controlling access	336
17.4.1	Traditional FTP security	336
17.4.2	Anonymous FTP security	336
17.5	Converting files	337
17.6	Administrative tools	337
17.6.1	The tcpd command	337
17.6.2	The FTP daemon	338
17.6.3	The ftpaccess file	339
17.6.4	The ftpusers file	341
17.6.5	The ftpgroups file	341
17.6.6	The ftphosts file	341
17.6.7	The ftpconversions file	342
17.6.8	The ftpcount command	343
17.6.9	The ftpshut command	343
17.6.10	The ftpwho command	344
17.7	Client notes	344
17.8	A different FTP server - ProFTPD	345
17.8.1	Obtaining ProFTPD	346

Chapter 18. Domain Name Service (DNS)	349
18.1 Introduction to DNS	349
18.1.1 Assumptions	349
18.1.2 Skills	349
18.2 How it works, in theory	350
18.2.1 In action	351
18.3 DNS solutions on S/390	352
18.3.1 Using OS/390	352
18.4 Hardware and software setup	353
18.4.1 Software not included	353
18.5 Caching-only name server	354
18.5.1 Forwarding out of a protected network	355
18.5.2 Configuration files	355
18.5.3 Starting named	359
18.5.4 Testing with nslookup	360
18.6 Tools	361
18.6.1 dig	361
18.6.2 The dnsquery program	362
18.6.3 The host program	363
18.6.4 The nslookup tool	364
18.6.5 The nsupdate command	364
18.7 Summary	365
Chapter 19. Network File System (NFS)	367
19.1 Installation of the server	367
19.2 Customizing	367
19.2.1 NFS Server	368
19.2.2 NFS Client	368
19.3 Operation	369
19.3.1 As a server for AIX	370
19.3.2 As a server to OS/2	370
19.3.3 As a server for OS/390	372
19.3.4 As a client to OS/390	374
19.3.5 As a client to VM/ESA	376
19.3.6 As a client to VSE/ESA	377
19.3.7 Data representation considerations	377
19.3.8 OS/390 access security	378
19.4 VM/ESA access security	385
Chapter 20. Samba	389
20.1 Installation	389
20.1.1 Installing from an RPM binary or source	389
20.1.2 Installing from source in the original package	389

20.2	Customization	393
20.2.1	Starting Samba automatically	393
20.2.2	Starting SWAT automatically	393
20.2.3	Using SWAT to customize Samba	394
20.2.4	Additional resources	395
20.2.5	Client-side operation	396
20.2.6	Finding a Samba server from Windows	396
20.2.7	Supplying the proper credentials from Windows	397
20.2.8	Accessing a Samba share from a DOS prompt	398
20.2.9	Accessing a Samba share from a GUI	399
20.3	Tools	399
20.3.1	SWAT	399
20.3.2	smbclient	402
Chapter 21. The Apache Web server		405
21.1	Installation	406
21.1.1	Obtaining a later Apache level	406
21.1.2	Exploding the Apache tar file	406
21.1.3	Building Apache	407
21.1.4	Customization	410
21.1.5	Server configuration settings	411
21.1.6	Virtual hosting	412
21.1.7	Operation	412
21.1.8	CGI and SSI	414
21.1.9	SSL	417
21.1.10	Web Server security considerations	417
Chapter 22. Firewall configuration		421
22.1	Installation	422
22.2	Customization	424
22.3	Operation	429
Chapter 23. Printing with Linux		431
23.1	Devices	431
23.2	Using VM resources	432
23.3	Using OS/390 resources	433
23.4	Using Linux as a print data hub	434
Chapter 24. Linux security issues		435
24.1	Consider using remote logging	435
24.2	Disable unnecessary services	435
24.3	Files and file system security	436
24.4	Disable remote login for root	437
24.5	Use encrypted connections	437

24.6	Use scp instead of FTP	437
24.7	Use a tcp wrapper (tcpd)	438
24.8	Use shadow passwords	438
24.9	X11 server access control	439
24.10	Consult the security-related Internet sites regularly	439
Chapter 25. Sources of help and information		441
25.1	man pages	441
25.2	info	442
25.3	help	442
25.4	howto	443
25.5	RFC	443
25.6	The Internet	444
25.7	Books	444
25.8	Finding a file	444
25.9	Determining the type of command	445
25.10	DAU	445
Chapter 26. Monitoring the system		447
26.1	Linux facilities and tools	447
26.1.1	log files	447
26.1.2	The proc file system	447
26.1.3	top	448
26.1.4	ps	448
26.1.5	pstree	448
26.1.6	who and w	449
26.1.7	xosview	450
26.1.8	xload	450
26.1.9	procmeter	450
26.1.10	uptime	450
26.1.11	free	450
26.1.12	du and df	451
26.1.13	fuser	451
26.2	VM tools	451
Appendix A. Intel architecture, S/390 architecture		453
A.1	Architecture description	453
A.1.1	IA32	453
A.1.2	S/390	456
A.1.3	I/O subsystem	459
A.2	Symmetric multiprocessing	460
A.2.1	Intel SMP	460
A.2.2	IBM SMP	462
A.3	RAS considerations	464

A.3.1 Intel Profusion chip set RAS considerations	464
A.3.2 S/390 RAS considerations	464
A.4 Comparing the IA32 and S/390 architectures	466
Appendix B. VM/ESA virtual machines	469
B.0.1 The CP directory	470
B.0.2 Processors	472
B.0.3 Storage	472
B.0.4 Minidisks	473
B.0.5 Reader, punch, printer	473
B.0.6 The console	474
B.0.7 Channel-to-channel device	474
B.0.8 Virtual I/O	474
B.0.9 CMS	475
Appendix C. Linux for S/390 I/O implementation	477
Appendix D. The parameter file	481
D.1 DASD	481
D.1.1 Syntax	481
D.1.2 Example	482
D.2 Mdisk	482
D.2.1 Syntax	482
D.2.2 Example	482
D.3 Root	482
D.3.1 Syntax	483
D.3.2 Example	483
D.4 Xpram	483
D.4.1 Syntax	483
D.4.2 Example	484
D.5 Ctc/Escon	484
D.5.1 Syntax	484
D.5.2 Example	484
D.6 IUCV	485
D.6.1 Syntax	485
D.6.2 Example	485
D.7 3215 Line mode terminal	485
D.7.1 Syntax	485
D.7.2 Example	486
Appendix E. Troubleshooting and avoiding pitfalls	487
E.1 Cannot boot big file system on VM - /etc/fstab not modified	487
E.2 Editing /etc/fstab with vi - be sure the last line has a newline	487
E.3 Irritating RPM messages/RPM update with RPM	488

E.4 Native Linux silo command - use the proper flags	488
E.5 Linux under VM won't boot - forgot to ftp files in FB80	489
E.6 Linux under VM won't boot after improper shutdown	489
E.7 Use the -c flag with the ping command.	490
E.8 Linux under VM - can't find the vertical bar on the keyboard	490
E.9 Rerun silo after changing the kernel parameter file	490
E.10 Linux under VM - reserve the minidisk	490
E.11 Linux under VM - format the minidisk	490
E.12 Linux under VM - IPL hangs	491
E.13 Device not registered by the kernel	491
E.14 Cannot mount file system - block sizes not the same.	491
E.15 Disk device ranges in kernel parameter file	492
E.16 RAM disk full	492
E.17 The Virtual CTC connection does not start.	492
E.18 Bad superblock	493
E.19 Error when running dasdfmt	493
E.20 minidisk.sh	494
E.21 The script with the networking questions is gone	495
E.22 MTU size problems	499
Appendix F. Special notices	501
Appendix G. Related publications	505
G.1 IBM Redbooks	505
G.2 IBM Redbooks collections	505
G.3 Other resources	506
G.4 Referenced Web sites	507
How to get IBM Redbooks	511
IBM Redbooks fax order form	512
Index	513
IBM Redbooks review	521

Figures

1. Infrastructure for e-business	6
2. The Application Framework for e-business	14
3. The open standards platform	15
4. Application infrastructure	20
5. Application deployment platform pyramid	22
6. IBM UNIX/Linux strategy	24
7. Linux on S/390 options	26
8. Linux on S/390 structure	40
9. Allocation attributes of the parmline file	49
10. Win95/NT FTP example	52
11. OS/2 FTP example	53
12. Size of parmline file on tape	56
13. HMC CPC Images Work Area	58
14. Load with unit address of tape drive	59
15. cat /var/log/dmesg	60
16. HMC display of network initialization	61
17. Delaying tr0 initialization	62
18. HMC Send Command dialog	63
19. cat /proc/cpuinfo command	64
20. cat /proc/meminfo command	64
21. cat /proc/interrupts command	65
22. Output of the df command	65
23. Output of the ifconfig command	65
24. cat /var/log/dmesg command	66
25. cat /proc/devices command	67
26. Major number 94 - DASD	68
27. cat /proc/dasd/devices command	69
28. mknodes already performed on Marist file system	69
29. Tar uncompress of big file system	72
30. cat of /proc/swaps	75
31. Customized fstab	75
32. cat /etc/sysconfig/network-scripts/ifcfg-tr0	76
33. cat /etc/resolv.conf	77
34. Silo command warning message	79
35. Successful silo command	80
36. Shutdown messages on the HMC	81
37. Telnet sessions on shutdown	81
38. Load profile for IPLing Linux for S/390 on DASD	82
39. DASD partitioning scheme	89
40. Sample CP directory definition for Linux for a S/390 virtual machine	91

41. Networking configuration with Linux for S/390 running in a virtual machine	94
42. Matching Linux for S/390 and VM TCP/IP definitions	104
43. Marist College Linux for S/390 download Web site	105
44. Paging in a three-tier storage model	129
45. YaST: warning message about a small terminal size	143
46. YaST: language selection	143
47. YaST: selection of the installation medium	144
48. YaST: entering the data for the NFS server.	145
49. YaST: selecting the installation menu	145
50. YaST: select swap partition menu	146
51. YaST: creating filesystems menu	146
52. YaST: the mount point menu	147
53. YaST: the format mode menu	147
54. YaST: confirmation before actually creating file systems	148
55. YaST: reading the description data	149
56. YaST: The main installation menu.	149
57. YaST: the load configuration menu	150
58. YaST: the various software packages	151
59. YaST: individual packages in series n	151
60. YaST: automatically resolving dependencies	152
61. YaST: some dependencies cannot be resolved automatically	153
62. YaST: the consequences menu.	153
63. YaST: the installing package menu	154
64. YaST: package installation finished.	154
65. YaST: selecting a kernel	155
66. YaST: list of time zones	155
67. YaST: local time or GMT	156
68. YaST: hostname and domain name	156
69. YaST: real network selection	157
70. YaST: DHCP server choice	157
71. YaST: network devices	158
72. YaST: network addresses	158
73. YaST: inetd	159
74. YaST: portmapper	159
75. YaST: nameserver configuration	160
76. YaST: network device and module	160
77. YaST: sendmail configuration	161
78. YaST: SuSEconfig is running.	161
79. Block usage w/ different block sizes, DASD format & file system make	187
80. Kernel configuration with menuconfig	247
81. OSA-2 ENTR, FDDI, ATM, and FENET features.	259
82. IP network topology	285
83. Where to enter network definitions in Linux for S/390 and VM TCP/IP	287

84. OSA-2 connections for Linux for S/390 running in a virtual machine. . . .	288
85. CTC connections for Linux for S/390 running in a virtual machine	290
86. IUCV connections for Linux for S/390 running in a virtual machine. . . .	293
87. Two-TC/IP-stack configuration for OS/390	301
88. Package build process.	325
89. Browser's view of Anonymous FTP server	340
90. Internet domain name space (partial representation)	351
91. The named.conf file	356
92. The 127.0.0 file	358
93. A dig sample	362
94. A dnsquery sample	363
95. A host example	364
96. Mounting file system on Linux for S/390 from OS/2 client.	371
97. Mount/unmount scenario Linux as a NFS Server for OS/390	373
98. Incremental backup of files on Linux from OS/390	374
99. Mount OS/390 data with a binary view	377
100. Mountings for text and binary view of data	377
101. Directory listing of hfs as seen from Linux.	381
102. Directory listing of /bin on hfs as seen from Linux.	382
103. Contents of /u/hdm directory on OS/390	383
104. Creating test data from the client	383
105. Mounting an OS/390 data set on Linux for S/390	384
106. Display of a mounted OS/390 data set	385
107. Building open source software packages	390
108. SWAT main window	395
109. SWAT password management interface.	401
110. SWAT password management without UNIX synchronization	402
111. SWAT with UNIX password sync setting.	402
112. Apache webserver testpage	411
113. Downloading from the Internet onto the Linux server	431
114. Printing from Linux to VM-managed printers.	432
115. Address translation on IA32 processors	454
116. Translation of a virtual address to a real one	458
117. The Profusion chip set.	461
118. The binodal cache	463
119. S/390 system resources	469
120. Virtual machines running under the control of a hypervisor	470

Tables

1. DASD devices	50
2. What you really get with “dasd” not specified	51
3. “dasd” specified with correct devices	51
4. Using a range on the “dasd” statement	51
5. OS/390 UNIX System Services HFS vs. Linux for S/390 customization	73
6. Device managers and emulated S/390 devices.	90
7. Record lengths for Linux for S/390 boot files	95
8. FTP subcommands for reblocking files	96
9. Details of Linux for S/390 minidisks.	101
10. Pairing device numbers in CP COUPLE commands	103
11. Description of files at Marist College Linux for S/390 download site	105
12. Files required by the install method	106
13. Linux for S/390 device names and S/390 device numbers	124
14. New Linux for S/390 device names and S/390 device numbers.	124
15. Required APARs and fixes	135
16. Run levels	165
17. Characteristics of S/390 block devices	179
18. Device node characteristics for S/390 devices	183
19. device node association with auto detect	184
20. Basic characteristics of the ext2 file system	186
21. Commands to create different types of swap spaces	190
22. From a raw device to processes accessing file systems	192
23. Comparison of backup commands	213
24. Backup level concept.	215
25. URL references to Linux for S/390 service	225
26. OSI model	268
27. Linux for S/390 network driver choices	283
28. Executables and documentation installed with bind8	354

Preface

The strengths of S/390 are well known: rock-solid reliability, the ability to run multiple diverse workloads, and highly scalable technology make S/390 an ideal choice for hosting key e-business applications. Now Linux has joined the S/390 family of operating systems, bringing a wealth of open source applications, middleware, and trained developers to help you respond to your business challenges quicker than ever before.

This IBM Redbook is aimed at beginners and intermediate Linux users with a S/390 operating system background. It covers Linux for S/390 Marist distribution (2.2.15) and a prerelease of the SuSE Linux for S/390 distribution. At the time of writing, there was no distribution available from TurboLinux.

The first four chapters offer an overview of Linux's origin, how it fits into the IBM strategy, what open source means, and why using Linux for S/390 is significant from an IBM perspective. These chapters are also suitable as a management overview.

The main part of the book will help you install Linux for S/390 in different environments. It discusses basic system administration tasks that can help you manage your Linux for S/390 system. It also introduces a wide range of services, such as Samba, NFS, and Apache. You will learn what each service is, what it is capable of, and how to install it. The services are not covered in detail, since they are very comprehensive; however, sources for more detailed information are documented.

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.

Erich Amrehn is a certified IT Specialist at the International Technical Support Organization, Poughkeepsie Center. Before joining the ITSO, he worked as technical consultant to the IBM System/390 division for e-commerce on S/390 in Europe, the Middle East and Africa. He also has 13 years of VM experience in various technical positions in Germany and other areas in Europe, as well as worldwide.

Joerg Arndt is a software developer who works for SuSE (Germany). He has 16 years of experience in software development, consulting and teaching, and has worked exclusively with Linux since 1994. He holds a degree in

theoretical physics from the Technical University of Berlin. His areas of expertise include Linux programming and scientific programming (see <http://www.jjj.de/>). He has written on time series analysis, Fast Fourier Transform (FFT) algorithms, and the number pi.

Mark Cathcart is technology strategist to IBM Corporate Technology Marketing where he has responsibility for Linux, Open Source, Java, and XML. Previously, he was Principal Consultant for OS/390 New Technology including Java, EJB, Component Software, e-business et al. His first presentation to IBM on XML was in late 1997, almost two years before the official IBM XML launch in 9/99. Mark did the first ever demonstration of linking Java to legacy systems at WWW5 in Paris in 1996, and more recently (8/98) was part of the IBM Academy of Technology project to redesign Java for IBM servers. Mark is a member of the IBM S/390 Software Design Council and an IBM UK Technical Staff Member (<http://www.ibm.com/s390/corner>).

Richard Higson is a consultant for Syskoplan AG (Germany). He worked as a system developer for mini-sized multiuser systems in the “pre-PC era” before becoming a systems programmer with VM/VSE. He then specialized in networking (SNA, TCP/IP) and UNIX. He has been using Linux since early 1993, put Syskoplan on the ‘net in 1994 with Linux, helped build “the world’s largest Linux Cluster” (Paderborn, December 1998), and spoke about this at the August 1999 Open Software Conference in Monterey. He became involved with Linux for S/390 in December 1999. He is an active supporter of the Debian GNU/Linux distribution.

Cliff Laking is a Senior Technical Specialist who works for IBM in the United Kingdom. He has over 25 years of experience in Information Technology. He holds a Bachelor of Arts degree from Victoria University of Wellington, New Zealand. His IBM career has been spent working primarily with the VM/ESA operating system. Other areas of interest include APPC programming, the IBM Network Station, and S/390 LAN integration tools such as LFS/ESA and Tivoli ADSM. He now focuses mostly on e-business solutions for S/390.

Richard Lewis is a Consulting I/T Specialist with the IBM Washington System Center. He has worked with the VM operating system for the past 16 years within IBM, both as a system programmer and technical support specialist. Richard has been active within the VM user community both as an IBM representative to the Guide VM Group, and as an IBM representative to the VM Cluster at SHARE. He has given numerous presentations at these venues over the years, covering a wide variety of topics related to the VM operating system and VM program products. Richard has been working with Linux for S/390 since late 1999, and created the first hands-on Linux for S/390 Installation Workshop offered by IBM.

Michael Maclsaac is a team leader for S/390 redbooks and workshops at the ITSO Poughkeepsie Center. He writes about and teaches classes on OS/390 UNIX and now Linux. Michael has worked at IBM for 13 years, mainly as a UNIX programmer.

Susan Matuszewski is a system programmer at GAD Gesellschaft für automatische Datenverarbeitung e.G. in Germany. She worked as a consultant and programmer on software development projects and has 11 years of experience on OS/390 (MVS), including UNIX System Services. Her areas of expertise include implementing AIX and different Linux distributions. Additional areas of interest include object-oriented programming and Internet solutions.

Hans Dieter Mertiens is a Senior Technical Marketing Specialist in Germany. He holds a diploma in physics from the University of Hamburg. He worked at several IBM installations as a system programmer and systems analyst before joining IBM in 1984. Since 1991 he has worked for the S/390 division, focusing mainly on new functions of MVS or later OS/390, like APPC/MVS and later POSIX. His areas of expertise include UNIX System Services on OS/390, application porting, and integrating UNIX Services into the traditional OS/390 world. He has participated in several residencies dealing with UNIX and OS/390 at the ITSO Centers in Poughkeepsie and Raleigh.

Eugene Ong is an Advisory Software Engineer who has worked for IBM since 1987. He spent 11 years in large system software design and development, which included products such as TSO/E, APPC/MVS, APPC/RRS, and System Logger. He joined IBM's Customer Enablement Lab (CEL) in 1998. His present areas of expertise in the CEL include benchmarking and proofs of concept in areas such as UNIX System Services, Webserver, DB2, and TCP/IP. He holds a masters degree in Computer Science from the Polytechnic University of New York. This was his first residency at the ITSO.

Eric Schabell is an IT Specialist working for IBM in the Netherlands. He has two years of experience in IBM working in the S/390 environment and more than four years with Linux. He is working on a degree in Computer Science from the Vrije Universiteit in Amsterdam, Netherlands. His areas of interest include Linux, UNIX, DB2, and networking. He is currently the Linux Knowledge Center leader for IBM Netherlands.

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

Boas Betzler, Dr. Holger Smolinski, Martin Schwidewsky
IBM Germany

Dave Bennin, Roy Costa
International Technical Support Organization, Poughkeepsie Center

Romney White, Stephen Record
IBM Endicott

Bernd Kaindl, Marcus Kraft, Joachim Schroeder
SuSE Germany

Sándor Bárány
M&M Investitionsberatungs GmbH Austria

Thanks also to Alfred Schwab, Terry Barthel and Alison Chandler for their editorial assistance, and to Ella Buslovich for her graphics assistance.
IBM Poughkeepsie

Comments welcome

Your comments are important to us!

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

- Fax the evaluation form found in “IBM Redbooks review” on page 521 to the fax number shown on the form.
- Use the online evaluation form found at <http://www.redbooks.ibm.com/>
- Send your comments in an Internet note to redbook@us.ibm.com

Chapter 1. Technology and business

Whether your company is a “dot-com” or a bricks and mortar company, the Internet and e-business have changed everything—from the way you interact with and reach your customers—to how you work with your partners, your suppliers, their suppliers, your contractors, and your subcontractors. In fact, the Internet and e-business have changed the products, processes, and procedures that make up our very organizations.

Because of these changes, we all face an era of business innovation, an era in which opportunity and complexity may go hand in hand. However, it doesn't have to be that way. Through the adoption of truly open industry standards, we can reduce and eliminate much of the complexity and allow business innovation to flourish. A discussion of the approaches to open industry standards can be found in 2.3, “Flourishing through open standards” on page 13.

Yet even with this explosion of change, we are witnessing only the *emergence* of e-business! The initial solutions and approaches resemble the initial steps of a child—random, somewhat uncoordinated, many taken very quickly with occasional stumbles. Yet, as it is with a small child, this initial stage is an essential part of the growth and maturity of e-business, and of the next generation of e-business solutions.

Some people feel that, as with a child, e-business will take 20 years to become fully mature, fully useful, and able to fulfill its destiny. Whether or not you agree with this assessment, it's apparent that we are at the crossroads of a number of interdependent generations of technologies. Web browser and Web server technologies, wireless and mobile devices, as well as Web application programming all serve the Internet/e-business arena.

Yet all these developments face a rapid succession of changes and progression as the new business requirements of the Internet and new technology force a never-ending succession of revisions, extensions, and reexamination of what is used, how it is used, and how it can be applied.

In this environment there appear to be a number of established and emerging technologies that will continue to evolve at a rate and in a way that can benefit users, developers, and suppliers, and which seem to offer the biggest and best opportunity to support the rapid progression and evolution of the Internet and e-business toward maturity.

1.1 The open source revolution

The technologies that offer the biggest benefit and most promise in this era of evolution and rapid change are firmly based in the world of *open source*. In Chapter 2, “Open source” on page 11, we examine what open source is, what it means, and what it doesn't mean. However, while open source software development has existed in one form or another since the first software programmable computers were invented, open source and its antecedents have become especially important with the advent of the Internet.

The *Internet* came into existence because of a mix of formally and informally agreed-to *standards* that are vendor-neutral and universally accessible. These standards, by definition, were a mix of established and new technologies such as the File Transfer Protocol (FTP), Gopher, the HyperText Transfer Protocol (HTTP) and the HyperText Markup Language (HTML). They all flourished on an established but growing base of TCP/IP networking protocols and network interoperability layers.

The explosion of the *World Wide Web* into popular consciousness came about through a set of standards implemented through an open academic development model that shared research work and provided for a series of rapid, successive improvements. Since the initial implementations were available in the form in which they were created (programmer source code), they could be easily adapted and adopted, and problems could be identified and resolved quickly.

Today, almost all implementations of common Web-based tools can be traced back to these early efforts. Many still use some source code from these original implementations, and some of these implementations have gone on to become market leaders and provide direction for that technology.

One example of this is the Apache Web server. Depending on the date and author of the latest Web server usage study, the Apache Web server runs something more than 60% of all Web servers. Apache is a direct descendent of the original Web server and owes much of its market share to the fact that it is available in source code and can be adapted, fixed, and extended by a large community of dedicated programmers based around the world (and thus around the clock), at academic, not-for-profit, and commercial organizations.

Another example of this evolutionary technology is the Linux operating system. Like Apache, Linux (or, more correctly, GNU/Linux) is available in programmer source code. This has enabled Linux to move from being an academic project by a single programmer, to being another key technology

enabler for current and future Internet and e-business projects. (For more information about GNU, refer to Chapter 2, “Open source” on page 11.)

This redbook examines Linux running on IBM System/390 hardware. Later, in 3.3, “Running Linux on S/390” on page 26, we review what it is and how it can be used, including what modes it can be run in on S/390-compatible hardware.

Also in Chapter 3, “Why Linux” on page 17, we examine why Linux has emerged, what purpose it serves and why it will continue to grow and evolve. Its use will continue to expand through the evolution and business innovation that will be the mainstay of technology changes in the next few years.

Is this because Linux is a better operating system? No. Is it because Linux is openly available across a wide variety of systems? Partially. Is it because Linux is available in source code? Yes. Essentially it is that Linux is available as a set of shared, open standards that will allow it to evolve and improve into a better operating system. Through its availability in source code format, it is possible for Linux to support a wide variety of hardware systems.

But most importantly, like the Internet that it so adeptly supports, Linux is a mixture of established open standards and protocols and emerging standards that are Linux itself.

This was positioned for analysts by IBM Chief Executive Lou Gerstner on May 9, 2000, when discussing IBM's e-business strategy and business performance.

When discussing imperatives, he said:

“The first is host or operating system software, which is now and always will be a continually shrinking percentage of our revenue. We're not fretting about this because, frankly, operating systems no longer hold the strategic importance they once had in our industry. In a world of open standards, which is where the world is going, the operating system platforms—ours or anyone else's in the open world—are not going to be control points anymore.

They won't be where the growth is, and they won't even command the margins they once did.”

We look at this further in Chapter 3, “Why Linux” on page 17, but the point here is to establish that it isn't an operating system, it is a set of truly open, shared standards that makes the further use and growth of Linux inevitable.

To fully understand the role of Linux and open, shared standards, we need to look at some technology directions before finally returning to look at some potential pitfalls.

1.2 Technology directions for business innovation

A number of things can be said to be “universal truths” about computer technology:

- The big keeps getting smaller
- The fast keeps getting faster
- Adaptation is as good as adoption

Who would have thought just a few years ago, when IBM took on and beat the World Chess champion, that the power provided by a highly complex, large server would be available at an affordable cost in a desktop configuration?

Looking even further back, it was at one time inconceivable that the world would need more than six mainframe computers, let alone that the power of those early mainframe computers would be surpassed by the computing power in a digital watch! It was also the case that it was once believed that just 640 thousand bytes of computer memory would be “more than enough,” when today’s inexpensive computer systems are routinely supplied with 64 million bytes!

As business innovation looks to technology for opportunities for greater market presence, improved shareholder value, and ultimately increased profitability, the exploitation of these smaller, faster and more powerful technologies is at the leading edge of this business innovation.

1.3 Edge of Network devices and services

These technologies will increasingly find their way into a wide variety of wireless, mobile, and *connected* devices. Support of these Edge of Network devices will require a new breed of servers, both local and remote, and a new breed of solution.

Businesses and technology companies have learned through the first generations of the Internet and recent technology innovations that one size does *not* fit all.

Regardless of what operating system you apply this to, IBM AIX, OS/400, OS/390; any commercial UNIX derivative, such as Sun Solaris, Hewlett Packard’s HP-UX, and especially Microsoft-based operating systems such as

Windows NT, Windows 95, Windows 98, Windows CE—when applied to the various computing models in current and emerging use, there were significant constraints or drawbacks when trying to apply an existing operating system to these models, let alone to span technology models.

Often the reason for this failure was an unwillingness to release, or give up, a significant part or portion of the operating system. Because the operating system was not a shared or open standard, including most UNIX derivatives, it could not be changed or redefined by anyone except the vendor or owner. This established a control point that ultimately worked against the best interest of both the operating system and the vendor.

Linux, by contrast, through its availability as open source code (that is, its implementation as a set of shared standards), offers the potential for the deployment of common tools and technologies for the next generation of e-business.

In Figure 1 on page 6, we see the evolution of an architecture for a *Next Generation e-business* infrastructure—a variety of types and classes of computing devices:

- Client devices—traditional desktop systems, mobile and wireless solutions, and appliance-type devices
- Server service providers—single function appliance, presentation, directory, security servers
- Web application servers
- Traditional business transaction and data servers

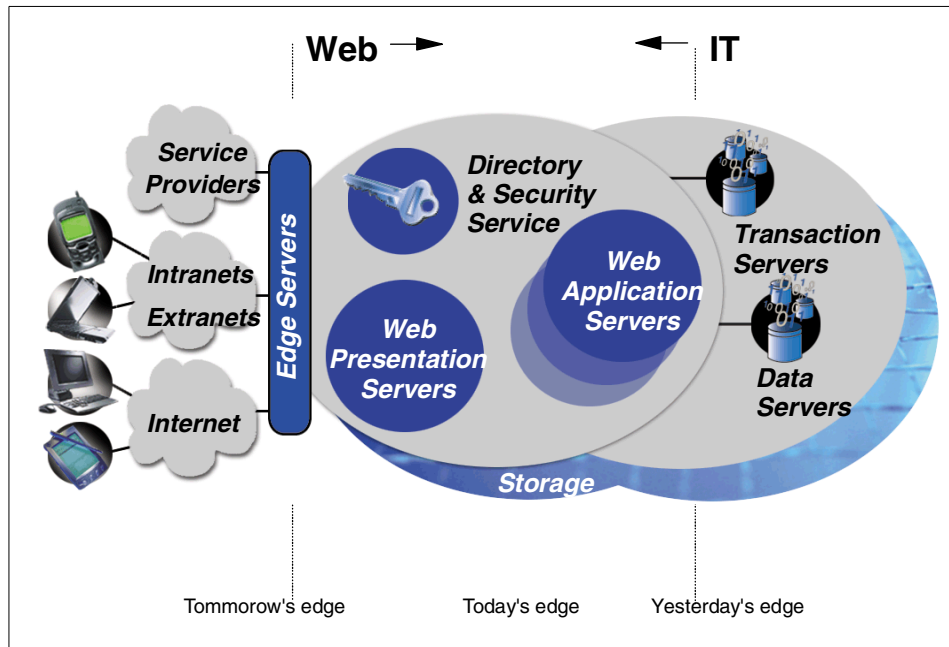


Figure 1. Infrastructure for e-business

Many of these will exist both internally in an organization and externally at partners and suppliers, and increasingly many more will also be on the other side of the Internet at consumers and customers.

The new devices, especially mobile, wireless and appliance-based devices, are sometimes referred to as being at the *edge of the network*. That is, they are loosely connected into what your organization would consider to be the very outside edge of its corporate network extended to the Internet.

To support these devices, a new class of simple, quickly implemented and often single-function servers will be needed. These are the *edge servers* seen in Figure 1. In some cases it is reasonable to expect these servers to be at the edge of your network. Again, some may be deployed directly at customer, consumer, and partner locations.

Many of these new types of servers will also exist on the border between your Information Technology infrastructure and your Web infrastructure, and will require qualities of service and availability traditionally associated with *transactional* systems. These edge servers will fulfill a number of roles, such as messaging servers that directly support appliances outside your network. They will also provide caching, content filtering, load balancing, and quality of

service routing. In addition there will be a role for application servers to respond to requests and messages from mobile, pervasive, and appliance-based systems.

These appliance servers will need to be integrated into your environment, but for the sake of business innovation this cannot afford to be a long, costly and complex process. You'll need to act quickly and decisively when it comes to deploying these servers.

So, where might Linux play in this scenario? Simply put, based on the organization and the business need, everywhere: clients, server service providers, Web, directory, security, application, database, and transaction servers.

Many organizations, though, will have requirements for components of this solution to run with different scalability, availability, and integration requirements. Supporting a few hundred Edge of Network devices is very different from supporting, say, ten thousand, one hundred thousand, or potentially millions and millions of devices.

This is where the IBM Linux strategy—a simple strategy to endorse and support the tools, technologies and open vendor-neutral standards of Linux—becomes very attractive. In 3.2, “IBM strategy for Linux” on page 22, we look at the role and opportunity for IBM S/390.

1.4 The Utility Service Provision (USP)

A key attribute required to support this new world of mobile, wireless and *edge* computing, is the provision of a wide range of services, services that have been converged into a single service entity that does not have the artificial separation of applications common in the computing marketplace today.

Often, people discuss Internet Service Providers (ISPs), Application Service Providers (ASPs), applications like Customer Relationship Management (CRM), Enterprise Resource Planning (ERP), Business Intelligence (BI), and processes such as server consolidation, as if they were entirely separate applications and attributes of computer systems. They are not. Convergence of these technologies and applications will be required as the service provision of computer processing becomes much more like that of a *utility*.

An important attribute of the Utility Service Provision (USP) is that it is *always on*. It is always on because that is what consumers will expect, since the clients will always be on. The provision of an “always on” service isn't just an

extension of 24 x 7 high availability; it is a statement about high availability not measured in hours, days, or weeks, but in years, perhaps even decades.

This high availability is then combined with applications that can process a request *straight through*, connecting directly and indirectly to back-end systems, to databases, to partners, suppliers, finance organizations, etc., so that a single request can be completely fulfilled within seconds or minutes, not in one or two days or even longer.

For example, when a refrigerator calls in for service, it is not a simple matter of just recording the call. It requires coordination with:

- The scheduling and dispatching system that sends out an engineer
- The parts supply system to make sure the parts are available, which in turn works with the supply chain to make sure a replacement is ordered if needed
- The product history data to record the call and do product trend analysis
- Billing, accounting, and more

The point is that what might seem a simple interaction can spawn hundreds of events, each event can spawn its own events, and so on, both internal to your organization and externally.

To support this type of an environment, servers need to provide attributes that are often associated with S/390 hardware and software: workload management to make sure that important work is carried out as required by service level agreements; coordination of priority with other concurrent processes; logical partitioning of resources to systems and subsystems so that appliance servers, Web application servers, transaction and database servers can themselves be serviced. Servers need to be able to be brought up and down as required for new software updates, to have system software upgraded, applications changed, and more. All this will need to be done without interrupting the service to the user and while continuing to provide straight-through processing (STP).

While many of these attributes can be achieved by S/390 hardware, it is when the hardware is used in conjunction with OS/390, VM/ESA, VSE/ESA, together with the ability to quickly deploy simple, single-function servers and services based on the open standards world supported by Linux, that this solution becomes compelling.

1.5 Complexity impedance

In order to be able to deliver on this vision, in fact in order to make any reasonable progress in e-business, you need to consider how you will address the integration required.

One approach is to build custom components, based on custom protocols (many of which are individually negotiated with suppliers), data formats and protocols; this would have been the approach often taken in the 1970s and 1980s. While this might be a reasonable process to support one or two new customers, individual appliances, or mobile devices, it simply is not sustainable with a wide range of mobile devices, and causes problems when working with more than a few partners and suppliers.

In the 1990s, with the push for a single UNIX, the advent of the Internet and a general acceptance of the benefits of standards, this started to change. The need for interoperability became more important than a single computer architecture becoming all-powerful, as some expected. The number of computer systems and architectures actually grew with the emergence of mobile and wireless devices, the adoption of standard networking protocols, etc. Also, after many years of understanding the benefit of getting programmers out of the business of *plumbing*, we started to see this become possible through the advent of technologies such as Java and its integrated application services and servers.

As we go forward, complexity will grow by orders of magnitude based on the numbers of interconnected systems needed to support applications. In this environment it becomes even more important to support and endorse truly open standards to minimize complexity. Vendor extensions of standards are to be avoided, especially those that are not available through open source, since you cannot rely on the implementation, cannot depend on ongoing vendor support and endorsement, and cannot afford to have to make changes when a vendor decides to change the implementation or extension.

Simply put, information technology should cooperate on shared standards and compete on corresponding implementations; for example, function, performance, quality, services, technical support, industry vertical value, integrated solutions, price, business partners, and so forth.

In this environment, tools and technologies such as Linux, XML, the emerging Simple Object Access Protocol (SOAP) and other open industry standards become essential. It is in this context that there is a continuing push to make Java and especially the enterprise definition, Java 2 Enterprise Edition, a truly open standard rather than one controlled by a single vendor.

Chapter 2. Open source

As explained previously, open, shared industry standards are essential for e-business to continue to grow and flourish.

Open source has existed in one guise or another for many years—in fact, since the inception of software programmable computers. It is not the purpose of this book to be the definitive guide to open source, but it is worth reviewing one of the key open source initiatives and providing a definition for the goals of this book.

For a detailed discussion of the types of open source and their pros and cons, as perceived by the Free Software Foundation, refer to:

<http://www.gnu.org/philosophy/free-sw.html>

(Note: IBM does not endorse or necessarily agree with the definitions and opinions expressed by the Free Software Foundation. The purpose of including this reference is that the Free Software Foundation, and its founder, Richard Stallman, are widely accepted to be an authoritative source on the subject.)

Various types of software available today include:

- Open source—with a number of different licenses including BSD, Apache, and GNU/Linux
- Royalty-free libraries
- Royalty-free executables
- Shareware
- Noncommercial
- Trial/test software
- Commercial

Within each of these categories is a wide range of types of software. For example, under commercial software, IBM has distributed commercial software under a number of different licensing conditions. This includes software with source code, software with source code that could not be modified, and software without source code. So, it's not always clear to which category some software actually belongs.

It is worth describing a few of these types.

2.1 The GNU Public License

To establish this right from the start: GNU stands for “GNU is Not UNIX.” The GNU General Public License, simply referred to by many as the GPL, provides for software distributed with source code and a series of rights that establish what a user can do with the software and source code. This is a significant difference from a traditional software license, which typically establishes the rights of the producer and tells the users what they cannot do with the software and source code.

The GNU license permits modification of the original program, provided that you retain the GPL and redistribute under the same terms and conditions. This means the modifications you build must also be modifiable by others: it requires you to distribute your source code.

This is important because it establishes a base from which a software design can be quickly and effectively extended, and then maintained on an ongoing basis independent of the success or failure of the original author.

As we will see in Chapter 3, “Why Linux” on page 17, Linux was developed (and continues to be developed) under the auspices of the GPL. Hence, the formal name for Linux is GNU/Linux. Any modifications you make to the source code that makes up the core Linux system, or any other component of the Linux system distributed under the GPL, must also be distributed with the GPL.

A key aspect of the GPL is that you cannot choose which parts to comply with and which to ignore. For example, one cannot say “I will take your source and modify it, but you can’t have mine for the same purpose”; instead, it’s all or nothing.

Key to the portability and availability of open source software and the Linux operating system is the GNU C Compiler, or GCC. GCC is often the starting point for many GNU programming projects, UNIX applications, and for Linux.

*(Open Source Software on OS/390, SG24-5944, contains many open source packages that also run on OS/390, as does *Porting UNIX Applications to OpenEdition for VM/ESA*, SG24-5458.)*

It is because of these open source, GPL-based tools and technologies that open standards can really thrive and flourish. This offers the ability to a programmer to build and extend the work of another, and to choose to use or discard the work of another.

In the Internet world, the availability of tools and technologies that encourage the distribution of and access to code, and the communication between programmers, has truly accelerated distributed development and made open source work.

2.2 The IBM Public License

An initial IBM Public License is now available and the first software packages have been distributed, with source code, under this license.

The IBM Public License is not meant to compete or replace the GPL. It is a license that grants rights to the user and at the same time protects the legal entity that is IBM from liability connected with the program.

One of the first programs released under the IBM Public License was the IBM/LOTUS 1.1 revision of the Microsoft Simple Object Access Protocol. The 1.1 level has been submitted to the W3C for ratification as a new open, shared standard for objects on the Web. The IBM implementation of SOAP has been offered through open source for inclusion in the Apache Web server.

Perhaps surprisingly, IBM has already released a package of *Tools and toys* for OS/390 written mainly by IBMers under the IBM Public License.

2.3 Flourishing through open standards

It is through this evolutionary process, and because of the availability of source code, that small teams of programmers, individual developers, and corporate teams from organizations like IBM can work together to drive forward shared standards, which will be an essential component of the next generation of e-business.

Given the availability of Linux source code, much of the innovation will be done, as it is now, using a version or derivative of Linux. This allows programmers to start from a known base, using a known toolset, to make modifications as appropriate and produce derivative works. These derivative works can then become available across all versions of Linux, ensuring a strong “food chain” for further innovation and, importantly, for the adoption of shared, open standards through access to source code.

This adoption of shared open standards through access to common source code will continue to cause interface and protocols to become commodities.

Adam Goodwin, publisher of Linux magazine, said “Anything that can be commodotized [in Linux] will be”.

2.4 The Application Framework for e-business

For programmers and users of IBM Software, IBM offers the Application Framework for e-business. This framework provides a guide for the standards that IBM is incorporating into its platform-independent middleware and application services and servers. It is a repository of design patterns, open source contributions, and reusable, code-based developer roadmaps.

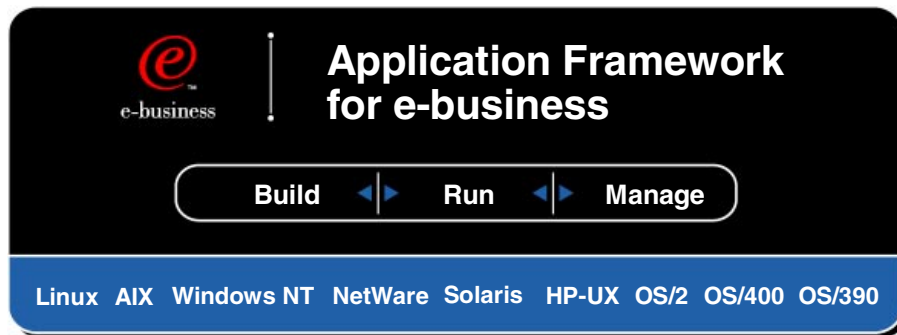


Figure 2. The Application Framework for e-business

Building applications based on standards incorporated into the framework means that your business applications can use and exploit those standards through IBM software across a wide range of platforms, including but not limited to Sun Solaris, HP-UX, Microsoft Windows NT, and the IBM operating systems OS/390, OS/400 AIX, and Dynix/PTX; the new and increasingly important member in this fold is Linux.

Even where these standards do not exist natively on a given framework platform, they will be provided by IBM software, thus protecting your investment in business application exploitations using the framework.

2.5 Summarizing open, shared standards

Building on the Linux standards base, combined with TCP/IP networking, Web technologies such as HTTP, XML for data interoperability, and programming reusability and portability and emerging standards such as SOAP for object access, open standards are providing the basis for current and future e-business applications.

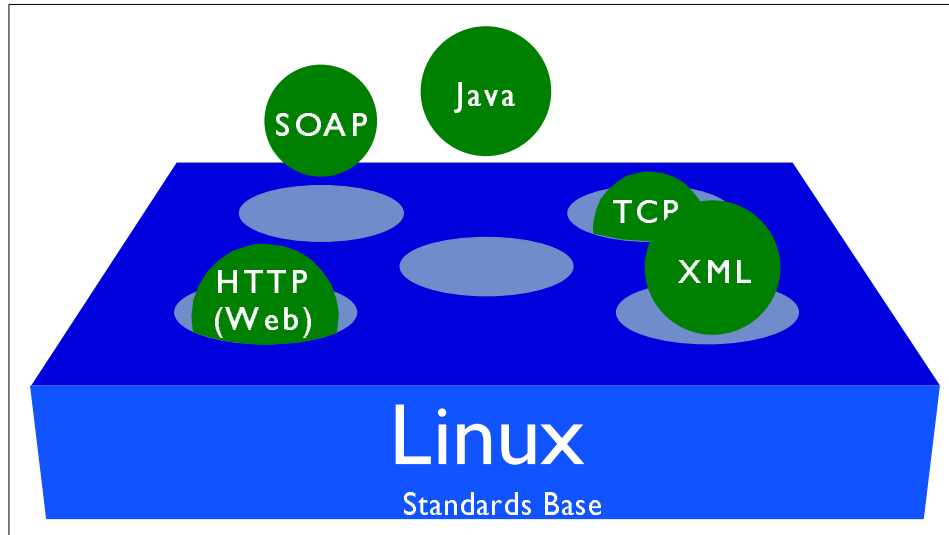


Figure 3. The open standards platform

Throughout this section the subject of cost, or price, has not been mentioned simply to avoid getting drawn into a debate that is much less important to the big picture: the cost of software compared to its value.

While some software may be available free, that does not imply it has no value. In fact, quite the contrary is often true. Software openly available can be quickly and easily adopted and adapted to meet individual and organizational needs. This, in turn, can be highly valuable to the organization or individual and to the organizations and individuals they work with, as it can enable rapid and compatible deployment.

As Richard Stallman, the Founder of the Free Software Foundation, succinctly put it, "To understand the concept, you should think of 'free speech', not 'free beer'."

Following is a list of useful URLs:

http://www.opensource.org	Open Source Initiative
http://www.gnu.org	Free Software Foundation
http://www.linux.org	Linux
http://www.apache.org	Apache Software Foundation
http://www.ietf.cnri.reston.va.us	Internet Engineering Task Force

Chapter 3. Why Linux

In Chapter 1, “Technology and business” on page 1, we explored the business and technical landscape that led to the adoption of shared, open standards as essential to greater exploitation of the Internet and e-business. Linux is one of the best examples of a shared open source standard. Now we look at:

- Why Linux?
- The IBM Linux strategy
- Applications and Linux
- An introduction to Linux and Linux on S/390

Finally, we describe what Linux is, how it is constructed, and some technical implementation considerations.

The TowerGroup (<http://www.towergroup.com>) estimated in a November 1999 report (*Linux in the Financial Services Industry: Not Just Your Developer's Operating System*) that “Worldwide, spending on Linux information technology (IT) by the top 100 financial institutions is expected to increase from \$50 million in 1998 to \$200 million by 2003, growing at a compound annual growth (CAGR) rate of 32%. Investment in Linux IT by the financial services industry represents only a small portion, less than 10% of all spending on Linux IT”.

International Data Corp. (IDC) reported at the same time (11/99) that the use of Linux consisted of:

- 45% Web servers
- 42% networking servers
- 38% e-mail/messaging
- 28% database
- 26% file/print

and that Linux was growing “bottom-up” and that many Chief Information Officers were only just discovering that they have Linux. Also that:

- The Linux server market grew 93.2% from 1998 to 1999.
- Approximately 1.3 million server licenses were shipped in 1999.
- Linux has over 11 million users worldwide.

True to its origins in education and academic research, Linux has garnered a significant following within the same communities, primarily because it's available at no charge, but also because of its availability in source code format. Add to this the fact that there is an ever-increasing range of tools and technologies available at little or no cost for Linux, many of which, like the

GNU tools, also come with source code, and you have a compelling platform for both educational value and cost savings.

While Linux may have come to the fore in the commercial IT press only over the past 18 months, it has been popular in education and research for at least the last 3-5 years. This means that there already exists a body of people who know Linux, understand how to use it, have experience with what it does well, and are able to put it to work—albeit in some limited form, such as a Web server or file server—in commercial organizations.

Through early successes in these limited fields, we can expect to see organic growth in the types of applications that are run with Linux. More on this later in 3.7, “Application scenarios with Linux on S/390” on page 30. However, Linux has also gained and will continue to grow in several other significant areas:

- Support - Traditional “community” sources, through online, Internet-based newsgroups, Web sites etc. will continue to support, extend and fix Linux, in addition to the arrival of commercial support options from dot-com Linux distributors such as TurboLinux and SuSe. Traditional software and service companies such as IBM Global Services also extend their support offerings to Linux.
- Self-healing software - Through the availability of source code, and an innate desire to carve out a niche for themselves in the Linux community (described in Eric Raymond’s work as *Homesteading the Noosphere*), the technical community is quick to find and resolve any problems and to extend Linux both for their own benefit and that of the community. While it is difficult to make direct comparisons, many extensions of Linux have gone from being an idea to final implementation in less than one month. This is unheard of in other standards.
- Efficiency - Various organizations have tried and tested Linux and found it to deliver excellent performance for the function it contains. For example, a Microsoft employee wrote in an analysis of Linux that “Linux outperforms many UNIXs in most major performance categories (networking, disk I/O, process ctx switch, etc.)”. The complete note is at www.opensource.org/halloween/halloween1.html. In addition, through the availability of source code, Linux can be pared down to the bare essential function needed to fulfill a specific business need, rather than having to address a general purpose computing need.
- Heterogeneous interoperability - Because Linux doesn’t seek to dominate computing, some of the key tools and technologies available for Linux are to allow it to share file systems, rich media, communications protocols, and programming environments with other industry software platforms,

such as UNIX, Microsoft Windows NT, and IBM OS/390 and OS/400 systems.

Through these initiatives, Linux has already been made available on some 35 different hardware platforms, including everything from embedded Linux systems - <http://embedded-linux.org> - to supercomputers, including a Beowolf cluster of 256 IBM Netfinity Servers with a total of 512 processors at the University of New Mexico at Los Lobos, and now on the IBM S/390 mainframe server, the subject of this book.

3.1 Applications and Linux

Although it is a fine line, and one which will generate some debate, generally Linux is well-supported by tools, technologies and middleware. However, business applications already running on Linux are few and far between. Where they are, they use the services of an application server, rather than depending on Linux directly for these services.

The same can be said for the open source world. While there are many open source tools and technologies, there are precious few business applications available through open source. While open source-based systems have started to make inroads into layer-3 (as shown in Figure 4 on page 20), they still have some way to go, despite being pervasive in layer-1 and layer-2 for Linux systems.

The reason for this can be seen by looking at Figure 4 on page 20. Here we see that in order to produce business applications quickly and efficiently (a benchmark for today's connected world), business applications depend on a software stack. There has been a concerted effort by software vendors since the mid-1990s to develop application generation tools and design application programming models that get business programmers out of the plumbing of a computer system so they can concentrate on producing the processes, rules and procedures that model a business.

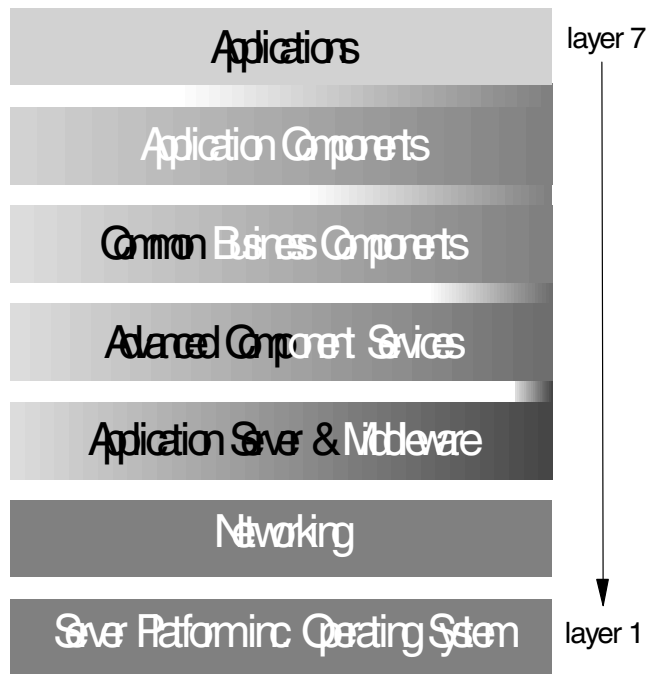


Figure 4. Application infrastructure

One example of this is the IBM WebSphere Application Server in its Advanced and Enterprise editions. This fulfills the role of an application server, layer-3 and layer-4 in Figure 4, and provides a structure for Independent Software Vendors and solution providers to plug in business components and component services in layers 5 through layer-7; see

http://www.ibm.com/software/webservers/appserv/download_linux.html

The IBM WebSphere Business Components, formally known as the San Francisco Frameworks, take this up another layer, providing:

- Reduced development time
- Improved time to market
- Flexible applications
- Simple portability
- Improved value to your organization
- Mix and match solutions
- Unique performance, availability and scale

This is accomplished through the availability of the IBM WebSphere Application Server on a wide variety of platforms, both with and without Linux.

Linux is a key enabler for WebSphere, especially in the fast growing world of e-business, but since WebSphere applications are written in Java and use the Enterprise Java specification, they can be moved between architectures without requiring recompilation through the Java “Write once, run anywhere” capability.

While Linux applications, adhering to the core Linux Application Programming Interfaces (APIs), allow application portability, this does not provide binary portability between different hardware platforms. Applications have to be recompiled to move from platform to platform.

It is also true that the typical corporate application developers have had little exposure to the open source and free software community and precious little opportunity to contribute to it, since the programming languages they write in are different. Whereas the open source world tends to write in C, C++ and Assembler or machine code, business programmers typically write in COBOL, Java, Visual Basic or other more common business languages.

Increasingly, Linux is becoming an application development reference platform for key software providers. There are already a number of Java Development Toolkits (JDKs) for Linux from IBM, Sun, Blackdown etc. and these are being supplemented with Integrated Development Environments (IDEs) from IBM and other vendors. IBM also provides an Application Development Environment, which includes VisualAge for Java (<http://www.ibm.com/vadd>), DB2 database and WebSphere, and more, for complete application development, unit testing and debugging in a Linux environment.

While there is still some way to go in the support for Linux on S/390, early indications are good. BMC Software announced initial deployment for Linux for S/390 system management, using PATROL technology. Software AG and IBM announced a joint initiative to bring Software AG’s Tamino XML Information Server to Linux for IBM’s S/390 platform.

For the latest status of Independent Software Vendor (ISV) products for use with Linux on S/390, see:

<http://www.ibm.com/s390/appsource/>

To further facilitate the adoption of Linux for S/390, members of *PartnerWorld for Developers*, S/390 will have porting/enablement programs in place for Linux for S/390. Both the San Mateo, California, and Waltham, Massachusetts, Solution Partnership Centers (SPCs), as well as both the Dallas Host Competency/Early Test Center and Boeblingen, Germany’s S/390 Technical Marketing Competency Center will be ready to accept and

schedule ISV requests for enablement, testing and verification of applications in the Linux for S/390 environment. For further information, see:

<http://www.ibm.com/spc>

3.2 IBM strategy for Linux

While this chapter sets out some of the strategic elements of IBM's Linux plan, it is not meant to be all-encompassing, especially since the strategy is still evolving.

Figure 5 shows the Application Deployment Platform Pyramid, which is based on three principles:

1. Application development on a widely available, volume platform
2. Volume platform for initial deployment with minimal barriers
3. Minimal barriers for scale from low to high and improved Quality of Service (QoS).

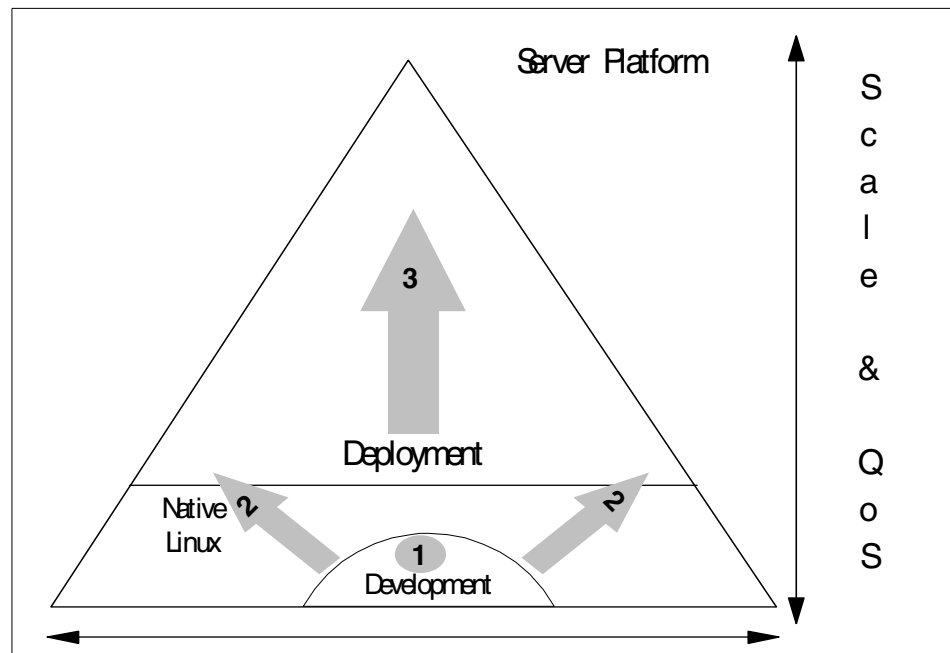


Figure 5. Application deployment platform pyramid

This strategy establishes Linux as a key development and deployment platform, especially when taking into consideration the point made in 3.1, "Applications and Linux" on page 19. Business logic is enabled through key

Java application servers such as WebSphere Application Server, and via CICS Transaction Server and Lotus Domino. Tools and technologies are enabled through the Linux standards base and APIs, including shared, open standards such as XML, HTTP and TCP/IP et al. For further information, see:

<http://www.lotus.com/home.nsf/welcome/dominolinux>

The IBM strategy focuses on five distinct but overlapping areas:

Hardware	Enable hardware for Linux including Netfinity, RS/6000, S/390, AS/400, NUMA-Q, ThinkPad, Intellistation clients and NetworkStations
Software	Key products and middleware ported to Linux including, but not limited to, WebSphere, DB2, Lotus Domino, VisualAge Java, MQSeries, etc.
Services	WW Support, Training, Professional and Consulting Services offerings
Alliances	WW Partners with Caldera, Red Hat, SuSE, TurboLinux on solution delivery and support
Open Source	Significant code contributions and technical resources working with the open source community

When looking specifically at IBM Server platforms, the strategy translates into an execution strategy, not an invention strategy. Simply put, it is to make Linux applications work.

To do this, IBM will provide an implementation of Linux for S/390, RS/6000, AS/400, Sequent NUMA-Q and Netfinity servers. From a software perspective, Linux should be able to fully interoperate with operating systems on these server platforms, including OS/390, VM/ESA, VSE/ESA, AIX and Windows NT. Additionally, the AIX and OS/400 operating systems will provide Linux interfaces (API) to support applications developed on Linux, and through AIX will provide an Application Binary Interface (ABI) to support Linux platforms in compiled format for both AIX on PowerPC and Monterey on IA64 and PowerPC.

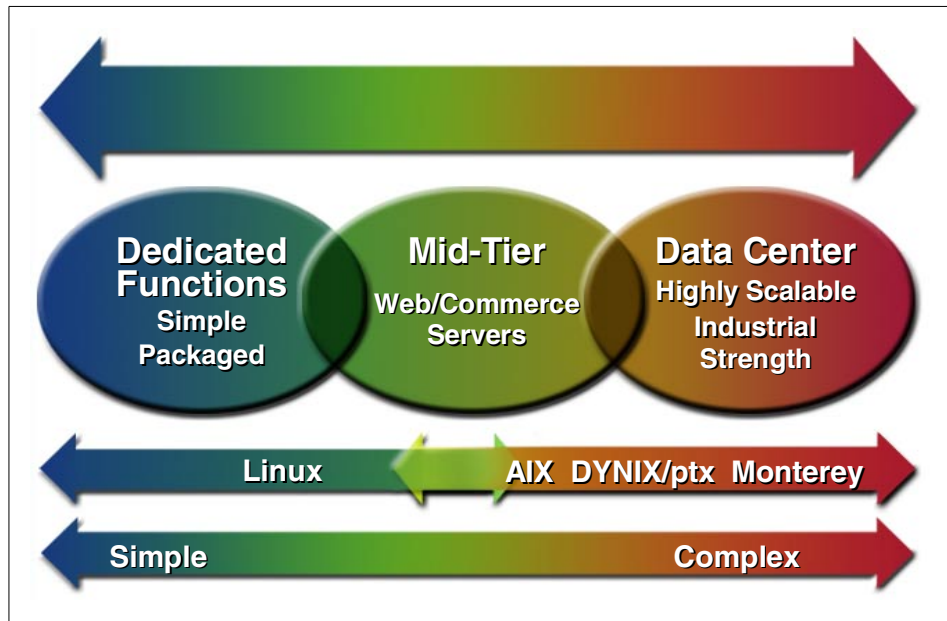


Figure 6. IBM UNIX/Linux strategy

Figure 6 shows the simplified dedicated function server, which, among other roles, is the *edge and appliance servers* discussed earlier, on the left. On the right are the high function data center and enterprise class transaction and database servers. On the extreme right one might expect to see OS/390 systems that provide this function and more. However, this diagram is focussed on IBM's UNIX/Linux strategy and you can see Linux fulfilling a growing role from the dedicated server space up through the middle-tier server. AIX, DYNIX/ptx and Monterey currently provide significantly more function than Linux, and they address the need for high function, enterprise UNIX servers.

Application portability is provided from left to right via the Linux operating system running on IBM hardware. Applications that need a more advanced infrastructure than this, but still need a UNIX operating system, can be written to run with the APIs of AIX, DYNAIX/ptx or Monterey. Beyond this, if applications need to be integrated with existing data and applications on VM/ESA or OS/390, they can be ported to those systems.

The key to application portability, platform independence and programmer productivity and other benefits already discussed, is to use Java. This is achieved through the portability of Java executable programs, or byte-code,

and through the services of an application server such as the IBM WebSphere Application Server. You develop on one platform and choose a deployment platform based on your requirements, not one chosen by any single vendor.

This application, server hardware and server software enablement encompasses both Linux itself and the shared, open standards available through Linux and through the IBM Application Framework for e-business, and establishes a link from level-1 to level-5 (as shown in Figure 4 on page 20) and beyond.

Following are useful Web links for further information:

<http://www.ibm.com/linux>

- Links to IBM Linux and open source sites
- IBM hardware for Linux
- IBM software for Linux
- IBM service and support for Linux
- IBM alliances

<http://www.ibm.com/developerWorks>

- Comprehensive online resource for the developer community
- Linux zone
- Open source zone
- Java, XML, security, Web architecture zones

3.2.1 An introduction to Linux for S/390

Finally we arrive at Linux for S/390, the subject of this book. When discussing Linux and S/390, an assumption is made that you have some knowledge of S/390. In fact, readers are expected to have some detailed knowledge of and experience with S/390 in order to proceed beyond these introductions and overview notes. However, for the purpose of this section, we summarize key attributes of S/390 hardware and software as needed.

First, why S/390 and Linux? Well, because you can! In order to understand Linux and its attraction, we embarked on a research project to see what it would take to make Linux available on S/390 hardware. This project, run by the IBM Boeblingen laboratory, soon came to the conclusion that not only was it possible, but a true version of Linux running on S/390 could be produced - not an implementation, not a Linux clone, not a set of Linux APIs on OS/390,

not an EBCDIC-based Linux, but a true, completely compatible Linux on S/390 running in ASCII mode.

This is significant because it bypasses many of the challenges associated with the UNIX System Services of OS/390 and VM/ESA. The UNIX System Services run in EBCDIC mode and often require UNIX applications not only to be recompiled, but sometimes to be extensively modified in order to work at all. So, Linux on S/390 is not *like* Linux, it *is* Linux!

3.3 Running Linux on S/390

There are three common alternatives for running Linux on S/390 hardware systems, and a fourth on the way. The three common options are illustrated in Figure 7.

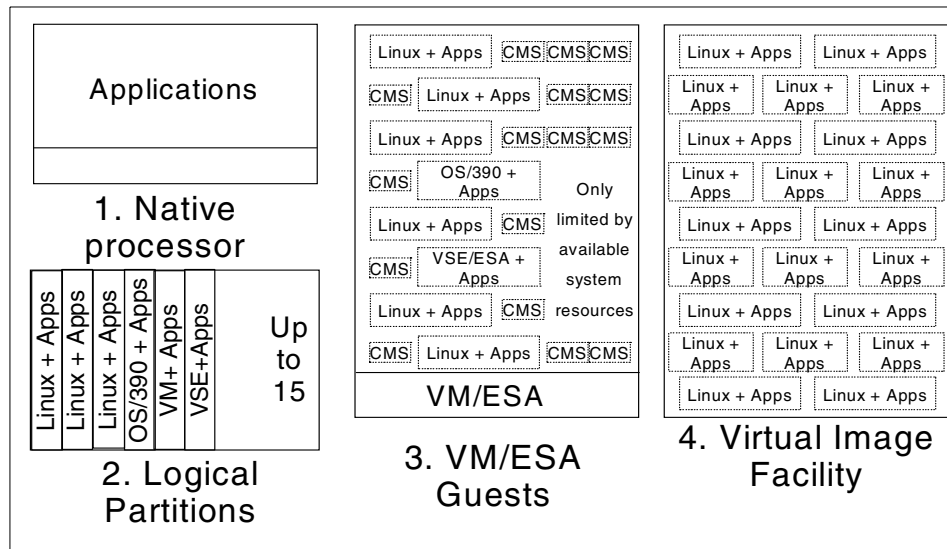


Figure 7. Linux on S/390 options

1. Native hardware support

Linux can be run natively on IBM or an IBM S/390-compatible processor that supports the *Relative* and *Immediate* Instruction support. These were first introduced on the IBM 9672 G2 processor and are also available on the P/390 and R/390 processors, and the Multiprise 2000 and Multiprise 3000 systems. Only one Linux system can be run at any one time.

Advantages

Provides full native access to the system without other prerequisite software. For a single application server that requires a large amount of memory, and high speed unconstrained channel and device support and access, this may be very interesting.

Disadvantages

Lacks full support for all S/390 attachable devices. Requires access to the machine hardware console to *boot* the system and for any Linux system debugging.

2. Logical Partition Support

Linux runs in one or more logical partitions. S/390 processors can support the partitioning of the native hardware support into 15 different *logical partitions*. Logical partitions are an allocation of the available processor resource, either shared or dedicated; and devices that are dedicated but can be serially switched between partitions and an allocation of memory. Processor sharing is possible on a system with any number of CPUs, even if there is only one.

This sharing is done by a sophisticated processor controller that gives a partition access to the processor for an amount of time. It also interrupts a partition when its time allocation or time-slice has been used up, or when the software running in the processor has no more work to do. The time-slice can also be interrupted when another, higher-priority partition has work to do, in which case the current partition will be suspended until the time-slice of the higher-priority partition is used up or this partition has no more work to do.

Memory is *not* shared between logical partitions but divided among the partitions. Each logical partition can be individually started and stopped without interfering with another logical partition.

Advantages

The ability to run and manage up to 15 individual Linux systems. Perhaps more compelling is the ability to run a number of Linux partitions concurrent with an OS/390 or VSE/ESA partition providing high value services such as transactions and access to existing applications and data.

Disadvantages

Access to a system console to start and restart a Linux partition, to define or change the definition of a partition or do any Linux system debugging; device support is the same as native execution.

3. VM/ESA GUEST Support

Linux runs as a *guest* operating system in one or more VM Virtual Machines. (If you are familiar with the concept of the Java Virtual Machine, where software is used to model a common software instruction set across different platforms, well, this is the converse. VM/ESA uses architectural hardware functions in S/390 to virtualize the S/390 instruction set, making each guest (similar to a process or address space) think it has its own dedicated S/390 processor.) Any number of Linux or other guests, including OS/390, VSE/ESA and the CMS Interactive System can run concurrently. The number of concurrent guests is limited by the resources available to virtualize them.

Advantages

The ability to use devices, both virtual and real, supported by VM and not directly supported by Linux, is a plus. This is especially true of communication between guest virtual machines, which can be done at extremely high speed. Included in this virtual device support is virtualization of the system console, one per virtual machine. The guest console can be accessed by any networking method supported by VM/ESA, including rlogin, telnet, 3270 via TCP/IP, and 3270 via SNA. This means you can start, restart, and debug remotely. Your virtual machine is protected by a user ID and password. You can automate, manage, and administer virtual machines using CMS.

Disadvantage

If you want to run a large number of Linux systems concurrently, VM/ESA can be complex. VM/ESA skills are needed, and a number of new things need to be learned.

4. Virtual Image Facility (VIF)

At the time of writing, IBM is planning to release a new product (VIF) that will extend the function of logical partitioning, combining it with some of the features of VM/ESA but without requiring the skill or knowledge of VM/ESA in order to install, build, and run a system that supports 15 or more concurrent Linux systems. At this stage it is too early to describe the pros and cons of this solution.

3.4 Communication and connectivity

Communicating from the outside of an S/390-based Linux system is achieved via TCP/IP through the Open Systems Adapter-2 (OSA-2), which supports Token-Ring, Ethernet, Fast Ethernet, Fiber Distributed Data Interface (FDDI), and Asynchronous Transfer Mode (ATM), or via an IBM Channel-to-Channel

Adapter (CTCA) that can be connected to other channels on the same or different IBM S/390 processors. (A channel can be thought of as a path to a device. However, it is actually much more than that and is managed by a processor-independent subsystem that can deliver high performance and support a large number of devices.)

These options are available when running in any of the modes shown in Figure 7 on page 26. In addition, when running Linux as a guest in a virtual machine, you can also use a virtual CTCA or another virtual driver provided by VM, called IUCV. Communication between these virtualized devices is extremely fast since you are effectively communicating at almost memory speed between virtual machines. It is worth stating, even though it might seem obvious, that since these devices are virtual, i.e., they do not really exist, you cannot use them to communicate with *external* systems of users. You must use either the OSA-2 or real CTCA. Of course, Linux systems can communicate via a mixture of real devices for external communication and virtual devices for internal communication.

In the future, we expect to deliver a virtual communications driver that works between logical partitions and delivers low-latency, high-speed communication that has performance similar to IUCV between VM/ESA guests.

Application programs running on Linux can communicate using standard TCP/IP facilities with no change, whether communicating between Linux systems running on S/390 or with external systems. So, for example, you can use TCP/IP sockets, Remote Procedure Call (RPC), and so on.

For more detail on connectivity, see Chapter 14, “Linux TCP/IP connectivity” on page 267.

3.5 Other device support

In addition to communication and connectivity, a number of other devices are supported when running in one or more of the modes supported by S/390 when running Linux. More information on specific support is included in later chapters. Of note is the virtual disk, or minidisk (MDISK), support provided by VM/ESA. This allows a portion of a real disk drive (device) to be used by a Linux system as if it were a real disk drive. VM/ESA delivers this support across a much wider range of real disk drives than are supported in LPAR or native mode.

3.6 Linux and S/390 benefits

Running Linux on S/390 in any of the supported modes inherits most of the IBM hardware benefits, such as the architecture and implementation of:

- Guaranteed data integrity
 - 100% error detection
 - Error-corrected memory, cache and I/O bus
 - Memory storage key protection
- Reduced customer cost
 - Dynamic activation of spare processor in event of a failure
 - Dynamic Storage Reconfiguration
 - Dynamic I/O Reconfiguration
- Minimized customer impact
 - Deferred repair
 - Deferred upgrade

These functions provide significant value, even when used alone with Linux. No other hardware platform brings all these functions to Linux. However, the real value of running Linux on S/390 is *how* you run it, and *what* you do with it.

3.7 Application scenarios with Linux on S/390

How you run Linux on S/390 depends on which of the supported S/390 modes you need. The selection of the mode depends on how many concurrent Linux systems you will need to run. As previously discussed, this can span from a single Linux system to perhaps hundreds or thousands of concurrent Linux systems.

We cannot predict all the ways you will want to run Linux on S/390. However, there are a number of predictable scenarios and categories.

Large scale application server

Running one or two large-scale application, database, or Web servers, thereby exploiting the reliability, availability, security, memory, and most importantly, the I/O subsystem, of S/390.

Network integrator

Consolidating distributed or rack-mounted Linux servers supporting a Web and network infrastructure consisting of APACHE, BIND and more onto a single, high-availability server and running multiple Linux systems concurrently. This minimizes the cost of running the configuration through

reduced hardware redundancy, reduced operational requirements, etc. S/390 provides *fine grain* sharing of processor resources. This might be attractive to Internet Service Providers and offers some interesting possibilities for running “edge and appliance servers” on S/390 hardware, providing high availability, unrivaled reliability and on-demand growth with S/390 nondisruptive hardware capacity growth.

Server consolidation

Taking the workload of two or more Linux directory, application, database or file server systems and running them as one or more Linux images on S/390. Combining multiple Linux systems into one is attractive for reduced cost of ownership and operational simplicity reasons. S/390 enables this by providing both higher reliability and increased I/O bandwidth, both of which can be common requirements for running multiple mirrored systems. This might be attractive to Application Server Providers.

Application, transaction and database integration

Where Linux systems, e-business, Web, or application servers need to integrate with existing OS/390, VM/ESA or VSE/ESA systems. Previously, this was only possible using network or, at best, S/390 I/O channel connections. This meant that any high-volume requests would bottleneck on the connection between the Linux system and the S/390 server. It would also introduce additional points of failure in software and hardware. A common solution to these points of failure is to replicate the *middle-tier* server. This in turn introduces new points of failure and grows the complexity and cost of the total solution. Running Linux in a logical partition or as a guest of VM/ESA means that the hardware and connectivity points of failure are eliminated and data can be transferred between the Linux and S/390 server application at very high speeds utilizing open industry standard interfaces.

This is especially interesting when considering the options for exploiting DB2 for OS/390. DB2 for OS/390 can, in real time (in read and write mode), share its relational database between multiple OS/390 systems using the OS/390 and S/390 Parallel Sysplex support. This provides for almost limitless scalability of the database and nondisruptive upgrades through the addition of entirely new OS/390 systems into the sysplex while sharing the same database.

Linux-based systems that would find this attractive include both ISP and ASP systems leveraging S/390 resources.

Testing and development servers and services

The ability to run a very large centralized server that can multiplex Linux systems might be very attractive. The ability to do nondisruptive restarts of

server images, that is, the ability to use the function provided by VM/ESA for remote system level debugging and problem determination could prove to be very attractive in an educational or large-scale Linux development environment. ("Hey, its Linux running on the mainframe but / can reboot and restart when / want and it doesn't affect anyone else!")

3.8 Tools and technologies

In the Linux arena there are a rapidly growing number of software implementations that span a broad range of technology areas. These tools and technologies include both open source and commercial software initiatives.

A great deal of open source software has already been recompiled and is running on S/390. Available are, for example:

- Apache HTTP server
- Bind Domain Name server
- GNU/GCC Toolset and compilers
- Journaling File System (JFS from IBM)
- Samba File server
- Logical Volume Manager (LVM from IBM)

IBM is porting its significant middleware and products to Linux. In a S/390 environment, some of the more significant requests we are getting are to enable Linux to exploit OS/390 and VSE/ESA database and transaction systems. Announced plans include e-business enabling *infrastructure* technologies such as DB2 UDB and a Java Virtual Machine. We include a brief description of the products, tools and technologies being made available by IBM to help you understand how you might use them.

The IBM WebSphere Application Server

WebSphere is the industry's leading Web application software platform. The Advanced Edition is a complete application server platform that leverages a Java Virtual machine to deliver a transactional infrastructure for running Enterprise Java Beans as well as servlets and Java Server Pages.

WebSphere is currently available in Standard and Enterprise Editions for OS/390. The Standard implementation uses the IBM HTTP server powered by the Domino GO Web server shipped as part of the OS/390 operating system. WebSphere Advanced Edition for Linux will be powered by the Apache Web server.

Applications can be developed using VisualAge for Java on any of its supported development environments, including Linux, and then deployed on either WebSphere Advanced Edition on Linux running on S/390, or on WebSphere Standard Edition (servlets and JSPs only) or WebSphere Enterprise Edition (servlets, JSPs and EJBs), both running on OS/390.

An additional advantage offered by WebSphere Advanced Edition on Linux for S/390 is that it will now be available to VM/ESA and VSE/ESA customers who previously didn't have access to WebSphere without running OS/390. This will include connector technology to allow WebSphere to connect to DB2 VM/VSE and at a later date to CICS Transaction Server for VSE. This brings IBM's leading development and deployment tools in a S/390 environment to customers who previously didn't have access to them or were unable to deploy applications on their preferred platform!

JAVA 2 support

IBM will provide an implementation of its industry-leading Java Virtual Machine (JVM) technology for use with the IBM WebSphere Application Server for Linux on S/390. At the time of writing, this was expected to be based on a Java 1.2.2 JVM. The JVM and associated base Java classes provide the basic portability and platform independence for Java-based applications. This is augmented and extended using the Enterprise Java-based services and classes contained in the IBM WebSphere Application Server Advanced Edition. These services include, but are not limited to, services such as: Name, Directory, Security, Transaction, Messaging, and Database and Application Connection.

The Java 2 support will provide Servlet, Java Server Pages (JSP) and Enterprise Java Bean support through the IBM WebSphere Application Server for Linux on S/390.

Connector technology

In support of WebSphere and the Enterprise Java initiatives, IBM will provide connector technology that can be used stand-alone, or in conjunction with the Common Connector Framework (CCF) that is embodied in VisualAge for Java and in the WebSphere Application Server and also is the foundation for the Java 2 Connector Framework.

Initially, IBM will provide connector technology for:

- DB2
- CICS
- IMS
- MQ Series
- DB2 Universal Database

- Tivoli Framework End Point support
- Tivoli Storage Manager Client

3.8.1 DB2 Connect

DB2 Connect will provide Java connectivity, JDBC, SQLJ connection to any DRDA-capable database. These include DB2 running in OS/90, VM/ESA and VSE/ESA. DRDA is an industry standard connectivity into relational databases. Other alternatives are also possible.

3.8.2 CICS

The CICS Transaction Gateway will provide Java connectivity into CICS Transaction Server running on OS/390 (and at a later date to VSE/ESA). It provides the ability to be used either through Servlets, JSPs or by stand-alone Java applications running on Linux for S/390. In whichever mode you use the CICS Transaction Gateway, you will be able to exploit and integrate with existing applications running under CICS. When using CICS for OS/390 you can use existing applications or write new applications, also in Java.

3.8.3 IMS

For customers using IMS as a transaction manager on OS/390, the IMS Connect feature will provide connectivity from WebSphere Application Server into existing IMS transactions.

3.8.4 MQ Series

Applications running on Linux for S/390 can be integrated with existing S/390 and non-S/390 applications by using the MQ Series Client. MQ Series Clients are provided on an unrivaled number of platforms, providing the most open messaging software in the industry. This support allows applications to be built and deployed for any platform, including S/390, and allows isolation between the client and the server. This isolation allows the client to be moved from platform to platform without requiring any change at the server. The server just receives its input in the form of a message on a queue.

3.8.5 DB2 Universal Database (UDB)

DB2 Universal Database (UDB) for Linux is a full-featured relational database, and part of the DB2 family. It is initially being offered on Linux for S/390 to address two main requirements. The first is that a relational database is required when using the IBM WebSphere Application Server Advanced Edition, that is, the ability to store local data for Session, Entity

beans, and to provide JDBC and SQLJ access to a local relational database. Running DB2 in Linux makes sense and provides a complete solution for stand-alone e-business Linux systems where there is no need to integrate with OS/390, VM/ESA or VSE/ESA.

The second reason DB2 is being offered for Linux on S/390 is so that local programs can exploit it for its relational database capabilities inside the Linux system, for example business intelligence, data mining, and customer relationship management solutions.

It should be noted that while DB2 UDB for Linux is a world-class Linux relational database, it does not have all the features and functions of DB2 running on OS/390, and has not been modified in any way to exploit S/390 features and hardware.

If you design a solution that needs DB2 for OS/390, or VM/ESA, or VSE/ESA features, or access to data or tables held in those systems, you can use DB2 Connect to access them remotely. Where the Linux system is running on the same S/390 hardware system, or where it is interconnected via a high-speed connection to the system, or where only low volume access is needed, this method avoids the need to replicate data and reduces management and additional hardware cost while not affecting overall application performance.

3.8.6 Tivoli Framework end-point support

Today's computing environment relies more and more on distributed systems for information system needs, where users at the client workstations perceive the network as one big server or service provider. The Internet, distributed computing, or network computing ties people, information, and resources more closely together, but brings a challenge when considering the management of these systems. Managers face the complex problem of maintaining many different types of hardware and operating systems.

The Tivoli Framework is a solution that addresses modern 3-tier computing environments providing support for management server/end-point manager, end-point gateway, and end-points. An end-point can be managed, but it cannot manage or be managed through.

This 3-tier architecture provides:

- Increased flexibility in planning or managing customers' enterprises
- Increased functionality
- Lower cost
- Lower maintenance
- Increased scalability

Adding Linux and specifically Linux for S/390 operating system platform support brings additional operating system platforms into the robust Tivoli management environment, that is, an increased ability to perform end-to-end enterprise management, regardless of the platform.

- Using Tivoli to leverage common services in the Tivoli framework

This means that when Tivoli improves a common service (for example, the scheduler), the other Tivoli applications “inherit” the improvements. Further, a set of common services allows the customer to lay down one source of systems management “plumbing” that does the integration work.

- Reduce the cost and complexity of using multiple management tools

With point products, the customer is forced to be the integrator. In other words, for many customers the act of integrating multiple systems management tools that don't work together is a major part of the problem. The Tivoli Management Framework serves as a single point of integration for Tivoli and third-party applications. Since all of these products can “plug into” the framework and exploit its capabilities, the customer is no longer forced to be the integrator.

Linux end-point support means that one or more Linux systems running on S/390, in any of the supported modes of operation, and running the Tivoli end-point support, can be managed and automated with, and benefit from, an enterprise-wide Tivoli-based systems management and automation solution.

3.8.7 Tivoli Storage Manager Client

The Tivoli Storage Manager is a hierarchical, client- and server-based backup and archive product. The clients and servers run on a wide range of platforms and connect to each other using industry standard communication and connectivity, such as TCP/IP over a 100 Mb Ethernet or FDDI.

The Tivoli Storage Manager Client, running in Linux on S/390, can be used to back up, archive, restore, and retrieve files and databases from a server that might typically be running on an OS/390 system, exploiting the high availability services and storage support, such as IBM Tape Library support. Tivoli Storage Manager Servers also run on a wide number of other platforms.

Availability of the Tivoli Storage Manager Client, as with the Tivoli Framework end-point support, brings Linux on S/390 into an enterprise-wide solution, allowing the files and databases used in Linux to be effectively managed by an existing storage management infrastructure.

3.8.8 Summary

The IBM middleware for Linux for S/390 will give customers the flexibility to use Linux as an e-business server to link mission-critical applications running on S/390 operating systems directly to the Web. Customers can take advantage of S/390 reliability, security, scalability and nonstop operation to run Linux applications concurrently with other line-of-business applications, such as enterprise resource planning, customer service, and supply chain management running on OS/390, VM/ESA and VSE/ESA, as well as other S/390 operating environments such as TPF and ALCS.

In addition, customers can take advantage of the new middleware capabilities to increase the integration and performance of the Linux applications running on S/390. For example, back-end mission-critical applications running on OS/390, VM or VSE will be accessible at near memory speeds, bypassing the network connection between the two applications.

Customers can also gain the cost savings resulting from the consolidation of workloads from many Linux systems onto a single S/390 system. Beta versions of the IBM e-business Enterprise Connectors for Linux for S/390, IBM DB2 Universal Database for Linux for S/390, and IBM WebSphere Application Server, Advanced Edition, with Java 2 support, and Tivoli Framework support will be available in the third quarter of 2000.

General availability will follow in the fourth quarter. The release date for Tivoli Storage Manager Client for Linux for S/390 will be announced later this year.

Chapter 4. Linux distributions

Linux has a number of unique characteristics. Not only is it available in open source, as discussed earlier, but it is also available from a number of different sources. Linux can be obtained in source format and built by an individual user or organization.

However, to support more complex hardware, for example a large number of PC servers from many different suppliers, or one or two major UNIX hardware servers, and you have a considerable task. This is further complicated when you consider the range of solutions one or more of those systems could be used for -- application development, office productivity, Web servers, e-business application servers, database servers, thin clients, kiosks, etc., etc. To address these and other requirements, including supplying a range of Linux-based tools and technologies, a number of standard Linux “distributions” have come about.

These distributions can typically be obtained from either the Internet itself via download, or on CD-ROM. Some come with little or no documentation, and no support, and cost little or nothing. Others come with some documentation and have some support available, others come with comprehensive documentation and a range of support and service offerings. In addition, distributions are targeted at a particular market, such as desktop PCs, Web servers etc. and provide the hardware drivers and other tools and technologies needed to support that environment.

IBM is working with four Linux distribution partners, Caldera, Red Hat, TurboLinux, and SuSE, as discussed earlier in 3.2, “IBM strategy for Linux” on page 22. For Linux on S/390, we are specifically working with SuSE and TurboLinux on S/390-specific distributions.

All Linux distributions share the same common Linux kernel and development libraries. Most also provide custom components that ease the installation and maintenance of Linux systems. You can see from Figure 8 on page 40 that Linux for S/390 contains the common Linux kernel, architecture-independent and hardware-dependent parts, including a number of S/390-specific functions and device drivers, and all the common, non-S/390 library and application interfaces.

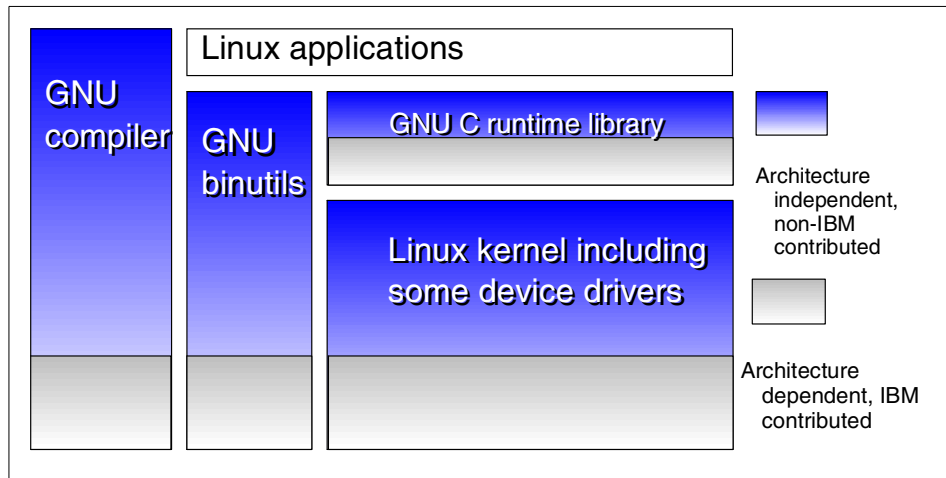


Figure 8. Linux on S/390 structure

A Linux system for almost any other hardware platform would typically look just the same. The key here is that tools and technologies programmed to use the common Linux interfaces will not see any difference, and it is very unlikely that they will need any platform-specific code or function.

Most of the interfaces in Linux are supported by nearly every version of the UNIX operating system, and Linux implements much of the industry standard IEEE POSIX specification and has a C language compiler available. Through the GNU tools that support the ANSI Standard C programming language, Linux is widely supported by applications. However, it is not just another UNIX. Through its open source availability, through its licensing making this source code freely available and modifiable, it is a good platform on which to build and run applications.

In addition to the POSIX and ANSI standards, Linux supports a wide range of other industry standards such as NFS, FTP, HTTP and more.

Linux for S/390 adheres to the standards and interfaces found on other Linux implementations. It runs in ASCII mode and supports common applications and development tools such as the GNU tools. This is key to deployment and use of Linux for S/390, since application porting is not required. Our experience has been that most applications simply need to be recompiled in order to run on S/390.

Today, Linux is supported by most major hardware and software vendors including IBM, as already discussed. Once considered a “hacker’s” system,

Linux is making inroads in the corporate world. Examples of Linux implementations can be found in both financial and non-financial business entities. The most common uses are found in enterprise network infrastructure services such as e-mail, printer, file, and Web servers.

4.1 What a distribution is

A *distribution* is a coherent collection of software packages built for a Linux kernel on a particular architecture. It includes a methodology for getting and installing other packages as needed.

In Linux folklore, distribution started with SLS (for Software Landing Systems) and Slackware; it was only considerably later that companies started producing Linux distributions commercially.

The word “distribution” is one of those words that can cause lengthy discussions, even skirmishes, between the “followers” of the various flavors of Linux. The leaders of the various distributions are working together constructively on the Linux Standards Base (LSB):

<http://www.linuxbase.org>

The goal of the LSB is to develop and promote a set of standards that will increase compatibility among Linux distributions and enable software applications to run on any compliant Linux system. In addition, the LSB will help coordinate efforts to recruit software vendors to port and write products for Linux.

4.1.1 Announced distributions

There is the Marist Distribution, which was the first publicly visible S/390 Linux distribution, and the one that this rebook is based on, see:

<http://linux390.marist.edu/>

Thinking Objects Software GmbH was among the first to get a system up and running:

<http://linux.s390.org/>

To date, there have been announcements from SuSE, Turbo-Linux and Debian that they are in the process of preparing distributions:

<http://www.suse.com/PressReleases/ibmsuse.html>

http://www.turbolinux.com/news/pr/ibm_s390.html

<http://www.debian.org/News/weekly/2000/15/>

4.1.2 Distribution media

This will be a decision by individual distributors, and will vary.

Obviously, from the point of view of an S/390 customer, a package on a distribution medium that customers are familiar with (such as SMP/E) would be the easiest to handle and integrate with existing procedures.

The Linux distributors are, of course, more familiar with the platforms they have running at the moment, such as i386, alpha, arm, m68k, mips, pa-risc, powerpc, sparc, and sparc-64. Two newcomers, ia64 and s390, are just joining the field, but will have a lot to contribute..

There is a lot of learning to be done by:

- Members of the Linux community who want to understand the strengths the mainframe is bringing to Linux
- Members of the S/390 community who want to understand what Linux and open source can do for the mainframe

4.1.3 Roll your own

There is nothing to stop you from building your own distribution. If you have needs that are not met by the current distributions, then building your own specialized distribution is one way of “scratching that itch” (to quote Eric Raymond, a popular open source advocate). This is what “freedom” is about, when talking about free software.

If this is something you want to consider, then you will be spending time studying the work done by the “Linux from scratch” site. It does not have anything about S/390 yet (but that will change); see:

<http://www.linuxfromscratch.org/>

4.2 Linux documentation

While the remainder of this book covers in depth many of the tasks and functions you may be required to perform to install and support a Linux on S/390, eventually you might need additional information not contained in this book.

One of the best places to look for this is in the Linux Documentation Project (LDP). The LDP creates and maintains the standard online manual pages (manpages) on Linux, as well as collections of documents on how to perform specific functions, some in great depth. You can find the LDP at:

<http://www.linuxdoc.org>

There are many servers around the world that contain the same contents as the sunsite.unc.edu site. These are called *mirrors*. Some of the Linux distributions also include some or all of the documentation from the LDP.

In addition to traditional documentation, Linux is extensively supported through newsgroups. In fact, Linux is *supported* through newsgroups more than any other operating system, and most software packages. This is because Linux “grew up” on the Internet through the same newsgroups. The main Linux newsgroups can be found in the `comp.os.linux.*` hierarchy.

Finally, for documentation and information on Linux there are also a number of mailing lists. One such mailing list, based out of Marist College in New York state, USA, was one of the original sources of information on Linux for S/390. This can be subscribed to by sending a note to `Listserv@vm.marist.edu` with this line in the body of the note:

```
SUBSCRIBE LINUX-VM your name
```

Chapter 5. Native S/390 installation and operation of Linux

In a native LPAR on S/390, Linux for S/390 can be installed on S/390 hardware via VM, VSE, OS/390, and on a native LPAR. This chapter steps you through the process of building an IPLable Linux for S/390 tape, IPLing from it, customizing a new file system, and IPLing once more, but from DASD.

5.1 Assumptions

At this point, we make the following assumptions:

- The installer has access, or can work with an individual who has access, to an OS/390 operating system in order to do the following:
 - Transfer (FTP) files from a workstation to OS/390.
 - Have a user ID capable of submitting JCL to copy code and create 3490 tapes.
 - Have authorization to vary devices online and offline.
- Access to a tape drive such as a 3490 and, of course, a 3490 (or equivalent) tape. Make sure you know the unit address/device number of the tape drive.
- Access to a Hardware Management Console (HMC).

Many of the instructions here are extensions of the instructions given in the document *Linux for S/390: Installation, Configuration, and Use*. We strongly recommend that you read this document before proceeding. This document can be obtained from the Linux for S/390 site hosted by Marist College at:

<http://linux390.marist.edu>

All downloaded code mentioned in this chapter came from the Marist distribution version 2.2.15, dated 05/19/00.

5.2 Skills and resources required

You may be saying, “I’m a Linux programmer, so why should I be worried about where, what, and how Linux for S/390 is installed as far as hardware is concerned?”. And you’d be correct—*after Linux for S/390 is activated*. But until then, planning and prior knowledge such as the following is required:

- You (or a system programmer) must first carve a partition for Linux for S/390 out of an S/390 Central Electronics Complex (CEC). You probably

will not have a dedicated machine for this. We have seen that a mixture of both Linux/UNIX and OS/390 skills is required for the Linux for S/390 installation.

- You need at least read access to the system IOCDs to see the subchannel path id (CHPID) layout of the Linux for S/390 partition.
- You need access to a DASD and the layout that will be used by the Linux for S/390.

Note: Linux for S/390 will only be able to access 64 DASD devices. If you only expect to IPL from tape to satisfy your curiosity, then DASD will not be needed.

- Some knowledge of the text editor vi is needed.
- If you want connectivity through a network, then knowledge of the network is required. The following information should be obtained before attempting a Linux for S/390 network-enabled installation:
 - Type of network device (OSA2 Token Ring or Ethernet, CTC, or ESCON)
 - Host name of the Linux for S/390 system
 - IP address of the Linux for S/390 system
 - Peer IP address (for CTC or ESCON)
 - Broadcast IP address (for OSA2, typically x.y.z.255)
 - Gateway IP address (for OSA2)
 - Domain name server IP address
 - Subnet mask
 - Network address (typically x.y.z.0)
 - DNS search domain (that is in our project, itso.ibm.com)

5.3 Hardware preparation

Linux for S/390 runs in a separate partition. It does not run on top of the OS/390 operating system the way that OS/390 UNIX System Services does.

Terminology

Throughout this chapter, the terms LPAR and partition are used interchangeably. Do not confuse a S/390 partition with a Linux for S/390 minor device partition.

- A minimum of 1 CPU is required. It need not be dedicated. However, in order to better monitor the partition and debug problems, we recommend that you first install and test on one dedicated processor until the system goes into production.
- Linux for S/390 has a maximum RAM size of 1920 MB. Therefore, it is unnecessary to allocate more than 1920 MB of central storage for your Linux for S/390 partition. The absolute minimum amount of storage required for Linux for S/390 is approximately 12 MB—but your possibilities are limited with such a system.

Instead, we recommend that you have a system with a minimum of 64 MB. This should be enough to do daily Linux for S/390 work for 1 to 5 users and have a Web server running. For further details on Web servers, see Chapter 21, “The Apache Web server” on page 405.

Important

At the time of writing, kernel version 2.2.14 was limited to a maximum number of devices. That is, if more than 1000 devices were configured online to the Linux for S/390 partition, the kernel would get confused on IPL and hang. No error messages would be displayed, nor would there be a disabled wait state.

This limitation was lifted with kernel version 2.2.15, which we IPLed and ran, but on an LPAR that did not have over 1000 devices online. Therefore, that aspect of 2.2.15 was not tested.

As a minimum, we recommend that only the required device CHPIDs be online for the Linux for S/390 partition during its first installation. They are:

- DASD (if more than a tape IPL is expected)
- 3490 tape drive
- OSA card or communication equivalent

5.4 The hardware we used

Following are the partition specifications we used to install and customize Linux for S/390:

System:	9672-X77
Central storage:	768 MB
Expanded storage:	64 MB ¹

¹ Expanded storage can be accessed via the XPRAM function. See 9.1.3, “XPRAM” on page 181 for details.

CPU:	2 (shared)
CHPIDs:	3 online (DASD, tape drive and OSA card)
OSA card:	OSA-2 Token Ring/Ethernet (10 Mb/s)

5.5 Activating Linux for S/390

To install Linux for S/390 on a native LPAR, you must do the following:

1. Build an IPLable tape from downloaded code.
2. Load your Linux for S/390 partition via tape.
3. Format a DASD and create file systems.
4. Uncompress a file system on the formatted DASD.
5. Customize files on the file system.
6. Create and activate swap space.
7. ReIPL from DASD.

5.5.1 Creating an IPLable tape

You will probably find yourself using a workstation to connect to a host server. In this section we describe how to download the Linux for S/390 binary files to a workstation, upload them to your host system, and create an IPLable tape.

Attention

It is assumed that you have access to an OS/390 operating system in order to create the IPL tape. Access the Linux for S/390 site hosted by Marist College at:

`http://linux390.marist.edu`

Download the Linux for S/390 kernel with a tape image and the initial RAM disk binaries. Then FTP the files up to your host system.

At the time of our installation, the file names for these were:

- `image.tape.bin` contained the Linux for S/390 kernel. It is configured to use an initial RAM disk as a root file system.
- `initrd.bin.gz` was the initial RAM disk root file system.

5.5.1.1 Parameter line file (parmline)

A third file, the parameter line file (parmline), is also needed on the IPL tape.

This file tells Linux for S/390 what devices are available for use and what root file system to mount. The parameter *names* in the parmline file must all be in *lowercase*, whereas the values for these parameters, such as a unit address, may be in uppercase.

The parmline file can be created on your local workstation and then FTPed up to the host. The parmline file data set may be preallocated; its attributes are shown in Figure 9.

```
Data Set Name . . . : userid.PARMLINE.TXT

General Data                               Current Allocation
Volume serial . . . : TOTTSQ                Allocated tracks . . : 1
Device type . . . . : 3390                 Allocated extents . . : 1
Organization . . . . : PS
Record format . . . . : F
Record length . . . . : 1024
Block size . . . . . : 1024                Current Utilization
1st extent tracks . . : 1                   Used tracks . . . . . : 1
Secondary tracks . . : 1                   Used extents . . . . . : 1

Creation date . . . . : 2000/05/01
Referenced date . . . : 2000/05/08
Expiration date . . . : ***None***
```

Figure 9. Allocation attributes of the parmline file

For a description of the possible parameter line file entries, see Appendix D, “The parameter file” on page 481. The parmline file should contain the following entries:

- `dasd=xxxx, (yyyy) | xxxx-yyyy`

Where `xxxx` and `yyyy` are the hexadecimal unit addresses for your DASD that you have reserved for your Linux for S/390 LPAR. Only one volume is needed to save the IPL information. Additional volumes may be specified for future expansion. A range of volumes can also be specified, that is:

`xxxx-yyyy`

- `root=/dev/ram0 ro`

Where `ro` specifies that the root file system `initrd` will be mounted read-only in the event an error condition occurs. An example of an error condition would be where a previous IPL of Linux for S/390 did not shut down completely, leaving either the file system still mounted or errors such as invalid or missing inodes, missing directories or files, and so on.

- `IPLdelay=xyz`

Some installations may encounter problems with the OSA-2 card. A time delay can be placed into the IPL with the `IPLdelay` keyword. This will allow the IPL to correctly sense the OSA-2 card. The value `xyz` is entered in seconds or minutes (that is, 30s or 1m, respectively). We did not use this parameter for our native install.

5.5.1.2 Implications of using “`dasd`” in `parmline`

Linux for S/390 behaves differently depending on whether the “`dasd`” statement is specified in `parmline`. Differences range from having all DASD detected (whether Linux for S/390 has access to them or not), to specifying each DASD that you want to use.

Following are examples of how DASD detection may differ. Assume that our installation has the attributes shown in Table 1. The 9xx devices will most likely be on the same CHPID but for this exercise, let’s assume that they are all different.

Table 1. DASD devices

DASD device number	CHPID	Contents
9AC	01B	IPL record, large file system
99A	01C	swap space
998	01A	empty
24AB	00B	not used by Linux

If the “`dasd`” statement is omitted from the `parmline` file, then all DASD devices defined to the partition will be detected, in order, by subchannel path identifier. Linux for S/390 assigns device names suffixed with letters starting

with “a” (that is: /dev/dasda, /dev/dasdb, and so on), so the DASD assignments in Table 2 will be made, which are *incorrect*:

Table 2. What you really get with “dasd” not specified

Linux device	DASD device number	CHPID	Assumed contents	Actual contents
/dev/dasda	24AB	00B	IPL record, file system	Not used by Linux
/dev/dasdb	998	01A	Swap space	Empty
/dev/dasdc	9AC	01B	Empty	IPL record, file system
/dev/dasdd	99A	01C	Undefined	Empty

The “dasd” statement is specified for only those devices defined to the Linux for S/390 partition in the correct order. For example, if `dasd=9AC,99A,998` is specified, the result would be the correct DASD assignments (as shown in Table 3), since the subchannel path identifier does not confuse the order.

Table 3. “dasd” specified with correct devices

Linux device	DASD device number	CHPID	Assumed contents	Actual contents
/dev/dasda	9AC	01B	IPL record, file system	IPL record, file system
/dev/dasdb	99A	01C	Swap space	Swap space
/dev/dasdc	998	01A	Empty	Empty

Using a range on the DASD statement needs special attention as well. For example, assume the `dasd` statement is `dasd=99A-99E`. The assignments in Table 4 are used, which may not be desirable.

Table 4. Using a range on the “dasd” statement

Linux device	DASD device number
/dev/dasda	99A
/dev/dasdb	99B
/dev/dasdc	99C

Linux device	DASD device number
/dev/dasdd	99D
/dev/dasde	99E

5.5.2 Getting the files to your host system

You will need the TCP/IP address of your host OS/390 system. All three files must be FTPed in binary, and must also use an lrecl of 1024, a blocksize suitable in your installation, and a fixed record format (RECFM=FB).

The FTP subcommand `site` is used to achieve this. When using an FTP client on Windows 95/98/NT, this subcommand needs to be prefixed with the `quote` subcommand. Two major parameters have to be set: the space values and the physical characteristics of the data set. An example of FTPing from Win95/NT is shown in Figure 10.

```
D:\Linux390>ftp wtsc47
Connected to wtsc47.itso.ibm.com.
220-FTPDMS1 IBM FTP CS V2R8 at wtsc47oe.itso.ibm.com, 17:07:46 on
2000-04-29.
220 Connection will close if idle for more than 5 minutes.
User (wtsc47.itso.ibm.com:(none)): userid
331 Send password please.
Password:
230 userid is logged on. Working directory is "USERID.".
ftp> quote site lrecl=1024 blksize=8192 recfm=fb ❶
200 Site command was accepted
ftp> quote site track pri=60 sec=10 ❷
200 Site command was accepted
ftp> bin
200 Representation type is Image
ftp> put initrd.txt
200 Port request OK.
125 Storing data set USERID.INITRD.TXT
250 Transfer completed successfully.
2991104 bytes sent in 78.00 seconds (38.35 Kbytes/sec)
ftp>
```

Figure 10. Win95/NT FTP example

FTP on OS/390 allocates new data sets with default parameters set in ftp.data. However, these parameters may not accomplish what you need in order to store the necessary Linux for S/390 files on OS/390.

Primarily, they are as follows:

- Data sets on OS/390 have a record format. We suggest you store in a record format similar to what the tape looks like. At ❶ in Figure 10 we use a logical record length (lrecl) of 1024, a block size (blksize) of 8192, and a record format (recfm) of FB. You may use others, but the final copy step must be able to convert them to the format required by the IPL process.
- New data sets on OS/390 must be allocated before they can be used. The size must be large enough to hold the data. In our case we are on disk model 3390-mod 3. We estimated the size to be about 60 tracks (one track being about 56 KB). We defined this as the primary size (pri=60), and allowed the data set to increase by 10 tracks (sec=10) at a time. The allocation unit is a track. This is shown at ❷ in Figure 10 on page 52.

If you have an OS/2-based workstation, the upload is very similar to the procedure described previously. Only the `site` command is entered differently, that is, without the `quote` prefix, as shown in Figure 11.

```
.
ftp> bin
200 Representation type is Image
ftp> site track pri=60 sec=10
200 Site command was accepted
ftp> site lrecl=1024 blksize=8192 recfm=fb
200 Site command was accepted
ftp> put initrd.txt
200 Port request OK.
125 Storing data set USERID.INITRD.TXT
250 Transfer completed successfully.
2991104 bytes sent in 78.00 seconds (38.35 Kbytes/sec)
ftp>
```

Figure 11. OS/2 FTP example

5.5.3 JCL to create the tape

The IPL tape must be created as an unlabeled 3490 tape. However, unlabeled tapes are rarely used today. Therefore, we have included a small job here to show how a standard label tape can be unlabeled:

```

//LINUXTP JOB (999,POK),NOTIFY=&SYSUID,
// CLASS=F,MSGCLASS=T,MSGLEVEL=(1,1)
//STEP0 EXEC PGM=IEBGENER
//SYSUT1 DD *
DDD
//SYSUT2 DD DISP=(NEW,PASS),LABEL=(1,BLP),DSN=DUMMY,
// DCB=(RECFM=F,LRECL=1024),
// UNIT=3490
//SYSIN DD DUMMY

```

The trick is to overwrite the label, which itself is a file on tape. When reading the tape as a standard labeled tape, it is interpreted as label data and not available to a program reading that tape. The *bypass label processing* option (option BLP) may not be available without intervention by your installation's security services.

After all files are uploaded to OS/390, the IPL tape can be created. The following is a JCL example:

```

//LINUXTP JOB (999,POK),NOTIFY=&SYSUID,
// CLASS=F,MSGCLASS=T,MSGLEVEL=(1,1)
//IMAGE EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD DISP=SHR,DSN=USERID.IMAGE.TXT
//SYSUT2 DD DISP=(NEW,PASS),LABEL=(1,NL),DSN=DUMMY,
// DCB=(RECFM=F,LRECL=1024),
// UNIT=3490
//SYSIN DD DUMMY
//PARMLINE EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD DISP=SHR,DSN=USERID.PARMLINE.TEXT
//SYSUT2 DD DISP=(NEW,PASS),LABEL=(2,NL),DSN=DUMMY,
// DCB=(RECFM=F,LRECL=1024),
// VOL=(,RETAIN,,REF=*.IMAGE.SYSUT2),
// UNIT=3490
//SYSIN DD DUMMY
//INITRD EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD DISP=SHR,DSN=USERID.INITRD.TXT
//SYSUT2 DD DISP=(NEW,KEEP),LABEL=(3,NL),DSN=DUMMY,
// DCB=(RECFM=F,LRECL=1024),
// VOL=(,RETAIN,,REF=*.IMAGE.SYSUT2),
// UNIT=3490
//SYSIN DD DUMMY

```

As the tape has to be written with fixed-length records, DCB parameters are applied to the SYSUT2 statements. Since the tape is written without labels,

the names of the data sets on the tape are irrelevant. Linux for S/390 only expects an order of three files on the tape. The order is:

1. The kernel image file *USERID.IMAGE.TXT*
2. The parmline file *USERID.PARMLINE.TEXT*
3. The initial root file system *USERID.INITRD.TXT*

If your host system is a VM-based environment, you can create the tape from there, as well. The commands to issue on a CMS virtual machine are as follows—but make sure that the files are correctly blocked on CMS before writing the tape:

```
rew 181
filedef outmove tap1 (recfm f block 1024 lrecl 1024 perm
filedef inmove disk image txt a
move
filedef inmove disk parm file a
move
filedef inmove disk initrd txt a
move
rew 181
```

Note: Remember to rewind the tape to the beginning.

Since it is unusual to handle text data as in parmline on OS/390 with a width of 1024 characters, we tested a parmline data set having a fixed record length of 80. The data set was allocated with attributes as shown:

```
Data Set Information
Command ==>

Data Set Name . . . : USERID.PARMLINE.TXT80

General Data                               Current Allocation
Volume serial . . . : TOTTS3                Allocated tracks . : 3
Device type . . . . : 3390                 Allocated extents . : 1
Organization . . . . : PS
Record format . . . . : FB
Record length . . . . : 80
Block size . . . . . : 27920                Current Utilization
1st extent tracks . : 3                     Used tracks . . . . : 1
Secondary tracks . . : 1                    Used extents . . . . : 1

Creation date . . . : 2000/05/23
Referenced date . . : 2000/05/23
Expiration date . . : ***None***
```

Remember to *set the numbering off* when editing a parmline data set with DCB parameters like these and in addition the data must be in lower case. The parmline data set now looks as follows:

```
File Edit Confirm Menu Utilities Compilers Test Help
ssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
EDIT          HDM.PARMLINE.TXT80                      Columns 00001 00072
Command ==>                                         Scroll ==> CSR
***** ***** Top of Data *****
000001 dasd=9AC,996,998
000002 root=/dev/ram0 ro
***** ***** Bottom of Data *****
```

In addition, we changed the job step to copy the parmline data set:

```
//PARMLINE EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD DISP=SHR,DSN=HDM.PARMLINE.TXT80
//SYSUT2 DD DISP=(NEW,PASS),LABEL=(2,NL),DSN=DUMMY,
//        DCB=(RECFM=FB,LRECL=80,BLKSIZE=960),
//        VOL=(,RETAIN, ,REF=*.IMAGE.SYSUT2),
//        UNIT=3490
//SYSIN DD DUMMY
```

We chose to write with a blocksize of 960 bytes to tape. This means that both lines—including the trailing blanks—are copied to tape. The data is written in *one* physical block of 160 bytes to tape. This is indicated in Figure 12.

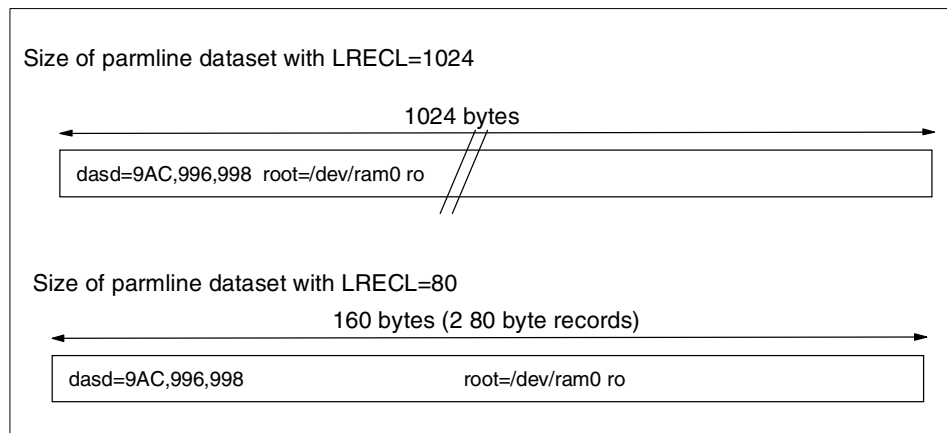


Figure 12. Size of parmline file on tape

Attention

Note the following regarding “how much data is read” from the parmline file:

- Linux for S/390 has a general read routine that reads in chunks of 1024 bytes. This routine detects short blocks. When reading the parmline file, this will normally be the case.
- Current kernel 2.2.15 handles only parmline data up to 896 bytes. This is a hard limit. You cannot exceed it.
- If you have a setting that ends exactly on 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 will be concatenated and presented to the kernel as *one* line of text.

5.6 Using the Hardware Management Console (HMC) to IPL

HMC connections vary from installation to installation. At the time of writing, our installation had an HMC talking over TCP/IP to the Support Element (SE) of each Central Electronics Complex (CEC) via a private Token Ring connection.

Linux for S/390 installers should be aware that the IPL of Linux for S/390, whether it be via tape or DASD, must be done on the HMC. There is no interface available to telnet into the box to perform the IPL, since Linux for S/390 is not active yet.

Terminology

When we speak of reactivating Linux for S/390, we normally use the terms *boot* or *reboot*. But since Linux for S/390 is activated from the HMC, it does not know anything about activation profiles (an HMC term), etc. So the terms used are *IPL* (meaning initial program load) or *reIPL*. Therefore, we do not use the terms boot and reboot in this chapter.

Let's go through the steps to IPL the Linux for S/390 partition from tape, as well as from DASD.

1. Highlight your Linux for S/390 partition. In our case, this was partition A5 on CEC SCZP701 (see Figure 13 on page 58).

2. Double-click **Reset Clear** on the right side to clear your storage.

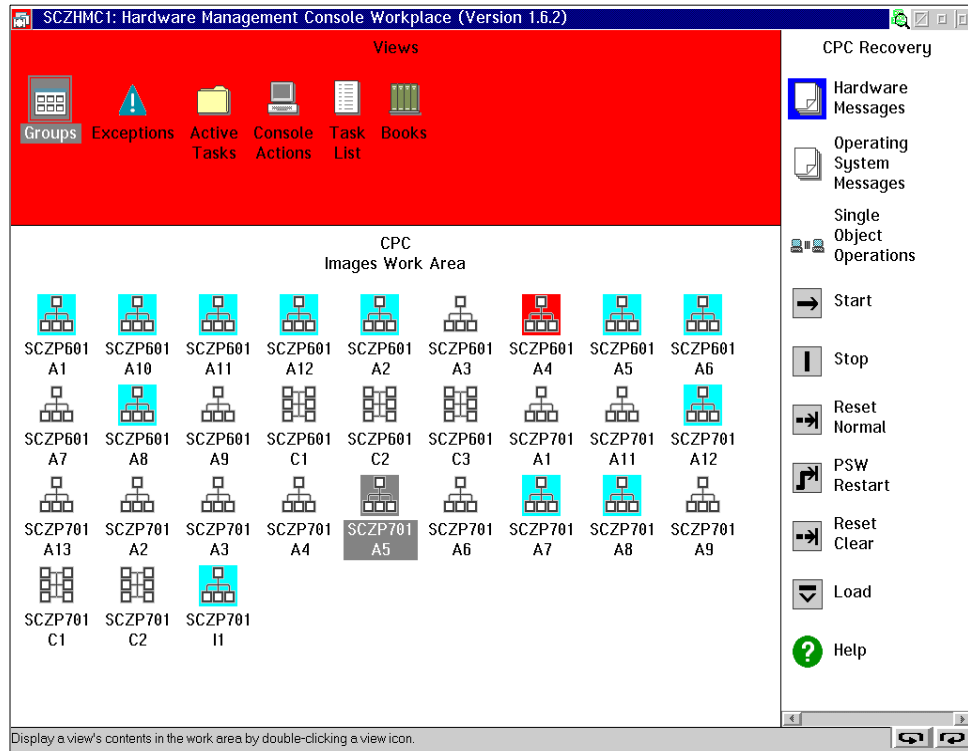


Figure 13. HMC CPC Images Work Area

3. After the Reset Clear is done:

- If IPLing from tape, place your tape into the appropriate tape drive. Double-click **Load** (also located on the right), and enter the tape drive unit address/device number as the load address. In our case, it was 0B31. See Figure 14 on page 59 for an example of the load dialog. There are no load parameters, so blank this field out.



Figure 14. Load with unit address of tape drive

- If IPLing from DASD, it is best to have an activation profile created with the address of the IPL volume in it. You can also have an activation profile for tape IPL. Once you have an activation profile, double-click the **Activation** task icon on the right of the HMC active window instead of performing the load task. Again, as on the load panel, the load address field is filled with the tape unit address/device number of the IPL DASD. The load parameter remains empty.
4. For both the load and the activate task, you will receive another panel to confirm the action. Click **Yes** to confirm.
 5. Once the load is complete, double-click **Operating messages** to wait for IPL messages. On our local system, these messages appeared within 30 seconds for the tape IPL and almost instantaneously for the DASD IPL. The messages you see on the HMC will also be captured in file `/var/log/dmesg` by Linux for S/390.

Figure 15 on page 60 shows the output of the last screen of the file `/var/log/dmesg` after Linux for S/390 has IPLed. For a detailed description of these messages, see Chapter 8, “Linux for S/390 bootup and shutdown” on page 165.

```

RAM disk driver initialized: 16 RAM disks of 8192K size
loop: registered device at major 7
dasd:initializing...
dasd(eckd):3390/a (3990/1) Cyl: 3339 Head: 15 Sec: 224
dasd(eckd):Estimate: 58786 Byte/trk 2074 byte/kByte 33 kByte/trk
dasd(eckd):Verified: 58786 B/trk 5202 B/Blk(4096 B) 12 Blks/trk 48
kB/trk
dasd:2404080 kB <- 'soft'-block: 4096, hardsect 4096 Bytes
dasd:devno 9ac added as minor 0 (ECKD)
channel: no Channel devices recognized
RAMDISK: Compressed image found at block 0
VFS: Mounted root (ext2 filesystem).
Freeing unused kernel memory: 0k freed
Starting lcs module

lcs: tr0 configured as follows read subchannel=7b0 write subchannel=7b1
hw_address=40:00:09:FF:71:C0 rel_adapter_no=0

lcs configured to use sw statistics,
ip checksumming of received packets 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)
[root@linuxx log]#

```

Figure 15. `cat /var/log/dmesg`

If a tape IPL was used, the tape may now be unloaded from the tape reader.

During the IPL from tape, you will be prompted to enter the information you gathered about your network interface at the beginning of this chapter. This is a script (located in `/etc/sysconfig/network-scripts`, and unfortunately deleted after running) that is kicked off by Linux for S/390.

After the appropriate information is entered, an `insmod` command is invoked, depending on the type of interface. For example, we used an OSA-2 Token Ring adapter. Linux for S/390 knows this as an LCS device and therefore invokes the `insmod` command followed by the LCS device parameters, also from the `netsetup` script. The prompts and your answers to these prompts will be displayed and entered on the HMC. Figure 16 on page 61 shows the display after you have successfully entered all input and the interface is initialized by Linux for S/390.

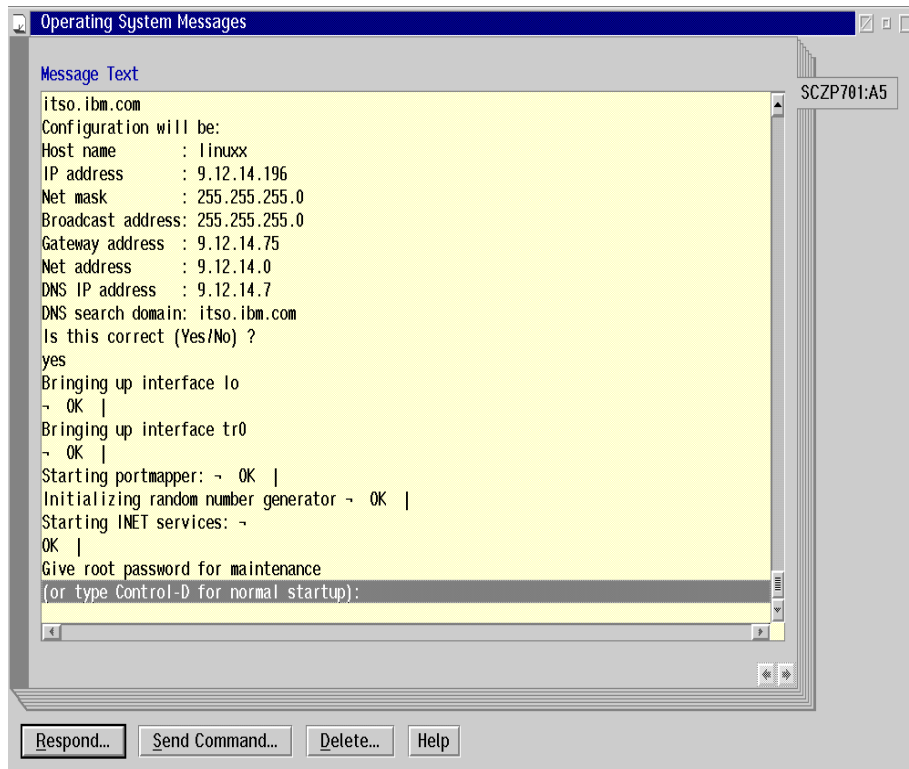


Figure 16. HMC display of network initialization

There is a possibility that you will run into problems with your network interface. You will notice this when you have responded to the prompts to the network script and the message indicating that the initialization of your LCS device (in our case the Token Ring tr0), has been delayed. See Figure 17 on page 62 for an example.

If this happens, double-check your network information. Make sure your OSA card is not shared by more than eight images/systems (kernel version 2.2.15 may have fixed this, but was not tested). In fact, when prompted for the LCS parameter during the network script, enter the `noauto=1` parameter to tell Linux for S/390 not to autosense the card. You achieve this by specifying the device number as:

```
noauto=1 devno_portno_pairs=0x21C0,0
```

where `21C0` is the device number of our OSA card, and we will be using port 0.

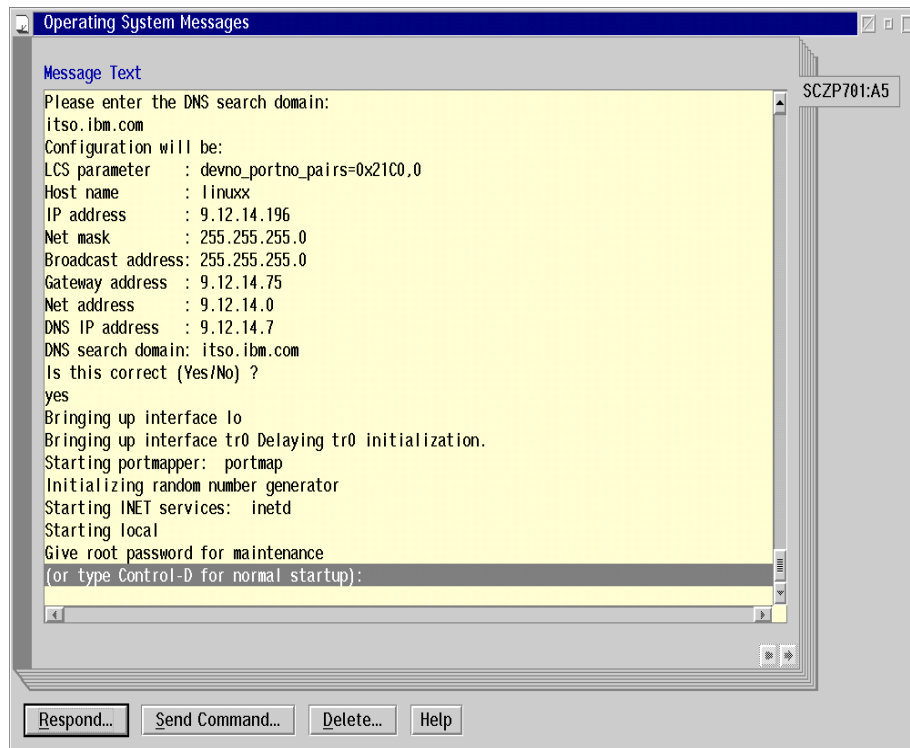


Figure 17. Delaying tr0 initialization

Unlike with OS/390, when Linux for S/390 is IPLed, there is no resulting master console to enter commands on. All Linux for S/390 commands are entered through the HMC using the Send Command button and the dialog box given. Replies to the network prompts in the previous figure are entered using this method. Figure 18 on page 63 shows an example.

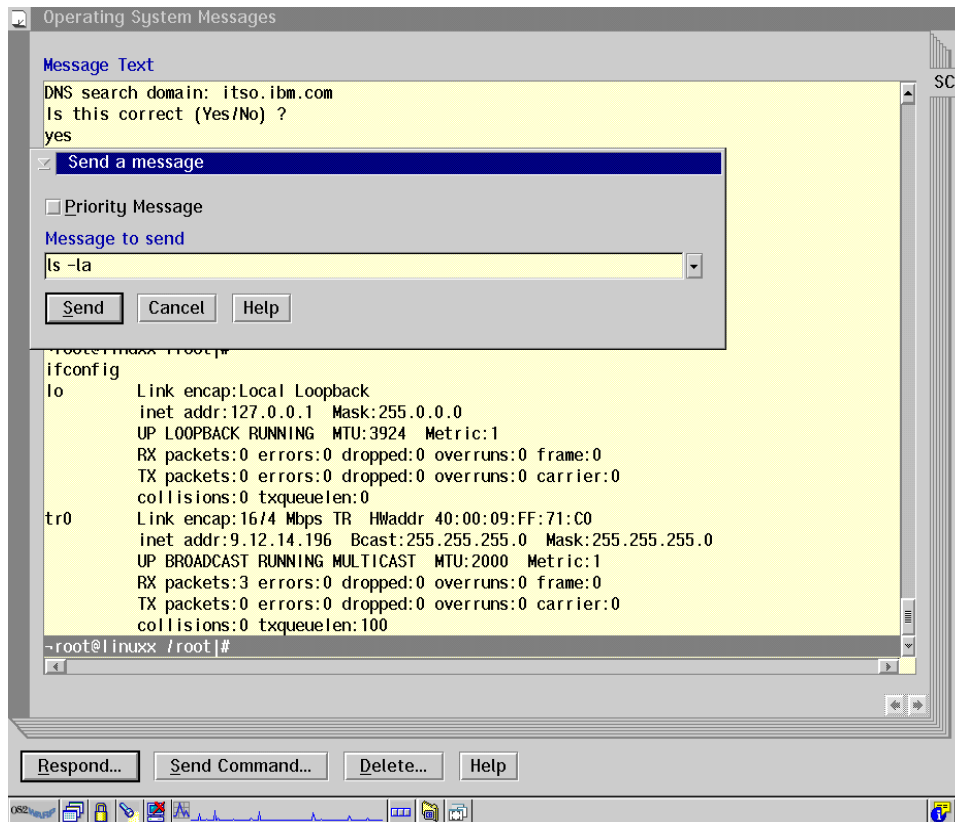


Figure 18. HMC Send Command dialog

When using the Send Command dialog, commands may be retrieved by using the up arrow key, just as you would do on your PC workstation with *doskey* active. But this only works for the last command issued. Use of the control key, such as Cntrl-C, is not supported on the HMC. Also, if you are working from the HMC only (which is not recommended when Telnet is available), be advised that when you enter the password for the system, it is not blanked out with asterisks or blanks; that is, *the password can be seen*.

The password for root on the tape IPL system is pass4root. It will also be the same password when you IPL from DASD. Since Cntrl-C is not supported, do not enter a command that will continually loop, such as `ping`. Use the `-c` flag to specify the number of packets to be sent; for example:

```
ping <ipadr> -c 3
```

5.7 Verifying the IPL from tape

Once the system is IPLed, check it by issuing the following commands and examining the output for accuracy. This will include things such as CPU information, network configuration, and file systems mounted.

The `cat` command concatenates the contents of a file to the console. The `more` command (meaning list the contents of a file page by page) is also available for files that are too large to fit on one screen. Let's look add some interesting informations:

```
cat /proc/cpuinfo
```

```
[root@linuxx /root]# cat /proc/cpuinfo
vendor_id      : IBM/S390
# processors   : 2
bogomips per cpu: 144.58
processor 0: version = 97,  identification = 050822,  machine = 9672
processor 1: version = 97,  identification = 150822,  machine = 9672
```

Figure 19. `cat /proc/cpuinfo` command

Figure 20 shows the output of the `cat /proc/meminfo` command:

```
[root@linuxx /root]# cat /proc/meminfo
          total:  used:  free:  shared: buffers:  cached:
Mem:  794341376 22081536 772259840 4055040 8388608 4161536
Swap:          0         0         0
MemTotal:    775724 kB
MemFree:     754160 kB
MemShared:    3960 kB
Buffers:     8192 kB
Cached:      4064 kB
SwapTotal:    0 kB
SwapFree:    0 kB
```

Figure 20. `cat /proc/meminfo` command

Figure 21 shows the output of the `cat /proc/interrupts` command:

```
cat of this file yielded a "bad file descriptor" message. This is due to
an error in the kernel. You should not receive this message. It seems as
though the filename was added to the directory but no actual file
existed.
```

Figure 21. `cat proc/interrupts` command

Figure 22 shows the output of the `df` command:

```
[root@linuxx /root]# df
Filesystem          1k-blocks      Used Available Use% Mounted on
/dev/ram0            7931           7931         0 100% /
```

Figure 22. Output of the `df` command

Figure 23 shows the output of the `ifconfig` command:

```
[root@linuxx /]# ifconfig
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

tr0         Link encap:16/4 Mbps TR  HWaddr 40:00:09:FF:71:C0
            inet addr:9.12.14.196  Bcast:9.12.14.255  Mask:255.255.255.0
            UP BROADCAST RUNNING MULTICAST  MTU:2000  Metric:1
            RX packets:6843 errors:0 dropped:0 overruns:0 frame:0
            TX packets:763 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:100
```

Figure 23. Output of the `ifconfig` command

5.7.1 IPL messages for Linux

All messages issued to the HMC are stored in the file `/var/log/dmesg`. Figure 24 on page 66 contains an example of its contents. We picked the portion that shows the DASD initialization. For a more detailed description of the IPL

messages, see 8.2, “Kernel initialization” on page 168. As you can see in our case, the device numbers 998, 99A, and 9AC are being made available for Linux for S/390.

```
dasd:initializing...
dasd(eckd):3390/a (3990/1) Cyl: 3339 Head: 15 Sec: 224
dasd(eckd):Estimate: 58786 Byte/trk 2074 byte/kByte 33 kByte/trk
dasd(eckd):Verified: 58786 B/trk 5202 B/Blk(4096 B) 12 Blks/trk 48
kB/trk
dasd:2404080 kB <- 'soft'-block: 4096, hardsect 4096 Bytes
dasd:devno 998 added as minor 8 (ECKD)
dasd(eckd):3390/a (3990/1) Cyl: 3339 Head: 15 Sec: 224
dasd(eckd):Estimate: 58786 Byte/trk 2074 byte/kByte 33 kByte/trk
dasd(eckd):Verified: 58786 B/trk 5202 B/Blk(4096 B) 12 Blks/trk 48
kB/trk
dasd:2404080 kB <- 'soft'-block: 4096, hardsect 4096 Bytes
dasd:devno 99a added as minor 4 (ECKD)
dasd(eckd):3390/a (3990/1) Cyl: 3339 Head: 15 Sec: 224
dasd(eckd):Estimate: 58786 Byte/trk 2074 byte/kByte 33 kByte/trk
dasd(eckd):Verified: 58786 B/trk 5202 B/Blk(4096 B) 12 Blks/trk 48
kB/trk
dasd:2404080 kB <- 'soft'-block: 4096, hardsect 4096 Bytes
dasd:devno 9ac added as minor 0 (ECKD)
```

Figure 24. `cat /var/log/dmesg` command

5.7.2 Formatting DASD for Linux

In this section we go through the process to configure a DASD for Linux for S/390 to store the root files system, and the IPL information to IPL from DASD later.

5.7.2.1 DASD major number/minor number

Linux for S/390 supports a variety of devices. The supported types can be found in the file `/proc/devices`. See Figure 25 on page 67 for an example of the output of the `cat /proc/devices` command.

```
[root@linuxx /root]# cat /proc/devices
Character devices:
 1 mem
 2 pty
 3 tty
 4 ttyS
 5 console
10 misc

Block devices:
 1 ramdisk
 7 loop
94 dasd
95 mmd
```

Figure 25. cat /proc/devices command

The device types are prefixed by their device major numbers. As you can see, the major number 94 represents DASD. The major number for DASD is broken down into four minor numbers, starting with 0. The first minor number represents the device and the remaining three represent the three possible partitions into which the DASD can be formatted.

For a more detailed discussion on major numbers, minor numbers, and their uses, see 9.1, “Devices” on page 177. The numbering scheme for two DASD is shown in Figure 26 on page 68.

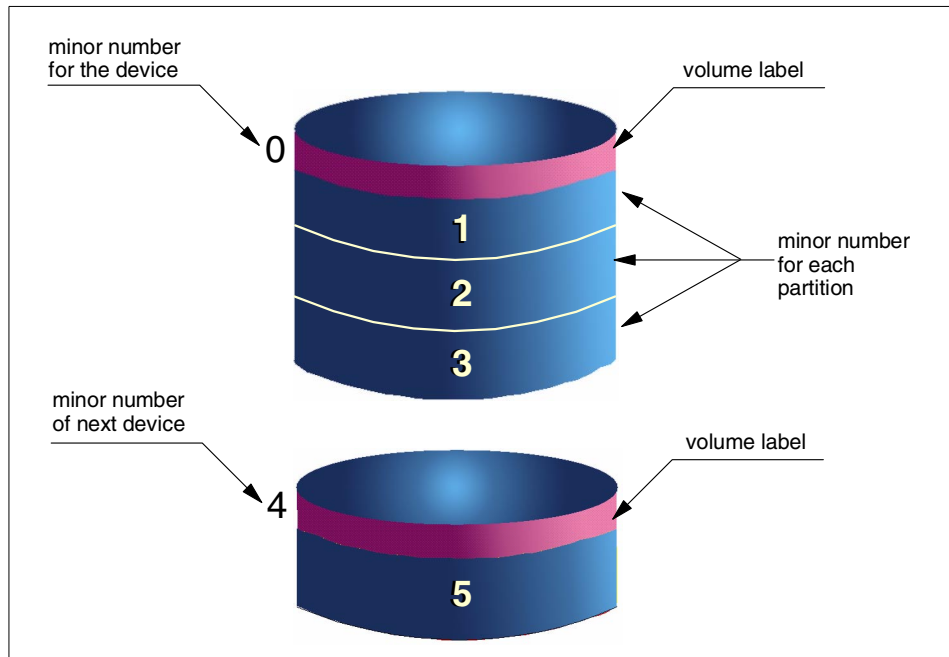


Figure 26. Major number 94 - DASD

5.7.2.2 DASD limitations

At the time of writing, Linux for S/390 supports up to 64 DASD. Also, though a major number is broken down into 3 minor numbers (each being a partition), Linux for S/390 at present supports the formatting of only one partition per DASD, and this partition takes up the entire volume.

If you want to use your DASD as storage media for your Linux for S/390 system (hard disks), you need to format them using the `dasdfmt` utility. First check the file `/proc/dasd/devices` (use the `cat` command) to see what devices have been reserved for your Linux for S/390 system.

There may be an instance where you may not see the major number of 94 for your DASD in the output of this file (that is, if the `mknod` commands were not issued for you in the file system). If this is the case, then issue the following command:

```
mknod devicenodename b 94 <minor_number>
```

where `minor_number` is contained in `/proc/devices`. The contents of our `/proc/dasd/devices` file are shown in Figure 27 on page 69.


```
[root@linuxxx /root]# cat /proc/dasd/devices
dev# MAJ minor node      Format
09AC  94      0 /dev/dasda  4096
099A  94      4 /dev/dasdb  4096
0998  94      8 /dev/dasdc  n/f
```

Figure 27. `cat /proc/dasd/devices` command

Fortunately, for the file system on the Marist site, the nodes “a” through “h” have already been created for the DASD and the first partition of the DASD. To see the definitions, look at the contents of your /dev directory. An example is shown in Figure 28.

```
[root@linuxxx /dev]# cd /root
[root@linuxxx /root]# cd /dev
[root@linuxxx /dev]# ls -la dasd*
brw-r--r--  1 root   root    94,   0 Jan  5 18:59 dasda
brw-----  1 root   root    94,   1 Jan  5 19:00 dasda1
brw-r--r--  1 root   root    94,   4 Jan  5 18:59 dasdb
brw-r--r--  1 root   root    94,   5 Jan  5 19:00 dasdb1
brw-r--r--  1 root   root    94,   8 Jan  5 19:00 dasdc
brw-r--r--  1 root   root    94,   9 Jan  5 19:00 dasdc1
brw-r--r--  1 root   root    94,  12 Jan  5 19:00 dasdd
brw-r--r--  1 root   root    94,  13 Jan  5 19:00 dasdd1
brw-r--r--  1 root   root    94,  16 Jan  5 19:00 dasde
brw-r--r--  1 root   root    94,  17 Jan  5 19:39 dasde1
brw-r--r--  1 root   root    94,  20 Jan  5 19:00 dasdf
brw-r--r--  1 root   root    94,  21 Jan  5 19:39 dasdf1
brw-r--r--  1 root   root    94,  24 Jan  5 19:00 dasdg
brw-r--r--  1 root   root    94,  25 Jan  5 19:39 dasdg1
brw-r--r--  1 root   root    94,  28 Jan  5 19:00 dasdh
brw-r--r--  1 root   root    94,  29 Jan  5 19:39 dasdh1
[root@linuxxx /dev]#
```

Figure 28. `mknods` already performed on Marist file system

As you can see in Figure 27, devices 9AC, 99A, and 998 have been reserved for the Linux for S/390 system. We specifically chose not to format the 998 volume to show you that the format is unknown and is identified by n/f. Their major nodes are all 94 and their minor numbers are assigned in groups of 4, starting with the number zero. See Figure 26 on page 68 for an illustration.

These volumes will later house the boot record, customized file system, and swap space. Since the initrd file system on tape is limited in space, we need to allocate space on DASD, create a file system on it, and mount it onto the initrd file system to continue customization.

We recommend that you dedicate DASDs for your Linux for S/390 system and not allow any other partition access to them. If not dedicated, another partition, perhaps an OS/390 system, could gain access to the pack and accidentally perform a destructive action such as a defrag or initialization of the pack.

The `dasdfmt` command is used to format space on the DASD reserved for Linux. The volume's Volume Table Of Contents (VTOC) and label area are erased. Linux for S/390 will know this DASD as `/dev/dasda`. The format of the `dasdfmt` command is:

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

The blocksize (`-b 4096`) can be any power of 2 between 512 and 4096. Since the `extf2fs` file system uses 1 KB blocks, there is a performance gain if you specify a blocksize of 1024 or higher. We recommend a blocksize of 4096. Once entered, Linux for S/390 will ask for your confirmation by issuing the following message:

```
I am going to format the device /dev/dasdc in the following way:
Device number of device : 0x998
Major number of device  : 94
Minor number of device  : 8
Start track              : 0
End track                : last track of disk
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:
```

A reply of `yes` will cause all data on the volume 0998 to be erased. This includes IBM volume labels, VTOC, and data. Depending on the size of the volume, the format may take some time. From our experience, a 3390 Mod 3 (2.1 GB) volume takes over 20 minutes to format. (This, of course, is contingent upon the type of hardware you are running on. Times at your installation may differ.)

If you need to view the `dasdfmt` process, you may telnet to the Linux for S/390 system, logon as root, and issue the `ps -ef` command. You will see the process running, but not its status.

The next thing to do is create a file system on this DASD. This cannot be done until the `dasdfmt` is completed. To create a file system, use the `mke2fs` command. At the time of writing, the `mke2fs` command did not support creating more than one file system on a DASD volume. In other words, the file system created with the `mke2fs` command will create a partition, e.g. `dasda1`, which will take up a full DASD volume. The format of the `mke2fs` command is:

```
mke2fs -b 4096 /dev/dasd<letter>1
```

where `<letter>` is the letter assigned to the device from the `dasdfmt` utility. The number `1` concatenated to the letter, resulting in `dasda1`, makes this a logical partition instead of an entire device `dasda`. The letter for the device is set when the DASD is detected.

Remember to access the logical partition when referring to the file system. For instance, execute `dasdfmt` on the entire device `dasda`, not the logical device `dasda1`. In the same manner, mount the logical partition (with a file system on it) `dasda1`, not an entire device `dasda`.

5.7.3 Upload and customize the new file system

If you want to be able to IPL from DASD, you will need IPL information written on DASD and a customized root file system. You may choose to download the large or small root file system. Their file names at the time of this writing were `initfs_big.tgz` for the large root file system, and `initfs_small.tgz` for the small root file system. The large file system contains more utilities and applications, such as the Apache Webserver. They both can be downloaded from the Linux390 Marist site:

```
http://linux390.marist.edu
```

Create a temporary mount point under `/mnt` for the DASD that was just formatted. In our case, we use the DASD device `dasdc` to write our file system and IPL record on. Issue the `mkdir` command to create the mount point. We chose to call it `/mnt/dasdc`. It can be called anything you like, such as `/mnt/device_num` or something that helps you easily recognize which piece of DASD you have mounted there.

```
mkdir /mnt/dasd<letter>
```

Mount the file system onto this mountpoint:

```
mount /dev/dasdc1 /mnt/dasdc
```

FTP the large file system to this mountpoint. The FTP should be done in binary mode. No uncompressing is necessary; that will be done once the file system has been transferred.

A file with the extension of .tgz is in compressed format. To uncompress this file, change your directory to /mnt/dasdc and issue the `tar` command:

```
tar xzfBp initfs_big.tgz
```

If you examine the contents of the /mnt/dasdc directory, you will now see a root file system similar to the one that dasdc is mounted on. In other words, there should be very little difference in the file structure between the directory list of / and the directory list of /mnt/dasdc. An example of the result of the `tar` command is shown in Figure 29.

```
[root@linuxx dasdc]# tar xzfBp initfs_big.tgz
[root@linuxx dasdc]# ls -la
total 115965
drwxr-xr-x 17 root    root      4096 May 17 01:35 .
drwxr-xr-x  5 root    root      1024 May 17 01:17 ..
drwxr-xr-x  2 root    root      4096 May 10 18:16 bin
drwxr-xr-x  2 root    root      4096 May 12 14:45 boot
drwxr-xr-x  2 root    root     12288 May  9 18:37 dev
drwxr-xr-x 16 root    root      4096 May 10 18:16 etc
drwxr-xr-x  3 root    root      4096 May 10 18:15 home
-rw-r--r--  1 root    root    118536619 May 17 01:27 initfs_big.tgz
drwxr-xr-x  4 root    root      4096 May 10 17:48 lib
drwxr-xr-x  2 root    root     16384 May 17 00:33 lost+found
...
```

Figure 29. Tar uncompress of big file system

Note to OS/390 System Programmers

As illustrated in Table 5, this Linux for S/390 customization process can be compared with HFS customization on OS/390 UNIX System Services. That is, space for an HFS is allocated on an SMS-managed volume. This HFS is then mounted on a system's temporary mountpoint. All customization on that HFS is done here. The HFS is then unmounted and mounted on the system for which it is customized. If this HFS is a root file system, the customization is done while the target system is down and will be IPLed when the root HFS is customized and unmounted.

Table 5. OS/390 UNIX System Services HFS vs. Linux for S/390 customization

OS/390 UNIX System Services HFS root file system customization	Linux for S/390 file system customization
Allocate PDSE for USS root HFS.	Dasdfmt of device.
Restore root HFS from dump using ADDRDSU.	Create file system with mke2fs. Uncompress the file system with the tar command.
Mount restored root HFS on a temporary mount point.	Mount file system on /mnt/dasd<letter>1.
Customize the restored HFS: - resolv.conf file - hosts file - profile - pty/tty pairs of /dev	Customize the Linux for S/390 file system: - resolv.conf file - ifcfg-tr0 or ifcfg-eth0 (Token Ring or Ethernet)
Update BPXPRM00 if necessary.	Customize /etc/fstab. Create the parmline file. Execute the silo command for boot device/record.
IPL or reIPL the system depending on whose root HFS it is.	ReIPL the Linux for S/390 partition.

If you do not need to create a swap space, refer to 5.7.5, "Customizing Linux for S/390 configuration files" on page 75.

5.7.4 Creating and activating swap space

Swap space is needed to run your system more efficiently. It can be created in two ways: either by taking up a complete DASD volume for swap, or by

allocating a swap file in a file system. As a rule of thumb, the size of the swap space should be the same as the size of memory; however, the effectiveness of the swap space depends greatly on memory size and workload.

1. Format DASD as your swap space.

- Enable a DASD device as your swap space with the `mkswap` command. In our case we used:

```
mkswap /dev/dasda1
```

- Activate the swap DASD with the `swapon` command. In our case we used:

```
swapon /dev/dasda1
```

2. Create a swap file in the file system. This can have the disadvantage of excess I/O due to swap file fragmentation. A swap file on a separate DASD partition would be ideal, but this is mentioned for future availability since DASD partitioning is not yet supported. Our recommendation is to create a swap file on each device being accessed (that is, a swap file for `dasda1`, `dasdb1`, etc.).

- Create a directory named `/swap` with permissions of 700. Allocate a swap file under `/swap`. We recommend starting with a size of 64 MB for the swap file and moving up or down according to your application and workload. There is no clear-cut answer as to what size is optimum. Swap sizes have been known to go up to twice the amount of central storage. Ensure that you have enough space for the swap file on the file system by issuing a `df -k` command.

- Issue the raw copy command `dd`, using the device `/dev/zero` to allocate space in the file system and create the swap file. In our case we used the following:

```
dd if=/dev/zero of=/mnt/dasda1/swap/swapfile bs=1M count=64
```

- Enable this file with the `mkswap` command. In our case, we used:

```
mkswap -c /mnt/dasda1/swap/swapfile
```

- Change the mode of the swap directory to 700 (read/write/execute for owner) and of the swap file to 600 (read/write for owner). In our case we used:

```
chmod 700 /mnt/dasda1/swap  
chmod 600 /mnt/dasda1/swap/swapfile
```

- Activate the swap file with the `swapon` command:

```
swapon /mnt/dasda1/swap/swapfile
```

- Verify your results, as shown in Figure 30 on page 75.

```
[root@linuxx /swap]# cat /proc/swaps
Filename                                Type              Size    Used   Priority
/mnt/dasda/swap/swapfile                file              65532   0      -1
[root@linuxx /swap]#
```

Figure 30. *cat of /proc/swaps*

Note

As you can see, the `swapon` command was used to activate the swapfile `/mnt/dasda/swap/swapfile`. It is important to note that it is *not* the `swapon` command that gives you your swap space later on with the DASD IPL. It is an entry in the file `/etc/fstab` that will cause `swapon` to be launched.

If needed, Linux for S/390 can access expanded storage for its swap space. This is much faster than DASD and is discussed in detail in 9.1.3, “XPRAM” on page 181 and in 9.3.4.1, “Swap space on a ramdisk” on page 190.

5.7.5 Customizing Linux for S/390 configuration files

Now you should be ready to customize the root file system that you have mounted on `/mnt/dasdx` (`/mnt/dasdc` in our case). Several steps are needed to customize configuration files.

The file `/etc/fstab` must be updated to have entries pointing to your root file system and your swap space if you created a swap file. Remember, the `fstab` you want is the one in the file system mounted on `/mnt/dasdc/etc`, not off the root of your ram disk in `/etc`. All your customization should be done on files in the mounted file system. The `fstab` file, after customization, should look something like the file shown in Figure 31.

```
[root@linuxx /etc]# cat fstab
/dev/dasdc1      /                ext2    defaults,errors=remount-ro 0 1
/swap/swapfile  swap             swap    defaults    0    0
none            /proc            proc    defaults    0    0
[root@linuxx /etc]#
```

Figure 31. *Customized fstab*

If you recall, when Linux for S/390 is first IPLed from tape on its ram disk (`initrd`), a network script is presented that asks questions about your network

environment. See Figure 16 on page 61 to refresh your memory. Most of that information is stored in the file `/etc/sysconfig/network-scripts/ifcfg-tr0`. `tr0` was used in our case since we were configuring a Token Ring interface. Copy this file to `/mnt/dasdc/etc/sysconfig/network-scripts/ifcfg-tr0` with the following command:

```
cp /etc/sysconfig/network-scripts/ifcfg-tr0 /
  mnt/dasdc/etc/sysconfig/network-scripts/ifcfg-tr0
```

An example of the contents of this file is shown in Figure 32 on page 76.

```
[root@linuxx network-scripts]# cat ifcfg-tr0
DEVICE=tr0
USERCTL=no
ONBOOT=yes
BOOTPROTO=none
BROADCAST=9.12.14.255
NETWORK=9.12.14.0
NETMASK=255.255.255.0
IPADDR=9.12.14.196
[root@linuxx network-scripts]#
```

Figure 32. `cat /etc/sysconfig/network-scripts/ifcfg-tr0`

During the IPL from DASD, the kernel looks for file `/etc/sysconfig/network`. This file is used as a switch to control network configuration. If the file exists, then the kernel will examine its contents and configure the network. If it does not exist, then the kernel will skip that step. Copy the network file from `/etc/sysconfig`. In our case we used:

```
cp /etc/sysconfig/network /mnt/dasdc/etc/sysconfig
```

Copy the file `/etc/conf.modules` to the mounted file system. In our case we used:

```
cp /etc/conf.modules /mnt/dasdc/etc
```

There are two files that you can create/update for nameserver support, `/etc/resolv.conf` and `/etc/hosts`. This is *not* required to get your Linux for S/390 and network interface up and running. You can defer this till later if you want to have a caching nameserver. For a complete description of setting up the nameserver, see 18.1, “Introduction to DNS” on page 349.

5.7.5.1 /etc/resolv.conf

The nameserver address and the domain name search order were stored into this file of the ram disk when the tape IPL network script was run. Copy this file into /mnt/dasdc/etc/resolv.conf. An example of the contents of /etc/resolv.conf is shown in Figure 33.

```
[root@linuxx /etc]# cat resolv.conf
search itso.ibm.com
nameserver 9.12.14.7
[root@linuxx /etc]#
```

Figure 33. `cat /etc/resolv.conf`

5.7.5.2 /etc/hosts

Your TCP/IP address and the name it resolves to can be placed into your host file. The ram disk does not contain this file, so it will have to be created. Once more, you can use the `echo` command to create the file since it is only two lines long. Echo can be used to pipe a line to a file and then append another line to the bottom of that file:

```
echo '127.0.0.1 localhost' > /mnt/dasdc/etc/hosts
echo '9.12.14.196 linuxx' >> /mnt/dasdc/etc/hosts
```

Both the large and small file systems contain the same network script that was used when you IPLed with the ram disk. When IPLing from DASD, this network script will again prompt you for your network environment. Once you have replied to all the prompts, the script will no longer run on subsequent IPLs. This gives you an opportunity to change any values that you might have changed in the interim.

If, however, you are satisfied with the values given from the first IPL and you don't want the network script to run again, then delete /mnt/dasdc/etc/rc.d/rc3.d/S00netsetup, which is a link to /etc/rc.d/init.d/netsetup.

5.7.6 Creating a new kernel

The large and small file systems already contain the kernel image in the /mnt/dasdc/boot directory. The file name is /boot/image. If you have another version of the kernel, it would be placed here. You can build/compile your own kernel, or copy a precompiled kernel into this directory. Doing so now places the kernel onto the DASD that you will IPL.

We recommend that you use a different name for your modified kernel, which you can later activate using the `silo` command. Before modifying your kernel, you should be familiar with 11.6, “Build and customize the kernel” on page 238.

5.7.7 Write IPL information to DASD

We assume that you will use the same disk where you untarred the file system as your IPL disk. This disk already has the needed system files for IPL in the boot directory, except the parameter file, which you have to create.

5.7.7.1 Create parameter file

Just as when we IPLed from tape, so when you IPL from DASD, the kernel will look for a parameter file. You should create one and place it into the `/mnt/dasdc/boot` directory. There should be an example of the parameter file in `/boot/parmfile`. If you do not want to go into the vi editor to edit this file, you can do the following to create a new file called `parmline`:

```
cd /mnt/dasdc/boot
```

```
echo 'dasd=9AC,996,998 root=/dev/dasdc1 ro noinitrd' > parmline
```

The `noinitrd` parameter is required since the kernel that came with the file system was compiled with initial RAM disk support, and you don't want that enabled when you are IPLing from DASD with the full root file system.

Note

Make sure that the `dasd=` statement in `/mnt/dasdc/boot/parmline` is the same as in the parameter file that was written to the IPL tape. The device assignments were based on that `parmline`. If this order is changed on the IPL DASD, then the assignments will also change. What you think may be `dasdc` may no longer be that in this case; additional DASD may be added to the end of the DASD list.

5.7.7.2 Write the IPL record to DASD

The final step before IPLing from DASD is to write the IPL record using the `silo` command:

```
cd /mnt/dasdc/boot
```

```
../sbin/silo -f <kernel> -d <DASD device>  
-p <parameter file> -b <boot sector>
```

For our system, we used:

```
../sbin/silo -f /mnt/dasdc/boot/image -d /dev/dasdc -p  
/mnt/dasdc/boot/parmline -b /mnt/dasdc/boot/ipleckd.boot
```

While testing kernel version 2.2.15, we ran into a warning message when the `silo` command was issued. It stated that the IPL would not be changed unless the `-t2` option was also specified. This is inconsistent since the `-t2` parameter is for testing purposes. See Figure 34 for an example.

```
[root@linuxx boot]# ../sbin/silo -f image -d /dev/dasdc -p parmline -b  
IPLeckd.boot  
o->image set to image  
o->IPLdevice set to /dev/dasdc  
o->parmfile set to parmline  
o->bootsect set to IPLeckd.boot  
Testlevel is set to 2  
IPL device is: '/dev/dasdc'  
bootsector is: 'IPLeckd.boot'...ok...  
bootmap is set to: './boot.aQ5G7z'...ok...  
Kernel image is: 'image'...ok...  
original parameterfile is: 'parmline'...ok...final parameterfile is:  
'parmline'.  
..ok...  
WARNING: silo does not modify your volume. Use -t2 to change IPL records  
ix 0: offset: 0201f7 count: 0c address: 0x00000000  
ix 1: offset: 020204 count: 80 address: 0x0000c000  
ix 2: offset: 020284 count: 80 address: 0x0008c000  
ix 3: offset: 020304 count: 56 address: 0x0010c000  
ix 4: offset: 020676 count: 01 address: 0x00008000  
Bootmap is in block no: 0x00000003  
[root@linuxx boot]#
```

Figure 34. Silo command warning message

The command was then issued with the `-t2` option and it ran successfully. See Figure 35 on page 80 for the messages you should see. Development is aware of this problem.

```

[root@linuxx boot]# ../sbin/silo -f image -d /dev/dasdc -p parmline -b
IPLckd.boot -t2
o->image set to image
o->IPLdevice set to /dev/dasdc
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/dasdc'
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: 0201f7 count: 0c address: 0x00000000
ix 1: offset: 020204 count: 80 address: 0x0000c000
ix 2: offset: 020284 count: 80 address: 0x0008c000
ix 3: offset: 020304 count: 56 address: 0x0010c000
ix 4: offset: 020676 count: 01 address: 0x00008000
Bootmap is in block no: 0x00020677
[root@linuxx boot]#

```

Figure 35. Successful silo command

5.7.8 ReIPL with the customized root file system on DASD

You are now ready to reIPL the system. Before you do this, you should reset the Linux for S/390 system by shutting it down.

5.7.8.1 Shutting down Linux for S/390

We highly recommend that you use the `shutdown` command to deactivate Linux. Do this before you reIPL, otherwise you may see messages like `File system still mounted on the next IPL`.

There are many variations of the `shutdown` command. These are discussed in 8.4, “Shutdown” on page 175. The command we used each time was an immediate shutdown:

```
shutdown -h now
```

This command can be entered either from the HMC or an authorized telnet connection such as one logged on as root. Figure 36 on page 81 shows an

example of the messages displayed on the HMC when the `shutdown -h now` command is issued.

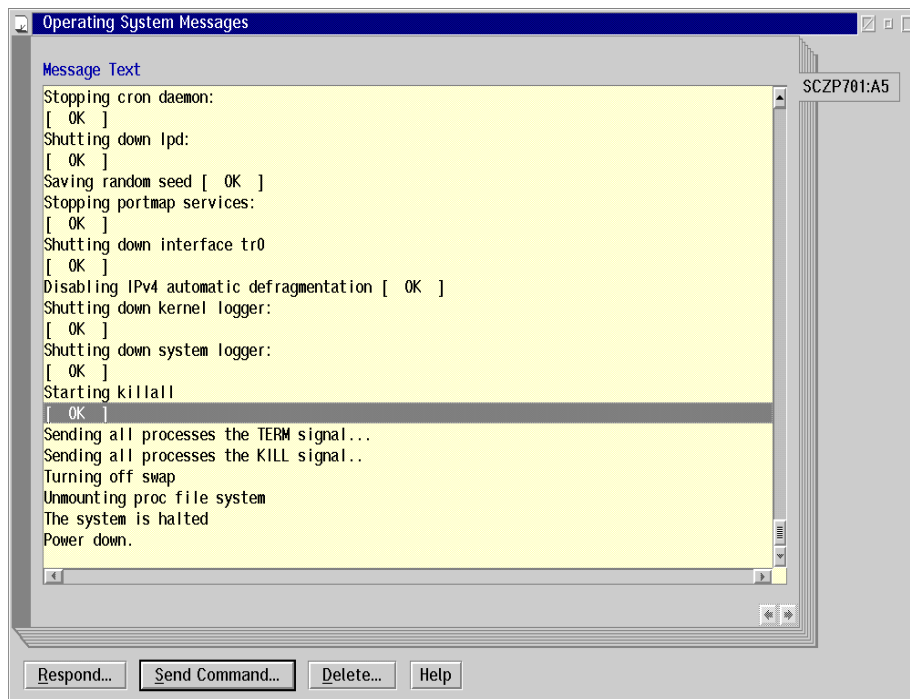


Figure 36. Shutdown messages on the HMC

Although you see the message `Power down`, this does not mean your LPAR has been powered off. You will receive an indication under the hardware messages tab that the partition has entered a disabled wait state condition.

When a shutdown is entered from the HMC, all telnet sessions will receive identical messages, as shown in Figure 37.

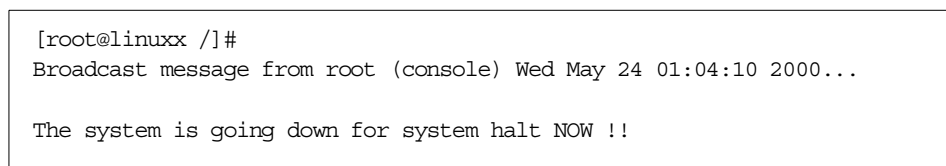


Figure 37. Telnet sessions on shutdown

5.7.8.2 ReIPL from DASD

Now that the system has been safely shut down, you can reIPL using the HMC. As mentioned earlier, a load profile can be created tailored to your Linux for S/390 partition. This is done from the task on the right carousel titled Customize, Delete Activation Profiles. To see an example of the load profile that we set up for our system here, refer to Figure 38:

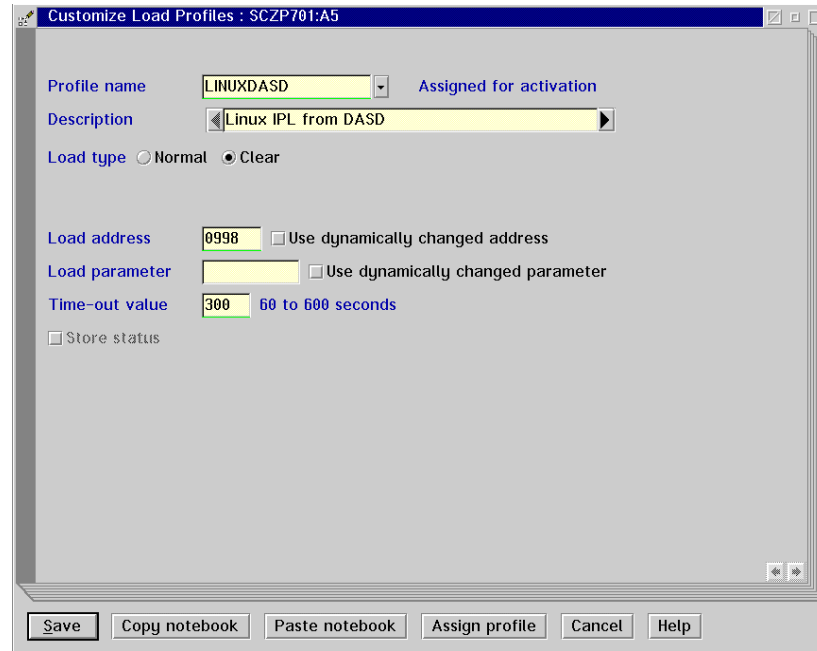


Figure 38. Load profile for IPLing Linux for S/390 on DASD

Your Linux for S/390 system should now be up and running.

5.8 Linux for S/390 on a P/390

Linux for S/390 could run on any of the P/390 family of systems. However, patches for 3215 support may be required.

5.8.1 Attempting to install on a P/390

Linux for S/390 offers two console drivers, a 3215 teletype line printer emulation, and support for the hardware console (HWC).

The HWC has no functional equivalent on the P/390 because the P/390 was an entry-level single-processor machine, not designed to do anything like an LPAR with PR/SM.

There is a driver (AWS3215.EXE) that supplies a 3215 emulation. However, it is not known to work for a native P390->L390 installation. The problem appears to be that the 3215 is not being detected/initialized properly during the boot sequence. There have been earlier kernel patches to address this problem, and a parameter has been added to the parameter file (condev=0x0009).

A fix has been placed on the developerworks Web site that addresses this problem.

Chapter 6. VM installation and operation of Linux for S/390

This chapter describes how to install and operate Linux for S/390 in a virtual machine running under the control of the VM/ESA operating system.

If you are intending to install Linux for S/390 to run either natively on a S/390 processor or in a logical partition (LPAR), then much of this chapter will not be relevant and you can skip reading this material.

Attention

The content of this chapter is based on the files stored on the Marist College Linux for S/390 download site after May 19, 2000. You can use the methods described here to install Linux for S/390 with the exception of the VM minidisk root file system (initmd.bin), which is now archived.

Note that the Linux for S/390 IUCV network driver does not work with these binaries. There is sometimes a problem with the DASD driver when you create a file system on a device partition. With larger minidisks that have been formatted and reserved by CMS, `mke2fs` tries to write past the end of the volume. The problem is under investigation.***

6.1 Linux for S/390 in a virtual machine (as a guest of VM)

An understanding of VM/ESA and the S/390 processor architecture is assumed in this chapter. If you have not used VM/ESA before, refer to Appendix B, “VM/ESA virtual machines” on page 469 for a brief introduction to S/390 virtual machine concepts and VM/ESA.

VM virtualizes S/390 hardware resources. You can install Linux for S/390 in a virtual machine using many of the techniques that you would use if installing on the basic S/390 hardware or in an LPAR. In practice it is easier to use some of the facilities provided by the Conversational Monitor System (CMS), a component of VM, to build the medium from which Linux for S/390 can be booted.

Before installing Linux for S/390 under VM/ESA, you should be running a supported level of VM/ESA with service applied where relevant.

As Linux for S/390 uses IEEE floating point arithmetic, we recommend to have the fixes for APARs VM62337 and VM62410 applied before installing

Linux for S/390 in a VM environment. IEEE Floating Point is available only on IBM 9672 Generation 5 and later processors.

If you are running Linux for S/390 as a guest of VM/ESA Version 2 Release 3, you will need APAR VM61762 as well. VM61762 is the initial IEEE FP support and has as prerequisites APARs VM62337 and VM62410. These are in the Version 2 Release 4 base.

For the VM/ESA TCP/IP product or component, APARs PQ34318 and PQ37002 are recommended.

6.2 Installing Linux for S/390

There are two different ways to install Linux on S/390, or indeed on any other processor architecture.

The more difficult method is to download the Linux source code from the Internet, apply the S/390 patches, and then build the Linux kernel and related binaries for execution on S/390. If you do not have a Linux for S/390 environment already installed and available to you, you would need to carry out these steps on some other Linux platform and then copy the resulting binaries over to your S/390 system. This method also requires a detailed technical understanding of Linux. Consequently, it is unsuitable for anyone without high-level Linux knowledge and skills.

We did not attempt this type of install.

The easier approach, adopted by most Linux users, is to install pre-built Linux for S/390 binaries, either packaged by a Linux for S/390 distributor or downloaded from an Internet site. As this is by far the most common approach, it is the one we chose to use and document.

We *strongly* recommend that you have a TCP/IP stack installed on your VM system if you are going to install Linux for S/390. If you do not have TCP/IP installed, you will not be able to use the File Transfer Program (FTP) to copy files from a PC to VM, in which case you will need to use an alternative method, such as the 3270 File Transfer Program, to upload the files.

Without a VM TCP/IP stack you have two options to connect Linux for S/390 to a network:

- Use a Linux for S/390 network driver, such as the Open Systems Adapter-2 (OSA-2) driver, to drive the physical network interface.

- Route through a TCP/IP stack running in another S/390 guest operating system, such as OS/390 or VSE/ESA.

The remainder of this chapter assumes that you have the VM/ESA TCP/IP Feature or the TCP/IP for VM program product (5735-FAL) installed on your system.

6.2.1 Installation steps overview

Installing Linux for S/390 binaries in a virtual machine can be broken down into the following steps:

1. Decide on the install method.
2. Prepare the virtual machine to run Linux for S/390.
3. Prepare the networking environment.
4. Obtain the binary files.
5. Copy the files to VM and reblock.
6. Create the initial kernel options file.
7. Boot the initial Linux for S/390 kernel.
8. Install the root file system tarball.
9. Complete the Linux for S/390 customization.

6.2.2 Decide on the install method

There are several ways to install Linux for S/390 in a VM environment. We tested most during this project.

Answering the following questions will help you decide on the most suitable approach for your installation.

Do you want to boot from the virtual reader or from tape?

We found it simpler to install Linux for S/390 by booting from the reader, rather than having to get a magnetic tape mounted. Your choice will depend on your particular operational environment and policies. Using tape makes the process almost identical to that followed when installing in a logical partition without VM. It is generally easier to use the reader if, as we found, you have to boot the downloaded kernel several times during installation.

Which file system do you want to install?

This is an installation-dependent decision. If you have enough disk space available (around 500 MB), our recommendation is to install the large file system. This delivers a much richer Linux for S/390 environment containing preinstalled packages such as the Apache Web server. It also contains the necessary tools to allow you to install other packages.

Which disk device drivers are you going to use?

When installing Linux for S/390 on a VM system, you may use either or both of two disk device drivers: the DASD driver or the VM minidisk driver. The DASD device driver is mandatory when you run Linux for S/390 in an LPAR or natively. It may also be used with VM dedicated DASD or minidisks.

The DASD device driver uses device names of the form `dasd<letter>`. Letters are allocated sequentially from “a” as the kernel registers devices to be managed by this driver.

Depending on how the kernel is compiled, either the Diagnose X'250' interface or S/390 Start Subchannel (SSCH) instruction can be specified to initiate disk I/O.

The VM minidisk device driver can only be used with CMS-format minidisks. These minidisks must be prepared for use by Linux for S/390 with the CMS `FORMAT` and `RESERVE` commands.

The VM minidisk device driver uses device names of the form `mnd<letter>`. Letters are allocated sequentially from “a” as the kernel registers devices to be managed by this driver.

The VM minidisk driver always uses Diagnose X'250' for I/O.

The DASD driver supports Extended Count Key Data (ECKD) and Fixed Block Architecture (FBA) devices. When Linux for S/390 runs in a virtual machine, it supports any DASD devices supported by VM/ESA.

Either the Diagnose X'250' interface or S/390 Start Subchannel (SSCH) instruction can potentially be used to drive disk I/O. If you want to boot Linux for S/390 from a disk device, that disk must use the DASD driver.

The DASD device driver can currently support a single partition on a device. The structure of the disk device is illustrated in Figure 39 on page 89.

DASD partitions are named `dasd<letter>1`.

The objective is to leave any boot sector, label or volume identifier (volid) untouched by the Linux for S/390 file system that you create on the partition.

Important

The kernel delivered with the Marist binaries is compiled with two kinds of support for minidisks: the old VM minidisk driver using Diagnose X'250' and the new DASD driver using SSCH. Diagnose X'250' support (`dasd_force_diag` option) is *not enabled* for the new DASD driver. The `dasd_force_diag` is ignored if you specify it. This maintains compatibility with previous versions of the kernel and disk device drivers. If you wish to use the new DASD driver with Diagnose X'250', you will need to recompile the kernel with the appropriate kernel option. Under S/390 block device drivers, select "Support for DIAG access to CMS reserved minidisk." This option is mutually exclusive with the "Support for VM minidisk" option.

If you enable Diagnose X'250' for the new DASD driver, you can no longer use the VM minidisk driver.

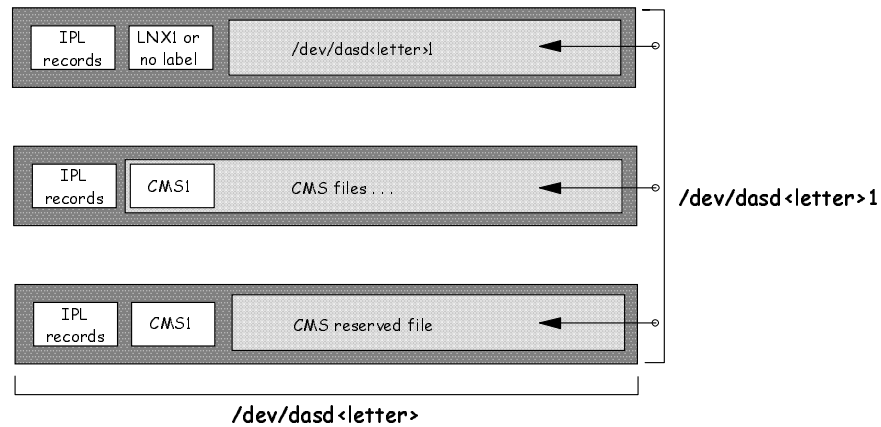


Figure 39. DASD partitioning scheme

We make the following recommendations:

- Try to use the DASD driver as far as possible for all dedicated Linux for S/390 disk devices.
- Use the Diagnose X'250' interface for minidisks as far as possible. This eliminates the need to deal with error recovery, which will be handled by the VM/ESA Control Program (CP).

- Prepare minidisks that use Diagnose X'250' for I/O with the CMS FORMAT and RESERVE commands, in order to create CMS label/volid information.
- Create a file system on a device partition wherever possible to preserve this information.

In our install we were unable to follow all these recommendations because we wanted to show the use of file system partitions. These can be created only with the DASD driver, and the options specified for S/390 block devices in the compiled kernel did not specify use of Diagnose X'250' with that driver.

For a discussion of the relationship between S/390 device numbers and Linux for S/390 device names, refer to 6.7, "Linux for S/390 device files and virtual device numbers" on page 122.

Installing on a P/390, R/390, Integrated Server or Multiprise 3000?

These machines allow you to use OS/2 or AIX files to emulate S/390 devices. The device managers in Table 6 can potentially be used to store the initial kernel and related boot files on OS/2 or AIX.

Table 6. Device managers and emulated S/390 devices

OS/2 or AIX device manager	Emulated S/390 device
AWSTAPE	magnetic tape
AWSOMA	magnetic tape
AWS2540	reader/punch

For further information on these device managers, refer to the relevant hardware product documentation.

6.2.3 Prepare the virtual machine to run Linux for S/390

Before beginning to install Linux for S/390 in a virtual machine, your VM system administrator must define a virtual machine suitable for running Linux for S/390.

Here is a sample virtual machine definition (CP directory entry) that is suitable for running Linux for S/390. It is the one we used to test the installation of Linux for S/390 in a VM environment.

```

USER LINUX5 XXXXXXXX 128M 256M G ❶
IPL CMS PARM AUTO CR ❷
MACHINE ESA 4 ❸
CONSOLE 0009 3215 ❹
SPOOL 000C 3505 A ❺

```

```

SPOOL 000D 3525 A ⑤
SPOOL 000E 1403 A ⑤
LINK MAINT 0190 0190 RR ⑥
LINK MAINT 019E 019E RR ⑥
LINK MAINT 019F 019F RR ⑥
LINK MAINT 019D 019D RR ⑥
MDISK 0201 3390 0001 1000 LINUX5 MR ⑦
MDISK 0202 3390 1001 1000 LINUX5 MR ⑦
MDISK 0203 3390 2001 200 LINUX5 MR ⑦
MDISK 0200 3390 3221 20 LINUX2 MR ⑦
MDISK 0191 3390 555 50 VMZU1A MR ⑧

```

Figure 40. Sample CP directory definition for Linux for a S/390 virtual machine

① The user ID that identifies this virtual machine is LINUX5. The virtual machine is defined with a default storage of 128 MB, but this can be redefined up to a maximum of 256 MB. A minimum of 64 MB is recommended.

The virtual machine has class G privilege, which is sufficient to run Linux for S/390. This is the same privilege class normally given to a CMS user.

② When you log on to the virtual machine, an IPL of the CMS operating system will occur automatically (IPL in S/390 terms is equivalent to a Linux for S/390 kernel boot). Once you have Linux for S/390 installed and ready to run in production, it is possible to change the IPL statement so that the Linux for S/390 IPL occurs directly from a disk device. However, by starting CMS first, you can invoke a profile that will tailor the virtual machine environment before the IPL of Linux for S/390. This is a more flexible approach.

③ This statement describes the processor architecture of the virtual machine. Specify ESA and not XC. The maximum number of processors that can be defined for the use of Linux for S/390 is four. The default is one.

④ The CONSOLE statement defines the operating console for the virtual machine. Although you will use a 3270 device or 3270 terminal emulation program to access this console, it will operate as a 3215 device. In other words, line-mode 3215 Channel Command Words (CCWs) are used for virtual I/O.

⑤ The next three statements define a reader, a punch, and a printer for the virtual machine.

⑥ These are read-only links to CMS minidisks owned by other virtual machines. They contain files that CMS users will need.

⑦ These statements define four minidisks on which Linux for S/390 file systems and data will be stored. The sizes of these disks were chosen based on their intended use. You can define as many minidisks for Linux for S/390 as you wish depending on your file system requirements.

These are read/write (R/W) minidisks.

⑧ The 191 minidisk will be accessed automatically by CMS when you log on. It will hold the CMS files that are needed to build a production Linux for S/390 environment in the virtual machine. The Linux for S/390 kernel does not use this minidisk.

6.2.3.1 Defining virtual devices

Another way to define virtual devices is with the CP DEFINE command. We used this command to define channel-to-channel (CTC) devices. For more information on the CP directory statements used to define a virtual machine, see *VM/ESA Planning and Administration, SC24-5750*.

Examples of some other CP directory statements that you might want to include in the definition of your Linux for S/390 virtual machine follow:

```
CPU 1 NODEDICATE ①  
SPECIAL 808 CTCA ②  
SPECIAL 809 CTCA ②  
IUCV ANY ③  
IUCV ALLOW ③
```

① Defining additional processors for a Linux for S/390 virtual machine allows you to test Linux for S/390 Symmetric Multiprocessor (SMP) support in a virtualized environment, even when your system has only one real processor installed. Do not define additional virtual processors unless you want to run the Linux for S/390 kernel in SMP mode.

The example defines one additional processor for the Linux for S/390 virtual machine. A virtual processor that is not assigned to a real processor can run on any available shared real processor.

If you have a S/390 system with multiple processors, you can dedicate one or more real CPUs to a Linux for S/390 virtual machine. For example, if you had an IBM 9672 R65 system running VM/ESA, you might dedicate two real CPUs to a Linux for S/390 virtual machine. To do this you would have CP directory control statements similar to these:

```
CPU 0 DEDICATE  
CPU 1 DEDICATE
```


Dedicating a processor to a virtual machine prevents its use by any other virtual machine.

❷ If you are intending to use a CTC connection to communicate from one Linux for S/390 virtual machine to another, or to the VM TCP/IP stack, you can define a pair of CTC devices as shown. Each CTC connection uses an even/odd pair of consecutive device numbers.

These special device definitions are not needed when you use the Inter-User Communications Vehicle (IUCV) driver to communicate with other Linux for S/390 virtual machines or the VM/ESA TCP/IP stack.

❸ These CP directory control statements allow the Linux for S/390 virtual machine to use IUCV communications. Refer to 6.6, “IUCV connections” on page 122 for more detail.

6.2.4 Prepare the networking environment

Careful consideration of the networking requirements for Linux virtual machines is needed since there are so many possibilities. These are discussed in detail in Chapter 15, “Linux for S/390 connectivity to VM, OS/390, VSE” on page 283.

If you are intending to use the VM TCP/IP stack as a gateway to your physical network, your network administrator will need to make suitable VM TCP/IP definitions for links to your Linux for S/390 virtual machine.

For information on defining links in the VM TCP/IP configuration file, refer to *VM/ESA V2R4.0: TCP/IP Function Level 320 Planning and Customization*, SC24-5847. Another useful reference is the redbook *TCP/IP Solutions for VM/ESA*, SG24-5459.

6.2.5 Typical connectivity configuration

A common way to connect multiple Linux for S/390 virtual machines to a physical network is to use virtual CTC or IUCV connections to the VM TCP/IP stack, which drives the physical network interface. All Linux for S/390 connections to the network are routed via this stack.

Figure 41 on page 94 illustrates this.

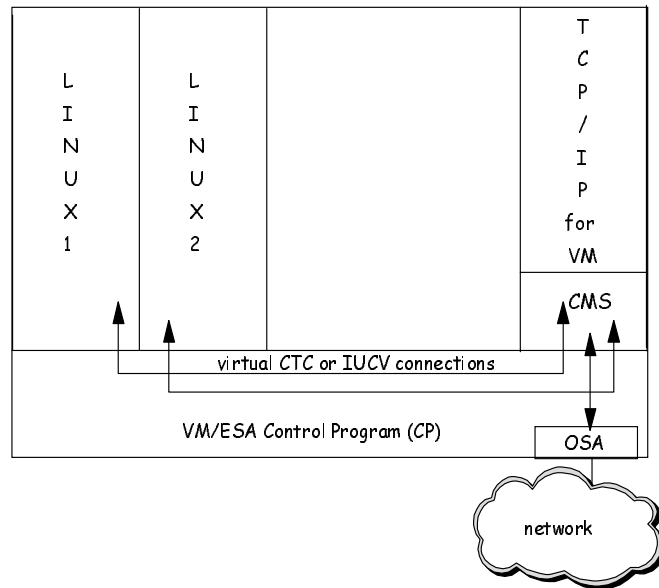


Figure 41. Networking configuration with Linux for S/390 running in a virtual machine

Advantages of this topology are:

- It reduces the number of physical network interfaces required.
- Both IUCV and virtual CTCs provide high bandwidth for TCP/IP traffic.

For TCP/IP hosts outside the VM system to communicate with the Linux for S/390 virtual machines, the local subnet router must have its routing tables updated. Any IP traffic destined for the IP addresses of the Linux for S/390 virtual machines is routed via the IP address of the VM TCP/IP stack's physical network interface.

Figure 83 on page 287 shows where you make TCP/IP definitions for a Linux for S/390 virtual machine and the TCP/IP stack. Linux for S/390 network definitions can also be entered using Linux for S/390 commands as described in 15.1, "Configuring the network" on page 283.

6.2.6 Obtain the binary files

For our project, we used binaries obtained from the Marist College Linux for S/390 download Web site. The date on which these binaries were released was 19 May, 2000. They contain a Linux for S/390 kernel at the 2.2.15 level.

Earlier levels of the binaries are archived at the same site.

Sometimes people experience problems transferring large binary files over the Internet. There can be a number of reasons for this, which are beyond the scope of this book to examine. Always try to verify that the number of bytes in the large files that you download matches the size of those files at the site from which they were transferred.

At the Marist College download site, the sizes of the root file system tarball files are displayed, so you can verify that the download has completed successfully.

Linux for S/390 binaries will typically be downloaded to a PC rather than directly to a VM user ID. Alternatively, using FTP or a 3270 text-based Web browser (Charlotte is a 3270 text-based browser that can be downloaded from the Web; there are also other commercially available 3270 text browsers), you can download the binaries from a Web site directly to your VM system without using an intermediate platform. If using FTP, connect to linux390.marist.edu and log in as user *anonymous*. The download files are stored in the /pub/download directory. Because of firewall restrictions, it may not always be possible to use FTP.

6.2.7 Copy files to VM and reblock

A Linux for S/390 file is just a stream of bytes with no implicit structure, whereas most S/390 files are record oriented. This implies that to boot the kernel on S/390, it must have a record structure imposed that is acceptable to the IPL device.

To boot the Linux for S/390 kernel from the card reader you need to reblock the kernel file to fixed 80-byte records. Similarly, to boot the Linux for S/390 kernel from a tape, you need to reblock the kernel file to fixed 1024-byte records.

The same rules apply to any other files that form part of your boot sequence, for instance a kernel parameter file or a RAM disk file.

Table 7 shows the files that need to be reblocked and the corresponding record length on VM.

Table 7. Record lengths for Linux for S/390 boot files

File description	Record length on VM (LRECL)
Linux for S/390 kernel for reader IPL	80
Linux for S/390 kernel for tape IPL	1024

File description	Record length on VM (LRECL)
Initial RAM disk for reader IPL	80
Initial RAM disk for tape IPL	1024

Do not reblock files that are not part of the initial installation steps. Specifically, do not reblock either the small or large root file system tarball files.

To reblock a bytestream file that you are transferring from a personal computer (PC) to VM, use FTP subcommands that create the target files on VM with the correct format.

Files can be transferred from a PC to VM running the FTP client either from VM or the PC. In practice, many PCs do not have an FTP server or daemon installed. In that case, files will need to be uploaded from the PC to VM by using the FTP client on the PC.

Table 8 summarizes the FTP subcommands to use when copying Linux for S/390 binary files from your PC to VM, reblocking during the file transfer. For the correct logical record length, see Table 7.

Table 8. FTP subcommands for reblocking files

Copying files to VM	FTP subcommands
FTP from PC to VM	bin quote site fix <i>lrecl</i> put
FTP from VM to PC	bin locsite fix <i>lrecl</i> get

Another way to reblock the kernel and associated files is to copy them to VM as binary files without reformatting and then reblock using a CMS tool. Use FTP to copy the files from PC to VM, but omit the `quote site` or `site` subcommands. A simple CMS Pipeline will perform the reblocking:

```
PIPE < fn ft fm | fblock lrecl 00 | > fn ft fm f lrecl
```

For example, to reblock the file LINUXVM BIN A to fixed length 80-byte records, enter:

```
PIPE < LINUXVM BIN A | fblock 80 00 | > LINUXVM BIN A f 80
```

If you downloaded the Linux for S/390 binaries without reformatting, use CMS Pipelines to reblock files to the correct record length and format.

6.2.8 Create the initial kernel parameter file

When the initial Linux for S/390 kernel is booted, it reads a set of values from a kernel parameter file. This file can be created on VM using the CMS XEDIT editor.

From the CMS command line prompt enter:

```
XEDIT fn ft
```

The name of the file is displayed along with the record format and record length. When booting from the reader, these should be F and 80, respectively. If not, enter either or both of the following XEDIT subcommands to rectify this:

```
recfm f
lrecl 80
trunc 80
```

Then enter:

```
serial off
add n
```

where n is the number of lines you need in your kernel parameter file. This creates an area on the screen where you can type in the initial Linux for S/390 boot parameter values. When you have finished entering the parameter line you can use the Enter or Tab key to return to the XEDIT command line. Enter:

```
file
```

to save and exit from the editor.

For information on the CMS XEDIT editor, consult *VM/ESA XEDIT User's Guide*, SC24-5779, or *VM/ESA XEDIT Command and Macro Reference*, SC24-5780.

Alternatively, you can create the kernel parameter file on a PC and copy it over to VM. Make sure that ASCII-to-EBCDIC codepage translation occurs. If using FTP, do not specify the `bin` subcommand. If using the 3270 File Transfer Program, do not request a binary transfer.

For details on the format and syntax of parameters in this file, refer to Appendix D, "The parameter file" on page 481.

6.2.9 Boot initial Linux for S/390 kernel

To boot (IPL) the initial Linux for S/390 kernel, a set of files is written to the IPL medium. Then the CP IPL command is issued with the boot device number as a parameter.

The CMS record format of these files depends on the type of IPL device, as explained in Table 7 on page 95.

6.2.9.1 Booting from the reader

If you are using a RAM disk for the initial root file system, enter the following sequence of commands from CMS:

```
CP PURGE RDR ALL ❶  
CP SPOOL PUNCH * RDR ❷  
PUNCH VM_reader_kernel_fileid (NOHEADER) ❸  
PUNCH parameter_fileid (NOHEADER) ❸  
PUNCH RAM_disk_fileid (NOHEADER) ❸  
CP IPL 00C CLEAR ❹
```

- ❶ This command purges the reader of any existing files, so first make sure nothing is stored there that you wish to keep.
- ❷ This command redirects the punch device output to the virtual machine's reader, since that is the IPL device.
- ❸ Write the three files that will be used by the kernel to the IPL device in succession. It is important to specify the NOHEADER option. The kernel boot will fail if you forget to do this.
- ❹ This command causes an IPL of the virtual machine from its reader and the Linux for S/390 kernel boot process begins. The CLEAR parameter clears the virtual machine's storage first before any input is taken from the reader.

To avoid having to reenter this set of commands, you can create a REXX program (analogous to a shell script in Linux for S/390) containing them. See 6.3.7, "Boot the kernel" on page 108 for an example.

6.2.9.2 Booting from tape

These instructions assume the use of a RAM disk for the initial root file system.

First request the system operator to mount a scratch tape on a real tape drive and attach the device to your virtual machine as device number 181.

From CMS enter the following sequence of commands or execute a REXX program that contains them:

```
CP REW 181
FILEDEF OUTMOVE TAP1 (REFM F BLOCK 1024 PERM
FILEDEF INMOVE VM_tape_kernel_fileid
MOVE
FILEDEF INMOVE parameter_fileid
MOVE
FILEDEF INMOVE RAM_disk_fileid
MOVE
CP REW 181
CP IPL 181 CLEAR
```

Ensure that the files you write to tape have been reblocked to fixed 1024-byte records.

6.2.10 Install the root file system

Once the kernel has booted and your network definitions are activated, you can use FTP to transfer the large or small root file system tarball onto a Linux for S/390 disk and unzip the contents. The tarballs are compressed archive file trees.

The Linux `tar` command is used to restore the file tree from the compressed archive file.

Once the root file system is installed, you can change your kernel parameter file to mount the disk on which your uncompressed file system resides as the root file system.

The next time you boot the kernel, you will be prompted again for network definitions, but thereafter they will be saved.

To make further changes to your network definitions, use Linux network commands such as `ifconfig` and `route`.

6.2.11 Complete the customization

You can create a boot sector on a disk managed by the DASD driver to allow Linux for S/390 to be booted from disk rather than the reader. Other activities you can perform include:

- Creating a swapfile
- Defining devices to be automounted

6.3 Installing Marist College binaries

This section documents the procedure we followed to install the Linux for S/390 binaries stored at the Marist College download site.

6.3.1 Install method

We found it most convenient to install the Marist binaries by booting from the reader and using RAM disk for the initial root file system. This is the method we describe in detail.

Earlier levels of the Marist binaries included a CMS file containing an initial root file system image that could be used instead of RAM disk. However, we did not find it particularly useful when the goal was to install the large root file system. The initial CMS minidisk root file system is not part of the current set of binaries.

The advantages and disadvantages of using the initial CMS minidisk image were as follows:

Advantages

- TCP/IP not essential to install.
- No unpacking of a tarball file is required.
- Initial network definitions can be saved.
- Can be used as the root file system for a future kernel install.

Disadvantages

- Linux for S/390 virtual machine needs a larger 191 minidisk.
- Redundant once large root file system installed.
- Minidisk must be formatted with a 512-byte block size.

6.3.2 Linux for S/390 virtual machine definitions

The virtual machine configuration we used is described in 6.2.3, “Prepare the virtual machine to run Linux for S/390” on page 90. Four minidisks were defined for use by Linux for S/390 as described in Table 9.

Table 9. Details of Linux for S/390 minidisks

Virtual device number	Size in 3390 cylinders	Number of 4K formatted blocks	Intended use	I/O method	Formatted by
0200	20	3,600	Boot disk	SSCH	Linux dasdfmt
0201	1000	180,000	big root file system	SSCH	Linux dasdfmt
0202	1000	180,000	empty spare volume	SSCH	Linux dasdfmt
0203	200	36,000	swap disk	SSCH	Linux dasdfmt

The minidisk sizes defined were sufficient for installing with RAM disk from reader and tape. It is not necessary to copy the large file system tarball onto the 191 minidisk, because the Linux FTP program copies it directly into the Linux for S/390 file system.

We defined a small minidisk to be used solely as boot device. This would allow the Linux for S/390 kernel to be booted from disk rather than the reader.

From CMS, we formatted and reserved all Linux for S/390 minidisks first. This was done for two reasons:

1. The Linux for S/390 `dasdfmt` command requires that a device must already have been formatted using another disk formatting utility. Failure to do this will generate kernel error messages.
2. If you enable Diagnose X'250' support for the DASD driver in the kernel, you prepare minidisks in the same way.

Minidisks formatted and reserved by CMS should require no other formatting before creating a file system. The only reason we used the Linux for S/390 `dasdfmt` command was to overcome the Linux bug discovered when creating a file system on a partition of a device formatted and reserved by CMS.

```
format 200 g (blksize 4096
DMSFOR603R FORMAT will erase all files on disk G(200). Do you wish to
continue?
Enter 1 (YES) or 0 (NO) .
1
DMSFOR605R Enter disk label:
lin200
DMSFOR733I Formatting disk G
```

```

DMSFOR732I 20 cylinders formatted on G(200)
Reserve lin200 mdisk g
DMSRSV603R RESERVE will erase all files on disk G(200). Do you wish to
continue?
Enter 1 (YES) or 0 (NO).
1
DMSRSV733I Reserving disk G

```

We repeated this sequence for the other three minidisks. You can use the same filemode letter (“G” in the example), but it is helpful to specify a disk label that readily identifies each minidisk volume.

The CMS FORMAT command performs a low level format of the disk device from the Linux for S/390 point of view. In that sense it is analogous to the Linux for S/390 `dasdfmt` command.

We also used a small CMS profile to customize our environment for Linux. The CMS PROFILE EXEC is executed after every IPL of CMS in the Linux for S/390 virtual machine.

```

/* PROFILE EXEC for Linux virtual machine */
'CP LINK TCPMAINT 592 592 RR' ❶
'ACCESS 592 U' ❶
'CP DEFINE CTC 808' ❷
'CP DEFINE CTC 809' ❷
'CP COUPLE 808 TCPIP 809' ❸
'CP COUPLE 809 TCPIP 808' ❸

```

❶ These two commands establish access to a minidisk owned by the TCP/IP service machine. It contains programs such as PING and FTP.

❷ These two commands define the CTC devices used by Linux for S/390 for TCP/IP communications. One device is for sending messages and the other for receiving.

You must couple the Linux for S/390 read channel to the VM TCP/IP write channel, and the Linux for S/390 write channel to the VM TCP/IP read channel. Whether you couple even to odd and odd to even, or even to even and odd to odd, depends on the value coded for adapter number in the relevant LINK statement in the TC/IP configuration file.

Table 10 shows the rule.

Table 10. Pairing device numbers in CP COUPLE commands

Device number pairing	CTC adapter number in TCP/IP LINK definition
even to odd odd to even	0
even to even odd to odd	1

③ These two commands establish the CTC link between the Linux for S/390 virtual machine and the TCP/IP service machine.

6.3.3 Networking definitions

We used the VM TCP/IP stack to route all Linux for S/390 TCP/IP communications, both to other Linux for S/390 virtual machines, and to the physical network.

We made our CTC definitions in the PROFILE EXEC. Alternatively, CTC definitions could be kept in the CP directory entry for the Linux for S/390 virtual machine by including the following control statements:

```
SPECIAL 808 CTCA  
SPECIAL 809 CTCA
```

To establish a connection to the other end of the link, we needed to issue two CP COUPLE commands. The COUPLE command connects a CTC device to a CTC device in the target virtual machine. The coupling statements were included in our PROFILE EXEC.

Figure 42 on page 104 shows how the network definitions for Linux for S/390 match the relevant definitions for the TCP/IP service machine.

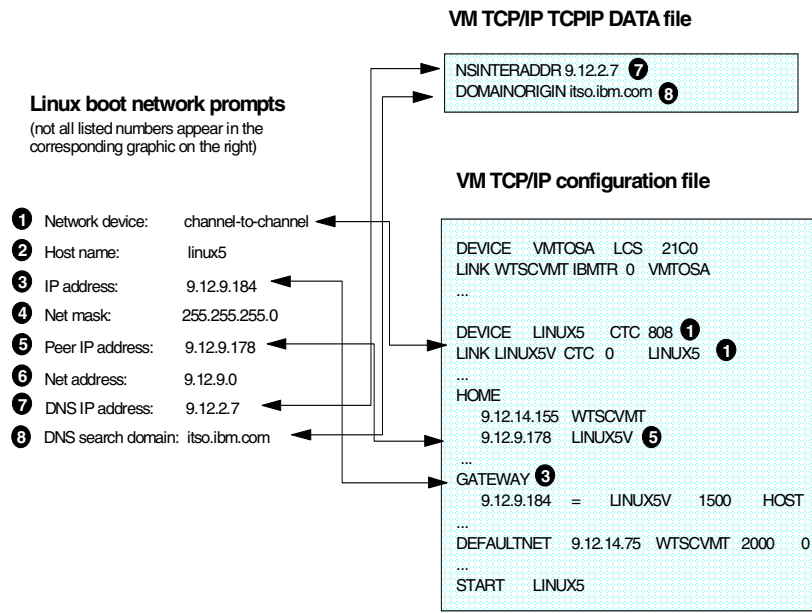


Figure 42. Matching Linux for S/390 and VM TCP/IP definitions

We did not make entries in the VM TCP/IP HOSTS LOCAL file for host name resolution, since we were using an external DNS server. If you did use the HOSTS LOCAL file, you would code an entry like this:

```
HOST : 192.12.9.184 : LINUX5 :::
```

Refer to *VM/ESA V2R4.0: TCP/IP Function Level 320 Planning and Customization*, SC24-5847, for further information on the syntax of statements in this file and how to generate the site table from it.

The site table enables name resolution and reverse name resolution without using a domain name server.

6.3.4 Downloading the binaries

The URL for the Marist College Linux for S/390 download site is

```
http://linux390.marist.edu/
```

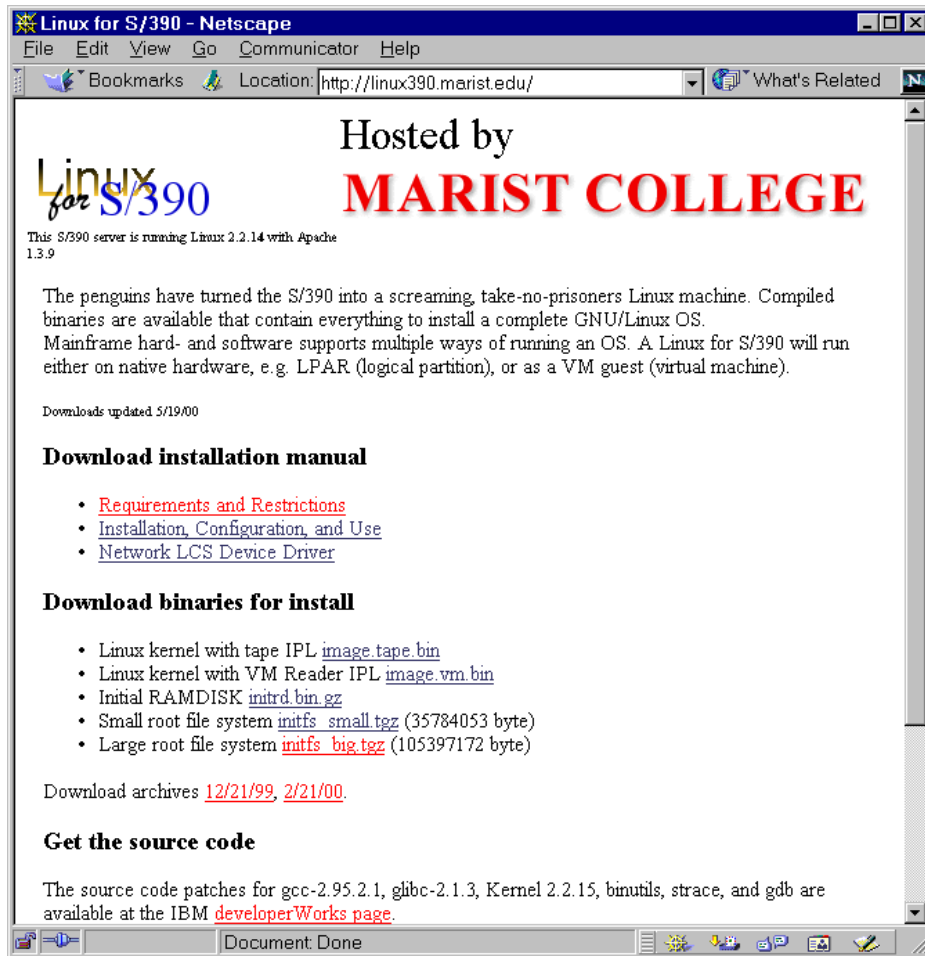


Figure 43. Marist College Linux for S/390 download Web site

Table 11 shows the binary files that can be found there.

Table 11. Description of files at Marist College Linux for S/390 download site

Description	File name
Linux for S/390 kernel with tape IPL	image.tape.bin
Linux for S/390 kernel with reader IPL	image.vm.bin
Initial RAM disk	initrd.bin.gz

Description	File name
Small root file system	initfs_small.tgz
Large root file system	initfs_big.tgz

You do not need to download all the binary files. The ones you choose depend on the method you intend to follow for installing Linux for S/390 on your VM system. Table 12 summarizes the required files.

Table 12. Files required by the install method

Download site file	Reader boot	Tape boot
image.tape.bin		✓
image.vm.bin	✓	
initrd.bin.gz	✓	✓
initfs_small.tgz	choose one	choose one
initfs_big.tgz		

Use your browser to download the files required to install Linux for S/390 on VM according to the install method you have chosen.

From a browser on our PC we downloaded all the files except the small root file system tarball (initfs_small.tgz).

This enabled us to test all the install methods available on VM.

We also downloaded copies of two manuals in Adobe Acrobat PDF format :

- Installation, Configuration, and Use
- Network LCS Device Driver

These manuals contain valuable information and guidance to help you install and customize Linux for S/390. As with the contents of this ITSO redbook, some information may become outdated in the future.

6.3.5 Copying Marist files to VM and reblocking

From our PC we started an FTP session to VM TCP/IP. Make sure that the target Linux for S/390 virtual machine is not logged on when you do this or the VM FTP server will be unable to obtain write access to the 191 minidisk of the Linux for S/390 virtual machine. (If the target Linux machine was logged

on, you could temporarily detach the 191 minidisk and relink it when you have finished the file transfer.)

```
ftp 9.12.14.155
Connected to 9.12.14.155.
220-FTPSERVE IBM VM Level at your_VM_host , 15:20:16 EDT THURSDAY
05/11/00
User (9.12.14.155: (none)): linux5
331 Send password please.
Password:
230 LINUX5 logged in; working directory = LINUX5 191
ftp> bin
200 Representation type is IMAGE.
ftp> lcd \temp\marist
Local directory now C:\temp\marist
ftp> quote site fix 80
200 Site command was accepted.
ftp> put image.vm.bin kernel.marist
200 Port request OK.
150 Storing file 'reader.marist'
250 Transfer completed successfully.
1453976 bytes sent in 7.11 seconds (204.50 Kbytes/sec)
ftp> put initrd.bin.gz initrd.marist
200 Port request OK.
150 Storing file 'initrd.marist'
250 Transfer completed successfully.
3022044 bytes sent in 16.17 seconds (186.86 Kbytes/sec)
ftp> quit
```

This file transfer session creates two files on the Linux for S/390 virtual machine 191 minidisk: KERNEL MARIST and INITRD MARIST.

These files were sufficient to boot the Linux for S/390 kernel from the reader of the Linux for S/390 virtual machine.

6.3.6 Creating the kernel parameter file

Next we logged on to the Linux for S/390 virtual machine and created an initial kernel parameter file called PARM MARIST to be read by the kernel during its boot sequence.

We had four minidisks defined for use as Linux for S/390 disk devices, with device numbers 0200-0203. Thus the kernel parameter line we entered was:

```
dasd=200-203 root=/dev/ram0 ro
```

`/dev/ram0` refers to the RAM disk that will be used as the initial root file system to be mounted by the kernel. The `ro` parameter indicates that it is mounted first as a read-only disk.

We were now ready to boot the Linux for S/390 kernel.

6.3.7 Boot the kernel

Although this task can be done manually, we found it productive to create a small REXX program (script) that carries out the steps automatically.

Our REXX program was called MARIST EXEC:

```
/* REXX EXEC to boot Linux for S/390 from VM reader */
/* using RAM disk as the initial root file system */
'CP CLOSE RDR'
'CP PURGE RDR CLASS L'
'CP SPOOL PUN * RDR CLASS L'
'PUNCH KERNEL MARIST A (NOHEADER' /* Kernel image */
'PUNCH PARM MARIST A (NOHEADER' /* Parameter file */
'PUNCH INITRD MARIST A (NOHEADER' /* RAM disk root file system */
'CP SPOOL PUN * RDR CLASS A'
'CP SPOOL RDR KEEP CLASS L'
'CP IPL 00C CLEAR'
```

This was the output on the console after executing the IPL command:

```
Linux version 2.2.15 (root@linux390.marist.edu) (gcc version 2.95.2
19991024 (release)) #5 SMP Fri May 19 07:49:25 EDT 2000
Command line is:dasd=200-203 dasd_force_diag 201-203 root=/dev/ram0 ro ❶
We are running under VM ❷
This machine has no IEEE fpu ❸
Initial ramdisk at: 0x02000000 (3012560 bytes)
Detected device 0009 on subchannel 0000 - PIM = 80, PAM = 80, POM = FF ❹
Detected device 000C on subchannel 0001 - PIM = 80, PAM = 80, POM = FF
Detected device 000D on subchannel 0002 - PIM = 80, PAM = 80, POM = FF
Detected device 000E on subchannel 0003 - PIM = 80, PAM = 80, POM = FF
...
Detected device 0201 on subchannel 0008 - PIM = F0, PAM = F0, POM = FF
Detected device 0202 on subchannel 0009 - PIM = F0, PAM = F0, POM = FF
Detected device 0203 on subchannel 000A - PIM = F0, PAM = F0, POM = FF
Detected device 0191 on subchannel 000B - PIM = F0, PAM = F0, POM = FF
Detected device 0200 on subchannel 000C - PIM = F0, PAM = F0, POM = FF
Detected device 0592 on subchannel 000D - PIM = F0, PAM = F0, POM = FF
Detected device 0808 on subchannel 000E - PIM = 80, PAM = 80, POM = FF
Detected device 0809 on subchannel 000F - PIM = 80, PAM = 80, POM = FF
Highest subchannel number detected: 16
SenseID : device 0009 reports: Dev Type/Mod = 3215/00 ❺
```



```
SenseID : device 000C reports: Dev Type/Mod = 3505/00
SenseID : device 000D reports: Dev Type/Mod = 3525/00
SenseID : device 000E reports: Dev Type/Mod = 1403/00
SenseID : device 0190 reports: CU Type/Mod=3990/E9, Dev Type/Mod=3390/0A
SenseID : device 019E reports: CU Type/Mod=3990/E9, Dev Type/Mod=3390/0A
SenseID : device 019F reports: CU Type/Mod=3990/E9, Dev Type/Mod=3390/0A
SenseID : device 019D reports: CU Type/Mod=3990/E9, Dev Type/Mod=3390/0A
SenseID : device 0201 reports: CU Type/Mod=3990/E9, Dev Type/Mod=3390/0A
SenseID : device 0202 reports: CU Type/Mod=3990/E9, Dev Type/Mod=3390/0A
SenseID : device 0203 reports: CU Type/Mod=3990/E9, Dev Type/Mod=3390/0A
SenseID : device 0191 reports: CU Type/Mod=3990/E9, Dev Type/Mod=3390/0A
SenseID : device 0200 reports: CU Type/Mod=3990/E9, Dev Type/Mod=3390/0A
SenseID : device 0592 reports: CU Type/Mod=3990/E9, Dev Type/Mod=3390/0A
SenseID : device 0808 reports: Dev Type/Mod = 3088/08
SenseID : device 0809 reports: Dev Type/Mod = 3088/08
dasd(Setup): added dasd range from 200 to 203.
```

- ❶ These are the boot parameters specified in the parameter file prepared in 6.3.6, “Creating the kernel parameter file” on page 107.
- ❷ The Linux for S/390 kernel detects that it is executing under the control of VM rather than natively or in a logical partition.
- ❸ If Linux for S/390 is running in a virtual machine, two conditions must be met to exploit the IEEE floating point instruction hardware:
 - The feature must be installed on the processor. Generation 5 or later models of the IBM 9672 processor family have this feature. So does the Multiprise 3000 processor family.
 - VM support for the hardware feature must also be installed. This was introduced by APAR VM61762 in VM/ESA Version 2 Release 3. If you intend to run Linux for S/390 in SMP mode with more than one virtual processor defined, you should also install APAR VM62410.

APAR VM61762 was incorporated into the 9903 Refresh Service Upgrade (RSU) for VM/ESA Version 2 Release 3. An easy way to check whether your VM/ESA system is at or beyond this level of service is to issue the CP command QUERY CPLEVEL.

The VM/ESA service tool provides a more general way to discover whether the fix for a particular APAR is applied to your system. More information can be found in *VM/ESA V2R4.0 VMSES/E Introduction and Reference*, GC24-5837.

- ❹ This message and the following group show Linux for S/390 sensing the devices currently defined to the Linux for S/390 virtual machine.

Refer to 6.7, “Linux for S/390 device files and virtual device numbers” on page 122 for a discussion of the relationship between Linux for S/390 device files and virtual device numbers.

⑤ Here Linux for S/390 determines the types of devices it has detected.

The next group of console messages shows the Linux for S/390 kernel creating and verifying its operational environment

```
Memory: 124504k/131072k available (1092k kernel code, 4k reserved, 2528k
data, 0k init) ❶
Dentry hash table entries: 16384 (order 5, 128k)
Buffer cache hash table entries: 131072 (order 7, 512k)
Page cache hash table entries: 32768 (order 5, 128k)
POSIX conformance testing by UNIFIX
Detected 1 CPU's ❷
Boot cpu address 0
cpu 0 phys_idx=0 vers=FF ident=0D0822 machine=9672 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
TCP: Hash tables configured (ehash 131072 bhash 65536)
Starting kswapd v 1.5
pty: 256 Unix98 ptys configured
RAM disk driver initialized: 16 RAM disks of 8192K size
loop: registered device at major 7
dasd:initializing...
16 areas reserved for debugging information
reserved 4 areas of 2 pages for debugging dasd
dasd(eckd):3390/a (3990/1) Cyl: 1000 Head: 15 Sec: 224 ❸
dasd(eckd):Estimate: 58786 Byte/trk 2074 byte/kByte 33 kByte/trk
dasd:0201 trying to access, irq 8, index 1
dasd:0201 is (dasdb) minor 4 (ECKD)
dasd(eckd):Verified: 58786 B/trk 5202 B/Blk(4096 B)12 Blks/trk 48 kB/trk
dasd:0201 (dasdb):720000 kB <- block: 4096 on sector 4096 B
dasd(eckd):3390/a (3990/1) Cyl: 1000 Head: 15 Sec: 224
dasd(eckd):Estimate: 58786 Byte/trk 2074 byte/kByte 33 kByte/trk
dasd:0202 trying to access, irq 9, index 2
dasd:0202 is (dasdc) minor 8 (ECKD)
dasd(eckd):Verified: 58786 B/trk 5202 B/Blk(4096 B)12 Blks/trk 48 kB/trk
dasd:0202 (dasdc):720000 kB <- block: 4096 on sector 4096 B
dasd(eckd):3390/a (3990/1) Cyl: 200 Head: 15 Sec: 224
dasd(eckd):Estimate: 58786 Byte/trk 2074 byte/kByte 33 kByte/trk
dasd:0203 trying to access, irq a, index 3
dasd:0203 is (dasdd) minor 12 (ECKD)
```

```

dasd(eckd):3390/a (3990/1) Cyl: 20 Head: 15 Sec: 224
dasd(eckd):Estimate: 58786 Byte/trk 2074 byte/kByte 33 kByte/trk
dasd:0200 trying to access, irq c, index 0
dasd:0200 is (dasda) minor 0 (ECKD)
dasd(eckd):Verified: 58786 B/trk 5202 B/Blk(4096 B)12 Blks/trk 48 kB/trk
dasd:0203 (dasdd):144000 kB <- block: 4096 on sector 4096 B
dasd(eckd):Verified: 58786 B/trk 5202 B/Blk(4096 B)12 Blks/trk 48 kB/trk
dasd:0200 (dasda):14400 kB <- block: 4096 on sector 4096 B
Partition check:
dasda:(CMS1)/LIN200: dasda1 (MDSK) ④
dasdb:(CMS1)/LIN201: dasdb1 (MDSK)
dasdc:(CMS1)/LIN202: dasdc1 (MDSK)
dasdd:(CMS1)/LIN203: dasdd1 (MDSK)
channel: 2 Parallel channel found - 0 ESCON channel found ⑤
ctc0: read dev: 0808 irq: 000e - write dev: 0809 irq: 000f
RAMDISK: Compressed image found at block 0
VFS: Mounted root (ext2 filesystem).
Freeing unused kernel memory: 0k freed
modprobe: can't locate module char-major-4
"INIT: version 2.74 booting"
!!!Welcome to Red Hat Linux
Mounting proc filesystem [ OK ]
/etc/rc.d/rc.sysinit: /proc/sys/kernel/sysrq: No such file or directory
modprobe: can't locate module char-major-4
hwclock: Can't open /dev/tty1, errno=19: No such device.
Cannot access the Hardware Clock via any known method. Use --debug
option to see the details of our search for an access method.
Setting clock : Mon May 29 15:35:15 EDT 2000 [ OK ]
/etc/rc.d/rc.sysinit: /etc/sysconfig/keyboard: No such file or directory
Activating swap partitions [ OK ]
Setting hostname [ OK ]
Checking root filesystem
/dev/ram0 was not cleanly unmounted, check forced.
ext2fs_check_if_mount: No such file or directory while determining
whether /dev ram0 is mounted.
/dev/ram0: 1066/2048 files (0.5% non-contiguous), 8005/8192 blocks
[PASSED]
Remounting root filesystem in read-write mode [ OK ]
Finding module dependencies [ OK ]
Checking filesystems
[ OK ]
Mounting local filesystems [ OK ]
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

```

- ❶ The Linux for S/390 kernel detects that it has 128 MB of memory available. From the point of view of the Linux for S/390 kernel, this is real memory. Its value equates to size of the memory defined for the Linux for S/390 virtual machine.
- ❷ As only one processor was defined for the Linux for S/390 virtual machine, the Linux for S/390 kernel will run in uniprocessor mode.
- ❸ The Linux for S/390 kernel registers the disks managed by the DASD device driver.
- ❹ The kernel detects that device numbers 200-203 have been formatted using CMS and displays the volume label for each.
- ❺ The pair of CTC devices we defined are registered by the kernel.

Now the Linux for S/390 kernel executes the netsetup script to prompt for initial network definitions:

```
Is your machine connected to a network (Yes/No) ?
yes
Select the type of your network device
1) for osa token ring
2) for osa ethernet
3) for channel to channel
Enter your choice (1-3):
3
Please enter your host name:
linux5
Please enter your IP address:
9.12.9.184
Please enter the net mask:
255.255.255.0
Please enter the IP address of your peer:
9.12.9.178
Please enter the net address:
9.12.9.0
Please enter the IP address of the DNS server:
9.12.2.7
Please enter the DNS search domain:
itso.ibm.com

Configuration will be:
Host name      : linux5 ❶
IP address     : 9.12.9.184 ❷
```

```

Net mask      : 255.255.255.0 ③
Peer IP address : 9.12.9.178 ④
Net address   : 9.12.9.0 ⑤
DNS IP address : 9.12.2.7 ⑥
DNS search domain: itso.ibm.com ⑦
Is this correct (Yes/No) ?
yes

Bringing up interface lo
Bringing up interface ctc0
Starting portmapper: portmap
Initializing random number generator
Starting INET services: inetd
Starting local
Give root password for maintenance
(or type Control-D for normal startup):
pass4root
[root@linux5 /root]#

```

It is useful to refer to Figure 42 on page 104 to see how Linux for S/390 and TCP/IP network definitions were matched.

- ① This is the host name of the Linux for S/390 virtual machine in the itso.ibm.com domain.
- ② This is the IP address of the Linux for S/390 virtual machine. In the VM TCP/IP configuration file it is specified in the GATEWAY statement. (Static IP routing is used in all the examples.)
- ③ Because our IP address is on the 9.12.9.0 subnet, we defined a subnet mask of 255.255.255.0. With a point-to-point connection there is normally no requirement for a subnet mask.
- ④ This the IP address in the VM TCP/IP stack associated with the CTC link to the Linux for S/390 virtual machine. In the VM TCP/IP configuration file, this link and the associated IP address are defined through the following entries:

```

HOME
9.12.9.178 LINUX5V

```

LINUX5V is the name of the CTC *link* over which connection is made to the Linux for S/390 virtual machine.

This link name is defined and associated with a network device through these two statements:

```
DEVICE LINUX5 CTC 808
LINK LINUX5V CTC 0 LINUX5
```

LINUX5 is the network *device name* for device number 808, the even device number of the CTC device pair 808/809 defined for the TCPIP service machine. These are the device numbers to which the Linux for S/390 virtual machine CTC device numbers are coupled.

Also required is this entry in the GATEWAY statement to tell the VM TCP/IP stack on which link it can reach the Linux for S/390 virtual machine:

```
9.12.9.184 = LINUX5V 1500 HOST
```

1500 is the maximum transmission unit (MTU) or packet size value defined for this link. Much higher values are often used for CTC links. The MTU value you use should be determined with your network administrator.

If any bridge or router does not perform IP-layer fragmentation of packets, you must select an MTU corresponding to the smallest MTU in use by that bridge or router. Selecting an MTU size that is too large may cause client applications to hang.

For MTU values recommended by IBM, refer to *VM/ESA V2R4.0: TCP/IP Function Level 320 Planning and Customization*, SC24-5847.

⑤ This is the subnetwork in which the Linux for S/390 virtual machine IP address is defined. The subnet mask must be consistent with the subnetwork address.

Subnet masks and subnet values are entered in the GATEWAY statement in the VM TCP/IP configuration file. Because this link is point-to-point, the parameter HOST is substituted.

⑥ The Domain Name Server (DNS) is often an external machine. In our environment the DNS machine was on a different subnet. If routing all Linux for S/390 IP traffic through the VM TCP/IP stack, ensure that the TCP/IP service machine can communicate with the DNS server.

In the VM TCP/IP service machine, the IP address of the DNS server is specified by the NSINTERADDR statement in the TCPIP DATA file.

⑦ The DNS search domain is also defined in the TCPIP DATA file.

The search order used to resolve domain names can be tailored to your needs through coding other statements in the TCPIP DATA file such as DOMAINLOOKUP, DOMAINORIGIN, DOMAINSEARCH and HOSTNAME.

The networking prompts will differ if you are configuring an OSA-2 network link. IUCV connections cannot be defined using the `netsetup` prompts.

6.3.8 Install the root file system

Once the root user was logged on to the Linux for S/390 console, we installed the large root file system previously downloaded onto a PC.

First we ran the Linux for S/390 `dasdfmt` command against device number 201 and created a file system structure on the partition for this device. Take care to distinguish between the device and the partition defined on it.

```
[root@linux5 /root]# dasdfmt -f /dev/dasdb -b 4096
I am going to format the device /dev/dasdb in the following way:
  Device number of device : 0x201
  Major number of device  : 94
  Minor number of device  : 4
  Start track              : 0
  End track                : last track of disk
  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.
[root@linux5 /root]# mke2fs /dev/dasdb1 -b 4096
mke2fs 1.15, 18-Jul-1999 for EXT2 FS 0.5b, 95/08/09
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
90048 inodes, 179997 blocks
8999 blocks (5.00%) reserved for the super user
First data block=0
6 block groups
32768 blocks per group, 32768 fragments per group
15008 inodes per group
Superblock backups stored on blocks:
32768, 98304, 163840
Writing inode tables: 0/6  1/6  2/6  3/6  4/6  5/6  done
Writing superblocks and filesystem accounting information:
done
[root@linux5 /root]# mkdir /mnt/dasdb
[root@linux5 /root]# mount /dev/dasdb1 /mnt/dasdb
```

Then we copied the root file system from the PC into our newly formatted disk.

From the PC:

```
C:\>ftp 9.12.9.184
Connected to 9.12.9.184.
220 linux5 FTP server (Version wu-2.4.2-VR17(1) Tue Nov 30 13:54:53 CET
1999) ready.
User (9.12.9.184:(none)): root
331 Password required for root.
Password:
230 User root logged in.
ftp> lcd \temp\marist
Local directory now C:\temp\marist
ftp> bin
200 Type set to I.
ftp> cd /mnt/dasdb
250 CWD command successful.
ftp> put initfs_big.tgz
200 PORT command successful.
150 Opening BINARY mode data connection for initfs_big.tgz.
226 Transfer complete.
102179165 bytes sent in 221.76 seconds (460.77 Kbytes/sec)
ftp> quit
```

Finally, we uncompressed the tar archive:

```
[root@linux5 /root]# cd /mnt/dasdb
[root@linux5 dasdb]# tar xzpBf initfs_big.tgz
```

6.3.9 Complete customization

If you would like to boot from disk rather than the reader, you can create a boot sector record on a disk managed by the DASD driver. We defined a separate small minidisk (device number 200) for this purpose. You must create a partition on the disk to leave room for the boot sector to be written at the front of the device.

6.3.9.1 Creating a boot sector

First, we did a low-level format:

```
[root@linux5 /]# dasdfmt -f /dev/dasda -b 4096
```

I am going to format the device /dev/dasda in the following way:

```
Device number of device : 0x200
Major number of device   : 94
Minor number of device   : 0
```



```
Start track          : 0
End track            : last track of disk
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.
root@linux5 /]#
```

Then we created the partition `dasda1` on the `dasda` device. In the next steps take care to distinguish between the device and the partition defined on it:

```
[root@linux5 /]# mke2fs /dev/dasda1 -b 4096
mke2fs 1.15, 18-Jul-1999 for EXT2 FS 0.5b, 95/08/09
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
3616 inodes, 3597 blocks
179 blocks (4.98%) reserved for the super user
First data block=0
1 block group
32768 blocks per group, 32768 fragments per group
3616 inodes per group

Writing inode tables: 0/1 done
Writing superblocks and filesystem accounting information: done
[root@linux5 /]#
```

We created a boot directory in the partition created on the boot device. Then we copied the boot files from the boot directory on device `dasdb` into it:

```
[root@linux5 /]# mkdir /mnt/dasda
[root@linux5 /]# mount /dev/dasda1 /mnt/dasda
[root@linux5 /]# cd /mnt/dasda
[root@linux5 dasda]# mkdir boot
[root@linux5 dasda]# mount
/dev/ram0 on / type ext2 (rw,errors=remount-ro)
none on /proc type proc (rw)
/dev/dasdb on /mnt/dasdb type ext2 (rw)
/dev/dasda1 on /mnt/dasda type ext2 (rw)
[root@linux5 dasda]# cd /mnt/dasdb/boot
[root@linux5 boot]# cp * /mnt/dasda/boot
```

```
[root@linux5 boot]# cd /mnt/dasda/boot
```

Then we copied the kernel from the VM 191 minidisk to ensure we were using the same version we had booted from the reader:

```
[root@linux5 boot]# ftp 9.12.14.155
Connected to 9.12.14.155.
220-FTPSERVE IBM VM Level at WTSCVMT.ITSO.IBM.COM, 18:45:08 EDT THURSDAY
05/18/00
220 Connection will close if idle for more than 5 minutes.
Name (9.12.14.155:root):
linux5
331 Send password please.
Password:
#####
230-LINUX5 logged in; working directory = LINUX5 191 (ReadOnly)
230 write access currently unavailable due to other links
Remote system type is VM.
ftp> bin
200 Representation type is IMAGE.
ftp> get kernel.marist image
local: image remote: marist.image
200 Port request OK.
150 Sending file 'kernel.marist'
250 Transfer completed successfully.
1447840 bytes received in 0.148 secs (9.6e+03 Kbytes/sec)
ftp> quit
```

Next we created a kernel parameter file in the boot directory on the dasda1 partition:

```
[root@linux5 boot]# ed image.parm
image.parm: No such file or directory
.a
dasd=200-203 root=/dev/dasdb1 ro noinitrd
.
wq
65
[root@linux5 boot]#
```

Next we built the boot sector record using the `silos` command:

```
[root@linux5 boot]# silos -f image -d /dev/dasda -p image.parm -b
./iplacekd.boot -t2
o->image set to image
o->ipldevice set to /dev/dasda
o->parmfile set to image.parm
```

```

o->bootsect set to ./ipleckd.boot
Testonly flag is now 0
Testlevel is set to 0
IPL device is: '/dev/dasda'
bootsector is: './ipleckd.boot'...ok...
bootmap is set to: './boot.map'...ok...
Kernel image is: 'image'...ok...
original parameterfile is: 'image.parm'...ok...final parameterfile is:
'image.parm'...ok...
ix 0: offset: 0000a4 count: 0c address: 0x00000000
ix 1: offset: 0000b1 count: 80 address: 0x0000c000
ix 2: offset: 000131 count: 80 address: 0x0008c000
ix 3: offset: 0001b1 count: 63 address: 0x0010c000
ix 4: offset: 000218 count: 01 address: 0x00008000
Bootmap is in block no: 0x00000219
[root@linux5 boot]#

```

Finally, we updated the file /etc/fstab to reflect the root file system device, dasdb1:

```

[root@linux5 /etc]# cd /mnt/dasdb/etc
[root@linux5 etc]# cat fstab
/dev/dasda1 / ext2 defaults,errors=remount-ro 0 1
none /proc proc defaults 0 0
[root@linux5 etc]# sed 's/dasda1/dasdb1/g' fstab > fstab2
[root@linux5 etc]# cat fstab2
/dev/dasdb1 / ext2 defaults,errors=remount-ro 0 1
none /proc proc defaults 0 0
[root@linux5 etc]# mv fstab2 fstab

```

After shutting down Linux for S/390, we were able to boot directly from the 200 minidisk with the CP command IPL 200.

6.3.9.2 Creating a swap file

We had defined a 200-cylinder minidisk (203) to be used as a swap file.

To create a swap file on the disk and activate it, we entered :

```

[root@linux5 /root]# dasdfmt -f /dev/dasdd -b 4096
I am going to format the device /dev/dasdd in the following way:
Device number of device : 0x203
Major number of device : 94
Minor number of device : 12
Start track : 0
End track : last track of disk
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).
dasd:called format ioctl
Finished formatting the device.
  dasdd:(nonl)/      : dasdd1
Rereading the partition table... done
[root@linux5 /root]# mkswap /dev/dasdd1
Setting up swapspace version 1, size = 147439616 bytes
[root@linux5 /root]# chmod 600 /dev/dasdd1
[root@linux5 /root]# swapon /dev/dasdd1

```

An entry in the `/etc/fstab` file will activate the swap space on a subsequent boot of the Linux for S/390 kernel. See 9.2.2, “The file system table `/etc/fstab`” on page 187 for how to do this.

6.3.9.3 Setting the time

The S/390 processor Time of Day (TOD) clock is set to Coordinated Universal Time (UTC). When VM/ESA is IPLed and the CP prompts for the date and time, it uses `TIMEZONE_DEFINITION` statements in `SYSTEM CONFIG` to work out the value to store in the real TOD clock.

By default, a guest virtual machine uses the real system TOD clock. Clock requests from a virtual machine are intercepted and their effect is limited to that virtual machine. The real system TOD clock is never changed by a guest virtual machine. When a `SET CLOCK (SCK)` instruction is issued in a virtual machine, an offset from the real system clock is computed and stored in an offset of the primary control block for that virtual machine. Specify `OPTION TODENABLE` in the directory entry for guests that need to use the `SCK` instruction to set the virtual machine's TOD clock. The guest virtual machine can also use its own facilities to establish the correct time zone. `OPTION TODENABLE` is not necessary to specify a time zone offset in a guest virtual machine.

We did not investigate whether Linux for S/390 tries to issue an `SCK` instruction. The Linux for S/390 virtual machine would need the `TODENABLE` option in its CP directory entry if it does.

Our experience was that Linux for S/390 would show the correct time by using the standard Linux for S/390 facility to specify the local time zone.

For example, this command sets the time to the US Pacific time zone:

```
mv /etc/localtime /etc/localtime.old
```

```
ln -s /usr/share/zoneinfo/PST8PDT /etc/localtime
```

The VM CP command QUERY OFFSET displays the difference between the system's current time zone and UTC.

6.3.9.4 Taking a backup

You should ensure that you always have the means to reboot the kernel in case you lose a disk device, or some other catastrophe occurs. VM gives you many options to safeguard your kernel.

Perhaps the simplest protection is to keep a copy of the kernel file, RAM disk and current kernel parameter file on a CMS minidisk (and tape), so that the kernel can be booted from the reader if necessary.

6.3.9.5 Other customization tasks

Now that you have a working Linux for S/390 environment, the remaining customization that you may wish to carry out is identical to when Linux for S/390 is running natively or in an LPAR. Refer to 5.7.5, "Customizing Linux for S/390 configuration files" on page 75 for information on how to tailor several important files in your root file system.

6.4 Logging into your Linux for S/390 system

Once you have established network connectivity for your Linux for S/390 virtual machine, you can use telnet from a workstation to log in. This means that the virtual machine console is not required and can be disconnected.

From the Linux for S/390 console prompt, enter the CP DISCONNECT command

```
#CP DISCONNECT
```

To log in to Linux for S/390 from your workstation, enter:

```
telnet linux5
```

From a VM CMS session you can also use the VM TCP/IP stack to telnet to Linux for S/390 .

6.5 3215 driver considerations

When using the Linux for S/390 virtual machine console to enter Linux for S/390 commands, there is no Ctrl key defined because it is operating as a 3215 device. This makes it impossible to enter control characters directly. The

character “^” in combination with certain other characters can emulate the Ctrl key:

- ^c is interpreted as Ctrl+C.
- ^d is interpreted as Ctrl+D.
- ^z is interpreted as Ctrl+Z.
- ^n is used at the end of the input line (on the terminal) to prevent the generation of a new line character.

If the special characters don't seem to work, make sure that you are using a suitable codepage on your terminal emulator. One that works is codepage 037 United States.

6.6 IUCV connections

The `netsetup` script packaged on the Marist binary distribution does not prompt for IUCV connections.

If you decide to use IUCV instead of CTC connections from Linux for S/390 to the VM TCP/IP stack, you will need to take the following actions:

- Make sure that the Linux for S/390 and VM TCP/IP service machines are suitably authorized for IUCV communications.
- Define an IUCV link definition in the TCP/IP configuration file.
- Make an IUCV entry in the kernel parameter file.
- Use Linux for S/390 `ifconfig` and `route` commands to define and activate the IUCV link from Linux for S/390.

Further details can be found in 15.3.3, “IUCV” on page 291.

As with CTC links to the VM TCP/IP stack, IUCV links are point-to-point connections. This means that your local router definitions must be enhanced to route IP traffic for the Linux for S/390 IP addresses through the VM TCP/IP network interface.

6.7 Linux for S/390 device files and virtual device numbers

The CP `QUERY ALL` command displays the device numbers, device types, and associated subchannel numbers for all the devices defined in the Linux for S/390 virtual machine. These are assigned in the order in which the devices were first defined to the virtual machine, either through the CP directory or after the virtual machine was logged on.

```

cp query all
STORAGE = 0128M
XSTORE = none
CPU 00 ID FF0D082296720000 (BASE)
CONS 0009 ON LDEV L0008 TERM START HOST TCPIP FROM 009.012.002.125
...
0009 SUBCHANNEL = 0000
RDR 000C CL A NOCONT NOHOLD EOF READY
000C 3505 CLOSED NOKEEP NORESCAN SUBCHANNEL = 0001
PUN 000D CL A NOCONT NOHOLD COPY 001 READY FORM STANDARD
...
000D SUBCHANNEL = 0002
PRT 000E CL A NOCONT NOHOLD COPY 001 READY FORM STANDARD
...
000E SUBCHANNEL = 0003
DASD 0190 3390 VMZRES R/O 107 CYL ON DASD 09B0 SUBCHANNEL = 0004
DASD 0191 3390 VMZU1A R/W 50 CYL ON DASD 09B7 SUBCHANNEL = 000B
DASD 019D 3390 VMZU1A R/O 120 CYL ON DASD 09B7 SUBCHANNEL = 0007
DASD 019E 3390 VMZP1P R/O 300 CYL ON DASD 09B2 SUBCHANNEL = 0005
DASD 019F 3390 VMZP1P R/O 400 CYL ON DASD 09B2 SUBCHANNEL = 0006
DASD 0200 3390 LINUX2 R/W 20 CYL ON DASD 09BA SUBCHANNEL = 000C
DASD 0201 3390 LINUX5 R/W 1000 CYL ON DASD 09BD SUBCHANNEL = 0008
DASD 0202 3390 LINUX5 R/W 1000 CYL ON DASD 09BD SUBCHANNEL = 0009
DASD 0203 3390 LINUX5 R/W 200 CYL ON DASD 09BD SUBCHANNEL = 000A
DASD 0592 3390 VMZU1R R/O 56 CYL ON DASD 09B5 SUBCHANNEL = 000D
CTCA 0808 COUPLED TO TCPIP 0809 SUBCHANNEL = 000E
CTCA 0809 COUPLED TO TCPIP 0808 SUBCHANNEL = 000F

```

There is a crucial difference between Linux for S/390 and traditional S/390 operating systems. In VM/ESA, when a virtual device is created, CP builds a virtual device block to represent that device. A subchannel number is associated with the virtual device block at that time. Redefining a device number using the CP DEFINE command does not create a new virtual device block. The new device number still has the original subchannel number.

If you detach a device with the CP DETACH command, the associated virtual control blocks are destroyed.

Linux for S/390 has a different view of devices. It expects devices to always appear in the same sequence. Linux for S/390 identifies a disk device based on the order in which it finds it. If you have a kernel parameter:

```
dasd=200 mdisk=201,202,203 root=/dev/mndc ro
```

Linux for S/390 builds a list of disk devices as shown in Table 13.

Table 13. Linux for S/390 device names and S/390 device numbers

Device name	Virtual device number
dasda	200
mnda	201
mndb	202
mndc	203

Virtual device number 203 is the device upon which the Linux for S/390 kernel will mount the root file system.

If you change the virtual device configuration, the boot parameter must be altered to show the new device number sequence *and* any consequential change to the device name of the Linux for S/390 root file system .

If device number 201 is detached and the kernel parameter is left unchanged, Linux for S/390 will make the associations between its device names and device numbers shown in Table 14.

Table 14. New Linux for S/390 device names and S/390 device numbers

device name	device number
dasda	200
mnda	skipped
mndb	202
mndc	203

To reflect the detached device, the kernel parameter must be altered as follows:

```
dasd=200 mdisk=202,203 root=/dev/mndb ro
```

We strongly advise that you list which disk devices Linux for S/390 will use with `dasd` and `mdisk` entries in the kernel parameter specification. If you fail to do this, Linux for S/390 will order *all* the virtual DASD devices it recognizes by subchannel number.

Avoid use of the `autodetect` parameter in the kernel parameter file, since it will automatically register all minidisks defined for the Linux for S/390 virtual machine as devices that Linux for S/390 might want to use. It is difficult to be

certain that you haven't inadvertently added virtual disk devices to your configuration, perhaps through linking to another virtual machine's minidisks to access a CMS application.

6.8 Operational considerations

Running Linux for S/390 in a virtual machine presents some unique opportunities to use VM/ESA for operational management, monitoring and control.

6.8.1 Starting Linux for S/390 virtual machines

Like any other virtual machine, a Linux for S/390 virtual machine can be autologged by an authorized CMS user. If you are running several Linux for S/390 virtual machines and want them all to start automatically, this can be done with either the CP AUTOLOG or XAUTOLOG commands.

See *VM/ESA V2R4.0 CP Command and Utility Reference*, SC24-5773 for further information.

The USER statement in the CP directory entry for a virtual machine has an AUTOONLY option that restricts its being started to autologging *only*.

See *VM/ESA V2R4.0 Planning and Administration*, SC24-5750 for further information.

Another useful VM/ESA facility is the CP LOGON BY command, which permits specified user IDs to log on to the Linux for S/390 virtual machine as a proxy user.

When a Linux for S/390 virtual machine is autologged, it is common practice for it to IPL CMS first and then execute a PROFILE EXEC that ends with a command to IPL from the Linux for S/390 boot device.

Here is an example we tested:

```
/* PROFILE EXEC for Linux virtual machine to be autologged */
'CP LINK TCPMAINT 592 592 RR'
'ACCESS 592 U'
'CP DEF CTC 808'
'CP DEF CTC 809'
'CP COUPLE 808 TCPIP 809'
'CP COUPLE 809 TCPIP 808'
'SET RUN ON'
'SET SECUSER LINUXMON'
'CP DISC' '15'X 'IPL 200'
```

Alternatively, if booting from disk, you could alter the IPL statement in the CP directory entry for the Linux for S/390 virtual machine to:

```
IPL boot_device_number
```

6.8.2 Stopping Linux for S/390 virtual machines

Always use the Linux for S/390 `shutdown` command rather than `halt` if you wish to stop your Linux for S/390 server.

You can also quiesce a Linux for S/390 virtual machine to a lower run level by using the Linux for S/390 `init` command.

After Linux for S/390 has shut down, you can reboot just by entering the CP IPL command from the Linux for S/390 virtual machine console. Linux for S/390 will boot from whichever device it was previous booted.

6.8.3 Secondary console interface

With suitable authorization, a disconnected Linux for S/390 virtual machine can have its console output redirected to another VM user ID. That user ID also has the ability to send commands to the disconnected Linux for S/390 virtual machine, which are treated as input from the Linux for S/390 virtual machine console.

Coupled with another standard VM facility, Programmable Operator, this provides a very powerful tool for managing multiple Linux for S/390 virtual machines. The Programmable Operator can filter and respond automatically to messages from multiple virtual machines based on a set of installation-defined rules. This facility is documented in *VM/ESA V2R4.0 Planning and Administration*, SC24-5750.

The secondary console user ID for a Linux for S/390 virtual machine may be specified in the CP directory entry for that machine. For example:

```
CONSOLE 0009 3215 T PROP
```

will redirect console messages to the PROP virtual machine when the Linux for S/390 virtual machine is running disconnected. It can also be specified in the PROFILE EXEC of a Linux for S/390 virtual machine by using the CP SET SECUSER command. The secondary user ID can enter console input on behalf of the Linux for S/390 virtual machine by using the CP SEND command. This is documented in *VM/ESA V2R4.0 CP Command and Utility Reference*, SC24-5773.

Using the CP SEND command from CMS converts messages to upper case, so we wrote a small REXX program called CPSEND to overcome this problem:

```
/* REXX EXEC to send console input to a Linux virtual machine */
parse arg linuxid cmd
upper linuxid
cpcmd = 'SEND' linuxid cmd
cprc = substr(Diagrc(8,cpcmd),1,9)
exit cprc
```

To test the secondary console facility with Linux for S/390, we booted a Linux for S/390 kernel in one virtual machine, set the secondary user ID, and then disconnected. Logged on as the secondary user ID, we entered a series of Linux for S/390 commands on behalf of the Linux for S/390 virtual machine. Here is one example:

```
cpsend linux5 df
Ready; T=0.01/0.01 16:12:45
LINUX5 : df
Filesystem          1k-blocks      Used Available Use% Mounted on
/dev/dasda1          707876         501304    170612    75% /
```

This technique could be used from a central point of control to manage many Linux for S/390 virtual machines.

You might wish to quiesce all your Linux for S/390 servers while taking backups, or use this method to shut down multiple Linux for S/390 guests prior to a shutdown of VM/ESA.

We found that after issuing a Linux for S/390 `shutdown -h now` command, we could re-IPL a Linux for S/390 virtual machine just by issuing `cpsend Linux_VM_userid IPL` from the secondary user ID.

6.8.4 Taking backups of Linux for S/390 file systems

VM offers a number of ways to take backups of Linux for S/390 file systems:

- CMS file backup
- DASD Dump Restore (DDR)
- SnapShot
- Tivoli ADSM client and NFS
- Proprietary VM backup products

6.8.4.1 CMS file backup

Linux for S/390 disks that are formatted and reserved by CMS are CMS files. They can therefore be backed up using the CMS TAPE DUMP command or some another proprietary VM backup product. This type of backup treats the whole Linux for S/390 volume as a single file. It does not provide granularity at the Linux for S/390 file level. To restore a single Linux for S/390 file you must restore the entire CMS reserved file.

6.8.4.2 DDR

This standard CMS utility backs up minidisks or full volumes on a block or track basis. It has no understanding of the file structure contained on the disk. This type of backup is device dependent and at a volume entity level. To restore a single file, you must restore the whole backed-up volume.

6.8.4.3 SnapShot

If you have a disk subsystem that supports IBM's RAMAC SnapShot for VM/ESA, you can take instantaneous snapshot logical copies of complete Linux for S/390 disk volumes. These can be backed up using the VM DASD Dump Restore (DDR) program to magnetic tape without affecting the source data.

6.8.4.4 Tivoli ADSM client and NFS

Using the Tivoli ADSM client for a workstation or OS/390 OpenEdition, and mounting the Linux for S/390 file system NFS from that client, you can back up Linux for S/390 data with file level granularity.

While this provides a comprehensive backup capability, it has the disadvantage that data flows from Linux for S/390 to the ADSM client and then directly to the Tivoli ADSM server.

6.8.4.5 Proprietary VM backup products

VM backup products from Independent Software Vendors (ISV) can also be used to back up Linux for S/390 data. Some have the ability not only to back up CMS files, but also to incrementally back up full volumes, irrespective of the file structures they contain. This is achieved by keeping track of changed tracks or blocks on the volume.

6.9 Performance considerations

The factors that affect the performance of a virtual machine running Linux for S/390 are the same as for any other virtual machine.

There are techniques to reduce paging, increase the share of system resources, and map data on minidisks into memory.

As with many performance improvements, there is frequently a trade-off involved. Any performance-related change should be made only after consideration and understanding of the likely effects on the rest of the system. For a more comprehensive discussion, refer to *VM/ESA V2R4.0 Performance*, SC24-5782.

6.9.1 Reducing Linux for S/390 swapping

When you have a swap file defined for Linux for S/390, double paging can potentially occur. VM/ESA may page Linux for S/390 memory onto its own paging devices and Linux for S/390 may swap pages of its own memory into a Linux for S/390 swap file.

This is illustrated in Figure 44.

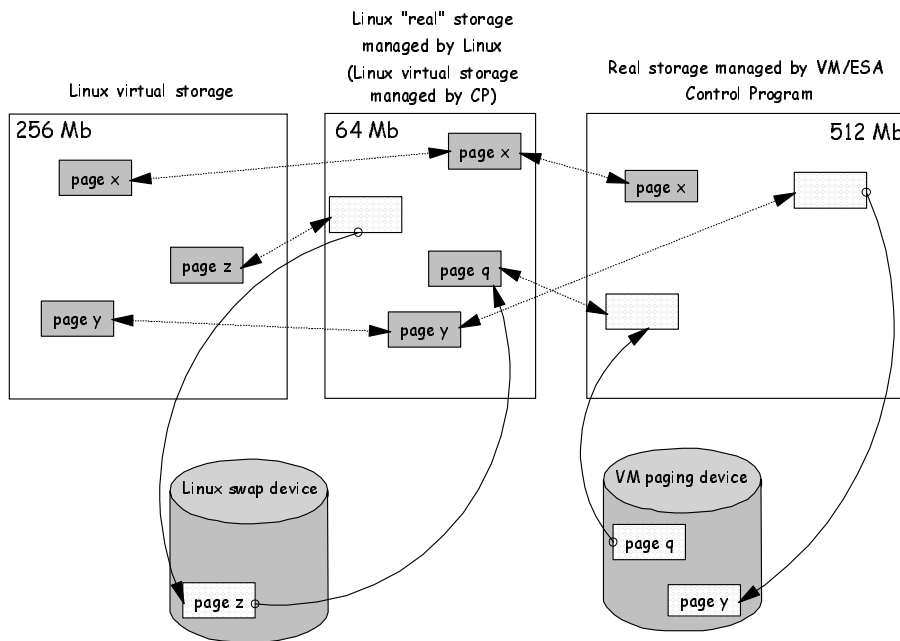


Figure 44. Paging in a three-tier storage model

Page x of Linux for S/390 virtual storage is resident in both the "real" storage of the Linux for S/390 virtual machine and the real storage of the system. Page y is resident in the "real" storage of the Linux for S/390 virtual machine

but has been paged out by the VM/ESA Control Program from real storage. Page z did not fit into the “real” storage of the Linux for S/390 virtual machine and was swapped out to the Linux for S/390 swap file. When Linux for S/390 tried to swap in page z to page q of its “real” storage, the target page q had been paged out by CP and must be paged in again. This leads to inefficient double paging.

Techniques that can reduce or eliminate double paging are:

- Make the storage of the Linux for S/390 virtual machine bigger to eliminate or minimize swapping. This can be a somewhat self-defeating exercise. As you increase the amount of storage available to the Linux for S/390 virtual machine, Linux for S/390 will tend to consume the additional capacity by caching more Linux for S/390 files in its storage.
- Dedicate real storage to the Linux for S/390 virtual machine by making it a V=R or V=F guest. CP will do no paging and leave memory management to the Linux for S/390 virtual machine.

Since VM/ESA has a highly efficient block paging mechanism, you may find it preferable to let VM page rather than have Linux for S/390 swap. In any situation, the trade-offs must be evaluated.

Another way to reduce the likelihood of a Linux for S/390 virtual machine’s storage being paged by CP is with the CP SET RESERVED command. You can reserve a number of pages that a virtual machine is entitled to have resident in real storage at all times.

The Linux for S/390 XPRAM driver allows the Linux for S/390 kernel to map a swap file onto S/390 expanded storage. While this technique would have the most obvious value when Linux for S/390 runs in an LPAR, it could be used by a Linux for S/390 virtual machine that had S/390 expanded storage dedicated to it.

6.9.1.1 Minidisk caching

VM/ESA provides a number of tools to map minidisks into real storage, thereby avoiding normal disk I/O.

Minidisk cache is a data-in-memory technique that attempts to improve performance by decreasing the I/O to DASD required for minidisk I/O. Minidisk cache trades increased use of real and expanded storage for decreased DASD I/O.

Since paging to DASD increases as the amount of available real and expanded storage decreases, you should expect some increase in paging I/O.

When planning what data should be enabled for minidisk cache, it is generally better to start with everything enabled and then eliminate poor candidates later.

The maximum benefit of minidisk cache is achieved with read-only data that is referenced multiple times. If you have multiple Linux for S/390 virtual machines that can share a read-only minidisk, that minidisk can be an ideal candidate for minidisk caching.

6.9.1.2 Virtual disks

Virtual disks in storage are temporary Fixed Block Architecture (FBA) minidisks allocated from real memory instead of requiring I/O to real disk devices. Because the I/O overhead is avoided, virtual disks in storage may be faster to use than other minidisks.

However, the improved I/O performance can be a trade-off for increased storage requirements or increased paging activity.

Virtual disk is a good candidate for the swap device of a Linux for S/390 virtual machine. A virtual disk can be defined in the CP directory entry for a Linux for S/390 virtual machine. It is created at logon time. The CMS PROFILE EXEC should then invoke the CMS FORMAT and RESERVE commands to prepare the minidisk for use by Linux for S/390 when it is booted.

6.9.1.3 Expanded storage driver

Linux for S/390 has a driver that allows S/390 expanded storage to be used to cache Linux for S/390 data. It is more likely to be used in an LPAR, but should work if expanded storage is dedicated to the Linux for S/390 virtual machine.

S/390 processors support an architectural maximum of 2 gigabytes of central storage. However, additional memory can be defined as expanded storage. Memory in expanded storage is addressable as 4 KB blocks.

The Linux for S/390 XPRAM device driver is a block device driver that allows Linux for S/390 to access expanded storage. Thus XPRAM can be used as a basis for fast swap devices and/or fast file systems.

6.9.2 Virtual machine priority

The share of the real processor and other resources available to a virtual machine can be controlled using CP commands such as SET SHARE. Shares are specified in weights, which can be absolute or relative and may also be limited or capped.

The CP SET QUICKDSP command ensures that a virtual machine does not need to compete for sufficient real storage whenever it has work to do, but can run immediately.

There are many other commands and settings that affect the performance of Linux for S/390 running in a virtual machine.

Chapter 7. Installing SuSE Linux on S/390

This chapter describes how to install SuSE Linux for S/390. Since it is based on a prerelease version, there might be minor differences between how screens are shown here and how they appear in the final release of SuSE Linux for S/390.

You will need the SuSE Linux for S/390 CD. We used the beta 2 version, dated July 28, 2000. At the time of writing of this book, it was available on the Web at:

`ftp://ftp.suse.com/pub/suse/s390/pre-suse-s390.iso`

7.1 Types of installation

You can install SuSE Linux on an LPAR. Use the hardware console (SE or HMC) to IPL and set up the network connection for Linux. This type of installation requires either booting from a tape (or emulated tape on MP3000) or using the **Load from CDROM or Server** task in the SE.

Alternatively, you can install SuSE Linux under VM/ESA, in which case a 3270 screen will be used for the initial steps. Linux will then normally be booted through the reader (although a tape IPL is also possible). This is the most flexible setup, especially if you plan to do development or testing, where the Linux system has to be IPLed more often.

Note: Although you can install SuSE Linux for S/390 in native mode (that is, using the whole machine), we do not recommend this method. Instead, use LPARs, VM/ESA, or VIF because these methods offer greater system flexibility.

After the network is set up, you must connect to the Linux system with a terminal that supports full-screen mode (a line mode terminal such as a 3270 does not work). A telnet session from a Linux (or Windows) workstation should be opened to proceed with the installation using YaST (which means Yet another Setup Tool, the SuSE menu-based installation and administration utility). At this point there is no difference between the types of installation, except for a few settings.

7.2 System requirements

This section describes the system requirements (hardware, microcode level and software) needed to install SuSE Linux for S/390.

7.2.1 Required hardware features

The following features must be on the system:

7.2.1.1 Memory and processor

You need at least 128 MB of RAM. The processor must be of the CMOS generation and be at least G2, although using a G5 or better is recommended.

G3, G4 and MP2000 do not support IEEE floating point in the hardware. Because of this, you might see performance degradation if you run Linux for S/390 and Linux applications on such machines, as IEEE floating point will be emulated by software.

However, since IEEE exception handling is not emulated, you'll get warning messages whenever a program tries to use it.

7.2.1.2 DASD volumes

If you install natively or in an LPAR, you need at least one dedicated 3390 model 3 disk; however, two are recommended so a swap space can be utilized.

Note: Real 3390 disks attached to real 3990 controllers are not supported.

If you install as a VM guest, you need one 3390 model 3 for the root file system, and 200 (or more) cylinders for the swap space. You also need a minidisk with 20 (or more) cylinders if you want to boot (IPL) from a disk and not from the VM reader.

7.2.1.3 Tape unit

If you install natively or in an LPAR, you need temporary access to a tape unit. On a Multiprise 3000, you need access to the emulated tape.

If you install as a VM guest, a tape unit is not required if you install using the VM reader.

7.2.2 Required APARs and fixes

Table 15 on page 135 lists the APARS and fixes needed.

Table 15. Required APARs and fixes

System	APAR or fix number
VM/ESA	VM61762 is required if the version of VM is V2R3.0. VM62337 is required when using IEEE FPU under VM. VM62410 is required when using IEEE FPU under VM. VM62520 is required to run LINUX guests with more than one virtual CPU. VM62573 is required to avoid FRE001 abend when telnetting into VM in line mode. PQ34318 is required when using TCP/IP under VM. VM62337 and VM62410 are required only to run on G5, G6, or Multiprise 3000 processors.
Multiprise 3000	Microcode fix EC F34643 MCL048. Microcode fix EC 34663 MCL087 load from CD.
9672 (G5/G6)	Microcode fix MCL025 EC 99918 load from CD.
All systems	OSA Express Fast Ethernet card LIC code level 324.

7.2.3 Software

If you install natively or in an LPAR, you need an S/390 operating system to create a tape (OS/390, VM/ESA or VSE/ESA). You also need a utility like DITTO or IEBGENER to copy the installation files onto a tape.

On a Multiprise 3000, you can use emulated I/O or the Load from CDRom or Server feature, so you don't need to create a tape.

If you install from a VM reader, you only need VM/ESA.

7.3 Connection requirements

This section describes the connection requirements, such as console and network access, needed to install SuSE Linux for S/390.

7.3.1 Console

If you install natively or in LPAR, use the console that is integrated with the Hardware Management Console (HMC) or the Support Element (SE).

If you install as a VM guest, use the virtual machine console of the Linux for S/390 guest, that is, the (real or emulated) 3270 terminal through which you

log on to VM. To connect from a SuSE Linux workstation, install the package x3270 from the series xap on that workstation.

The intended use of the console is solely to launch Linux. After Linux is running, use a telnet connection directly to Linux for S/390 to log into Linux and access the shell and other applications such as vi or YaST.

7.3.2 Network connection

A TCP/IP network connection is required in order to get files from the FTP or NFS server and to telnet into the Linux system. The connection can be one of the following:

OSA (OSA-2, OSA Express)

CTC (virtual or real)

ESCON channels

PCI adapter (emulated I/O, only on MP3000)

All network connections require the correct setup on both Linux for S/390 and the remote system, and a correct routing between both ends.

7.3.3 The telnet client

You need telnet in full-screen mode (as previously mentioned, a line mode terminal such as a 3270 does not work). With Windows, for example, you can use PuTTY, a freeware telnet/SSH client that can be found on the Internet at:

<http://www.chiark.greenend.org.uk/~sgtatham/putty/>

Note that Windows terminal programs tend to have the following problems, listed here with their possible workarounds:

- They can put meaningless values into the `TERM` variable.

As a workaround, type the following on the command line before you start YaST:

```
export TERM=vt220
```

- They can intercept certain keys (like the function keys); pressing one of the function keys often triggers actions like displaying copyright notices instead of just handing the keystroke to telnet. Even when some function keys work normally, others may deliver the escape character, which causes YaST to abort the current dialog.

As a workaround, use `Ctrl-f 1` for F1, `Ctrl-f 2` for F2 and so on (`Ctrl-f 0` for F10). If the tab key does not work, use the arrow up and arrow down keys to switch between the entries or actions.

- They use nonstandard character sets; for example, the characters used for the subwindow borders of YaST might not be displayed correctly.

No workaround is needed, since this problem is only cosmetic.

Because of these problems, we strongly recommend that you use SuSE Linux (on a PC or workstation) since it comes with a telnet client and terminals that work as expected.

Note: With YaST, the terminal has to be at least 25 lines by 80 columns. Therefore you may have to resize the terminal before launching YaST.

7.3.4 NFS or FTP server

The installation loads the software packages from the CD via NFS or FTP. You can use SuSE Linux to supply the installation files, because both an NFS server (as package `aaa_base` in series a) and an FTP server (as package `nkitb` in series a) are included in the SuSE Linux distribution. You will probably want to use the same workstation that is used to connect via telnet.

Note that there is no NFS server included in Windows. The Rockridge extension (the standard for long filenames on an ISO9660 file system, used for CDs) is not supported by Windows. This, plus the fact that Windows applies case conversions on filenames, may cause installations from a Windows NFS or FTP server to fail.

Testing the server

To test the NFS server, mount the CDROM directory on the server (often done via the command `mount /dev/cdrom /cdrom`) and set up the `/etc/exports` file so NFS clients can access the CD. Now mount the CDROM directory from your local workstation with an NFS client (for example; `mount nfsserver:/cdrom /mnt/susecd`) and try copying a file from the CD.

To test the FTP server, *connect to it as an ordinary user* and download a file. (Do not try to connect as root, because by default on Linux, the user root is not allowed to connect.)

7.4 IPLing the install system

The installation of SuSE Linux begins with IPLing the installation system, a RAM disk-based system that contains all programs necessary for the actual

installation. Before connecting via telnet to the system, of course, you must set up the network.

7.4.1 IPLing from the VM reader

To IPL from the reader, first connect to the FTP server with the installation CD and transfer the kernel, parameter file, and RAMdisk as fixed block 80 byte records. Following are example FTP commands:

```
bin
locsite fix 80
cd /cdrom/suse/images
get vmrdr.ikr vmlinux.txt
get parmline linux.parm
get initrd initrd.txt
quit
```

Then create a REXX EXEC that punches the files into the reader and IPLs Linux from the reader (**Note:** be sure you don't have any important files in your reader as all files are first purged):

```
type lin exec
/* */
'close rdr'
'purge rdr all'
'spool punch * rdr'
'punch vmlinux txt a (noh'
'punch linux parm a (noh'
'punch initrd txt a (noh'
'change rdr all keep nohold'
'ipl 00c'
```

Alternatively, you may want to create a file that just IPLs from the reader. This EXEC can be used if the proper files are already in your reader:

```
type lipl exec
/* */
'change rdr all keep nohold'
'ipl 00c'
```

7.4.2 IPLing from tape

To IPL from tape, you have to use a tape description file. Some sample files (named `awsoma*.tdf`) are on the CD in the directory `/suse/images/`. These files contain lists of files that are to be transferred to tape. It is possible the file may have to be modified:

```
@TDF
H:\SUSE\IMAGES\TAPEIPL.IKR UNDEFINED RECSIZE 1024
```

```
H:\SUSE\IMAGES\PARMLINE      UNDEFINED RECSIZE 1024
H:\SUSE\IMAGES\INITRD        UNDEFINED RECSIZE 1024
TM
TM
EOT
```

7.4.3 IPLing from the CD-ROM (emulated tape)

With Multiprise 3000, you can IPL directly from the CD-ROM; the SE emulates a tape with the data on the CD-ROM.

To do this, you first have to set up emulated I/O in the IOCDS. Then you “rewind” the emulated tape by issuing the following on the OS/2 console (assuming that 080 is the emulated I/O device number):

```
F:\> awsmount 080 /rew /D /R
```

Then select **LOAD** with the same device.

Note: You may have to try the rewind and load task several times before it actually works, there will be a fix (provided by IBM) for that in the future.

7.4.4 The Load from CDROM or server task

With Multiprise 3000 and Parallel Enterprise Server (Generation 5 or 6), you have the option of IPLing from the local or Hardware Management Console (HMC) CD-ROM or an FTP server. To be able to use this new function you have to be on the following microcode level:

- Multiprise 3000 MCL 087 EC 34663
- Parallel Enterprise Server (Generation 5 or 6) MCL025 EC 99918

After starting the task from the HMC, a dialog appears that lets you choose whether you want to load from CDROM or an FTP server. If you load from an FTP server, you’ll have to enter the hostname (or IP) of the server, your user ID, and the password for FTP access.

The next menu lets you select among the existing files having a .ins extension. These files contain a list of the files that actually get loaded (in a certain order).

For the SuSE installation CD, the corresponding files are:

- F:\suse\images\linux.ins (on the HMC), or
- /cdrom/suse/images/linux.ins (on an FTP server, assuming that the mount point of the CD-ROM is /cdrom).

Loading this file causes the (Linux-) install system to be IPLed.

7.5 Setting the network parameters in Linux

Note that in the dialogs shown in this section, user responses are printed in bold.

Initially you are asked which network device is to be used, as follows:

```
First, select the type of your network device:
0) no network
1) for osa token ring
2) for osa ethernet
3) for channel to channel
4) for escon channel
Enter your choice (1-4): 3
```

In our case, we entered 3 for a CTC connection. The script continues:

```
Please enter your full host name (e.g. s390.suse.com): suse390.mydom.com
Please enter your IP address: 123.123.123.7
Please enter the net mask: 255.255.255.0
```

For CTC, which is a point-to-point connection, the IP of the peer (which will also serve as a gateway) is requested:

```
Please enter the IP address of your peer: 123.123.123.88
```

For non-point-to-point connections, the broadcast IP and the gateway IP must be given. You will not see these two lines for CTC or IUCV:

```
Please enter the broadcast address: 123.123.123.255
Please enter the gateway address: 123.123.123.77
```

For all network types, you must give the DNS information:

```
Please enter the IP address of the DNS server: 123.123.55.66
Please enter the DNS search domain (e.g. suse.com): mydom.com
```

The data is summarized and you can either confirm it or redo the input in case of an error. After accepting the network information, SuSE Linux should IPL onto the RAMdisk. Many lines of output will go by. Watch to verify that the networking device comes up successfully.

Next, you continue by loading the DASD device driver.

7.6 Loading the DASD device driver

Now that networking is up, you should be able to telnet into the new system:

```
telnet 123.123.123.7
```

Then login as user root:

```
Connected to localhost.  
Escape character is '^]'.  
Welcome to SuSE Linux 7.0 (S/390) - Kernel 2.2.16 (3).  
suse390 login: root  
Password:
```

Press Enter after you are prompted for the password (the root password is empty for the install system). Note that no characters are echoed in any way when a password is entered.

A prompt like the following indicates that the login was successful:

```
suse390:~ #
```

To load the DASD device driver, first perform a check by issuing the following command:

```
insmod dasd dasd=probeonly
```

That will load the driver in probeonly mode, which means all DASD is probed, but not accessed. If you issue the following command, you can print a list of all accessible DASDs:

```
cat /proc/dasd/devices
```

For the prerelease we used, only the following error message was displayed:

```
cat: /proc/dasd/devices: Cannot allocate memory
```

Therefore, we chose the alternative method of issuing this command from the VM control program (this command can only be issued from the 3215 console session):

```
#cp q dasd  
00: CP Q DA  
00: DASD 0190 3390 VMZU2R R/O      107 CYL ON DASD 09B6 SUBCHANNEL = 0004  
00: DASD 0191 3390 VMZU1R R/W      10 CYL ON DASD 09B5 SUBCHANNEL = 0008  
00: DASD 019D 3390 VMZU1A R/O      102 CYL ON DASD 09B7 SUBCHANNEL = 0007  
00: DASD 019E 3390 VMZP1P R/O      300 CYL ON DASD 09B2 SUBCHANNEL = 0005  
00: DASD 019F 3390 VMZP1P R/O      400 CYL ON DASD 09B2 SUBCHANNEL = 0006  
00: DASD 0200 3390 LINUX2 R/W      200 CYL ON DASD 09BA SUBCHANNEL = 0009  
00: DASD 0222 3390 LINUX4 R/W      2000 CYL ON DASD 09BC SUBCHANNEL = 000A  
00: DASD 0224 3390 LINUX2 R/W      1000 CYL ON DASD 09BA SUBCHANNEL = 000B  
00: DASD 0235 3390 LINUX7 R/W      3000 CYL ON DASD 0999 SUBCHANNEL = 000C  
00: DASD 05FA 3390 VMZU1A R/O      100 CYL ON DASD 09B7 SUBCHANNEL = 0014
```

```
00: DASD 05FB 3390 VMZU1A R/O      100 CYL ON DASD 09B7 SUBCHANNEL = 0013
00: DASD 05FC 3390 VMZU1A R/O      100 CYL ON DASD 09B7 SUBCHANNEL = 0012
00: DASD 05FD 3390 VMZU1A R/O      100 CYL ON DASD 09B7 SUBCHANNEL = 0011
00: DASD 05FE 3390 VMZU1A R/O      100 CYL ON DASD 09B7 SUBCHANNEL = 0010
```

To unload the driver again, issue:

```
rmmod dasd
```

Note: The probeonly step was introduced as a “stop and think” point in order to make absolutely sure that no DASDs that belong to other operating systems are accessed. It is not strictly necessary, so its use is optional.

The actual load of the device drivers is done by the command:

```
insmod dasd dasd=222,224,235
```

The numbers represent the subset of device numbers that you, after double-checking, decided to really use for SuSE Linux for S/390.

The following command provides a final check, if needed:

```
cat /proc/dasd/devices
```

```
0222(ECKD) at (94:0) is  dasda:active  at blocksize: 4096, 360000 blocks, 1406 MB
0224(ECKD) at (94:4) is  dasdb:active  at blocksize: 4096, 180000 blocks, 703 MB
0235(ECKD) at (94:8) is  dasdc:active  at blocksize: 4096, 540000 blocks, 2109 MB
```

7.7 Installing with YaST

Having logged in as root, you can check that the TERM variable is meaningful by issuing:

```
echo $TERM
```

In return, you might receive something like this:

```
jeosverycoolterm
```

You can fix it by issuing the following (note the absence of the '\$'):

```
export TERM=vt220
```

Now you can start YaST:

```
yast
```

Should YaST issue a warning about the terminal having less than 25 lines of at least 80 characters, as shown in Figure 45 on page 143, you can stop YaST, resize the terminal, and start YaST again.

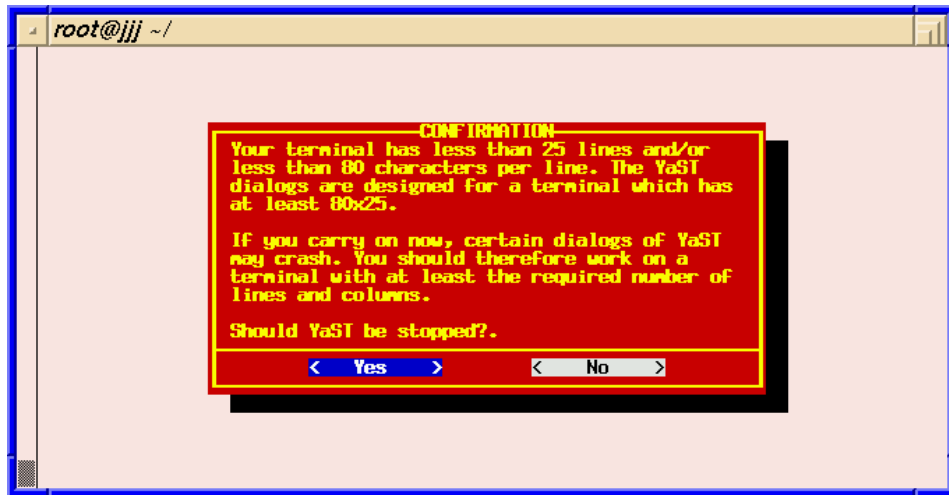


Figure 45. YaST: warning message about a small terminal size

However, some Windows terminals just can't be resized. In that case, a different telnet client must be used. A good free one is Tera Term. See their home page at:

<http://hp.vector.co.jp/authors/VA002416/teraterm.html>

After starting YaST, you are asked which language you want to use:

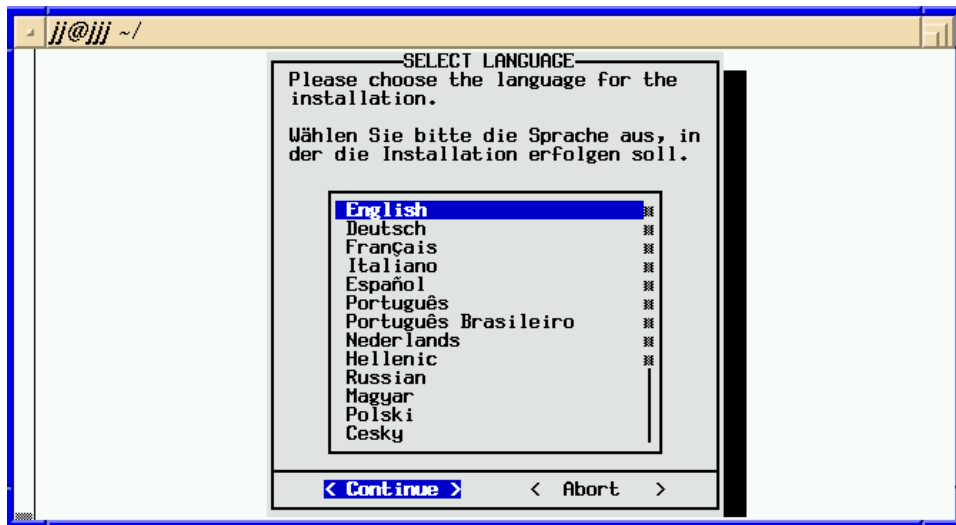


Figure 46. YaST: language selection

The next screen lets you choose the installation medium:

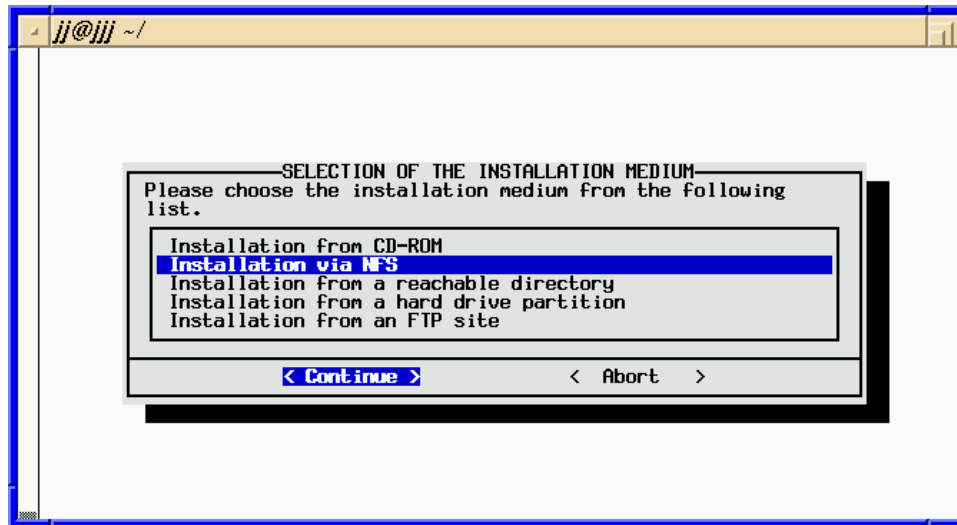


Figure 47. YaST: selection of the installation medium

In this example we choose NFS, but FTP would also be a valid option.

If you are using FTP, you have to enter the server IP and the directory. In addition you are asked for user name (login) and password you want to use. As you may remember, by default the user root is not allowed to connect, so you should create an ordinary user to connect to the FTP server.

Next, as shown in Figure 48 on page 145, the NFS Server data has to be entered, including the IP address (or DNS name) of the server and the directory (mount point) of the SuSE CD-ROM.

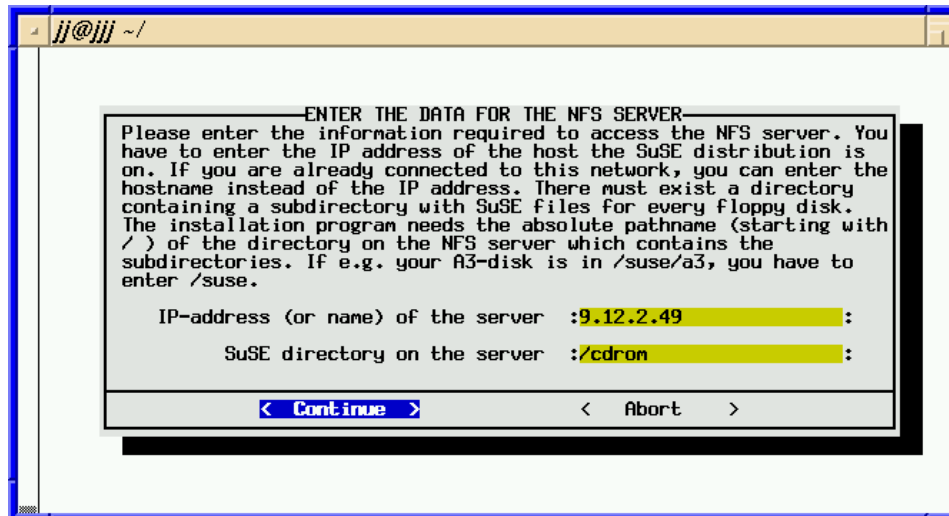


Figure 48. YaST: entering the data for the NFS server

In this case the mount point is /cdrom.

Choose the type of installation in the following screen:

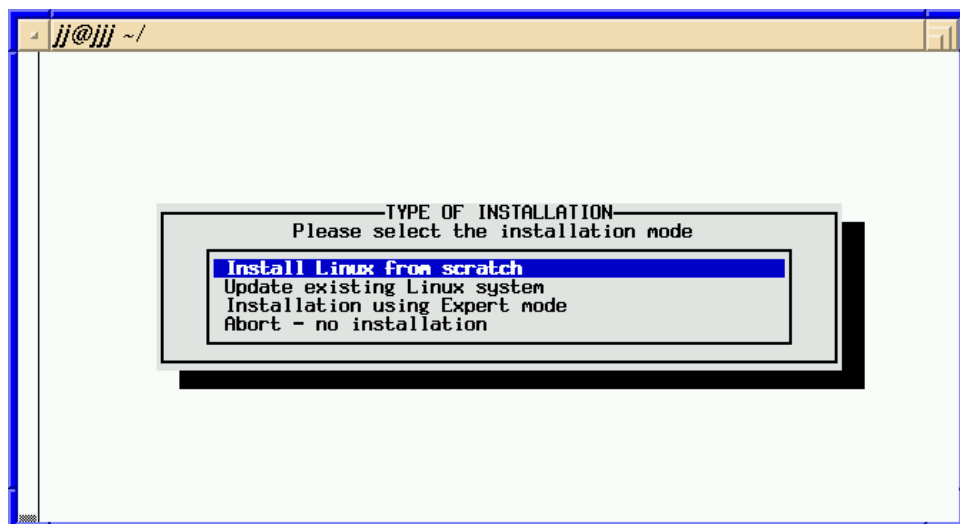


Figure 49. YaST: selecting the installation menu

In our case, we chose **Install Linux from scratch**, since this was a new installation.

Selecting the swap partition is the next task:

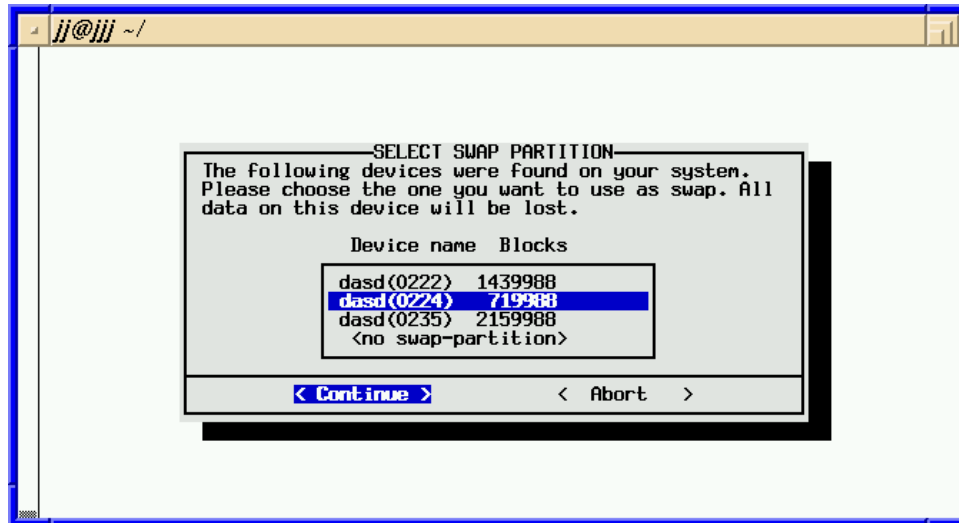


Figure 50. YaST: select swap partition menu

The screen reflects the specific situation for our setup; you will see different sizes and device numbers. Note that using swap space, while recommended, is not absolutely necessary.

Now that swap space has been set up, the file systems have to be created:

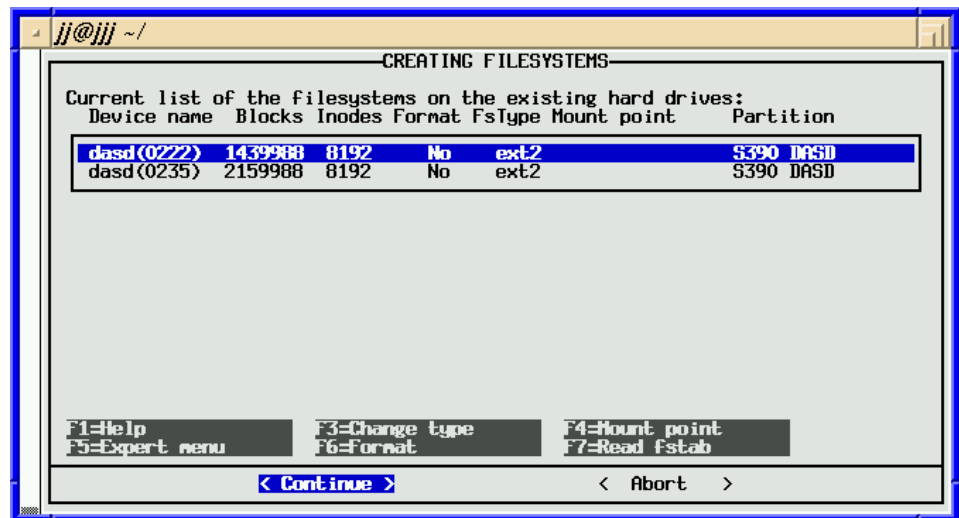


Figure 51. YaST: creating filesystems menu

You need to define a mount point for each partition that SuSE Linux will access; the F4 key brings you to the corresponding menu (**Note**: you may have to press Ctrl+F4):

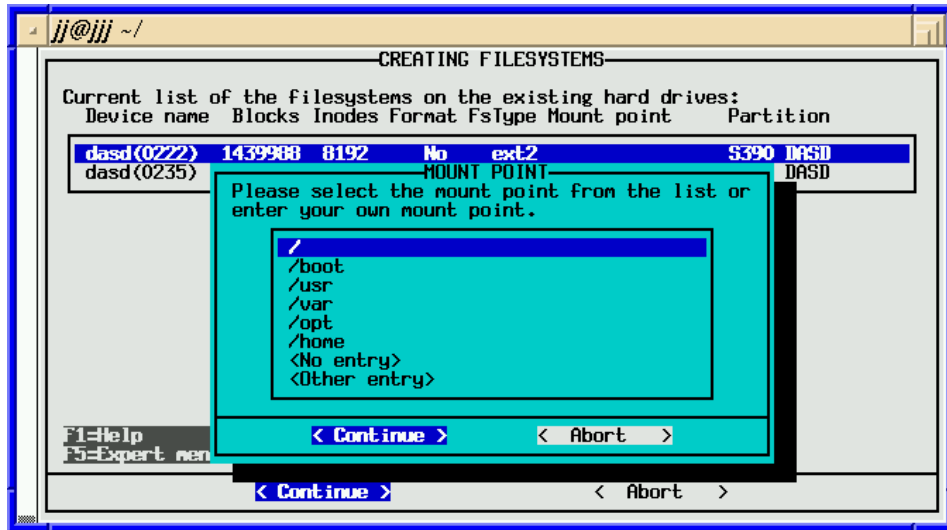


Figure 52. YaST: the mount point menu

At least one partition must be mounted as the root file system (mountpoint "/"). In this example, we install the entire file system over root. Next, you have to determine the format method:

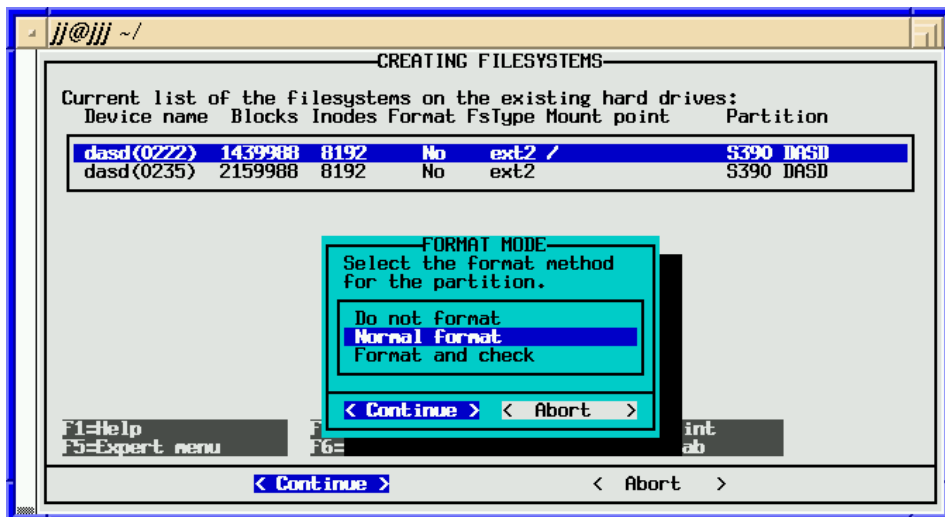


Figure 53. YaST: the format mode menu

You will be prompted for the format method of each partition. Here, **Normal format** is the usual choice. The Format and check option should not be necessary on the hardware we use and would take long time. Choosing the **Do not format** option only makes sense when a (Linux-) file system already exists on the partition; the files from the previous installation will stay on the disk in this case.

In Figure 53 on page 147, the DASD at address 235 was not attached to the Linux installation; however, we still can format and mount (permanently or temporarily) the partition later from the installed system.

A confirmation request follows:



Figure 54. YaST: confirmation before actually creating file systems

With the kernel versions we used at the time of writing, only the ext2 type of file system is available. Later on, other file system types, notably the reiser file system, will be available. The F3 key allows you to modify the type of file system.

Having created the file systems, YaST then proceeds by reading the description of the packages available on the installation medium:

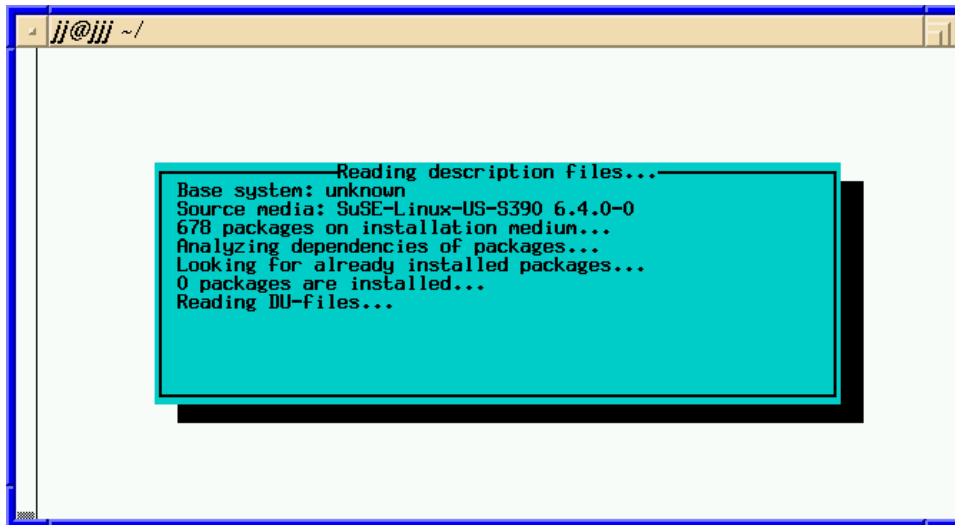


Figure 55. YaST: reading the description data

Select the software packages you want to install:

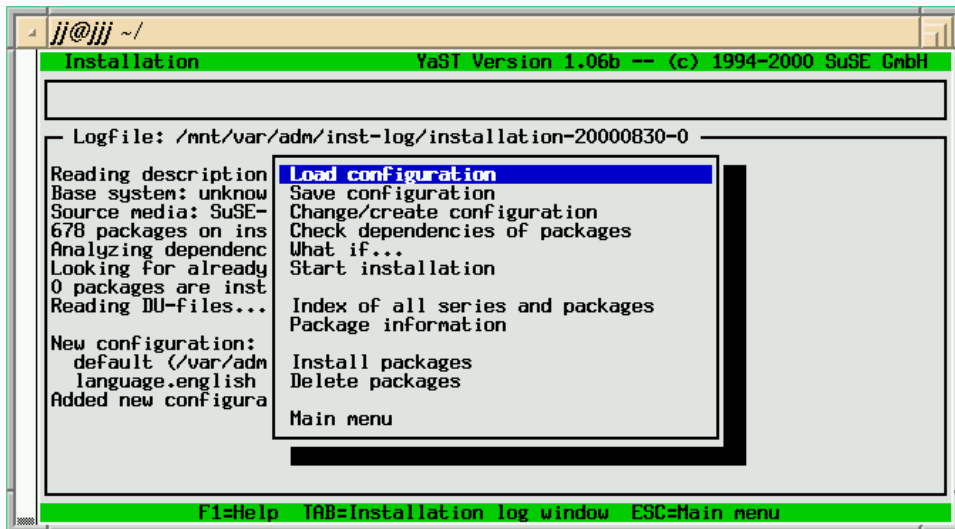


Figure 56. YaST: The main installation menu

To use a preconfigured selection as the base for your installation, select **Load configuration**.

A menu with several predefined configurations appears:

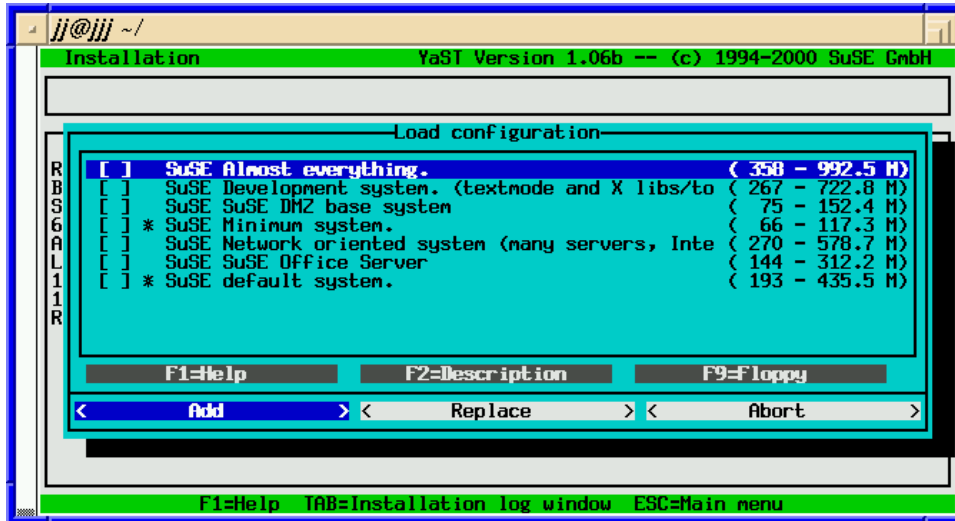


Figure 57. YaST: the load configuration menu

The asterisks indicate previous configurations (here, the default installation configuration). Pressing the space bar on any configuration will mark it as user-selected. When leaving the menu, you can decide to add your selections to the previously chosen ones, or to replace the previous configuration with the one you made.

You should ensure that the choice selected will fit on the available disk space. If you don't like one of the predefined sets of Linux configurations, you can adapt the software selection by selecting individual packages. This possibility is shown in Figure 58 and Figure 59 on page 151. Even after the installation is complete, you can still use YaST to install or deinstall packages from the CD-ROM.

Back in the main installation menu, choose the **Change/create configuration** option to select or deselect individual software packages:

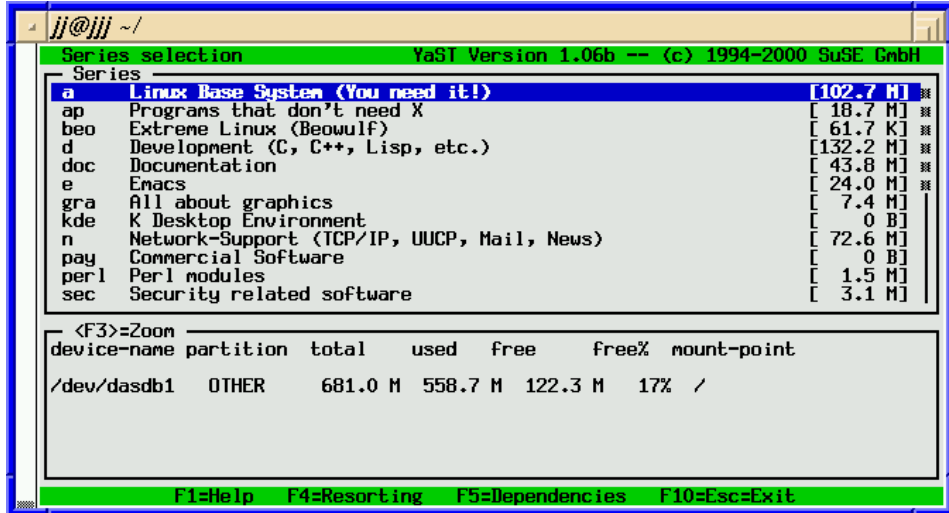


Figure 58. YaST: the various software packages

Select a series to browse the packages (here, we chose series n):

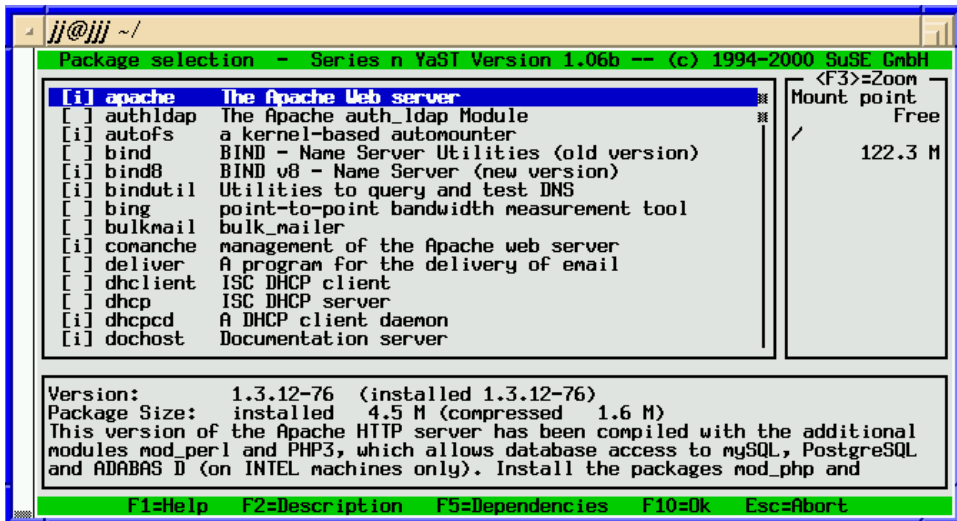


Figure 59. YaST: individual packages in series n

You can mark a package for installation (indicated by **[i]** or **[X]**), or for replacement with the package on the installation medium (indicated by **[R]**), or for deletion (indicated by **[D]**). A package not installed is tagged as **[]**.

Note that some software packages have dependencies on other software packages, and certain packages (those of the base system) are absolutely necessary. YaST will warn you when you are likely to make a mistake and give you a chance to correct it.

Pressing F5 will check the dependencies for the configuration selected so far:

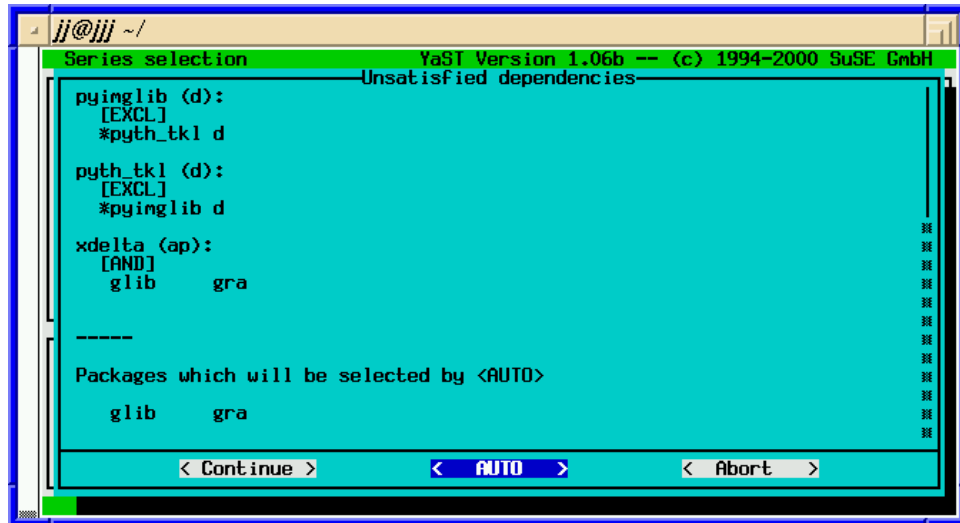


Figure 60. YaST: automatically resolving dependencies

If YaST offers to automatically resolve the dependencies, allow it to do so by choosing **AUTO**.

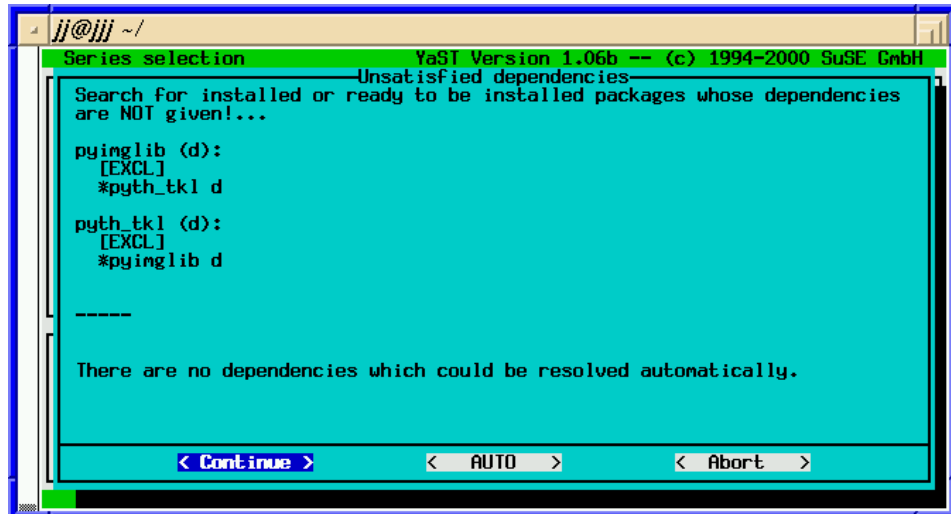


Figure 61. YaST: some dependencies cannot be resolved automatically

You needn't be concerned about dependencies that YaST can't resolve because these are redundant packages, so you can select **Continue** if you get the message that dependencies could not be resolved automatically.

In our case, after returning to the main installation menu, we chose **what if ...** to check the size of the installation:

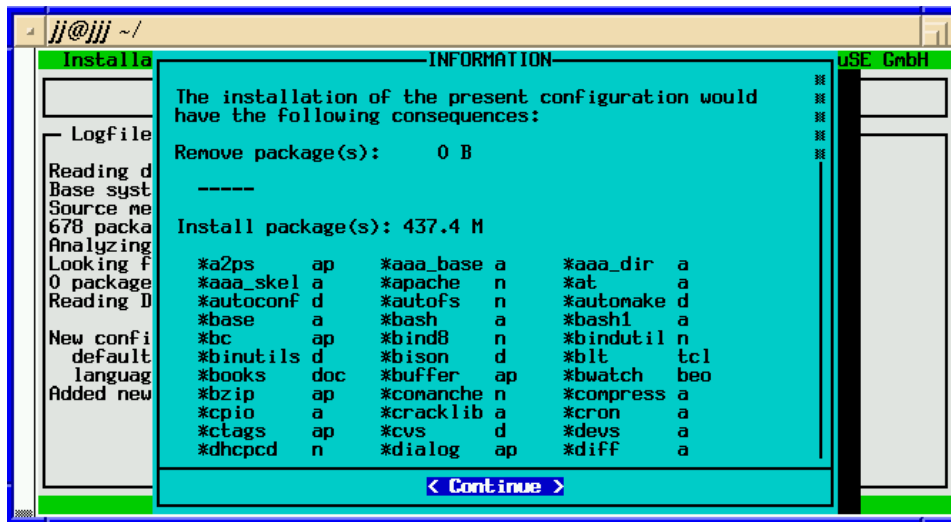


Figure 62. YaST: the consequences menu

Next, we started the installation as shown in Figure 63 (this step may take 20 minutes or more):

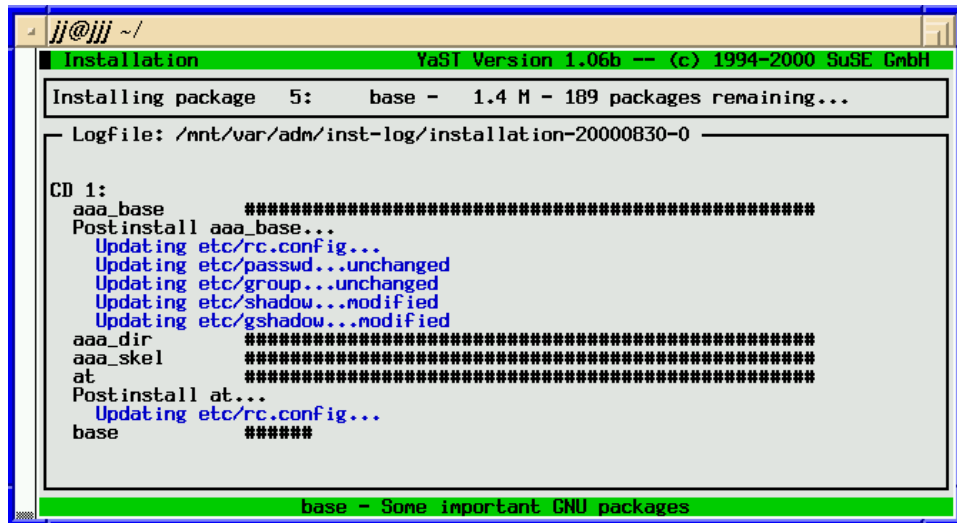


Figure 63. YaST: the installing package menu

After the installation completes, you are returned to the main installation menu:

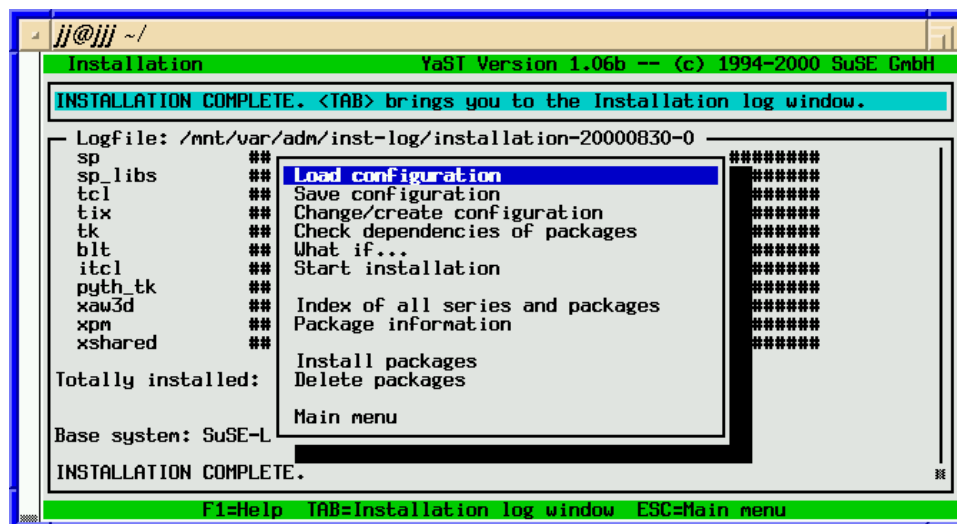


Figure 64. YaST: package installation finished

After choosing **Main menu** to proceed, you are asked which kernel to install. Select the **default kernel** as shown in Figure 65:

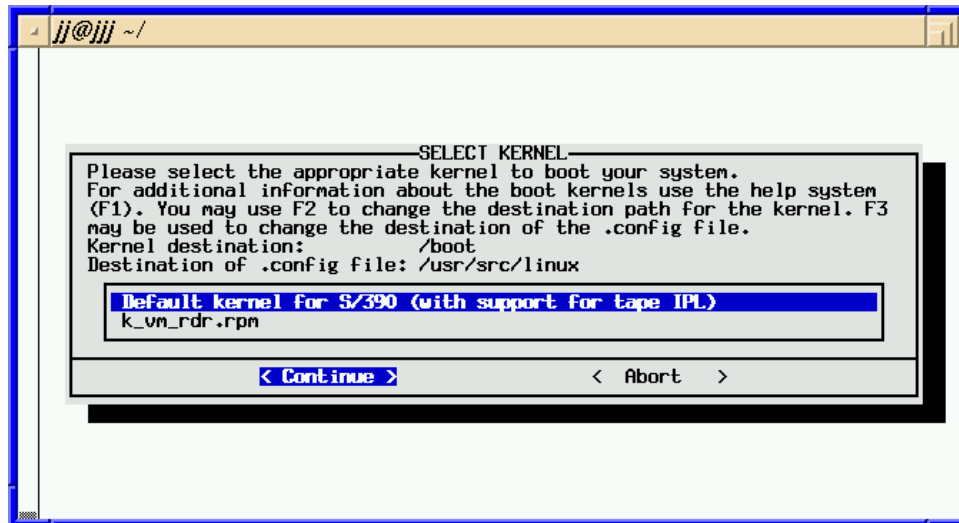


Figure 65. YaST: selecting a kernel

You are presented with a list of time zones; choose the one you need:

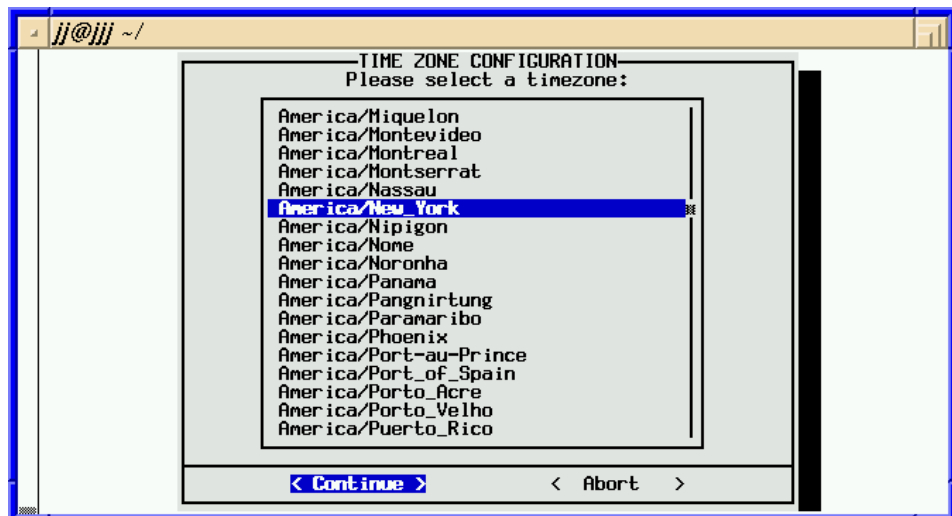


Figure 66. YaST: list of time zones

You can choose to set your clock to local time, or to Greenwich Mean Time (GMT):

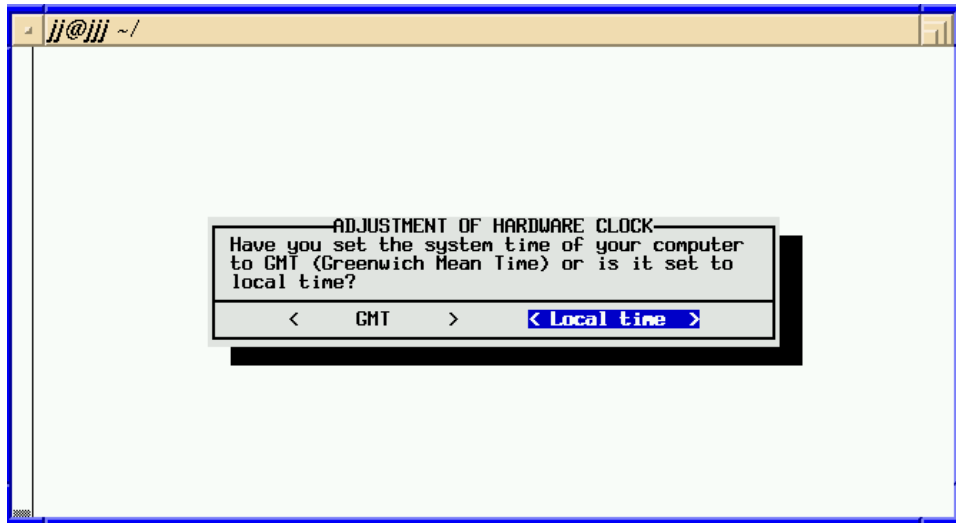


Figure 67. YaST: local time or GMT

Ensure your choice corresponds to the actual setting of the hardware clock.

The hostname and DNS domain defaults to what you entered when the initial netsetup script ran:

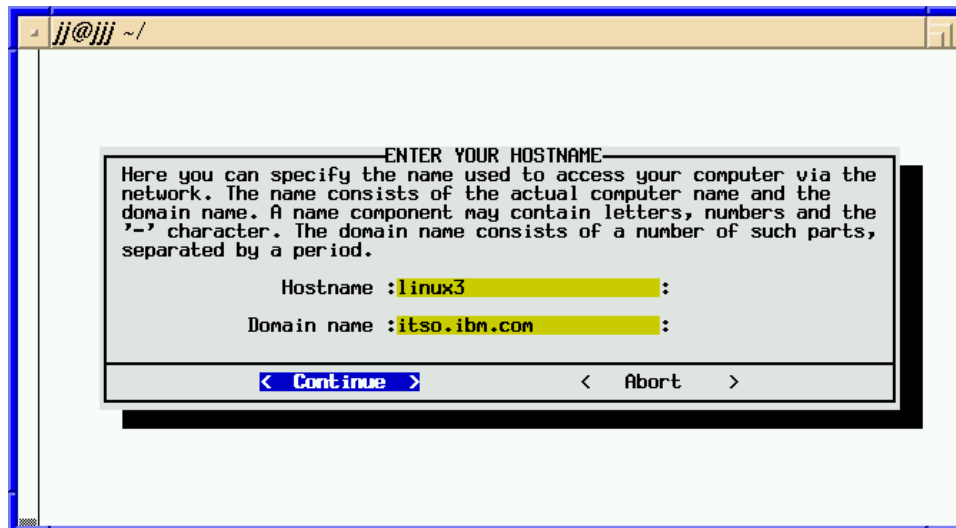


Figure 68. YaST: hostname and domain name

You should not need to make changes here.

The next menu relates to connectivity; choose **Real network**:

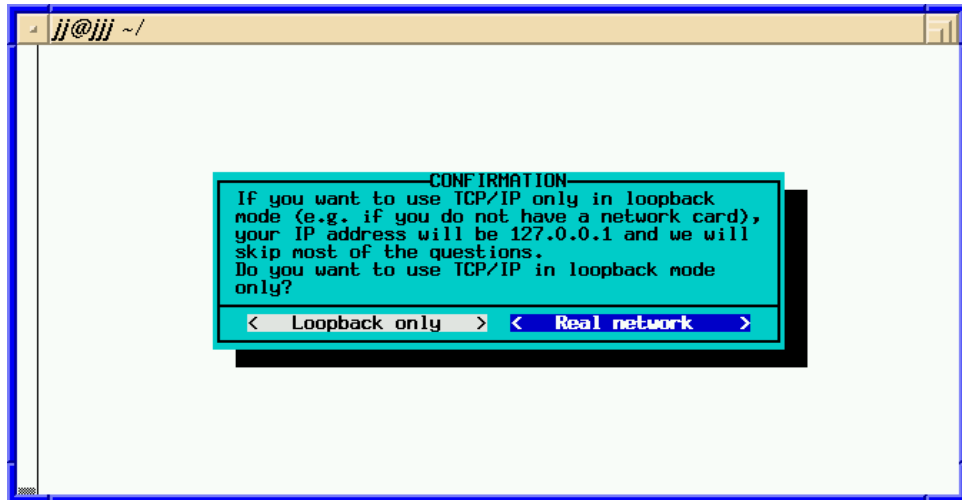


Figure 69. YaST: real network selection

Although you have the option to set up your system as a DHCP server, we did not do so:

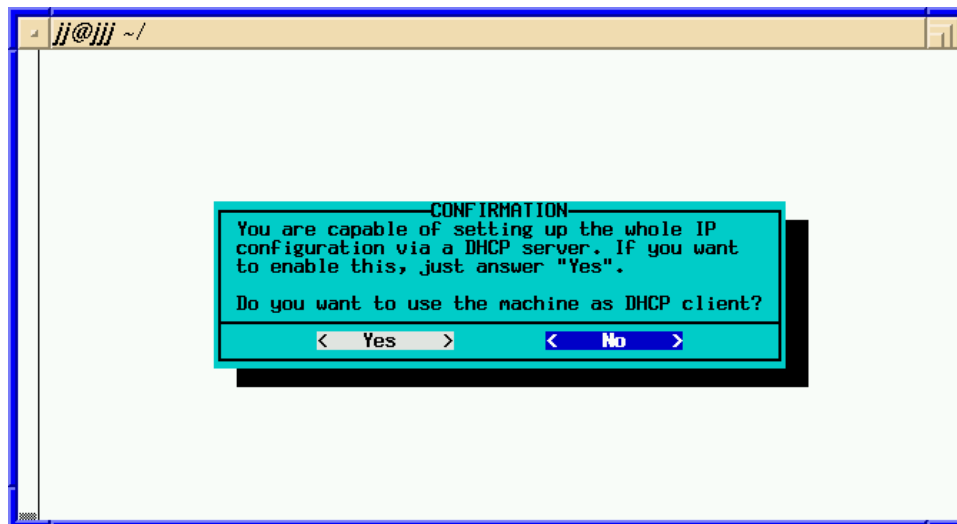


Figure 70. YaST: DHCP server choice

At the time of writing, the prerelease version did not present CTC/IUCV in the network device section for the network setup; therefore, we chose **eth0** which

will have to be fixed later as described in 7.7.1, “Finishing the install when using a CTC network device” on page 162.

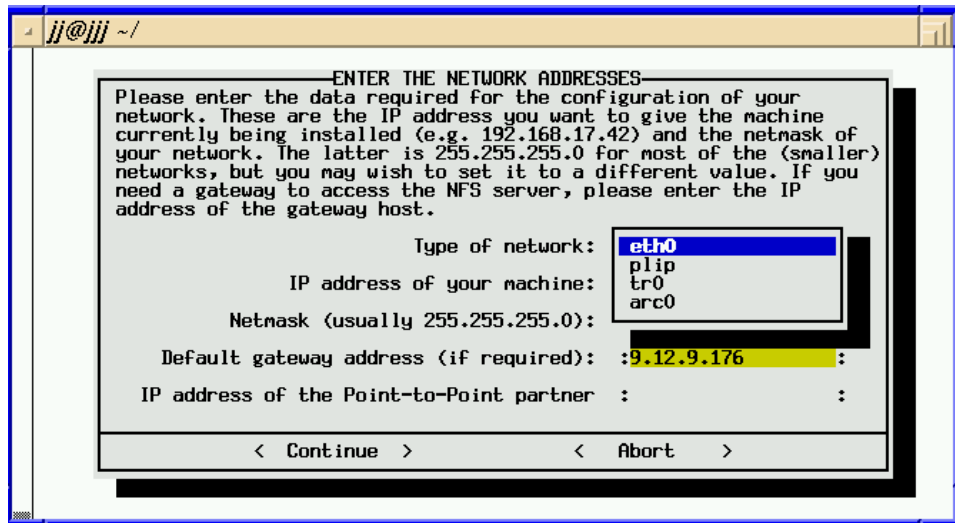


Figure 71. YaST: network devices

The address, netmask, and gateway (from the netsetup script) should be correct:

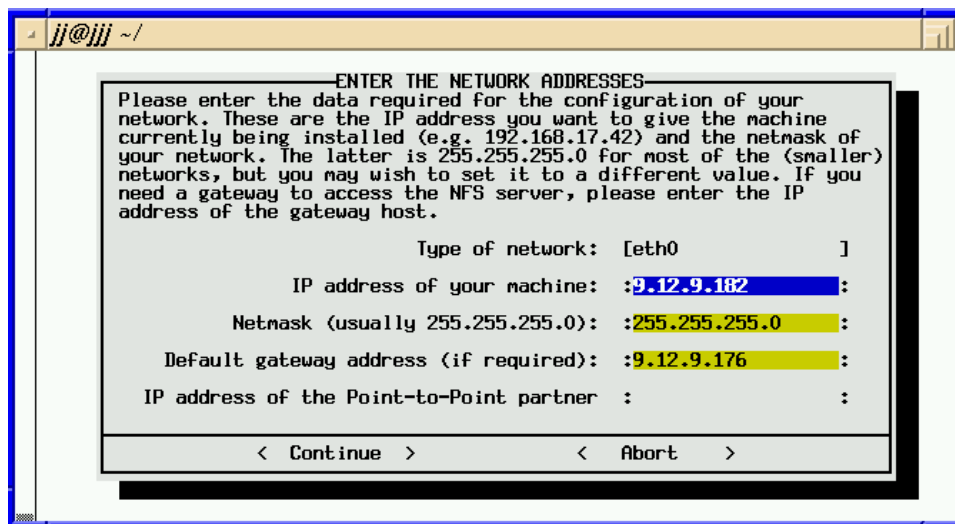


Figure 72. YaST: network addresses

We needed to start `inetd` in order to open a telnet connection later, so we chose **Yes**:

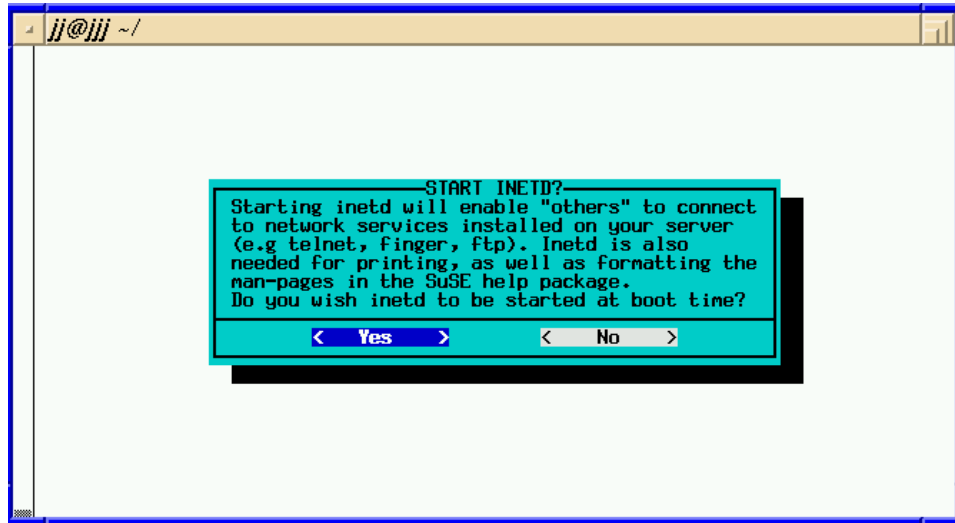


Figure 73. YaST: `inetd`

The `portmapper` is needed for NFS servers, but we decided not to set up the system as an NFS server:

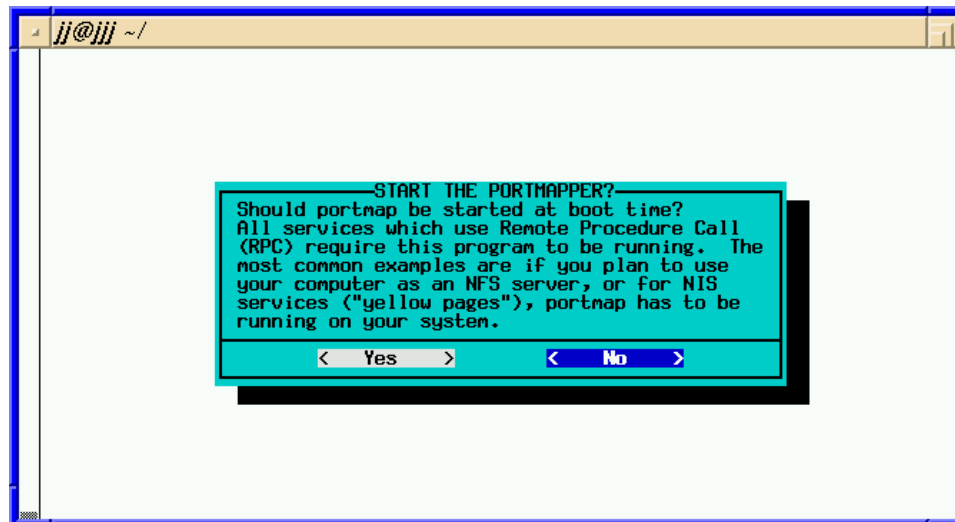


Figure 74. YaST: `portmapper`

The `nameserver` configuration has to be entered:

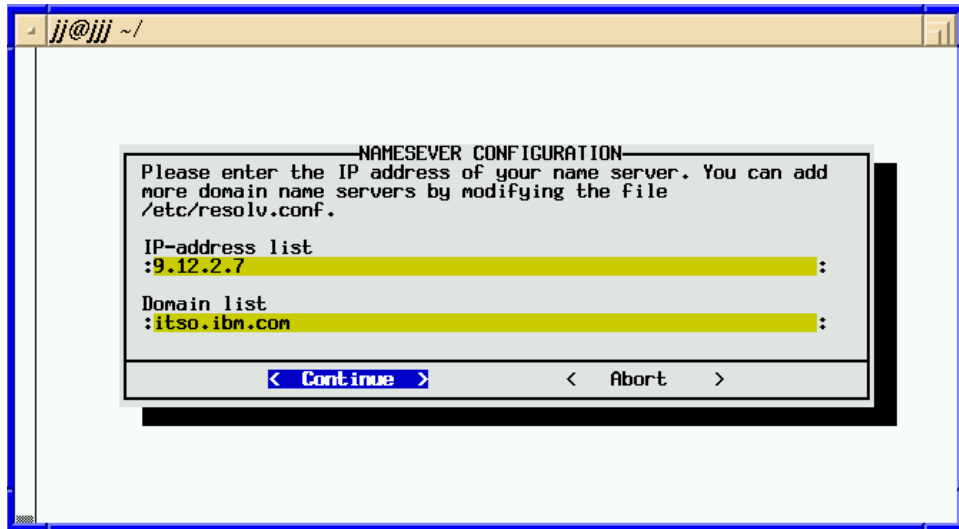


Figure 75. YaST: nameserver configuration

You can enter up to three nameserver IPs.

The following menu allows you to name a module for the networking device, as some network devices are implemented as kernel modules:

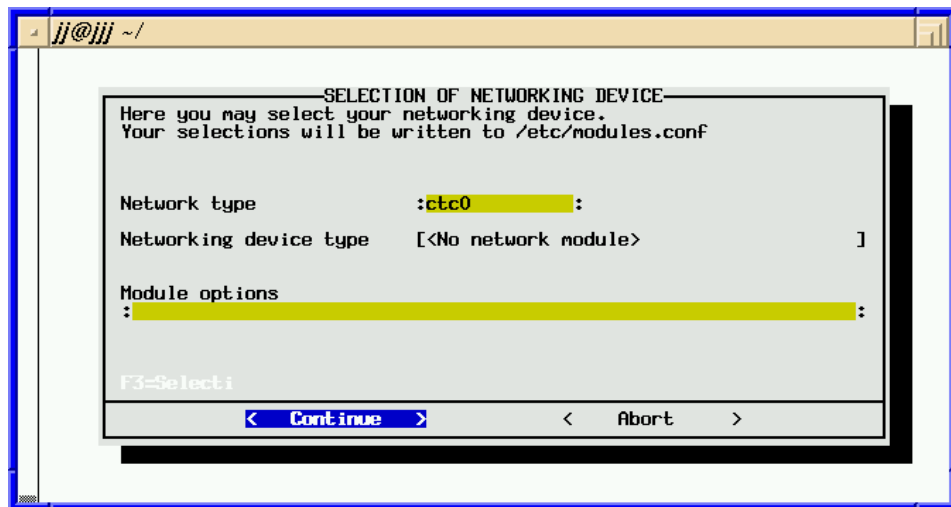


Figure 76. YaST: network device and module

A sendmail configuration can be selected from the next menu:

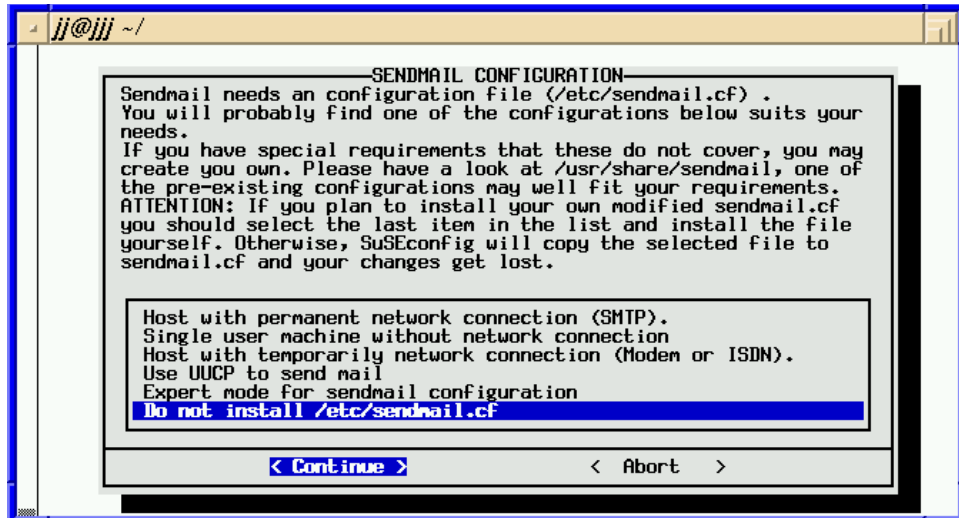


Figure 77. YaST: sendmail configuration

We decided not to have sendmail running on the system.

When all necessary steps have been finished, SuSEconfig is launched as is shown in Figure 78. This script automatically finishes the install by creating and modifying the proper files in the Linux file system.

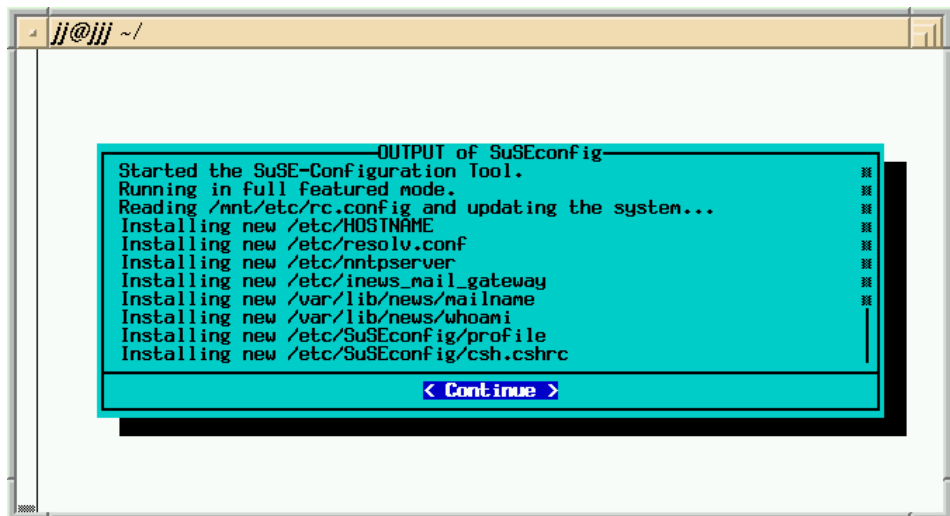


Figure 78. YaST: SuSEconfig is running

YaST will finish after this step.

7.7.1 Finishing the install when using a CTC network device

At the time of this writing, YaST cannot set up a Channel to Channel (CTC) network device, so if you are using one you have to modify the `rc.config` file on the newly created disk. This was the case in our example install.

We had to mount the newly created DASD device as YaST unmounts after finishing:

```
mount /dev/dasdb1 /mnt
```

Remember to edit `/mnt/etc/rc.config`, because this file will become `/etc/rc.config` after we re-IPL. We edited `rc.config` file on DASD:

```
vi /mnt/etc/rc.config
```

We changed the line beginning with `IFCONFIG_0=...` to the following:

```
IFCONFIG_0="9.12.9.182 pointopoint 9.12.9.176 netmask 255.255.255.0 up"
```

Note that the statement is: `pointopoint` not `pointtopoint`. The first IP is ours, and the second is that of the peer.

After saving, you can leave the editor.

Note: We sometimes experienced broken network connections with Linux (the terminal “froze” after a few minutes). In such cases, the Maximum Transfer Unit (MTU) has to be decreased; see Appendix E.22, “MTU size problems” on page 499.

You can decrease the MTU by adjusting the `mtu` parameter (ask your network administrator for the correct setting) by changing the `IFCONFIG_0` line to:

```
IFCONFIG_0="9.12.9.182 .... mtu 1492 up"
```

Unmount the DASD (note that the statement is `umount`, not `unmount`):

```
umount /mnt
```

If the `umount` fails, it is probably because you forgot to leave the mounted file system before trying to unmount it (a directory cannot be unmounted if any process has that directory as its current directory), so issue the following:

```
cd /
```

Then try the `umount` command again.

7.8 Booting the installed system

In the 3270 terminal, IPL the installed system:

```
#cp ipl 222 clear
```

After you receive the usual bootup messages, you'll see the following lines at the first bootup of a freshly installed system:

```
Started the SuSE-Configuration Tool.  
Running in full featured mode.  
Reading /etc/rc.config and updating the system...  
Executing /sbin/conf.d/SuSEconfig.groff...  
Executing /sbin/conf.d/SuSEconfig.perl...  
Executing /sbin/conf.d/SuSEconfig.sendmail...  
Executing /sbin/conf.d/SuSEconfig.susehilf...  
Executing /sbin/conf.d/SuSEconfig.susehilf.add...  
Executing /sbin/conf.d/SuSEconfig.ypclient...  
Creating /usr/share/info/dir...  
Processing index files of all manpages...  
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...
```

At this point, you should press RETURN.

Have a lot of fun!

Your SuSE Team

```
INIT: Entering runlevel: 2  
Master Resource Control: previous runlevel: N, switching to runlevel: 2  
Setting up network device ctc0 done  
Setting up routing (using /etc/route.conf) done  
Re-Starting syslog services done  
Initializing random number generator done  
Starting service httpd done  
Starting service at daemon: done  
Starting INET services (inetd) done
```

```
Starting CRON daemon done
Starting Name Service Cache Daemon done
Master Resource Control: runlevel 2 has been reached
Give root password to login: XXXX
```

In our case, we entered XXX as our root password here.

```
bash-2.04#
```

At this point, you should be logged in to your new SuSE Linux for S/390 system.

Chapter 8. Linux for S/390 bootup and shutdown

At this point we assume you have a Linux for S/390 system up and running and have seen many messages during the initial load of Linux for S/390, so now let's have a closer look at what happens at system startup and shutdown.

8.1 Linux run levels

After basic initialization, the system switches to a so-called *run level*, which means that a certain set of processes are started. Run levels 0 and 6 are reserved for system use. Run level 1 should be kept unchanged for emergency assistance. Run levels 2 to 5 have a predefined behavior, but they are open for modification by the system administrator. Table 16 lists the run levels based on the Marist file system, which are identical to most distributions.

Table 16. Run levels

Run level	Description
0	Halt the system
1	Single user mode
2	Multiuser mode without NFS
3	Multiuser mode
4	Unused
5	Multiuser mode with graphical login (X11)
6	Reboot the system

You can find the scripts executed by the rc script in the directory `/etc/rc.d/init.d/` (`/sbin/init.d/` for SuSE). One example is the `netsetup` script started at the first Linux boot to create the network parameter files like `/etc/HOSTNAME`, `/sysconfig/network` and `/sysconfig/network-scripts/ifcfg-<ip-dev>`.

These scripts, which manage startup and stoppage of certain services (typically daemons), have a common structure that defines a set of possible actions like start, stop, status, restart, and reload. The `syslog` script that starts the `syslog` daemon may serve as an example:

```
[root@linux6 init.d]# cat syslog
#!/bin/sh
#
```

```

# syslog      Starts syslogd/klogd.
#
#
# chkconfig: 2345 30 99
# description: Syslog is the facility by which many daemons use to log \
# messages to various system log files. It is a good idea to always \
# run syslog.

# Source function library.
. /etc/rc.d/init.d/functions

[ -f /sbin/syslogd ] || exit 0
[ -f /sbin/klogd ] || exit 0

RETVAL=0

# See how we were called.
case "$1" in
  start)
    echo -n "Starting system logger: "
    # we don't want the MARK ticks
    daemon syslogd -m 0
    RETVAL=$?
    echo
    echo -n "Starting kernel logger: "
    daemon klogd
    echo
    [ $RETVAL -eq 0 ] && touch /var/lock/subsys/syslog
    ;;
  stop)
    echo -n "Shutting down kernel logger: "
    killproc klogd
    echo
    echo -n "Shutting down system logger: "
    killproc syslogd
    RETVAL=$?
    echo
    [ $RETVAL -eq 0 ] && rm -f /var/lock/subsys/syslog
    ;;
  status)
    status syslogd
    status klogd
    RETVAL=$?
    ;;
  restart|reload)
    $0 stop
    $0 start

```

```

        RETVAL=$?
        ;;
*)
    echo "Usage: syslog {start|stop|status|restart}"
    exit 1
esac

exit $RETVAL

```

To get the status of the syslog service call the script with entry point `status`:

```
/etc/rc.d/init.d/syslog status
```

The following messages indicate the status:

```

syslogd (pid 308) is running...
klogd (pid 319) is running...

```

The entry points `start`, `stop`, and `restart` can be used in a similar fashion.

Symbolic links to scripts like this determine which services are started (in which order) for each run level. The separate run levels correspond to the links in the directories `/etc/rc.d/rc<runlevel>.d`. The name `Snn<script>` refers to the script called with the start (`S = start`) parameter, while `Knn<script>` refers to the script called with the stop (`K = kill`) parameter. The sequence in which to run the scripts is determined by the `nn` number in the name of the link. On our system, the links for run level 3 were as follows:

```

[root@linux6 rc3.d]# ls -l /etc/rc.d/rc3.d
total 0
lrwxrwxrwx 1 root root 16 May 2 10:57 K55routed -> ../init.d/routed
lrwxrwxrwx 1 root root 14 May 2 10:57 K80nscd -> ../init.d/nscd
lrwxrwxrwx 1 root root 17 May 2 10:57 S00netsetup -> ../init.d/netsetup
lrwxrwxrwx 1 root root 17 May 2 10:57 S10network -> ../init.d/network
lrwxrwxrwx 1 root root 17 May 2 10:57 S11portmap -> ../init.d/portmap
lrwxrwxrwx 1 root root 16 May 2 10:57 S20random -> ../init.d/random
lrwxrwxrwx 1 root root 15 May 2 10:57 S25netfs -> ../init.d/netfs
lrwxrwxrwx 1 root root 16 May 2 10:57 S30syslog -> ../init.d/syslog
lrwxrwxrwx 1 root root 14 May 2 10:57 S50inet -> ../init.d/inet
lrwxrwxrwx 1 root root 15 May 2 10:57 S85httpd -> ../init.d/httpd
lrwxrwxrwx 1 root root 13 May 2 10:57 S90xfs -> ../init.d/xfs
lrwxrwxrwx 1 root root 11 May 2 10:57 S99local -> ../rc.local

```

The script `netsetup` is called first (`S00*`), followed by `network` (`S10*`), and the last script to be executed is `rc.local` (`S99*`).

One can add script calls by adding a symbolic link; for example to add a service (started by `myscript`) that should be started as the last one in this run level, run the following:

```
ln -s /etc/rc.d/init.d/myscript /etc/rc.d/rc3.d/S95myscript
```

S99local points to a special script where the system administrator can specify all actions that do not belong elsewhere. (Not all distributions use S99local. SuSE, for example, uses script `boot.local`, which is called before the first run level is entered.)

8.2 Kernel initialization

To boot the system, the Linux for S/390 kernel is loaded into memory from either `intrad-Image` (ramdisk) or from the hard disk (`dasd`) you created with the `silos` command.

At initialization time, the kernel prints messages to the system console documenting the process. Most of the messages are saved in the system log files.

```
Linux version 2.2.15 (root@linux6) (gcc version 2.95.2 19991024
(release)) #4 SMP Tue May 17 19:45:19 EDT 2000
Command line is: mdisk=400 dasd=190,19e,19f,19d,191,200,300,400,192
root=/dev/dasdfs1 ro noinitrd ❶
We are running under VM ❷
This machine has no IEEE fpv
Initial ramdisk at: 0x00800000 (8388608 bytes)
Detected device 0009 on subchannel 0000 - PIM = 80, PAM = 80, POM = FF ❸
.....
Detected device 080A on subchannel 000E - PIM = 80, PAM = 80, POM = FF
Detected device 080B on subchannel 000F - PIM = 80, PAM = 80, POM = FF
Highest subchannel number detected: 16
SenseID :Device 0009 reports: Dev Type/Mod = 3215/00
SenseID :Device 000C reports: Dev Type/Mod = 3505/00
SenseID :Device 0190 reports: CU Type/Mod = 3990/E9, Dev Type/Mod = 3390/0A
.....
SenseID :Device 0191 reports: CU Type/Mod = 3990/E9, Dev Type/Mod = 3390/0A
SenseID :Device 0200 reports: CU Type/Mod = 3990/E9, Dev Type/Mod = 3390/0A
SenseID :Device 0300 reports: CU Type/Mod = 3990/E9, Dev Type/Mod = 3390/0A
SenseID :Device 0400 reports: CU Type/Mod = 3990/E9, Dev Type/Mod = 3390/0A
SenseID :Device 0192 reports: CU Type/Mod = 3990/E9, Dev Type/Mod = 3390/0A
SenseID :Device 080A reports: Dev Type/Mod = 3088/08
SenseID :Device 080B reports: Dev Type/Mod = 3088/08
dasd: added dasd range from ..... ❹
Calibrating delay loop... 147.05 BogoMIPS ❺
Memory: 119424k/131072k available (1052k kernel code, 4k reserved, 2400k
data, 0k init)
Dentry hash table entries: 16384 (order 5, 128k)
Buffer cache hash table entries: 131072 (order 7, 512k)
Page cache hash table entries: 32768 (order 5, 128k)
POSIX conformance testing by UNIFIX
Detected 1 CPU's
Boot cpu address 0
cpu 0 phys_idx=0 vers=FF ident=0D0822 machine=9672 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
TCP: Hash tables configured (ehash 131072 bhash 65536)
Starting kswapd v 1.5
RAM disk driver initialized: 16 RAM disks of 8192K size
mnda: register device at major 5F with 179803 blocks 4096 blksize ⑥
mndb: register device at major 5F with 179803 blocks 4096 blksize
loop: registered device at major 7
dasd:initializing...
dasd(eckd):3390/a (3990/1) Cyl: 107 Head: 15 Sec: 224
dasd(eckd):Estimate: 58786 Byte/trk 2074 byte/kByte 33 kByte/trk
dasd(eckd):Verified: 58786 B/trk 5202 B/Blk(4096 B) 12 Blks/trk 48
kB/trk
dasd:77040 kB <- 'soft'-block: 4096, hardsect 4096 Bytes
dasd:devno 190 added as minor 0 (ECKD)
.....
Partition check: ⑦
dasda:(nonl)/      : dasda1
dasdb:(nonl)/      : dasdb1
dasdc:(CMS1)/Z-DISK: dasdc1(CMS)
dasdd:(CMS1)/HELP! : dasdd1(CMS)
dasde:(CMS1)/LIN191: dasde1(CMS)
dasdf:(nonl)/      : dasdf1
xpraminfo:initializing:
xpraminfo: number of devices (partitions): 1
xpraminfo: size of partition 0 to be set automatically
xpraminfo: hardsector size: 4096B
xpraminfo: 20480 kB expanded memory found.
xpraminfo: automatically determined partition size: 20480 kB
channel: 2 Parallel channel found - 0 ESCON channel found
ctc0: read dev: 080a irq: 000e - write dev: 080b irq: 000f ⑧
VFS: Mounted root (ext2 filesystem) readonly. ⑨
Freeing unused kernel memory: 0k freed
INIT: version 2.74 booting ⑩

```

- ① Display of the current active kernel parameters from the parameter file. These parameters are:
- root= Device corresponding to the root file system. The parameter `ro` (read-only) tells that the root file system should be mounted read-only for file system check.

Later, during system startup, it is remounted read-write. `Noinitrd` has to be specified when the kernel is compiled with ramdisk support, but there is no ramdisk. Here are two examples:

```
/dev/ram0 ro points to the ramdisk.
```

```
/dev/dasda ro noinitrd points to a disk.
```

`mdisk=` minidisks that are associated to the system in the order of device node assignment. For more information on minidisk, see 9.1.2, “VM minidisk” on page 180.

`dasd=` DASD devices that are associated to the system in order of device node assignment

A complete list of parameter options and restrictions is documented in Appendix D, “The parameter file” on page 481.

- ❷ Detection of the environment Linux for S/390 is running in, which can be VM, native, or LPAR.
- ❸ Detection of devices and device characteristics by subchannel.
- ❹ Honoring the `dasd=` specifications from the kernel parameter.
- ❺ Measurement of processor speed which is used to calculate delay loops for several device drivers; in a shared environment like S/390, you may see different BogoMIPS, depending on the workload within the S/390 complex.
- ❻ Associate Linux device node, major number and minor number to the block devices, `mnd` = minidisk, `dasd` = DASD devices.
- ❼ Display of all devices specified by the `dasd=` kernel parameter. If these devices are formatted and reserved by VM/CMS, the type and volume information is displayed. Devices formatted and reserved by CMS will be displayed as:

```
dasd<letter>:(CMS1)/<volid>      : dasd<letter>1 (MDSK)
```

Devices formatted by CMS but not reserved will be displayed as:

```
dasd<letter>:(CMS1)/<volid>      : dasd<letter>1 (CMS)
```

`xpram*` messages show information about expanded storage used with the `xpram` device driver. For more information, see 9.1.3, “XPRAM” on page 181 in this chapter.

- ❽ Initialization of the network devices like `ctc`, `tr`, or `eth`.

- ⑨ Mount the root file system as specified in the kernel parameter `root=` statement and free up some of the memory used by the memory-loaded kernel.
- ⑩ Enter the init process basic startup; also refer to 8.3 on page 171.

The kernel initialization ends here and should not be modified. The next step in bringing up the system is the init process.

8.3 The init process and run level

The init process is always the first process started by the kernel. The following messages appear after entering the init process at boot time:

```
INIT: version 2.74 booting ①
Starting lcs module ②
No lcs capable cards found
/lib/modules/2.2.14/net/lcs.o: init module: Device or resource busy
Mounting proc filesystem [ OK ] ③
/etc/rc.d/rc.sysinit: /proc/sys/kernel/sysrq: No such file or directory
unrecognized option `-S'
Setting clock (srm): Thu May 12 18:42:04 EDT 2000 [ OK ]
Activating swap partitions swapon: warning: /dev/dasdl has insecure
permissions 0644, 0600 suggested
swapon: cannot stat /mnt/swap/swapfs1: No such file or directory ④
swapon: cannot stat /mnt/swap/swapfs2: No such file or directory
[FAILED]
Setting hostname linux6 [ OK ]
Checking root filesystem
/dev/dasdf1: clean, 31367/225344 files, 163545/449997 blocks
[ OK ]
Remounting root filesystem in read-write mode [ OK ]
Finding module dependencies [ OK ]
Checking filesystems
/dev/dasdg1: clean, 13/27008 files, 23918/26997 blocks
[ OK ]
Mounting local filesystems [ OK ]
Enabling swap space [ OK ]
INIT: Entering runlevel: 3 ⑤
Entering non-interactive startup
Bringing up interface lo [ OK ]
Bringing up interface ct0 [ OK ]
Starting portmapper: [ OK ]
Initializing random number generator [ OK ]
Mounting other filesystems [ OK ]
Starting system logger: [ OK ]
```

```
Starting kernel logger: [ OK ]
Starting INET services: [ OK ]
Starting httpd: [ OK ]
Starting X Font Server: [ OK ]
Give root password for maintenance
(or type Control-D for normal startup):
```

- ❶ Start the `init` process which runs `rc.sysinit`; at this point the boot process switches to a point where it can be customized.
- ❷ Load the lan channel station (`lcs`) driver module for OSA-card enablement; this is done with the `insmod` or `modprobe` command.
- ❸ More basic settings are made as defined in the shell script `rc.sysinit`, which executes tasks like clock setting and file system checking and mounting. See 8.3.2, “Basic system initialization” on page 174 for more information.
- ❹ Startup swap devices (swap files and swap devices as listed in `/etc/fstab`). See 9.3, “Linux swap space” on page 188 for more information.
- ❺ Enter run level 3, described in more detail in 8.1, “Linux run levels” on page 165.

8.3.1 System `init` and `inittab`

The kernel starts the `init` process (which always has process ID 1), which first reads `/etc/inittab`. This file describes which processes are started at bootup and which processes should be started and or restarted after termination of a running Linux environment. The syntax for each line is specified in the fixed format `code:runlevel:action:command`.

<code>code</code>	A 2 to 4 character identifier (usually only 2).
<code>run level</code>	Specifies by run level number where this command is executed.
<code>action</code>	Determines what to do. Possible actions are: <code>initdefault</code> Sets default run level <code>sysinit</code> Basic system init that runs only once at boot time <code>respawn</code> Restart the command if terminated <code>wait</code> Wait for completion before doing anything else
<code>command</code>	The command to execute.

Lines that begin with a # (pound sign or hash) are comments and are not processed. Following is an example of the inittab:

```
# inittab          This file describes how the INIT process should set up
#                  the system in a certain run-level.
#
# Author:          Miguel van Smoorenburg, <miguels@drinkel.nl.mugnet.org>
#                  Modified for RHS Linux by Marc Ewing and Donnie Barnes

id:3:initdefault: ❶

# System initialization.
si::sysinit:/etc/rc.d/rc.sysinit ❷

10:0:wait:/etc/rc.d/rc 0 ❸
11:1:wait:/etc/rc.d/rc 1
12:2:wait:/etc/rc.d/rc 2
13:3:wait:/etc/rc.d/rc 3
14:4:wait:/etc/rc.d/rc 4
15:5:wait:/etc/rc.d/rc 5
16:6:wait:/etc/rc.d/rc 6

# Things to run in every runlevel.
ud:once:/sbin/update

# Run gettys in standard runlevels ❹
1:2345:respawn:/sbin/sulogin /dev/console
#1:2345:respawn:/sbin/mingetty tty1
#2:2345:respawn:/sbin/mingetty tty2
#3:2345:respawn:/sbin/mingetty tty3
#4:2345:respawn:/sbin/mingetty tty4
#5:2345:respawn:/sbin/mingetty tty5
#6:2345:respawn:/sbin/mingetty tty6

# Run xdm in runlevel 5
# xdm is now a separate service
#x:5:respawn:/etc/X11/prefdm -nodaemon
```

❶ Default run level line

You can switch a running system to another run level by issuing the `init` command as follows:

```
init 1
```

This would bring the system to run level 1. The command `runlevel` shows the current and previous run level:

```
[root@linux6 /root]# runlevel
N 3
```

The current run level is 3, and N indicates that no run level was active before.

❷ System initialization line

`rc.sysinit` is a shell script (in Linux for S/390) to handle basic system initialization such as activating modules for the lcs device driver,

synchronizing the system clock and mounting file systems. See 8.3.2, “Basic system initialization” on page 174 for a more detailed description.

3 Set run levels

The script `/etc/rc.d/rc` executes other scripts as appropriate for the specified run level. The run level scripts are found in the directory `/etc/rc.d/rc<runlevel>.d/`. This may be different for other distributions (e.g. `/sbin/init.d/` for SuSE).

4 Provide a virtual console

Normally on Linux, six or more *virtual consoles* are started. This enables the system administrator to hop between sessions by pressing Alt-F_n (where *n* is the console number). However, due to the limited capability of the 3215 data stream, there is only one virtual console on Linux for S/390. The `respawn` action makes sure that this virtual console is restarted if the `sulogin` process has died for some reason.

8.3.2 Basic system initialization

For basic system initialization `init` calls `rc.sysinit` (`/sbin/init.d/boot` for SuSE) as specified in the `sysinit` row of `/etc/inittab`. This shell script activates the necessary modules like the `lcs` device driver, synchronizes the system clock with the underlying layer (TOD or simulated by VM), and checks and mounts the file systems. Our `rc.sysinit` looked like this:

```
[root@linux6 init.d]# cat rc.sysinit
. /etc/sysconfig/network
# Source functions. /etc/rc.d/init.d/functions
insmod /lib/modules/2.2.14/net/lcs.o
# Print a banner. ;)
action "Mounting proc filesystem" mount -n -t proc /proc /proc
. /etc/sysconfig/clock
action "Activating swap partitions" swapon -a
action "Setting hostname ${HOSTNAME}" hostname ${HOSTNAME}
# Set the NIS domain name if [ -n "$NISDOMAIN" ]; then action "Setting NIS domain name
  $NISDOMAIN" domainname $NISDOMAIN
if [ -f /fsckoptions ]; then fsckoptions=`cat /fsckoptions`
STRING="Checking root filesystem" initlog -c "fsck -T -a $fsckoptions /"
# Add /proc to /etc/mtab mount -f -t proc /proc /proc
# Enter root and /proc into mtab., mount -f / , mount -f /proc
# Mount all other filesystems (except for NFS and /proc, which is already# mounted).
action "Mounting local filesystems" mount -a -t nonfs,smbfs,ncpfs,proc
# Configure machine if necessary.
if [ -x /usr/bin/passwd ]; then /usr/bin/passwd root
if [ -x /usr/sbin/netconfig ]; then /usr/sbin/netconfig
if [ -x /usr/sbin/timeconfig ]; then /usr/sbin/timeconfig
if [ -x /usr/sbin/authconfig ]; then /usr/sbin/authconfig --nostart
if [ -x /usr/sbin/ntsysv ]; then /usr/sbin/ntsysv --level 35
# Reread in network configuration data. . /etc/sysconfig/network
# Reset the hostname. action "Resetting hostname ${HOSTNAME}" hostname ${HOSTNAME}
# Reset the NIS domain name. action "Resetting NIS domain name $NISDOMAIN" domainname
  $NISDOMAIN
# Delete X locks, rm -f /tmp/.X*-lock
```

```
# Right, now turn on swap in case we swap to files. swapon -a >/dev/null 2>&1
cat > /boot/kernel.h << EOF make kernel headers
# Now that we have all of our basic modules loaded and the kernel going,
# let's dump the syslog ring somewhere so we can find it later
dmesg > /var/log/dmesg
```

8.4 Shutdown

To bring the system down properly, you issue the `shutdown` command. This sets the run level to 0. During shutdown, the file system buffers are written to the disk, the file systems are marked as clean and finally unmounted. Improper shutdown (like simply deactivating via the HMC) would leave the file systems marked as not clean and cause a file system check at the next reboot. The commands `shutdown -r <time>` or `reboot` enter run level 6 to halt, and then restart, the system.

With the `shutdown` command, one must always specify the time (now, +mm, time of day) at which to halt the system. For example:

```
shutdown -h now           Halt the system immediately
shutdown -h +10          Halt the system in 10 minutes
shutdown -h 13:00       Halt the system at 1:00 pm
```

If you want to cancel the shutdown before the process has started, you can issue a `shutdown -c` command.

Following is an example of shutdown messages sent to the console:

```
[root@linux6 /root]# shutdown -h now
INIT: Switching to runlevel: 0 ❶
INIT: Sending processes the TERM signal
INIT: Sending processes the KILL signal ❷
Shutting down X Font Server: [ OK ]
Shutting down http: [ OK ]
Stopping INET services: [ OK ]
Saving random seed [ OK ]
Stopping portmap services: [ OK ]
Shutting down interface ctc0 [ OK ] ❸
Disabling IPv4 automatic defragmentation [ OK ]
Shutting down kernel logger: [ OK ]
Shutting down system logger: [ OK ]
Starting killall [ OK ]
Sending all processes the TERM signal...
Turning off swap ❹
Unmounting filesystems
Unmounting proc filesystem
```

- ❶ The `init` process switches to run level 0. The processes that are running receive a `TERM` signal, which should bring them down gracefully.
- ❷ If there are processes still running, they are sent the `KILL` signal.

- ③ The network devices and kernel modules are deactivated.
- ④ The swap spaces are deactivated and the file systems are unmounted.

Chapter 9. Linux for S/390 administration

The first part of this chapter describes devices (including DASD, minidisk, XPRAM and swap space) and the tasks you need to perform to make them ready for use within a Linux for S/390 system. The second part lists tasks to administer the usage of the system (like user administration, shells, and basic security settings).

9.1 Devices

An important concept to remember is: *to Linux, everything is a file.*

The hardware (such as disks and printers) attached to the system and many software mechanisms (e.g. the virtual garbage can, `/dev/null`) are represented as *devices* within Linux. The devices correspond to special files (device files or device nodes) usually found in the directory `/dev/`.

With this layer of abstraction, it is possible to operate on everything using file operations. As an example, one does not need special commands to fill the entire disk represented by `/dev/dasdf` with zeros; instead, the following simple statement does it:

```
dd if=/dev/zero of=/dev/dasdf
```

The device `/dev/zero` is the Linux source of bytes with the value zero.

The preceding, rather destructive, command may seem somewhat useless, as it does not leave anything useful on the whole disk. However, in situations where data has to be physically erased (for security reasons), this is the way to do it. Removing a file only unlinks it from its inode; the contents are still somewhere on the disk surface.

There are two types of devices, block devices (block-oriented operation) and character devices (character-oriented operation). With block devices (most notably disks) the data transfer happens in multiples of their block size. Character devices transfer data in units of bytes.

Devices have a *major number* and a *minor number*. The major number identifies the device type and the minor number identifies the unit or instance of that device.

The file `/proc/devices` can be used to get a list of device types supported by the kernel; the output is a list of major numbers followed by the base name of the appropriate device file:

```
[root@linux6 /proc]# cat /proc/devices
Character devices:
 1 mem
 2 pty
 3 ttyp
 4 ttyS
 5 ptmx
10 misc
128 ptm
136 pts
```

```
Block devices:
 1 ramdisk
 7 loop
35 xpram
94 dasd
95 mnd
```

To work with anything via its device, the following three conditions must be met:

1. If it is a piece of hardware, it must be attached.
2. The Linux kernel must support it. With Linux, the kernel device drivers can be compiled either into the kernel binary itself or compiled as modules that can be loaded after boot.
3. The appropriate device file must be present. The type of the device is determined by its major number. The minor number typically selects one particular device of several identical devices, or a certain part of one device (like a partition on a disk).

The major and minor numbers are listed in the file `/usr/src/linux/Documentation/devices.txt`.

The major and minor numbers of the device files are listed by simply using the `ls -l` command in the `/dev` directory:

```
brw-r--r-- 1 root root 94, 0 Feb 20 09:14 dasda
brw-r--r-- 1 root root 94, 1 Feb 20 09:14 dasda1
brw-r--r-- 1 root root 94, 4 Feb 20 09:14 dasdb
brw-r--r-- 1 root root 94, 5 Feb 20 09:14 dasdb1
brw-r--r-- 1 root root 94, 8 Feb 20 09:14 dasdc
brw-r--r-- 1 root root 94, 9 Feb 20 09:15 dasdc1
```

Note the `b` in the first column of the permission bits; this indicates that the file represents a block device.

Due to the great importance of the DASD, minidisk, and XPRAM devices, we give an overview of these in Table 17 and dedicate the next few sections to describing them.

Table 17. Characteristics of S/390 block devices

Device type	Format method	Device node (x = a-z)	Partition able	Boot record/special considerations
Native DASD or hardware emulated DASD	Linux <code>dasdfmt</code> command	<code>/dev/dasd<n></code>	yes	Device must be partitioned to contain boot record, kernel parameter <code>dasd=</code> or autodetected.
VM minidisk (DASD driver)	CMSFORMAT and RESERVE or Linux <code>dasdfmt</code>	<code>/dev/dasd<n></code>	yes	Kernel must be recompiled to enable <code>dasd_force_diag</code> support.
VM minidisk (minidisk driver)	CMSFORMAT and RESERVE	<code>/dev/mnd<n></code>	no	Default for the Marist distribution 2.2.15.
XPRAM	No format needed	<code>/dev/xpram<n></code>	no	File system must have a block size of 4K (or multiple) on this device.

Note: Although not normally necessary, you can use the Linux `dasdfmt` command on a minidisk, but it must be formatted by CMS first.

9.1.1 DASD (direct access storage device)

The Linux for S/390 DASD driver manages all S/390 disk devices as block devices with a major number of 94. By default, Linux for S/390 probes all available disk devices (auto-detects) at startup time. Additionally you can select specific devices using the `dasd=` kernel parameter.

The device itself is separated into 4 addressable areas that are associated with minor numbers. The first minor number represents the physical volume, while the other 3 are reserved for the partitions, as shown in the following example.

- The first DASD gets a major number of 94 and starts with minor 0, so:
 - minor 0 = address physical device
 - minor 1 = first partition
 - minor 2 = second partition (future design)
 - minor 3 = third partition (future design)
- The second DASD gets a major number of 94 and starts with the next free minor number to get a unique major/minor number associated:

minor 4 = address physical device
minor 5 = first partition
minor 6 = second partition (future design)
minor 7 = third partition (future design)

As you only have one major number for DASD devices, theoretically a maximum of 64 DASDs can be associated to the system.

At the moment a device can only contain one partition per volume, but this may be enhanced in the future. The location of the first partition is behind a reserved area for the boot record. This record is needed for the initial load of the Linux kernel.

Linux accesses the devices through the `/dev/dasd<n>` device node, the partition on the device through `/dev/dasd<n>1`. We discuss device nodes in more detail later in this chapter.

9.1.2 VM minidisk

Under VM you can partition a real disk volume into minidisks. This gives great flexibility to the size of devices. As you can see from Table 17 on page 179, minidisks can be low level-formatted in two different ways for use by Linux for S/390:

1. By using the Linux `dasdfmt` command (not accessible as a CMS file).
 - This is not recommended for minidisks.
 - `dasdfmt` will destroy any CMS label/volid information.
2. By using CMS `format` and the `reserve` command (accessible as a CMS file).
 - This is recommended for minidisks.
 - CMS label/volid information is preserved, provided the file system is created on a Linux device partition.

To format and reserve a minidisk, use the CMS commands:

```
format <device number> <filemode> (blksize <blksize>
reserve linux mdisk <filemode>
```

When enabled in the kernel, as a general rule you should specify the `dasd_force_diag=` kernel parameter in combination with the `dasd=` parameter for minidisks.

The old minidisk driver (before 2.2.15)

To specify that you want a minidisk managed by the old Linux for S/390 VM minidisk device driver, use the kernel parameter `mdisk=`.

Linux associates device nodes (names) `/dev/mnd<letter>` with the devices and assigns a major number of 95. If you have the 2.2.15 level of the kernel and have been using the `mdisk=` kernel parameter, you might consider using `dasd=` and `dasd_force_diag=` for minidisk devices. This necessitates recompiling the kernel.

The DASD driver

This is the same driver as for the type 2 minidisks. Use the `dasd_force_diag=` kernel parameter in combination with the `dasd=` parameter. Linux associates device node `/dev/dasd<letter>` with the devices and the major number 94.

The file `/proc/dasd/devices` lists both CMS-formatted and Linux-formatted devices. However, apart from referring to the boot messages, there is no way to detect if the device was formatted by CMS! Remember, if you use the Linux `dasdfmt` command, it will destroy any VM/CMS information.

Refer to Appendix E.20, “minidisk.sh” on page 494 for the `minidisk.sh` script that creates a file like `/proc/dasd/devices` but includes the original CMS label information.

9.1.3 XPRAM

S/390 hardware usually has *expanded storage*. This is not as fast as central storage, but is much faster than accessing DASD; therefore, it is great for swap space. Expanded storage has no analogy on the PC.

Although Linux addresses only about 2 GB of memory, you can access expanded storage as file system or swap space under Linux. The `xpram` driver will map a file system or a swap space into the expanded storage.

A `xpram` device has major number 35 and can be partitioned, starting with minor number 0 for the first partition. You can have up to 32 partitions. The associated device node is `/dev/xpram<letter>`. `Xpram` devices can be assigned by kernel or module parameters for use by the `insmod` or `modprobe` commands (specified in `/etc/modules.conf` or in `rc.sysinit`).

Following is an explanation of the `xpram` parameter:

- The `xpram` kernel parameter is as follows:

```
xpram_parts=<number_of_partitions>[,size[,...]]*
```

Number_of_partitions can contain a number from 1 to 32, and the default is 1.

The size specification of the kernel parameter is divided in three parts: <hex/decimal><number><unit>. For the first part, the value 0x means the second part will be a hexadecimal value.

For the third part, the unit can be k or K for kilobyte, m or M for megabyte, and g or G for gigabyte. A size of 0 requests the driver to allocate the rest of expanded storage that is available. See the following examples:

xpram_parts=1,0x200m Refers to one partition (/dev/xpram0) with a size of (hex) 200 megabytes. In decimal, that is 512 MB

xpram_parts=2,200m,1g Refers to two partitions, one with a size of 200 MB (/dev/xpram0) and a second with a size of 1 GB (/dev/xpram1)

- The xpram module parameter is as follows:

```
devs=<number_of_partitions>,sizes=<size>[,size....]
```

Number_of_partitions can contain a number from 1 to 32, and the default is 1.

The size for the module parameter can only contain decimal numbers. The specification of a unit is not allowed. The size is interpreted as kilobyte specification.

At boot time, the xpram driver auto-detects the expanded storage and associates it with the /dev/xpram0 node (up to 2 GB) if there is no kernel parameter active. Following is an example of some xpram driver output:

```
xpraminfo:initializing:
xpramdebug: major 35
xpraminfo: number of devices (partitions): 1
xpraminfo: size of partition 0 to be set automatically
xpramdebug: memory needed (for sized partitions): 0 kB
xpramdebug: partitions to be sized automaticallys: 1
xpraminfo: hardsector size: 4096B
xpraminfo: 20480 kB expanded memory found.
xpraminfo: automatically determined partition size: 20480 kB
```

9.1.4 Creating a device node with mknod

Device nodes are associated with a major number which identifies the device type and a minor number which identifies the unit of a device. They allow programs to access hardware devices through the kernel device drivers. All

device nodes are located in the directory /dev. Table 18 gives an overview of the block devices with their numbers in Linux for S/390.

Table 18. Device node characteristics for S/390 devices

Device node (n = 1 letter)	Major number	Minor number	Maintained in directory
/dev/dasd<n>	94	4 numbers for each disk	/proc/dasd/devices
/dev/xpram<n>	35	0-31	nowhere; look in startup messages with command <code>dmesg</code>

Use the `mknod` command (as superuser) to create a new device node. Following is the syntax of the command

```
mknod [-m permissions] name type(c=char,b=block) major minor
```

The permission option `-m` is optional, but consider what permissions are really needed on the device. Incorrectly set permissions may give unwanted access to a device (through raw reads on the device node), bypassing file security. A recommended value is 600.

For example, to create a device node for the ninth DASD device, issue the command:

```
mknod /dev/dasdi b 94 32
```

For an explanation on the association of minor numbers, see 9.1.1 on page 179.

9.1.5 Linux for S/390 device node assignment

For all examples in this chapter we use the device node-dependent messages from Chapter 8, “Linux for S/390 bootup and shutdown” on page 165.

The kernel parameters influence the association of device nodes. This could be important, as your file systems (including the root file system) are accessed through a device node mapping (e.g. `root=/dev/dasda1`).

When no DASD kernel parameters are specified, the system will auto-detect the DASD devices. The order is determined by subchannel; see the extract of the boot messages:

```
Detected device 0190 on subchannel 0004 - PIM = F0, PAM = F0, POM = FF
Detected device 0191 on subchannel 0008 - PIM = F0, PAM = F0, POM = FF
Detected device 0200 on subchannel 0009 - PIM = F0, PAM = F0, POM = FF
Detected device 0300 on subchannel 0009 - PIM = F0, PAM = F0, POM = FF
```

Detected device 0400 on subchannel 0009 - PIM = F0, PAM = F0, POM = FF
 Detected device 0192 on subchannel 000C - PIM = F0, PAM = F0, POM = FF

Table 19. device node association with auto detect

Unit address	Expected device node depending on device number sequence	Device node assignment through auto detect
190	/dev/dasda (minor 0)	/dev/dasda (minor 0)
191	/dev/dasdb (minor 4)	/dev/dasdb (minor 4)
192	/dev/dasdc (minor 8)	/dev/dasdf (minor 20)
200	/dev/dasdd (minor 12)	/dev/dasdc (minor 8)
300	/dev/dasde (minor 16)	/dev/dasdd (minor 12)
400	/dev/dasdf (minor 20)	/dev/dasde (minor 16)

In this example, based on the unit addresses, the conclusion can be drawn that a device node association will be as shown in the “Expected device node...” column of Table 19. As you can see from the boot messages above, the system uses a different method for assigning device nodes based on the subchannel number when auto detecting the devices. See the Table 19 column entitled “Device node assignment through auto detect” for the result.

The file /proc/dasd/devices reports the DASD device numbers with their associated major and minor numbers:

```
# cat /proc/dasd/devices
dev# MAJ  minor node      Format
0190 94    0  /dev/dasda  n/a
0191 94    4  /dev/dasdb  n/a
0200 94    8  /dev/dasdc  4096
0300 94   12  /dev/dasdd  4096
0400 94   16  /dev/dasde  4096
0192 94   20  /dev/dasdf  4096
```

Imagine you want to format the device with unit address 192 without first looking in the /proc/dasd/devices file. You would specify the device node /dev/dasdc since the device is the third DASD in the device number sequence, but this will format the device 200 instead!

To avoid this and have the DASD ordered correctly, you can specify the kernel parameter `dasd=`. The order of appearance in the `dasd=` parameter line(s) determines the assignment of the device nodes.

```
dasd=190-192,200,300,400
```

The previous parameter line example will result in a device node assignment as shown in the column entitled “Expected device node...” in Table 19 on page 184.

Note

When you specify a DASD range like `dasd=190-400` in the parameter line, a set of minor numbers is reserved for each device (up to the maximum minor numbers of 64) regardless of whether the device exists.

Therefore, the system will reserve minor numbers for devices 190 - 22F and then print a message saying that the specified range is too large, and ignore devices 230-400 for minor number assignment.

From our experience with the above scenarios, we recommend that you specify the device containing the root file system first in the `dasd=` kernel parameter, to avoid having an invalid root file system when adding additional DASDs.

9.2 File system types

Linux supports various kinds of file systems, among them:

- | | |
|-----------------------|---|
| <code>ext2</code> | A second extended file system, mainly developed for Linux, the de facto standard Linux file system. |
| <code>reiserfs</code> | A journalled file system (all changes to the file system are logged). Though considered superior to a conventional file system, this file system is not commonly used. |
| <code>JFS</code> | Journalled File System (JFS) provides fast file system restart in the event of a system crash. Using database journaling techniques, JFS can restore a file system to a consistent state in a matter of seconds or minutes, versus hours or days with non-journalled file systems |
| <code>nfs</code> | The network file system, the de facto standard for accessing remote file systems over the network (see Chapter 19, “Network File System (NFS)” on page 367). |
| <code>swap</code> | The swap “file system”, used to page out currently unused memory pages. This file system is not mounted. |

- procfs A virtual file system (it actually exists in memory) where the kernel provides system information accessible with file operations.
- smbfs The Samba file system, allows file sharing with windows clients.

With Linux for S/390 you will most likely use the ext2 or the reiser file system. The limitations of the ext2 file system are listed in Table 20.

Table 20. Basic characteristics of the ext2 file system

Maximum file system size	4 terabytes
Maximum file size	2 gigabytes
Maximum file name length	255 characters
Default inode allocation	1 inode per 4096 bytes

More detailed information on the ext2 file system can be found on the Web at:

<http://web.mit.edu/tytso/www/linux/ext2intro.htm>

File systems read and write operations are buffered in storage. Invoking the `sync` command forces the system to physically write all file data to the device. Normally this automatically happens periodically, and also during unmount of a file system.

9.2.1 Block size relation between device and file system

During formatting of the device, a block size is specified. For example, the following command specifies a block size of 4098 bytes:

```
dasdfmt -b 4096 -f /dev/dasd<n>
```

The block size of the file system specified during `mke2fs` should be larger than or equal to the block size given to the `dasdfmt` command. If you `format` with block size 4 k and create an ext2 file system with block size 1 k, you can address (from a file system point of view) 1 k, but this 1 k will be stored on the device in a 4 k block, leaving 3 k unaddressable.

Figure 79 illustrates this scenario.

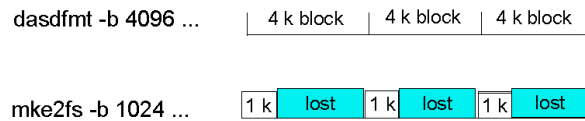


Figure 79. Block usage w/ different block sizes, DASD format & file system make

According to our experience we recommend that you specify 4 k as the block size when formatting devices.

9.2.2 The file system table `/etc/fstab`

The file `/etc/fstab` contains information about the file system type that some devices contain and where they should be mounted. All entries that do not contain the `noauto` option are mounted during boot.

```
#<device>      <mountpoint><fs type><mount opts><to-be-dumped><fsck order>
/dev/dasdf1    /                ext2    defaults,errors=remount-ro 0 1
none          /proc           proc    defaults          0 0
/dev/dasdg1    /mnt            ext2    defaults          0 2
/dev/dasdi1    swap            swap    defaults          0 0
```

The `<device>` specifies the partition on the block device (`/dev/dasd<n>1`) or a file containing a file system. The `<mountpoint>` represents the directory on which the file system is to be mounted over. Typically the mount point is an empty directory. Swap spaces have the entry `swap`. The field `<fs type>` determines the file system type. The field `<mount opts>` (mount options) is a comma-separated list of options specific for the file system type. The number `<to-be-dumped>` is used for dump processing. It tells the kernel whether to dump (1) or not to dump (0) the file system. The second number `<fsck order>` specifies the order in which the file systems are checked at boot time. The root file system should have a 1 in that field, all others a 2. file systems such as swap space and `/proc` that do not need checking have a value of 0.

The kernel mounts file systems at boot time listed in `/etc/fstab` in order of appearance, therefore the root file system must be listed first.

After booting you can use `mount -a` command to mount all listed file systems that do not have the option `noauto`. This could be used to activate the changes made in the `/etc/fstab` file.

9.2.3 Checking and repairing an ext2 file system: e2fsck

An ext2 file system can be in an inconsistent state after an unclean shutdown or after being mounted (read-write) from two different Linux partitions at the same time. During boot up, the system does a file system check on file systems that are marked unclean and have an entry in `/etc/fstab`. However, you can manually initiate a file system check with the `e2fsck` command:

```
e2fsck /dev/dasdh1
```

For file systems that are marked clean, you have to issue the `e2fsck` command with the force option:

```
e2fsck -f /dev/dasdh1
```

`e2fsck` checks and repairs file system inconsistencies. It performs a check over all inodes and checks the directories and their connectivity. Directories that have lost their association will be connected to the `/lost+found` directory. Next `e2fsck` checks the reference count for all inodes, verifying the links and validating the file system summary information.

Note: A file system must be unmounted or mounted read-only for this task in order to prevent any other process from changing the contents of the file system.

9.3 Linux swap space

When more memory is required than physically exists on the system, swap spaces are used. This is the difference between real memory (physical) and virtual memory (physical memory plus swap space).

The total size of the virtual memory that can be accessed by Linux for S/390 is slightly less than 2 GB (1919 MB). With less than 2 GB of physical RAM, swap areas can be used to expand the available memory.

Linux uses the technique of *paging*, which means that blocks of memory (memory pages, whose size is 4 kilobytes) that are not currently in use are temporarily moved to the swap device. In contrast to paging, *swapping* is the process of exporting all pages of a process at once. Even though the device is called swap space, Linux does not really swap out a process.

A swap space can be a block device (disk or partition on disk) or a regular file. For Linux for S/390, expanded storage can be used as a block device through the XPRAM driver. If you have expanded storage available, this might be the preferred solution (see 9.3.4.1, “Swap space on a ramdisk” on page 190).

If the system already has swap space assigned, the free command will display this information in a separate row:

	total	used	free	shared	buffers	cached
Mem:	127616	22808	104808	22488	868	6832
-/+ buffers/cache:		15108	112508			
Swap:	236132	0	236132			

The decision whether to use disk partitions or swap files is highly dependent on your environment. See 6.9.1, “Reducing Linux for S/390 swapping” on page 129 and the discussion in each swap space description for more information.

9.3.1 Creating swap spaces

The `mkswap [-c] device` command is used set up a swap area. The device can be a device like `/dev/xpram<n>`, `/dev/dasd<n>1`, or a file.

The option `-c` requests a check for bad blocks on a block device before creating a swap area. If any are found, the count is printed. The following example creates a swap file on the first partition on the device `dasdf`:

```
mkswap /dev/dasdf1
```

9.3.2 Activating and deactivating swap spaces

To activate a swap area use the `swapon` command. The `swapon -a` command activates all swap areas listed in `/etc/fstab`.

Active swap spaces can be deactivated with the `swapoff` command.

9.3.3 Displaying information on swap spaces

There are two possibilities to verify that a swap space is active: using the `swapoff` command together with the option `-s`, or using the information stored in the `/proc` file system.

```
$ cat /proc/swaps
```

Both produce the following identical output:

Filename	Type	Size	Used	Priority
/dev/dasdf1	partition	143980	0	-1
/dev/xpram0	partition	0	4	-2
/mnt/swap/swap001	file	51196	0	-3
/mnt/swap/swap002	file	40956	0	-4

The listed swap spaces are a partition on a DASD drive, a partition in expanded storage, and two swap files in the directory `/mnt/swap`.

9.3.4 Preparing swap space

The commands to create a swap space are nearly the same for all implementations. See Table 21 for an overview.

You can have up to 8 swap areas comprising the swap space. You can concatenate different styles (swap files with different block size and swap partitions).

Table 21. Commands to create different types of swap spaces

	xpram	swap partition		swap file	
Device	/dev/xpram _n	/dev/dasdn1	/dev/dasdn1 (minidisk)	/dev/dasdn1	/dev/dasdn1 (minidisk)
Command to prepare	Check availability of expanded storage	dasdfmt	none needed	dasdfmt mke2fs mkdir /swap chmod 700 <dir> mount /swap dd	mke2fs mkdir /swap chmod 700 <dir> mount /swap dd
Command to activate	mkswap <partition> chmod 600 <partition> swapon <partition> add entry in /etc/fstab check with swapon -s			mkswap <swap> chmod 600 <swap> swapon <swap> add entry in /etc/fstab check with swapon -s	

We recommend adjusting the file permission bits of the device node or file to 600 using the `chmod` command (explained in 9.6, “File ownership and access permissions” on page 202).

To reserve space for a swap file, use the `dd` command. For example:

```
$ dd if=/dev/zero of=/dev/swap1 bs=1M count=128
```

Option `bs` of the `dd` command specifies the block size and `count` specifies the number of blocks to copy. So in this example we copy 128 blocks of 1 megabyte from the input file (`if`) to the output file (`of`). Always double-check the `dd` command before you actually enter it; a wrong device given as `of=` argument will be overwritten with zeroes!

9.3.4.1 Swap space on a ramdisk

To create a swap device on a ramdisk, you need to have access to expanded storage through the `xpram` driver.

1. Set up as swap area. For example:

```
$ mkswap /dev/xpram0
```

2. Adjust the access permissions for the device:

```
$ chmod 600 /dev/xpram0
```

3. Activate the swap space:

```
$ swapon /dev/xpram0
```

4. Optionally, add the entry to the file `/etc/fstab`.

5. Verify that the swap space is active:

```
$ swapoff -s
```

9.3.4.2 Swap space on a partition

For a swap space on a partition you need a formatted device with a block size of 4 k, the size of Linux's memory pages.

1. Prepare to create the swap space:

```
$ dasdfmt -b 4096 -f /dev/dasda1
```

```
$ mke2fs -b 4096 /dev/dasda1
```

2. Create a swap file system:

```
$ mkswap /dev/dasda1
```

3. Adjust the access permissions:

```
$ chmod 600 /dev/dasda1
```

4. Activate the swap space:

```
$ swapon /dev/dasd<n>1
```

5. Optionally, add the entry to the file `/etc/fstab`.

6. Verify that the swap space is active:

```
$ swapoff -s
```

9.3.4.3 Swap file

In order to use a swap file you need a mounted file system with a block size of 4 k, the size of Linux's memory pages. The free space on the file system must be larger than the size of the swap file.

1. Create a directory for the swap file. For example:

```
$ mkdir /swap
```

2. Create a swap file of size 128 MB:

```
$ dd if=/dev/zero of=/swap/swapfs1 bs=1M count=128
```

3. Set up as swap area:

```
$ mkswap -c /swap/swapfs1
```

4. Adjust the access permissions of the file:

```
$ chmod 600 /swap/swapfs1
```

5. Activate the swap file:

```
$ swapon /swap/swapfs1
```

6. Optionally, make an entry in /etc/fstab.

9.4 File systems and devices

In the next sections we cover commands to go from a raw hardware device to file systems which can be used with Linux. The list of commands and terms used is shown in Table 22. The background information needed is provided in Chapter 8, “Linux for S/390 bootup and shutdown” on page 165.

Table 22. From a raw device to processes accessing file systems

Aspect of file system	Component	Command
Hardware device	DASD (3380, 3390, 9345) Expanded storage	N/A
Driver	DASD ECKD driver Minidisk driver Xpram driver	Kernel parameter Module parameter
Device node	/dev/dasd* /dev/mnd* /dev/xpram*	mknod
Partition	/dev/dasd,n.1	dasdfmt
File system	ext2 file system swap file system	mke2fs mkswap
Boot	Boot sector	silo

9.4.1 Formatting a block device: dasdfmt

Prior to creating a file system on a device, it must be formatted with the `dasdfmt` command. See Table 17 on page 179 for information on how to determine the devices that need to be formatted with `dasdfmt`. Following is the syntax of the `dasdfmt` command:

```
dasdfmt [-tvy] [-s start_track] [-e end_track]  
        [-b blocksize] -f dev_filename | -n 390_devno
```

Using the `-v` (verbose) flag, more messages are displayed. The `-y` flag omits the prompt to reconfirm the format request. The `-t` flag sets the command to test mode; the device will not actually be formatted. To format only a part of the device, you can specify a range with `-s start_track` and `-e end_track` flags.

The block size can be 512, 1024, 2048 or 4096 bytes. As the ext2 file system uses a minimum of 1024 bytes for a block, 1024 bytes or higher is recommended. For a discussion on block sizes to use see 9.2.1, “Block size relation between device and file system” on page 186.

In most cases only a few options are used. For example:

```
dasdfmt -b <blocksize> -f <device node>
dasdfmt -b <blocksize> -n <s390-devnr>
```

In order to format the device `/dev/dasdc` (with an S/390 device address of 192 in our example), the command can be specified in these two ways:

```
dasdfmt -b 4096 -f /dev/dasdc
dasdfmt -b 4096 -n 192
```

The system will prompt you to reconfirm the request:

```
$ dasdfmt -b 4096 -f /dev/dasdc
I am going to format the device /dev/dasdc in the following way:
Device number of device : 0x192
Major number of device  : 94
Minor number of device  : 8
Start track              : 0
End track                : last track of disk
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
```

The `dasdfmt` command will erase all data on the volume, including the volume label and volume table of contents (VTOC) of other operating systems. Depending on the size of the device, it may take some time to complete.

The `dasdfmt` command can only format previously formatted devices (i.e. devices that were formatted with any other format utility at least once).

9.4.2 Creating a file system: `mke2fs`

A file system can be created on a formatted device. The command `mke2fs` creates an empty ext2 file system on a device. Following is the syntax:

```
mke2fs [-b <block size>] /dev/dasd<n>1 | /dev/xpram<n>
```

For example, to create an ext2 file system of /dev/dasdc1, you would issue the following command:

```
mke2fs -b 4096 /dev/dasdc1
mke2fs 1.15, 18-Jul-1999 for EXT2 FS 0.5b, 95/08/09
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
27008 inodes, 26997 blocks
1349 blocks (5.00%) reserved for the super user
First data block=0
1 block group
32768 blocks per group, 32768 fragments per group
27008 inodes per group

Writing inode tables: done
Writing superblocks and filesystem accounting information: done
```

The logical block size can be 1024, 2048 or 4096 bytes. The block size influences device use and performance. As the kernel reads in blocks, a big block size speeds up I/O (fewer I/O requests), but the smallest addressable unit to write to is a block so even a 1 k file will reserve 1 block (i.e., 4 k on the device).

For a discussion of block sizes to use see 9.2.1, “Block size relation between device and file system” on page 186.

9.4.3 Accessing a file system: mount

To automatically mount a file system on boot up, an entry in /etc/fstab is needed (see 9.2.2, “The file system table /etc/fstab” on page 187). Manual mounting is done via the `mount` command:

```
mount -t type dev mountpoint
```

See 9.2, “File system types” on page 185 for valid file system types.

A mount point is a directory. Normally the directory is empty. If it is not, any files it might contain are hidden as long as a file system is mounted over it. The following command mounts the file system that is on /dev/dasdg1 over the directory /mnt:

```
mount -t ext2 /dev/dasdg1 /mnt
```

The `mount` command without any options or parameters shows the mounted file systems:

```
$ mount
/dev/dasdf1 on / type ext2 (rw,errors=remount-ro)
none on /proc type proc (rw)
/dev/dasdg1 on /mnt type ext2 (rw)
```

9.4.4 Making a device bootable: silo

To load a Linux for S/390 image from a device, the device has to contain a boot record (this is known as the `IPL record` in OS/390 terminology). Only devices formatted with the `dasdfmt` command can have a boot record. The block size of the device must be either 2 k or 4 k.

The `silo` command is used to create a boot record (Linux on the PC uses the `lilo` command, while Linux on some other platforms use the `milo` command) The syntax of `silo` is as follows:

```
silo -d /dev/dasd<letter> -f <image_file> -p <parmfile> -b <bsfile>
```

The input files for the `silo` command have to be on the same device as the boot record. Do not rely on defaults; instead, you should always specify the following four flags, which are recommended for use with `silo`:

- d The device on which the boot record will be written. The boot record is written on the first blocks of this device. Therefore the device node representing the whole disk has to be specified.
- f The name of the kernel image.
- p The file containing the kernel parameters (for example `dasd=` or `root=`). For a list of possible values in the parameter file, see Appendix D, “The parameter file” on page 481.
- b The boot sector file. This file is created together with the image file during kernel build. This data will be written to the first blocks of the device.
- t Set the command in test mode. A test number of 1 specifies do-not-test, while a number > 1 specifies a test level. It is not expected, but the flag `-t2` is required with the 2.2.15 kernel. This is confirmed by the following warning message:

```
WARNING: silo does not modify your volume. Use -t2 to change IPL
records
```

Additional silo flags are:

- F Specify a configuration file to set silo defaults.
- V Display the version of the silo command.
- v Set a deeper message level (verbose).
- B Set the name of the boot map created by the silo command.

We assume a directory `/boot`, together with the files needed to create a boot record. For a discussion on how to create a boot record on another device, see 11.6.1, “Preparing a second bootable device” on page 239.

For example, you should see the following files in the directory `/boot`:

```
$ cd /boot
$ ls -l
total 1444
-rwxr-xr-x  1 root    root      1453976 Feb 20 09:43 image
-rwxr-xr-x  1 root    root         2048 Feb 20 09:43 ipleckd.boot
-rw-r--r--  1 root    root           80 May 18 09:15 parm.line
```

Following is an example of creating a boot record with `silo`:

```
silo -f image -d /dev/dasdb -p parm.line -b ipleckd.boot
```

The `silo` command creates files `boot.map` and `kernel.h`. Messages about the process are displayed in the following example:

```
o->ipldevice set to /dev/dasdb
o->image set to image
o->bootsect set to ipleckd.boot
o->parmfile set to parm.line
Verbosity value is now 2
IPL device is: '/dev/dasdb'...ok... (94/4)
bootsector is: 'ipleckd.boot'...ok...
Kernel image is: 'image'...ok...
parameterfile is: 'parm.line'...ok...
ix 0: offset: 00007f count: 0c address: 0x00000000
ix 1: offset: 00008c count: 80 address: 0x0000c000
ix 2: offset: 00010c count: 80 address: 0x0008c000
ix 3: offset: 00018c count: 57 address: 0x0010c000
ix 4: offset: 0001e4 count: 0e address: 0x00163000
ix 5: offset: 0001f3 count: 01 address: 0x00008000
Bootmap is in block no: 0x000001f4
```


It is recommended that the configuration file be named `/etc/silo.conf`. In our tests we used this file:

```
[root@linux6 /etc]# cat /etc/silo.conf
ipldevice = /dev/dasd00
image = /boot/image
bootsect = /boot/ipleckd.boot
map = /boot/boot.map
parmfile=/boot/parm.line
testlevel=2
```

Besides the `silo` command specific defaults, you can set kernel parameters in this file. These are normally in the file specified with the `-p` option of `silo`.

These kernel parameters are:

<code>ramdisk=</code>	Name of the ramdisk
<code>root=</code>	Name of the root partition
<code>readonly</code>	Mount the root device read-only before reading <code>/etc/fstab</code>
<code>append=[parameterlist]</code>	parameters like <code>mem=</code> , <code>mdisk=</code> , <code>dasd=</code> , <code>dasd_force_diag=</code> , <code>xpram=</code> , <code>ipldelay=</code> as described in Appendix D, “The parameter file” on page 481

9.5 Users and groups

While the UNIX user enters his user name (or login) and password to log on to the system, his identity is represented by a unique integer. This number is called the user identifier (UID). By convention, a lower UID range (e.g. 0..499) is reserved for accounts that exist for system-level services.

The UID zero is reserved for the so called superuser or “root”. Daemon (a UNIX term for a process that is designed to run continuously) users have UIDs typically in the range 1 to 99. These users are defined for programs running in the background that offer certain services. Examples are: `named` (name service); `wwwrun` (http server script execution); and database back ends like `postfix`. Which daemon users are actually present on a particular system depends on which services have been installed. Some daemons also run with root privileges.

The user `nobody` (usually UID 65534) exists for certain tasks where no special rights are required (or wanted), such as the automatic update of the file database.

Every user belongs to an initial (or login) group. Groups, similar to users, consist of a clear text name and an integer identifier, the group identifier (GID). One group can contain more than one user and one user can belong to more than one group.

Groups are a means of granting certain rights to users. As an example, some users of might be added to a group which grants the right to access a common working directory. Similarly, a user might make some files readable for the members of his login group.

The command `su` allows a user to run a shell with another UID (and GID). Authentication (entering the password of the account that one switches to) is required except for root.

9.5.1 Creating a user account: `useradd`

The `useradd` command creates a new user. It is in the `/usr/sbin` directory which may not be in your `PATH`.

```
useradd [-d <home>] [-g <group>] [-G additional groups>] [-m]
        [-s <shell>] [-u <uid>] [-p passwd] <username>
```

The options are:

- c Comment (usually full name of the user).
- d The home directory to be created.
- g The default group for the user by name or by GID. If you do not specify a group, Linux will create a group with the same name as the user and the next free group id ≥ 500 . The user is then the only member in this new group.
- G Additional groups the user should belong to.
- m Create the home directory specified by the `-d` option if it does not exist.
- s Basic shell, i.e. program to run at login time. For more information about shells, see 9.8, “Shells” on page 203.
- u Numeric user ID associated with the user name. If this value is not specified, the system will associate the next free user id ≥ 500 .
- p Set initial password.

For example, to add a user named `user1`, issue the command:

```
useradd -m -d /home/user1 -p dummypw -c 'Sam Adams' user1
```

This user will have the password `dummypw` which he can change using the `passwd` command. An entry will be created in the file `/etc/passwd`:

```
[root@linux6 /etc]# grep user1 /etc/passwd
user1:!tyj61STZId5Iw!:501:501:Sam Adams:/home/user1:/bin/bash
```

The format of the passwd entries is:

```
user:password:uid:gid:description:home:login_shell
```

The defaults used when creating a user are defined by settings in the file `/etc/default/useradd`. To display or change the defaults, use `useradd` with the `-D` option:

```
[root@linux6 /etc]# useradd -D
GROUP=100
HOME=/home
INACTIVE=-1
EXPIRE=
SHELL=/bin/bash
SKEL=/etc/skel
```

9.5.2 Modifying a user account: `usermod`

To change the characteristics of a user, use the `usermod` command. Its options are:

- c** New comment
- g** New initial group ID or group name for the user
- G** Comma-separated list of supplementary group the user should be in. The user will be removed from groups not listed here.
- s** Change login shell
- u** Specify the user ID for which to change any values
- l** Change the login name of the user

Some attributes can only be changed when the user is not currently logged in. The `usermod` changes file ownership of the `/home` directory of the user when changing any relevant data like the group id. The following command will assign a new name to the user with UID 500.

```
$ usermod -u 500 -l name2
```

9.5.3 Deleting a user account: `userdel`

The `userdel` command deletes a user. Following is the syntax:

```
userdel -r user2
```

The option `-r` causes the home directory of the user to be deleted. `Userdel` does not delete any files owned by the user that are not stored in the user's

home directory. Files owned by the user can be found (*before* deleting the user) with:

```
find / -user user2
```

Deletion of all those files can be achieved with the following command (use care when using `find` and `rm`):

```
find / -user user2 -exec rm -f {} \;
```

The groups the user belongs to will not be deleted even if the user is the last one in that group.

9.5.4 Verifying the integrity of the passwd file: `pwck`

The command `pwck` does a consistency check for the file `/etc/passwd`. It checks for:

- Correct number of fields
- Unique user names
- Valid user and group identifiers
- Valid primary groups
- Valid home directories
- Valid login shells

It also prints a warning for any user who has no password.

9.5.5 Creating a new group: `groupadd`

The `groupadd` command adds a group. Following is the syntax:

```
groupadd [-g <groupid>] [-f] <groupname>
```

`-g` Specifies the group ID. Group IDs 0 - 499 are reserved for system usage.

`-f` Abort with an error if the group already exists.

For example, the following command creates a group named `group1` with the next free GID ≥ 500 :

```
$ groupadd group1
```

The groups are listed in the file `/etc/group`:

```
[root@linux6 /etc]# cat /etc/group
root::0:root
bin::1:root,bin,daemon
```

```
daemon::2:root,bin,daemon
sys::3:root,bin,adm
adm::4:root,adm,daemon
tty::5:
disk::6:root
lp::7:daemon,lp
mem::8:
kmem::9:
wheel::10:root
mail::12:mail
news::13:news
uucp::14:uucp
man::15:
games::20:
gopher::30:
dip::40:
ftp::50:
nobody::99:
users::100:
utmp:x:22:
xfs:x:101:
group1:x:500:
group2:x:610:
```

This file has the following format:

```
name:password_of_group:gid:list_of_users
```

9.5.6 Modifying a group: `groupmod`

To change a group, use the `groupmod` command. Following is the syntax:

```
groupmod [-g <gid>] [-n <newname>] <groupname>
```

`-g` Change the group ID.

`-n` Change the name of the group.

The following example changes the GID of `group1` to 603:

```
groupmod -g 603 group1
```

Be aware that if you change the GID of a group, `groupmod` does not change the GID of the users and files. This can result in problems accessing files.

9.5.7 Deleting a group: `groupdel`

The command `groupdel` deletes a group. Following is the syntax:

```
groupdel group2
```

The initial or primary group of any user cannot be deleted with `groupdel`. With the `groups` command you can determine to which groups a user belongs:

```
groups user1
```

9.5.8 Verifying the integrity of the group file: `grpck`

The `grpck` command checks the `/etc/group` file for inconsistencies. It works similar to the `pwck` command described in 9.5.4, “Verifying the integrity of the passwd file: `pwck`” on page 200.

9.6 File ownership and access permissions

Files created by a user belong to him and his login group. File ownership can only be changed by the superuser by using the `chown` command. The group can also be changed by the owner of the file, but only to a group that he is a member of. The following command makes `jj` the new owner of the named file, and makes `math` its group.

```
chown jj.math /somewhere/fft.c
```

The following command makes `fbi` the new owner of the `/elsewhere/xfiles/` directory and all the files and directories in it:

```
chown -R fbi/elsewhere/xfiles/
```

The `chmod` command changes the access permissions of files. The access permissions (write, read, execute) for different domains of users (owner, initial group of owner, everybody) can be granted by the owner of the file. For example, the following command adds the permission to execute the file `progfile` to the owner (`u` stands for user) and members of the file’s group (`g`):

```
chmod ug+x progfile
```

The following command allows everybody (`a` stands for all) to read the file `message.txt`:

```
chmod a+r message.txt
```

The following command withdraws rights to read, write, or execute (or enter directories) for `mysecrets` and any file or directory within (option `-R` stands for recursive) to everybody except the file owner:

```
chmod -R go-rwx mysecrets/
```

See the `chmod` manpage for more details:

```
man chmod
```

9.7 Changing passwords

The `passwd` command allows a user to change his password interactively by simply entering `passwd` without arguments. The old password must be entered first, and then the new password has to be entered twice (to avoid problems caused by typos). The superuser can change the password of any user (without having to know the current password) by using the `passwd` command followed by the username.

9.8 Shells

A *shell* is an interface that allows a user to work with the operating system. Several different shells are available with Linux; some of the more important are:

- `/bin/sh` Bourne shell (now a link to `bash`).
- `/bin/bash` Bourne Again Shell. It is the default shell of almost all existing Linux installations.
- `/bin/csh` C shell (now a link to `tcsh`); its syntax is somewhat similar to C
- `/bin/tcsh` enhanced C shell.
- `/bin/ksh` Korn shell; compatible to the bourne shell. It combines the characteristics of bourne and C shell.
- `/bin/zsh` Z shell; enhanced version of the Korn shell.

To determine which shell one is currently using, enter:

```
[root@linux6 /root]# echo $SHELL
/bin/bash
```

During login, the system calls the shell that is set in the `/etc/passwd` file for this user. The login shell can be changed interactively using the `chsh` command.

9.9 System logs

The system log daemon `syslogd` logs various kinds of system activity. It is started during runlevel processing at boot time (`/rc<N>.d/S30syslog`).

The `/etc/syslog.conf` file contains the configuration of `syslogd`. It describes which kind and level of information is logged into which file. The configuration file on our system looks like this:

```
[root@linux6 /etc]# cat syslog.conf
```

```

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none           /var/log/messages
authpriv.*                                /var/log/secure
# Log all the mail messages in one place.
mail.*                                     /var/log/maillog
*.emerg                                    *
# Save mail and news errors of level err and higher in a
# special file.
uucp,news.crit                             /var/log/spooler
# Save boot messages also to boot.log
local7.*

```

The first information in the configuration files contains messages in the form `systemapplication.level`. It can be repeated several times, separated by semicolons. Possible system applications can be:

<code>auth</code>	Used by user authentication (login) programs.
<code>cron</code>	Used by the cron daemon.
<code>daemon</code>	Used by miscellaneous daemons.
<code>kern</code>	Used by the Linux kernel itself.
<code>lpr</code>	Used by the line printer daemon.
<code>mail</code>	Used by the mail daemon.
<code>news</code>	Used by the news daemon.
<code>syslog</code>	Used by syslog itself.
<code>uucp</code>	Used by UUCP daemon.
<code>local0-7</code>	Used by miscellaneous daemons and applications; for example the application <code>chat</code> writes its messages to facility <code>local2</code> .

The level determines the severity of the message and can contain:

<code>none</code>	Do not write messages.
<code>debug</code>	Debugging messages.
<code>info</code>	Miscellaneous information messages.
<code>notice</code>	Something may be wrong.
<code>warning</code>	A condition that may cause trouble if not checked.
<code>err</code>	An error condition.
<code>crit</code>	A critical error.
<code>alert</code>	A severe error.

`emerg` An irrecoverable error has occurred within the kernel, often followed by `kernel panic` on your screen.

The second field contains the *location* to store the messages. This can be any filename or `*` for the current virtual console or terminal (xterm) started with the `-c` option.

You can change the `syslogd` configuration by editing the configuration file. To tell `syslogd` to reread the file after changes, send the HUP signal to it:

```
$ kill -HUP `cat /var/run/syslogpid`
```

The command `cat /var/log/syslog.pid` gives the current process ID of the `syslog` daemon. You can also determine the process ID with the `ps -ef` or `ps auxw` command.

Log rotation

The `syslog` and other daemons normally append the log files. Therefore, over time, log files can grow very big. To avoid this you can use the `logrotate` command. It rotates, compresses, and mails system logs as specified in the configuration file. The main `logrotate` configuration file is `/etc/logrotate.conf`:

```
[root@linux6 /etc]# cat logrotate.conf
# rotate log files weekly
weekly
# keep 4 weeks worth of backlogs
rotate 4
# send errors to root
errors root
# create new (empty) log files after rotating old ones
create
# uncomment this if you want your log files compressed
#compress
# RPM packages drop log rotation information into this directory
include /etc/logrotate.d
# no packages own lastlog or wtmp -- we'll rotate them here
/var/log/wtmp {
    monthly
    create 0664 root utmp
    rotate 1
}
/var/log/lastlog {
    monthly
    rotate 1
}
# system-specific logs may be configured here
```

Other software will put its logrotate configuration in a file in the `/etc/logrotate.d/` directory. The `cron` mechanism is used to start the logrotate command automatically. You can find a shell script doing the log rotation in the directory `/etc/cron.daily`.

9.10 Cron

Cron is a simple scheduler that executes predefined commands at specified times. It searches the directory `/var/cron/tabs` for crontab files (lists of scheduled commands; see *man 5 crontab*) which are named after user accounts in `/etc/passwd`. In addition, the `/etc/crontab` file and files in `/etc/cron.d/` are searched for system crontab entries. System crontab entries have an additional user field that determine the user id under which the respective command is started.

A typical system crontab file might look like the following:

```
[root@linux6 /etc]# cat /etc/crontab
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/

01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
```

In the first few lines, the environment is set, while in the remainder of the file there is one row for each command to be scheduled. The format of the fields is as follows:

```
minute hour day month dayofweek user command.
```

Possible values are 0-59 for `minute`, 0-23 for `hour`, 0-31 for `day`, 0-12 for `month` and 0-7 for `dayofweek`. Ranges (for example 1-3) and comma-separated lists (for example 1,3,5,7 or 0-3,5-7) are allowed. An asterisk (“*”) stands for “all values”. A forward slash specifies periodic operation: “*/2” would mean “every two minutes” if specified in the minute field. For further information see the `crontab` manpage *man 5 crontab*.

Note that crontabs are *not* shell scripts; many of the comfortable features of your favorite shell (for example the backtick mechanism) are not available in crontabs. If you like to do more complex things than a single command, you

will typically create a shell script for the job and call that script from the crontab.

As the cron daemon sends standard output (stdout) and error messages (stderr) via mail to the owner of the crontab (or the account specified as MAILTO in the crontab), it is sometimes necessary to redirect unwanted output into the virtual garbage can (/dev/null); the following command discards messages on stdout but only allows error messages into the mail:

```
somecmd > /dev/null
```

The following command appends any output to the file somefile. The standard error is redirected to standard output by the `2>&1` statement:

```
somecmd >> somefile 2>&1
```

To list your crontab as a user, simply enter:

```
crontab -l
```

To edit your crontab, use the command:

```
crontab -e
```

Do *not* directly edit the crontab files; the above command will launch the editor specified in the VISUAL (or EDITOR) environment variable, let you edit the entries, and finally schedule your jobs with cron (see the crontab man page).

9.11 Pluggable Authentication Module (PAM)

The Linux Pluggable Authentication Module (PAM) is a set of libraries that handle the authentication tasks of services on the system. The library provides a general Application Programming Interface (API) that privilege-granting programs such as `login` and `su` defer to perform standard authentication tasks. The principal feature of the PAM approach is that the nature of the authentication is dynamically configurable.

With PAM, one can easily configure account verification, user authentication, password management and session management for different services. The configuration files for PAM are located in the directory `/etc/pam.d/`. These are named after the services whose authentication they configure; one file, named “other” is for the default rules. A line of the configuration files is of the form:

```
type control module-path module-arguments
```

The `type` field can be a value of `account`, `auth`, `password` or `session`; it addresses the authentication task to be configured. The `control` field can be a value of `requisite`, `required`, `sufficient` or `optional`; it sets the behavior in case of authentication failure. The `module-path` and `module-arguments` fields name the (path and) binary of the PAM module to be used, and possible calling arguments.

9.12 Interactive administrative utilities

The Marist Linux big file system does not include an interactive administrative utility; therefore, you may want to get a separate application. These types of applications often give the Linux administrator a graphical, rather than a command line interface, and allow the administrator to do activities which are task-based rather than command-based. Two commonly used utilities of this type are `linuxconf` and `YAST`.

9.12.1 Linuxconf

The `linuxconf` tool includes a user interface for configuration tasks and an activator. On the Web see:

<http://www.solucorp.qc.ca/linuxconf/>

There are different user interfaces:

Text-based	This works everywhere on a terminal or using a Telnet session.
Web interface	This works through any Web browser and has no need for a http server. <code>linuxconf</code> handles the http protocol itself and is started from the <code>inetd</code> server.
Graphical interface	There are two GUI front ends (Java or wxXT toolkit).
Command line	Configuration via a shell script. It is mainly used to handle major changes with lots of repetitive tasks.

9.12.2 YAST

`YAST` is an acronym for Yet Another Setup Tool. It is a tool to install, configure, and administer the SuSE Linux distribution. It offers the ability to install or update software packages and perform many system tasks like network configuration, user administration, and security. `YAST` has a text-based user interface.

9.12.3 YAST2

YAST2 is Yet Another Setup Tool, next generation. It is the successor of YAST and comes with both a text-based and graphical (X11) user interface. Compared to YAST, it offers some enhanced functionality, most notably remote installation and administration.

Chapter 10. Backup

Backing up your system is an essential, but often tedious, task. In this chapter we discuss general methods of doing backups and detail the commands you can use to back up the system (or part of it).

10.1 The general concept

Before you start a backup, first decide which errors you want to recover from.

- Physical errors, such as:
 - Total loss of the system and its data, so that you need a disaster recovery plan
 - Loss of a single device
- Logical errors, such as:
 - Users who accidentally delete files
 - Failure of software that forces you to go back to a previous level of a data file or software

10.1.1 Backup strategies

For a backup that covers physical errors, you will want to do a per device or per file system backup. Together with your backup, you should also store information about the system (that is, location of the data and customization of the system). For disaster recovery, the backup files have to be at a remote location.

For a backup that covers logical errors, it may be sufficient to back up the file system or directory. Optional information about the image of the system may also be helpful.

Both backup strategies have the need for meaningful backup names that include information on what, when, and how the backup was taken.

10.1.1.1 Automating backup

Do not rely on a person to make your backups; instead, automate the steps so you do not miss a backup cycle. An easy way to do this is by using cron; see 9.10, “Cron” on page 206.

Include a script to check the success of the backup itself. For example, use `grep -v` to analyze the backup log. The script can be used to inform the administration users of errors or uncommon messages via e-mail.

10.1.1.2 Storing backup data

Note

Linux for S/390 does not support tapes (3480/3490) at the time of writing. For more information about tape support check the Marist List Server and the Linux for S/390 distributions (SuSE and TurboLinux).

As tape is not supported today, there are limited ways to store backup data. These are:

- On DASD owned by the same Linux. For the S/390 architecture, this DASD can be at a remote location.
- On a remote file system using NFS (see Chapter 19, “Network File System (NFS)” on page 367).
- On tape through another system (UNIX or OS/390 USS, VM/ESA) with access to tape drives and to the Linux for S/390 data (using FTP, NFS, or Samba, to name a few options).
- VM/ESA offers additional flexibility to back up Linux for S/390. For more information see 6.8.4, “Taking backups of Linux for S/390 file systems” on page 127 .

To keep track of the backups—where they are stored and how they are created—we recommend that you take some additional steps:

- Create a list of backup files on a second system.
- Store a layout of the backed-up system on a second system.
- Keep track of basic changes that may influence the backup commands, such as the installation of new kernel versions. Test the compatibility of the commands after each change.

Knowing how to recover the data is as important as the backup itself. Be familiar with the restore commands. This will make your task a lot easier if a disaster occurs.

You should also give some consideration to how long you want to keep old backup files. Such planning has the potential of saving a considerable amount of space and may help make backups to DASD devices more attractive for your installation.

A useful discussion on backup concepts and utilities for UNIX platforms can be found in *UNIX Backup and Recovery* by W. Curtis Preston, published by

O'Reilly & Associates, ISBN 1565926420. Some chapters of the book, including helpful scripts, can be found on the Internet at:

<http://www.backupcentral.com/>

This Web page also includes links for downloading free backup utilities.

10.2 Native backup commands

In general, you have four possibilities to back up the system with native Linux for S/390 commands. The command names for backup and restore are the same except for the `dump` command. Backups created with the `dump` command have to be restored with the `restore` command.

Which possibility (command) you should use depends on:

- What kind of backup you want to create (quick backup or disaster recovery backup, logical or physical backup)
- The commands you are familiar with

Table 23 compares the commands you can use for backup.

Table 23. Comparison of backup commands

Characteristic/Task	Dump/Restore	GNU cpio	GNU tar
Complexity	Complicated but good for multilevel backup	Mostly simple, needs find	Simple, needs find
Incremental backup	Yes, through levels	Yes, but has to use find	Yes, but has to use find and has limits
Multivolume archive	Yes	Yes	Yes
How to find a list of archived files	Simple, index in front (restore -r)	Search entire file (cpio -it)	Search entire file (tar -t)
Find specific files	Interactive (supports <code>ls</code> and <code>cd</code> commands)	Complex, search entire file, wildcards allowed	Complex, search entire file, no wildcards allowed
Backup protocol	Generate after backup with restore -t dumplog	<code>cpio -v 2> cpiolog</code>	<code>tar cvf 2> tarlog</code>
Restore archive with absolute path to different location	Always relative to current work directory	Limited functionality with <code>cpio -l</code>	Complicated, only by using <code>chroot</code>

Characteristic/Task	Dump/Restore	GNU cpio	GNU tar
Uses	System backup	System backup, transfer files between systems	Quick backup, transfer files between systems

There is another archive command, `pax`, that is similar to `tar`. The `pax` command creates a portable archive. It can read other file formats such as the `tar` and `cpio` formats. If you already use the `pax` command on other platforms, it is worth installing it on Linux for S/390. By default it is not included in the Marist big file system.

10.2.1 dump/restore

The `dump` command is the only one that directly supports incremental backup. The `restore` command has powerful functionality, such as interactive restore and indexing.

Hint

If the `dump` and `restore` commands are not included in your file system, you can download the `dump` package (`dump-0.4b4-11.s390.rpm`). This package provides the `dump` and `restore` commands for the `ext2` file system.

10.2.1.1 Backup using dump

To back up the file system, issue the `dump` command:

```
dump <level> unBf blocking_factor records archive_file fs_to_save
```

The options are:

- <level> A number from 0 to 9 (level 0 = full backup, 1 to 9 = incremental backups).
- b The blocksize; the number of kilobytes per dump record.
- B The number of dump records per volume; this option overrides the calculation of tape size based on length and density. Specify a very high number, for example 10000000, to inactivate tape size calculation when dumping to DASD (otherwise `dump` will prompt for a new tape during backup).
- u Update the file `/etc/dumpdates` after a successful dump.
- n Notify the members of a specified group whenever `dump` requires attention.

f Write the backup to filename (`f archive_file`).

There are two further options, used for informational purposes:

w List information on all file systems.

W List those file systems that need to be backed up (dumped).

When you run the `dump` command for the first time on a system, you have to create an empty `/etc/dumpdates` file. This file must be owned by root.

```
touch /etc/dumpdates
```

Dump writes a table of contents at the beginning of the archive file. This index is created before the backup data is written. It does not honor changes made while the dump task is active.

The *level* of the `dump` command specifies how often a full backup shall be executed and how often an incremental backup shall be executed. Use level 0 to do a full backup and levels 1 to 9 for the incremental backups, where each level backs up whatever was changed since the last backup with the next lower level.

Table 23 presents a concept using incremental backup:

Table 24. Backup level concept

Day	Level	Data-Backup
1	0	Full backup
2	3	All changes since day 1
3	2	All changes since day 1
4	5	All changes since day 2
5	4	All changes since day 2
6	7	All changes since day 3
7	6	All changes since day 3

This example is for a large system with lots of data:

- Once a week a full backup is taken; in our example we call it day 1.
- On day 2, a level 3 backup is taken that saves all changes since the last full backup on day 1.
- On day 3, the level 2 backup saves all changes done after day 1. In a recovery situation after day 3 where you need to restore the latest files, you would only need the backups of day 1 and day 3.

- Day 4 to 7 do the same switching of the backup levels as day 2 and 3. This enables you to have more than two backup levels, without having to restore up to 7 backups in case of a crash. At worst, you would need the levels 0, 2, 4, and 6 for a recovery on day 7.

10.2.1.2 The restore command

To restore data saved with the `dump` command, use the `restore` command:

```
restore [trxi] vbfy blocking_factor archive_file fs_to_restore
```

The restore type/function can be specified with the following options:

- t Means display the content of a backup
- r Means restore entire archive file (recursive)
- x Means extract only listed files from the archive file
- i Means interactive restore

The restore behavior is influenced by the following options:

- v The verbose option - displays detailed information during restore.
- b Specify the blocking factor (block size) used for the dump command. If this option is not specified, restore tries to determine the block size dynamically.
- f Specify the archive file, `-f archive_file`. If you do not specify the archive file, the restore will expect the input from the default tape drive
- y Attempt to recover from read errors (skip over bad blocks).

Option `-r` requires starting with the level 0 archive file. You have to restore in *level order*; otherwise, the restore will fail.

As the backup of the files is done with relative paths, you have to change to the directory where you want to restore the file into before starting the restore.

For option `-x`, specify the exact path and file name. To restore two files named `file1` and `file2` from the directory `dir1`, enter:

```
restore rbvfy 126 /archives/archive.file.dump ./dir1/file1 ./dir1/file2
```

To determine what's in the archive file, you can create a table of contents with:

```
restore tbfy 126 /archives/home.day.dump
```

For the interactive option of restore, call restore with `-i`. This will provide a shell with limited functionality. You can use the following commands: `cd` for change directory; `ls` to list files; `pwd` to list the current directory name. In addition, you can issue restore-specific commands like `add`, `delete` and `extract`.

To *add* a file to the restore list, use `add filename` or `add *pattern*`. This will mark the selected file with an asterisk (*) when displaying the filelist with `ls`. To *delete* a file from the restore list, use `delete filename` or `delete *pattern*`. After having marked all the files you want to restore, initiate the restore with the `extract` command.

Following is an example of a dump with an interactive restore session.

- Dump the directory `/home/sujoma` in `/tmp/dump.home`:

```
[root@linux6 /]# dump 0ubBf 64 10000000 /tmp/dump.home /home/sujoma
DUMP: Date of this level 0 dump: Thu Jun  2 13:50:41 2000
DUMP: Date of last level 0 dump: the epoch
DUMP: Dumping /dev/dasdf1 (/) to /tmp/dump.home
DUMP: mapping (Pass I) [regular files]
DUMP: mapping (Pass II) [directories]
DUMP: estimated 125378 tape blocks on 0.01 tape(s) .
DUMP: Volume 1 started at: Thu Jun  2 13:50:41 2000
DUMP: dumping (Pass III) [directories]
DUMP: dumping (Pass IV) [regular files]
DUMP: DUMP: 125811 tape blocks on 1 volumes(s)
DUMP: finished in 129 seconds, throughput 975 KBytes/sec
DUMP: Volume 1 completed at: Thu Jun  2 13:52:50 2000
DUMP: Volume 1 took 0:02:09
DUMP: Volume 1 transfer rate: 975 KB/s
DUMP: level 0 dump on Thu Jun  2 13:50:41 2000
DUMP: DUMP: Date of this level 0 dump: Thu Jun  2 13:50:41 2000
DUMP: DUMP: Date this dump completed: Thu Jun  2 13:52:50 2000
DUMP: DUMP: Average transfer rate: 975 KB/s
DUMP: Closing /tmp/dump.home
DUMP: DUMP IS DONE
```

- Create a table of contents to verify the dump:

```
[root@linux6 /tmp]# restore tbfy 64 dump.home | less
Level 0 dump of / on linux6:/dev/dasdf1 (dir home/sujoma)
Label: none
Dump date: Thu Jun  2 13:50:41 2000
Dumped from: the epoch
      2      .
112686      ./home
```

```

113991      ./home/sujoma
113992      ./home/sujoma/.bash_history
114231      ./home/sujoma/rpms
114232      ./home/sujoma/rpms/THE-3_0-1_s390.rpm
114233      ./home/sujoma/rpms/sed-3_02-4_s390.rpm
114234      ./home/sujoma/rpms/ncurses-4_2-25_s390.rpm
.....

```

- Change into the destination directory for restore:

```
cd /tmp
```

- Call the interactive restore, select a file to restore, and extract it from the dump file (all actions are in bold letters):

```

[root@linux6 /tmp]# restore -if dump.home
restore > ls
.:
home/
restore > cd home/sujoma/rpms
restore > ls
./home/sujoma/rpms:
Regina-0_08h-1_s390.rpm          ncurses-4_2-25_s390.rpm
Regina-devel-0_08h-1_s390.rpm  ncurses-devel-4_2-25_s390.rpm
THE-3_0-1_s390.rpm             pam-0_68-7_s390.rpm
bzip2-0_9_5c-1_s390.rpm       rpm-3_0_3-3_s390.rpm
crontabs-1_7-7_noarch.rpm      samba-2_0_5a-12_s390.rpm
dump-0_4b4-11_s390.rpm        samba-client-2_0_5a-12_s390.rpm
fileutils-4_0-8_s390.rpm      samba-common-2_0_5a-12_s390.rpm
gpm-1_17_9-3_s390.rpm         sed-3_02-4_s390.rpm
joe-2_8-22_s390.rpm           sh-utils-2_0-1_s390.rpm
knfsd-1_4_7-7_s390.rpm        sysreport-1_0-1_2_noarch.rpm
knfsd-clients-1_4_7-7_s390.rpm textutils-2_0-2_s390.rpm
logrotate-3_3-1_s390.rpm      tmpwatch-2_0-1_s390.rpm
lvm_0_8final_tar.tar         xosview-1_7_1-2_s390.rpm
mc-4_5_40-2_s390.rpm         xsysinfo-1_7-1_s390.rpm
mcserv-4_5_40-2_s390.rpm

restore > add THE-3_0-1_s390.rpm
restore > ls
./home/sujoma/rpms:
Regina-0_08h-1_s390.rpm          ncurses-4_2-25_s390.rpm
Regina-devel-0_08h-1_s390.rpm  ncurses-devel-4_2-25_s390.rpm
*THE-3_0-1_s390.rpm            pam-0_68-7_s390.rpm
bzip2-0_9_5c-1_s390.rpm       rpm-3_0_3-3_s390.rpm
crontabs-1_7-7_noarch.rpm      samba-2_0_5a-12_s390.rpm
dump-0_4b4-11_s390.rpm        samba-client-2_0_5a-12_s390.rpm
fileutils-4_0-8_s390.rpm      samba-common-2_0_5a-12_s390.rpm
gpm-1_17_9-3_s390.rpm         sed-3_02-4_s390.rpm

```

```

joe-2_8-22_s390.rpm          sh-utils-2_0-1_s390.rpm
knfsd-1_4_7-7_s390.rpm      sysreport-1_0-1_2_noarch.rpm
knfsd-clients-1_4_7-7_s390.rpm  textutils-2_0-2_s390.rpm
logrotate-3_3-1_s390.rpm     tmpwatch-2_0-1_s390.rpm
lvm_0_8final_tar.tar        xosview-1_7_1-2_s390.rpm
mc-4_5_40-2_s390.rpm        xsysinfo-1_7-1_s390.rpm
mcserv-4_5_40-2_s390.rpm
restore > extract
You have not read any tapes yet.
Unless you know which volume your file(s) are on you should start
with the last volume and work towards the first.
Specify next volume #: 1
set owner/mode for '.'? [yn] n
restore > quit

```

- Take a look at the restored file:

```

[root@linux6 /tmp]# ls -l
drwxr-xr-x  3 root  root          4096 May 16 15:12 home
-rw-----  1 root  tty           9720 Jun  2 13:57
rstdir959881841-XXXXVJrA9i

```

The restore information is created in the current directory, and the selected file for restore is THE-3_0-1_s390.rpm in home/sujoma/rpms/ of the current directory.

10.2.2 cpio

The archive utility `cpio` packs files together like `tar` does. It creates and reads file archives. The restore processing includes recovery from data corruption in an archive file.

Because `cpio` has many options, it is not easy to handle. You should get some practice with the `cpio` command so you know how to use it efficiently in the case of a recovery situation.

10.2.2.1 Backup using cpio

By default, `cpio` reads the list of files to be archived from standard input (stdin) and writes the archive to standard output (stdout):

```
cpio -o [aBcv] [-C block_value]
```

Create the list of files to be archived with either the `ls` or `find` command.

```
ls | cpio -oacvB > /archives/backup.cpio
find . -print | cpio -oacvB > /archives/backup.cpio
```

The `find` command is more powerful and can be used to do a partial archive.

Note: It is important to archive the files *relative to the current working directory*. This provides greater flexibility when restoring files. Archives created with absolute paths can only be restored to the original path (except with use of `chroot` or `cpio -I` commands).

If you have a script that determines which files need to be archived, you can pass the output file (by using pipelines) of this script to the `cpio` command using the following command example:

```
cat /tmp/filelist | cpio -oacvB > /archives/backup.cpio
```

For the `cpio` archive function (create a backup), use the `-o` flag with any of the following useful options (`cpio -o<options>`):

- a Reset the access time to the value before the backup executed. This is important if you have processes that rely on a real access time. With the backup, all files would otherwise get the time of the last backup as access time.
- c This is the default for the GNU `cpio` used by Linux. Specify ASCII header format, as this is the most compatible format across platforms.

 If you archive and restore only with GNU `cpio`, you can use the `-newc` option instead. This is the new ASCII header format which supports bigger file systems.
- v Print list of files that are archived to standard error (as standard output is reserved for the archive file itself)
- B Block size for input and output; we recommend you use 5120 bytes per record. The default is 512 bytes per record. You have to remember the block size for the restore process.
- C Like -B, but block size can be any positive integer (`-C<value>`)
- O Only available in the GNU `cpio`, it allows you to specify an output archive file instead of just writing to standard output (`-O filename`).

10.2.2.2 Restore using `cpio`

After the archive is done to restore a `cpio` archive, you need to know the exact block size of the archive (backup). Because Linux uses GNU `cpio`, the header format is detected by `cpio` automatically.

If you do not know how the archive was created, try to validate it with the following command:

```
cpio -itv -C <block_size_you_expect> < /tmp/unknown.archive.cpio
```


For the `cpio` restore, use the `-i` flag with any of the following options .

```
cpio -i [options] [-C block_value] [patterns]
```

Here is a detailed list of the options used to restore files:

t	Generate a table of contents.
d	Create directories as needed.
m	Restore files with original modified-time instead of restore time.
u	Unconditional overwrite all files.
“pattern”	Restore files matching the pattern. Pattern can be repeated several times. Wildcards like an asterisk (*) can be used at any position, including the first character. A pattern */myfile will restore myfile from any path in the directory.
r	Interactively rename files during restore.

When you only need specific files of the archive `/archives/backup.cpio` to be restored, you should first create a table of contents:

```
cpio -ipt < /archives/backup.cpio
```

To restore only the needed files, use the following command, where pattern can be repeated several times and can contain wildcards:

```
cpio -iBdmuv "pattern1" "pattern2" < /archives/backup.cpio
```

To restore all files of the archive, change to the specific directory and use:

```
cpio -iBdmuv < /archives/backup.cpio
```

10.2.2.3 Incremental backup with `cpio`

Because `cpio` uses a file list to determine which files have to be archived, incremental backup is possible. With a little helper file it is even possible to create different concept levels for the backups, as we show in Table 24 on page 215.

We started with a full backup of the directory to be archived, for example, an application directory named `/appl`:

```
cd /appl
```

We created a dummy file in `/appl` to save the time of the backup:

```
touch dummy.cpio.level.0
```

We backed up the files of the directory and all its sub-directories:

```
find . -print | cpio -oacvB > /archives/appl.level.0.cpio
```

For the next backup we create a level 2 backup, which leaves level 1 free for consolidation:

```
touch dummy.cpio.level.2
find . -newer dummy.cpio.level.0 -print | cpio -oacvB >
/archives/appl.level.2.cpio
```

The `find` command with the option `-newer` looks for files that were modified more recently than the file specified, which in our case was the dummy file `dummy.cpio.level.0`.

We perform a level 3 and level 4 backup on the next days. Each backup refers to the dummy file level - 1 (for example, level 4 refers to `dummy.cpio.level.3`). Now we create a level 1 backup:

```
touch dummy.cpio.level.1
find . -newer dummy.cpio.level.0 -print | cpio -oacvB >
/archives/appl.level.1.cpio
```

The `appl` directory itself indicates the sequence of the backups through the timestamps of the dummy files:

```
[root@linux6 /appl]# ls -l
total 0
-rw-r--r--  1 root    root          0 May 10 10:30 dummy.cpio.level.0
-rw-r--r--  1 root    root          0 May 17 10:33 dummy.cpio.level.1
-rw-r--r--  1 root    root          0 May 12 10:31 dummy.cpio.level.2
-rw-r--r--  1 root    root          0 May 14 10:29 dummy.cpio.level.3
-rw-r--r--  1 root    root          0 May 16 10:32 dummy.cpio.level.4
```

10.2.3 tar

The `tar` command is an easy-to-use archive command that is used in everyday system work, not just for backup purposes. It should be used for quick backups, but it is not the first choice for everyday backups.

Tar is more portable than the other backup commands we discussed so far. Even non-UNIX operating systems like Windows can read `tar` archives, for example, with programs like WinZip.

Linux for S/390 provides the GNU version of tar, so in this section we discuss the functionality of GNU tar.

10.2.3.1 Backup using the tar command

To create a backup with the `tar` command using the `-c` option, enter:

```
tar -cvpf /archives/backup.tar "pattern"
```

The options are:

- c Create a new archive.
- v Print information while archiving (verbose).
- f Specify an output file name `-f archive.tar`.
- p Save all file attributes, e.g. file permission bits, ownership bit.
- pattern Archive files matching the pattern. Wildcards are allowed except for the first character.

A command to back up the entire file system would look like:

```
tar -cvpf /archive/full-backup-mydate.tar -directory /  
-exclude=mnt -exclude=proc -exclude=archives
```

Exclude all directories that are mounted temporarily, special file systems like `/proc`, and the directory containing the archives with the `-exclude` option of `tar`.

GNU `tar` provides the useful option `-w` to verify the archive after it was written:

```
tar -cvpWf /archives/backup.tar <pathname><filename>
```

A table of contents for a `tar` file can be created with:

```
tar -tf /archives/backup.tar > /archives/backup.tar.toc
```

10.2.3.2 Restore using the `tar` command

To restore an archive file with the `-x` option, type:

```
tar -xvpf /archives/backup.tar ["pattern"]
```

To preserve the original owner of the files, you have to call `tar` from a superuser id.

The options are:

- x Restore from an archive file.
- v Print information while restoring (verbose).
- f Specify an archive name as input `-f archive.tar`.
- p Restore all file attributes, e.g. file permission bits, ownership bit.
- pattern Restore files matching the pattern. The pathname and filename must *exactly* match the name in the archive file. Wildcards are not allowed. To work with wildcards during restore, use `tar` in combination with the `grep` command.

10.2.3.3 Incremental backup with tar

There are only limited ways to use `tar` to create an incremental backup. You can create a list of files to be archived with the `find` command. This can be done similar to the `cpio` incremental scenario on page 221. For the following example we chose a kind of manual incremental backup based on the `find` command.

The `find` command can produce a list of files that have been changed since a certain period of time (in this example, 24 hours):

```
find / -mtime -1 \! -type d -print > /tmp/mylist
```

All directory entries are excluded with `\! -type d`, and the output of `find` is written to `/tmp/mylist`. If NFS file systems are connected to your system, the file `mylist` will also include files of connected NFS file systems. If you want to exclude the NFS file systems from your backup, you have to edit `mylist` and delete the NFS-specific entries.

Specify the `-T` option of `tar` to read the file name list as input for the backup:

```
tar -cv -T /tmp/mylist -f /archives/backup.incremental.tar
```

10.3 Backup programs and tools

There are some “free” tools that assist and manage the backup process available on the Internet. In addition, you may find backup programs offered by different software vendors. One problem with most of them is that they are not yet ported to Linux for S/390. You can consider porting the “free” tools yourself, or check out the Internet from time to time to see if anyone has ported backup tools for Linux for S/390.

Chapter 11. System maintenance and upgrade

In this chapter we explain the steps needed to set up a new system (build a kernel) or install new software. We look at the installation software using Red Hat Package Manager (RPM). For those who are familiar with OS/390, RPM is a limited SMP/E, and for VM folks, it is VMSES/E.

In addition, we briefly discuss installation from source code, as well as installing or dealing with libraries.

11.1 Where to obtain software

At present, you can build a kernel with all tools on your own, or use the Marist file system. In the future you will have distributions available from companies like SuSE and TurboLinux.

Table 25. URL references to Linux for S/390 service

What you need	Where to find it
kernel source	http://www.kernel.org/
patches	http://oss.software.ibm.com/developerworks/opensource/linux390/index.html
Marist file system	http://linux390.marist.edu/
S/390 RPMs	http://www.linux.s390.org/
SuSE distribution	ftp://ftp.suse.com/pub/suse/s390/pre-suse-s390.iso

At the time of writing, the Marist Web page provides the latest documentation on Linux for S/390 and the associated drivers. You should obtain this documentation as a complement to this redbook.

For more documentation on the SuSE distribution, check the following Web sites:

<ftp://ftp.suse.com/pub/suse/s390/SuSEbookS390.pdf>
<http://www.s390.ibm.com/linux/installfest/l390fpr3.pdf>
<http://www.s390.ibm.com/linux/installfest/l390fin2.pdf>

Even though Linux for S/390 updates are not included, you can track recent updates to software at:

<http://www.freshmeat.net/>

For the Linux for S/390 distribution URLs, see 4.1.1, “Announced distributions” on page 41.

11.2 Overview of upgrade strategies

To upgrade the Linux kernel to a new version, you can do one of the following:

- Download the new kernel version and the associated patches from the Internet, and then compile the new kernel and activate it. This strategy is discussed in 11.6, “Build and customize the kernel” on page 238.
- Completely upgrade to a new Linux version (e.g. by installing a new distribution). This will not only upgrade Linux to a new version, but also create a full root file system. If installed on a device other than your running system, you can copy your modifications to the new file system.
- Use one of the distribution’s tools to upgrade the Linux kernel and the associated software. This will be the most comfortable method in the future.

To update the software on your system you can use packages in different formats like those for Red Hat Package Manager (RPM), .deb (the Debian package manager), or simply tar archives. Using RPM (or .deb) is generally more comfortable than using “plain” tar archives. For one thing, the package managers keep a database of the installed software and check the dependencies before actual installation. Another plus is the comfortable deinstallation that is available with the package managers. In contrast, software installed from tar archives can only be removed manually.

The software packages can come either as precompiled binaries or as source code archives. In the latter case, you’ll have to compile the software before you can use it.

11.3 Software installation with RPM

In this section we describe how to install software using Red Hat Package Manager (RPM), which has become the most widely used package manager in the Linux world.

11.3.1 RPM overview

RPM, used by most Linux distributions, keeps track of all kinds of information about packages.

Among other things, RPM does the following:

- It remembers what packages you have installed.
- It remembers which files came with which packages.
- It allows you to install software (packages) with one command.
- It removes software (packages) with a single command.
- It writes warning messages if installing a package is going to overwrite needed files.
- It warns if removing a package will remove a file needed by other packages.

RPM has five basic functions: installing, uninstalling, upgrading, querying, and verifying software packages.

Every package has a common structure (design) for the package name e.g. `bzip2-0.9.5c-1_s390.rpm`, which includes the package name (`bzip2`), version (`0.9.5c`), release (`1`), and architecture (`s390`).

Although the software itself is the same, each distribution will tailor and ship its own RPM packages of the same software. The differences between distributors are mostly in the area of file locations and possible patches for RPM packages. Always try to get the RPM package adapted for the distribution you use.¹

11.3.2 The RPM database

The RPM database contains information about each piece of software installed with RPM. Database information can be accessed through the `rpm` command.

The `initdb` option creates a new, empty RPM database:

```
rpm --initdb
```

No information can currently be migrated or copied into a RPM database. To insert information about packages, you have to install the software using the `rpm` command.

To rebuild a broken RPM database you can use the `rebuilddb` option; this can be a lifesaver when RPM fails due to a broken database:

```
rpm --rebuilddb
```

¹ The Marist big-file system is not a distribution in any formal sense, it is a dump of a running Linux for S/390 image. You can find some RPMs for S/390-specific Linux on the Web page at: <http://www.linux.s390.org/>

Note that the `-rebuilddb` command restores a previous backup of the RPM database; it does not validate the whole system for installed packages.

11.3.3 Querying package information

Querying the RPM database of installed packages is accomplished with `rpm -q`. This query function examines the contents of a package (list of files, space needed, destination of files) and also checks dependencies and possible conflicts with other packages.

RPM offers many ways to specify what information to display about a queried package. The following list of options can be used to select the information you are interested in:

```
rpm -q[ildc] [-p package] [-f <file>]
```

The options are:

- i Display detailed package information (name, description, release, size, build date, install date and other helpful information)
- l List all files within this package
- d List all files marked as documentation files
- c List all files marked as configuration files
- a Query all currently installed packages
- f <file> Query the package owning <file>
- p <packagefile> Query the package <packagefile>

When you enter the `rpm -q` command without the `-p<packagefile>` option, specify the software name (e.g. `bzip2`):

```
rpm -qi bzip2
```

The following information about the package is displayed. (If the package is not installed, the message `package bzip2 is not installed is displayed`.):

```
Name      : bzip2                      Relocations: /usr
Version   : 0.9.5c                    Vendor: Thinking Objects
Software
Release   : 1                          Build Date: Wed Apr 27 08:18:21
2000
Install date: Fri May 6 17:38:25 2000  Build Host: piranha.think
Group     : Applications/File          Source RPM:
bzip2-0.9.5c-1.src.rpm
Size      : 278402                      License: GPL
```



```
Packager      : Fritz Elfert <felfert@to.com>
Summary       : A file compression utility.
```

Description :

Bzip2 is a freely available, patent-free, high quality data compressor. Bzip2 compresses files to within 10 to 15 percent of the capabilities of the best techniques available. However, bzip2 has the added benefit of being approximately two times faster at compression and six times faster at decompression than those techniques. Bzip2 is not the fastest compression utility, but it does strike a balance between speed and compression capability.

Install bzip2 if you need a compression utility.

To list the files that belong to the bzip2 package, use `rpm -ql`. All files owned by this package are displayed with full pathnames.

To list all RPM installed packages, issue the command:

```
rpm -qa
```

11.3.4 Checking dependencies

Whenever you install new software, you should always check the dependencies. (Otherwise, if dependencies are not completely resolved, you may end up with a system that does not support basic commands.) This is especially important when updating basic software like RPM itself, compilers, or packages with shared libraries.

The `--test` option of RPM checks whether a package is installable and which components are missing, if any. It also checks for conflicts with other software packages:

```
[root@linux6 rpms]# rpm -Uv --test samba-2_0_5a-12_s390.rpm
error: failed dependencies:
    pam >= 0.64 is needed by samba-2.0.5a-12
    samba-common = 2.0.5a is needed by samba-2.0.5a-12
    /usr/bin/killall is needed by samba-2.0.5a-12
    fileutils is needed by samba-2.0.5a-12
    sed is needed by samba-2.0.5a-12
    /bin/sh is needed by samba-2.0.5a-12
    libpam.so.0 is needed by samba-2.0.5a-12
    libtermcap.so.2 is needed by samba-2.0.5a-12
    /bin/csh is needed by samba-2.0.5a-12
    /bin/sh is needed by samba-2.0.5a-12
    /usr/bin/awk is needed by samba-2.0.5a-12
```

In cases where you want to install a software package and ignore any dependencies, use the option `--nodeps` to deactivate dependency checking. This option can be useful if non-RPM software was installed and you have manually checked that all the dependencies will correctly resolve. In the big file system from Marist distribution, not all installed software packages are RPM packages.

11.3.5 Install and update a package

RPM automates the process of installing binary software on the system. For source packages, you have to perform post-installation steps.

Two options can be used to install a package. The `-i` option installs a new package:

```
rpm -ivh samba-common-2_0_5a-12_s390.rpm
```

The `-U` option updates (and installs) a package. In general, use the `-U` option for installing and upgrading RPM packages.

```
rpm -Uvh samba-common-2_0_5a-12_s390.rpm
```

To specify the verbose operation, use the `-v` option. By specifying `-h`, a progress indicator (hash marks) is displayed while the package is being installed.

Configuration files that are part of the package are stored with the extension `.rpmorig`. Check these configuration files for new parameters. A warning is displayed for every new configuration file:

```
[root@linux6 rpms]# rpm -Uvh --nodeps sh-utils-2_0-1_s390.rpm
warning: /etc/pam.d/su saved as /etc/pam.d/su.rpmorig sh-utils
```

11.3.6 Post-installation steps for source RPMs

A source RPM contains source code in a tar file, along with specifications and possibly additional patches. Because the Marist file system is partly based on Red Hat structure, the RPM installation step will place these files in subdirectories under `/usr/src/redhat/`.

In a non-Red Hat Linux system, they might be stored in subdirectories under `/usr/src/packages/`.

As a simple example to demonstrate this, we use the `linux_logo` source file and install it using the following RPM command:

```
rpm -iv linux_logo-3_0-2_src.rpm
```

This will create two directories, SOURCES and SPECS:

```
/usr/src/redhat/SOURCES/linux_logo-3.0.tar.gz
/usr/src/redhat/SPECS/linux_logo.spec
```

Next, we change to the SPECS directory:

```
cd /usr/src/redhat/SPECS/
```

We issue the RPM command with the build option `-bi`, specifying the specs file:

```
rpm -bi linux_logo.spec
```

The `rpm -bi` command will do some preparation steps such as untar the tar file, apply patches if necessary, and compile the software:

```
Executing: %prep
+ umask 022
+ cd /usr/src/redhat/BUILD
+ cd /usr/src/redhat/BUILD
+ rm -rf linux_logo-3.0
+ /bin/gzip -dc /usr/src/redhat/SOURCES/linux_logo-3.0.tar.gz
+ tar -xvzf -
drwxr-xr-x vince/users      0 1999-04-02 14:21 linux_logo-3.0/
-rw-r--r-- vince/users    13661 1999-04-02 01:29
linux_logo-3.0/linux_logo.c
.....
-rw-r--r-- vince/users     2949 1999-03-26 19:36
linux_logo-3.0/linux_logo.1
+ STATUS=0
+ [ 0 -ne 0 ]
+ cd linux_logo-3.0
++ /usr/bin/id -u
+ [ 0 = 0 ]
+ /bin/chown -Rhf root .
++ /usr/bin/id -u
+ [ 0 = 0 ]
+ /bin/chgrp -Rhf root .
+ /bin/chmod -Rf a+rX,g-w,o-w .
+ exit 0
Executing: %build
+ umask 022
+ cd /usr/src/redhat/BUILD
+ cd linux_logo-3.0
+ CFLAGS=-O2 -fomit-frame-pointer
+ touch dummy.tmp~
+ make clean
rm -f *.o
```

```

rm -f linux_logo
rm *~
+ make
Compiling for Linux
Edit the Makefile to change Platform
Edit defaults.h to change Default Values
gcc -O2 -Wall -DLINUX_ANSI -c linux_logo.c
gcc -O2 -Wall -DLINUX_ANSI -c sysinfo.c
In file included from sysinfo.c:23:
sysinfo_ix86.c: In function `get_hw_info':
sysinfo_ix86.c:60: warning: `mem' might be used uninitialized in this
function
gcc -O2 -Wall -DLINUX_ANSI -c bogomips.c
gcc -O2 -Wall -DLINUX_ANSI -o linux_logo linux_logo.o bogomips.o
sysinfo.o
+ exit 0
Executing: %install
+ umask 022
+ cd /usr/src/redhat/BUILD
+ cd linux_logo-3.0
+ mkdir -p /var/tmp/linux_logo-root/usr/bin
/var/tmp/linux_logo-root/usr/man/man1
+ install -s linux_logo /var/tmp/linux_logo-root/usr/bin/linux_logo
+ install linux_logo.1
/var/tmp/linux_logo-root/usr/man/man1/linux_logo.1
+ exit 0
Processing files: linux_logo
Executing: %doc
+ umask 022
+ cd /usr/src/redhat/BUILD
+ cd linux_logo-3.0
+ DOCDIR=/var/tmp/linux_logo-root/usr/doc/linux_logo-3.0
+ export DOCDIR
+ rm -rf /var/tmp/linux_logo-root/usr/doc/linux_logo-3.0
+ /bin/mkdir -p /var/tmp/linux_logo-root/usr/doc/linux_logo-3.0
+ cp -pr ANNOUNCE.logo BUGS CHANGES README USAGE.FAQ samples
/var/tmp/linux_log0
+ exit 0
Finding Provides: (using /usr/lib/rpm/find-provides)...
Finding Requires: (using /usr/lib/rpm/find-requires)...
Requires: ld.so.1 libc.so.6 libc.so.6 (GLIBC_2.0) libc.so.6 (GLIBC_2.1)

```

Instead of doing prep and build in one step by using the `-bi` option, you can do a two-step process by using the `-bp` and `-bc` options, as follows.

For the %prep phase:

```
rpm -bp
```

For the %build phase:

```
rpm -bc
```

After the build is done, you will find a new directory called `/usr/src/redhat/BUILD/linux_logo-3.0`.

Refer the to RPM man page for more information. For complete information on the necessary post-installation steps for source code RPMs, refer to the installation information provided with the package itself.

The build-root set in the %install phase is:

```
/var/tmp/linux_logo-root/
```

Finally, you can test the just-installed software with the command:

```
/var/tmp/linux_logo-root/usr/bin/linux_logo
```

11.3.7 Removing a package

Uninstalling (removing) a package is just as simple as installing if you use the `-e <packagename>` option with the package name:

```
rpm -ev joe
```

Be aware that you may encounter a dependency error when uninstalling a package if other installed packages depend on the one you trying to uninstall. To force RPM to ignore that type of error and uninstall the package anyway, use the `--nodeps` option. However, this may not be desirable since the dependent package may fail after your uninstall is done.

11.4 Software installation with tar

Apart from using the RPM method, you have other options for downloading tar files containing the desired software in source or binary format. Using these options, however (and in contrast to RPM), you have to check dependencies and prerequisites manually. Therefore, we recommend that you use the RPM software packages whenever possible.

In this section we briefly describe how to deal with the tar installation method. A tar file is similar to a package; it contains all files and directories, including their file permissions and ownership. Tar files can contain either binaries

(sometimes accompanied by an installation program like setup or install), or source (including a configure script made by the GNU autoconf system).

The tar packages are usually gzip-compressed with the file ending .tar.gz. Use the `-z` option with the tar command because otherwise, tar will not recognize the file.

Before extracting a tar file, check the contents with the following command:

```
[root@linux6 hdm]# tar -tzf samba-2_0_7.tar.gz | more
samba-2.0.7/
samba-2.0.7/COPYING
samba-2.0.7/docs/
samba-2.0.7/docs/faq/
samba-2.0.7/docs/faq/Samba-Server-FAQ-2.html
samba-2.0.7/docs/faq/Samba-Server-FAQ-1.html
samba-2.0.7/docs/faq/Samba-meta-FAQ.txt
samba-2.0.7/docs/faq/sambafaq-2.html
samba-2.0.7/docs/faq/sambafaq-3.html
samba-2.0.7/docs/faq/sambafaq-1.html
...
samba-2.0.7/source/groupdb/groupfile.c
samba-2.0.7/source/groupdb/aliasfile.c
samba-2.0.7/source/configure
```

To get even more detailed information, you can include the verbose option (`tar -tzvf <myfile.tar.gz>`). From the example above you can see that every file in the tar file has samba-2.0.7 as its first directory. Therefore, you do not need to make a separate directory, since tar will create one.

Now untar the package with `tar -xzvf <myfile.tar.gz>`.

You should find a README or INSTALL file that describes the steps to install the software you have just untared from the source tar files. This file normally point to a configure script. This script will auto-detect information about the system and automatically configures the build environment for the source code.

After running `configure`, the next step is to compile the software.

Note

Because the S/390 architecture in Linux is very “young”, you may find some configure scripts that do not include the S/390 architecture. If you encounter this problem, check the `config.guess` and `config.sub` files for the appropriate sections on the S/390 architecture. If you don't have the S/390 sections in these files copy them from a package that installed correctly in your Linux for S/390 system.

After you have prepared for the compile, use the `make` command to compile the software and the `make install` command to copy the executables into the appropriate directory for usage.

A more complete example on installing software from a source tar file is documented in 20.1.2, “Installing from source in the original package” on page 389.

11.5 Updating libraries

There are two kinds of libraries: static and dynamic (meaning dynamically loadable). *Static* libraries (filenames `*.a` for Archive) are linked to program binaries at compile time. *Dynamic* libraries are loaded at runtime.

Dynamic libraries are naturally implemented as shared libraries (filenames `*.so` for Shared Object). When more than one process using a certain shared library is running, these processes share the memory pages containing the library code.

Note: Although the terms *shared* and *dynamic* are often used interchangeably, we use the term *shared* in this discussion.

The dynamic feature means not having the same library code in multiple executables, which saves disk space. The shared feature means not holding multiple instances of the library code in memory, which saves memory space.

For statically linked libraries, the executable holds its own copy of the library routines. The program has to be relinked to use a new version of the library. Updated shared libraries will take effect the next time the dynamic loader `ld.so` is called by a program to load these libraries. This is normally the time a program is started, but it can also occur later.

With shared libraries, new versions automatically take effect. A significant advantage of this is that corrected versions of shared libraries can magically “repair” all corresponding binaries without the need of recompilation. Note, however, that new major version of a dynamic library *cannot be used with an executable built with a previous version*.

As an example, let’s look at the bzip2 library files, which can be found in the directory `/usr/lib`:

```
libbz2.a ❶  
libbz2.so -> libbz2.so.0.0.0 ❸  
libbz2.so.0 -> libbz2.so.0.0.0 ❸  
libbz2.so.0.0.0 ❷
```

- ❶ This is the static version in the library package.
- ❷ This is the dynamic version in the library package (shared library).
- ❸ These are symbolic links for the dynamic linker `ld.so`.

When a program is started, the dynamic linker looks for a library name `libname.so.major_version` or `libname.so`. `Ld.so` always searches the directories `/lib` and `/usr/lib` first. Then it searches the directories listed in the files `/etc/ld.so.conf`.

If an environment variable `LD_LIBRARY_PATH` is set, the linker will search in the directories specified in this variable *first*, and then continue with the standard search order.

The `ldd` command examines which shared libraries are required by an executable:

```
[root@linux6 bin]# ldd /bin/rpm  
librpmbuild.so.0 => /usr/lib/librpmbuild.so.0 (0x40019000)  
libdb.so.2 => /lib/libdb.so.2 (0x40048000)  
libz.so.1 => /usr/lib/libz.so.1 (0x40058000)  
libbz2.so.0 => /usr/lib/libbz2.so.0 (0x40068000)  
librpm.so.0 => /usr/lib/librpm.so.0 (0x40076000)  
libc.so.6 => /lib/libc.so.6 (0x400c3000)  
/lib/ld.so.1 => /lib/ld.so.1 (0x40000000)
```

11.5.1 Upgrading shared libraries

When upgrading a shared library, it is important that the symbolic link `libname.so.major_version` always points to a library file during the process.

Copy the new library file to the correct directory:


```
cp -a libbz2.so.0.0.1 /usr/lib/
```

Both versions of the dynamic shared library are now in /usr/lib:

```
libbz2.so -> libbz2.so.0.0.0
libbz2.so.0 -> libbz2.so.0.0.0
libbz2.so.0.0.0
libbz2.so.0.0.1
```

Change the symbolic links to the new library. It is important to *update the symbolic link in one step* by issuing `ln -sf`. Otherwise, commands using the library will no longer work after the link is removed:

```
ln -sf /usr/lib/libbz2.so.0.0.1 /lib/libbz2.so.0
ln -sf /usr/lib/libbz2.so.0.0.1 /lib/libbz2.so
```

Now the symbolic links point to the new library version:

```
libbz2.so -> libbz2.so.0.0.1
libbz2.so.0 -> libbz2.so.0.0.1
libbz2.so.0.0.0
libbz2.so.0.0.1
```

Run `ldconfig` to regenerate the shared library cache used by `ld.so`. The command `ldconfig` is used to maintain the shared library system. If a new shared library is installed on the system, a new entry has to be created in `/etc/ld.so.cache`. In addition, `ldconfig` sets the symbolic links.

When you install shared libraries with RPM, the post-installation script of RPM runs the `ldconfig`.

11.5.2 Resolving incompatibilities

Note

After the distributions for Linux for S/390 become generally available, you should not expect incompatibilities with the library structures on your system. There is, however, a slight chance that you might run into library incompatibilities with the Marist “distribution”, due to the speed at which these packages were hammered together, just to get running code up, working and out to people, so you should not expect the usual standards of quality assurance, etc. as would be found with a regular distribution.

Shared libraries with different major numbers are incompatible. This is no problem for programs that need the `libname.so.major_version` library, as you can simply keep both versions on the system.

In case this is no solution, you should either upgrade the programs that have this incompatibility problem or use a different `LD_LIBRARY_PATH` environment variable.

However, different environment variables will only work for a limited number of programs that suffer from incompatibility. For each of these affected programs, you should rename the executable and write a kind of wrapper script with the original executable name. The script should first set the environment variable `LD_LIBRARY_PATH` and then execute the binary. The old shared library must be in a different directory than the actual version.

In our example the wrapper script for the program `mypg` (which we renamed to `mypg.prog`) could look like this:

```
#!/bin/sh
export LD_LIBRARY_PATH=/usr/lib/diffdir
exec mypg.prog
```

So if the user calls program `mypg`, the script `mypg` is called instead, which will export and use the old library and then call the program (`mypg.prog`) with the right (old) library.

11.6 Build and customize the kernel

Installing a kernel from source sounds like a complicated task, but it isn't really that difficult. If the kernel you have supports all the options you want, then there's no need to change it.

However, what if you want a feature that was not selected, or want to remove something that you're not going to use at all? Or what if there is a patch or workaround needed to remove a bug or vulnerability discovered in the kernel version you are running, and you need to compile a kernel from source?

Follow these steps to create a new kernel from source:

1. Start by establishing a second bootable device in case something goes wrong. This is always sound advice, and should be part of your site administration policy anyway.

Note that `siloboot-2.2.x` only allows one kernel per boot device, unlike the linux loaders on most other Linux platforms, which often allow one of several

different kernels to be selected at boot time. Thus, for s390-ibm-linux, it is even more important that you have an alternate kernel boot path.

This is easier on VM, but can also be done in an LPAR.

2. Get the source.
3. Install the source.
4. Patch it with S/390 specifics.
5. Clean up all extraneous files.
6. Check the basic configuration.
7. Configure the kernel.
8. Check all dependencies.
9. Compile the kernel.
10. Activate the image.
11. Create the boot record.
12. Boot with the new boot record.

All distributions should give you the opportunity to install the Linux system including the source code and compiler tools, and therefore enable you to customize the current kernel to your needs. In that case you would start with step 5.

11.6.1 Preparing a second bootable device

Regardless of which operating system you run, you should always have a standby method available to get the system back up and running in case something goes down and refuses to come back up the way you want it to.

Note

This section was written from the perspective of an LPAR installation. The differences between PR/SM and VM are subtle, but one of the most striking (and underestimated) differences is that the “unit of allocation” for disk space on an LPAR is 3390-x Full-Pack, whereas on VM it would be very reasonable to assign a couple of 20-cylinder (3390) minidisks to act as boot devices. The same restriction applies to swapdevices.

There are two ways to ensure that you can boot a valid Linux system:

- Keep a bootable tape or VM initrd image available. This will load the starter file system you used during your first installation of Linux, so you

may be able to repair a system that cannot get up. However, be aware that your production Linux will be down for that time.

- Make a second device bootable. Then you will may be able to switch boot devices and keep the production root file system accessible.

Since there should already be a bootable tape or ramdisk, let's discuss how to set up a second bootable device. You need a Linux-formatted device with a file system on it:

```
[dasdfmt -b 4096 -f /dev/dasdc]
mke2fs -b 4096 /dev/dasdc1
```

Create a mount point and mount the device on it:

```
cd /
mkdir /appl
mount /dev/dasdc1 /appl
mkdir /appl/boot.alternate
```

Now a second boot directory is needed, because it has to be on the same device as the silo boot record. In the example we first created a directory /appl. Because the boot directory is very small, you can use the /appl directory, for example, for your own applications without losing a whole device.

Note: You can skip these steps and create a boot.alternate directory on any Linux-formatted device with a file system that is already mounted.

Copy your existing /boot directory to the /appl/boot.alternate directory:

```
cd /boot
cp -a /appl/boot.alternate
```

Now you have the same kernel on both boot directories. Because you know that this kernel boots cleanly, you can use `silo` to create a boot record on the new device, and because you use the same root file system, the kernel parameters stay the same:

```
cd /appl/boot.alternate/
silo -f image -d /dev/dasdc -p parm.line -b ipleckd.boot
[root@linux6 /boot.alternate]# silo -f image -d /dev/dasdc -p parm.line
-b ipleckd.boot
o->image set to image
o->ipldevice set to /dev/dasda
o->parmfile set to parm.line
o->bootsect set to ipleckd.boot
IPL device is: '/dev/dasdc'
bootsector is: 'ipleckd.boot'...ok...
```

```
Kernel image is: 'image'...ok...
parameterfile is: 'parm.line'...ok...
ix 0: offset: 00007e count: 0c address: 0x00000000
ix 1: offset: 00008b count: 80 address: 0x0000c000
ix 2: offset: 00010b count: 80 address: 0x00008c000
ix 3: offset: 00018b count: 57 address: 0x00010c000
ix 4: offset: 0001e4 count: 01 address: 0x00008000
Bootmap is in block no: 0x0000007d
```

Now make the `appl` directory permanently visible in the root file system. For `boot.alternate` itself, this is mandatory because the boot record holds all-important data:

```
vi /etc/fstab
/dev/dasdf1      /          ext2  defaults,errors=remount-ro 0 1
/dev/dasdg1     /mnt/swap  ext2  defaults      0   2
/dev/dasdi1     swap      swap  defaults      0   0
/mnt/swap/swapfs1  swap      swap  defaults      0   0
/mnt/swap/swapfs2  swap      swap  defaults      0   0
none           /proc     proc  defaults      0   0
/dev/dasdc1     /applext2  defaults,errors=remount-ro 0 3
```

After these steps, you are ready to test the second boot device:

```
shutdown -h now
```

When the system is down, reload the system, specifying the new IPL address during load from HMC or via the VM command `#CP IPL <device_number>` CLEAR.

11.6.2 Get the Linux kernel source

The kernel and the Linux for S/390 specific patches are available on the Internet. You can download the kernel itself from the official kernel Web site or from the country mirror nearest you:

```
http://www.kernel.org/
ftp://ftp.kernel.org
ftp://ftp.de.kernel.org/ (for example, in Germany)
```

In our scenario we downloaded the Linux 2.2.15 file `linux-2.2.15.tar.gz`. Note that the S/390 code has now been integrated into the rest of the mainline kernel, so you probably won't need to apply the S/390 patches unless there is a specific patch which you require which hasn't been integrated into the main kernel tree yet.

The version number of the kernel follows a specific order. An even number like 2.0.x or 2.2.x represents a stable Linux release, while an odd number like 2.1.x or 2.3.x is a development release.

Note that the S/390 was initially written to fit into the 2.2.x (stable) Kernel Tree. It is unusual to integrate a change as large as a completely new architecture into the stable kernel tree, but this was the design decision taken by the Böblingen team and accepted by the kernel maintainers, and in retrospect, it worked very well. It gave the developers a more stable platform to develop with, and the assurance that any bugs they were seeing were more likely to be caused by the new too chain or implementation, and not because the kernel itself might be having growing pains due to the ongoing development work of the other (unstable) kernel developments.

The Linux for S/390 specific patch files have the ending `diff.tar.gz` and can be found on the following Web site:

```
http://oss.software.ibm.com/  
developerworks/opensource/linux390/download_src.html
```

In our case, we downloaded the patch file `linux-2.2.15-s390.diff.tar.gz`.

You will also need the object code only (OCO) modules from the Internet that are compatible with the new kernel. We downloaded `lcs-2_2_15.tar.gz` for the module `lcs.o` from:

```
http://oss.software.ibm.com/  
developerworks/opensource/linux390/download_obj.html.
```

One problem specific to the upgrade from Linux 2.2.14 to Linux 2.2.15 was an enhancement in the `binutils` and `gcc` compiler. The versions on the Marist-2.2.14 file system had no support for the new CSP instruction used in Linux kernel 2.2.15.

Without the new utilities we got an error during kernel make, saying that the compiler does not know the new CSP instruction:

```
{standard input}: Assembler messages:  
{standard input}:1654: Error: Unrecognized opcode: `csp'
```

You can ignore the error for the moment but the long-term production solution should be to get the right set of the new utilities which fits the new 2.2.15 kernel.

11.6.3 Recompiling the S/390 tool chain (binutils and gcc)

For this task, we first had to compile the binutils and the gcc compiler itself because of the new CSP instruction. (You will probably not have to rebuild the compiler tool chain yourself unless there are further changes to the binutils and gcc components and the distribution you are using has not yet provided you with a newer gcc.) Normally you would just install the newer RPMs for binutils and gcc instead.

However, if you need to do it “the hard way”, here’s how.

1. Install the binutils assembler to support the CSP instruction:

- To keep all data in one place, install it into the source directory:
`cd /usr/src`
- Check the contents of the tar file: `tar -tvzf binutils.tar.gz`
- Unpack the tar file: `tar -xvzf binutils.tar.gz`
- Create a directory for the build process to keep it separate from the source:
`mkdir /usr/src/binutils-build`
- Change to the new directory: `cd binutils-build`
- Call the script command to capture the terminal messages of the next step into a file: `script binutils.config.out`
- Call configure to prepare for compilation:
`../binutils/configure --prefix=/usr/local`
- Exit the script command: `exit`

Review the messages in the file `binutils.config.out`, and then compile binutils by using `make`. You can capture the terminal messages with the `script` command, as done before.

Copy the binaries into `/usr/local` (the prefix specified during configure processing) with `make install`. Note that you have to be root for this task. It may replace the binutils, including the shared libraries, that are currently in use. Decide if you want to do a backup of these files before you call the `make install`.

- Change the PATH environment variable to reflect `/usr/local/bin` (if not already present).
- Check the PATH settings with: `echo $PATH`
- Set the path variable (example in bash syntax) for the current shell with:
`export PATH=/usr/local/bin:$PATH`

2. Install the gcc compiler:

- As with binutils, start in the source directory: `cd /usr/src`
- Check contents of the tar file: `tar -tvzf gcc.tar.gz`

- Unpack the tar file: `tar -xvzf gcc.tar.gz`
- Create a directory for the build process to keep it separate from the source: `mkdir /usr/src/gcc-build`
- Change to the new directory: `cd gcc-build`
- Call the script command to capture the terminal messages of the next step into a file: `script gcc.config.out`
- Call the configure to prepare for compilation:


```
../gcc/configure --prefix=/usr/local \
--enable-languages="c" \
--with-newlib
```
- Exit the script command: `exit`

Then review the messages in the file `gcc.config.out`

- Compile gcc: `make`
- You can capture the terminal messages with the `script` command, as done before.

The compile lasts a while. It will end with one error in `libiberty`, which can be ignored:

```
make[1]: *** [strerror.o] Error 1
make[1]: Leaving directory
`/usr/src/gcc-build/s390-ibm-linux/libiberty'
make: *** [all-target-libiberty] Error 2
```

- Copy the binaries into `/usr/local` (the prefix specified during configure processing) using: `make install`
- You have to be root for this task. It may replace the `gcc`, including shared libraries that are currently in use. Decide if you want to do a backup of these files before you call the `make install`.

Now you are positioned to install the new kernel.

11.6.4 Preparing `/usr/src/linux`

For this task, you have to be in the `/usr/src` directory. If you already have a kernel source in this directory, you should have a symbolic link from `linux` to `linux-2.2.14`. Remove this link to avoid overwriting `linux-2.2.14`.

```
cd /usr/src
```

In the tar file, locate the name of the tar root directory:

```
tar -tvzf /usr/src/download/linux-2_2_15_tar.gz
```

Untar the file:

```
tar -xvzf /usr/src/download/linux-2_2_15_tar.gz
```


Now you have a directory Linux that contains the linux-2.2.15 source. Rename it, as follows:

```
mv /usr/src/linux /usr/src/linux-2.2.15
```

Then reestablish the Linux link:

```
ln -s /usr/src/linux-2.2.15 /usr/src/linux
```

Ensure that both these links point to the new kernel:

```
ln -sf /usr/src/linux/include/linux /usr/include/linux
ln -sf /usr/src/linux/include/asm /usr/include/asm
```

Note

If you are using the kernel NFS daemon `knfsd`, you will experience problems when setting the link to the new Linux version. The NFS daemon reads the file `/usr/src/linux/System.map` and detects a higher version of the kernel. You will not be able to start the daemon with the new kernel source directory. In this case, establish the link to `Linux-2.2.14` and work for the next steps with the `Linux-2.2.15` directory instead of the symbolic link `Linux`.

Next untar the s390 diff tar file:

```
tar -xvzf /usr/src/download/linux-2_2_15-s390_diff_tar.gz
```

This step will provide the file `linux-2.2.15-s390.diff` that is used with the patch command.

From the `/usr/src` directory, patch the original kernel with the S/390 architecture specifications. This patch will assume that the files are in directory `linux-2.2.15`:

```
/usr/bin/patch -sp0 < /usr/src/download/linux-2.2.15-s390.diff
```

Patch is a very “silent” command, so if you get no feedback, it means it was successful.

After these steps, your `/usr/src` directory should look similar to this :

```
drwxr-xr-x 14 tux1      tuxgrp    4096 Apr 15 08:29 binutils
drwxr-xr-x 10 root      root      4096 May 17 18:47 binutils-build
drwxr-xr-x 17 tux1      tuxgrp    4096 May  4 09:44 gcc
drwxr-xr-x  7 root      root      4096 May 17 15:05 gcc-build
lrwxrwxrwx  1 root      root          21 May 17 15:53 linux ->
/usr/src/linux-2.2.15
drwxr-xr-x 14 1046     xfs       4096 Jan  4 13:12 linux-2.2.14
drwxr-xr-x 14 sujoma  sujoma    4096 May  4 20:16 linux-2.2.15
```

11.6.5 Configure and compile the kernel

Note

This section discusses compiling a standard 2.2.xx kernel assuming that all the S/390-specific code and patches have already been applied.

For the next steps, change to the directory with the Linux-2.2.15 source:

```
cd /usr/src/linux
```

Clean up all extraneous files from any previous kernel builds; that is, remove all object modules to force a complete rebuild of the kernel:

```
make clean
```

If you do not know how to set the parameters, just start with these defaults. It worked for our installation.

There are three basic frontends provided for you to select the kernel options you would like to have for this kernel. It doesn't really matter which one you choose, although most people use `make menuconfig`.

All of these frontends produce a file called `/usr/src/linux/.config`, which is then used to actually generate the kernel you have specified. If you are unsure of which values to choose, note that the defaults (provided in `/usr/src/linux/arch/s390/config.in`) will produce a working kernel for most environments.

1. Do a `make config` (and `make oldconfig`)

This is the no-frills bash script that prompts for every option and goes through it step by step. (There is nothing fancy about it, and it should even work with the 3215 teletype.) The possible answers are specified in brackets after the question. Default settings are in capital letters. If you type a wrong answer you cannot go back, but you can exit with Control-C and restart the config dialog from the beginning. Here is a short example of what it looks like:

```
[root@linux6 linux-2.2.15]# make config
rm -f include/asm
( cd include ; ln -sf asm-s390 asm)
/bin/sh scripts/Configure arch/s390/config.in
#
# Using defaults found in .config
#
*
* Code maturity level options
```

```

*
Prompt for development and/or incomplete code/drivers
(CONFIG_EXPERIMENTAL) [Y/n/?] Y
*
* Processor type and features
*
Symmetric multi-processing support (CONFIG_SMP) [Y/n/?]

```

There is a slight variation on this called `make oldconfig` which will regenerate a kernel using the same settings as found in the (existing) `.config` file. This is normally only used when you want to recompile a new (2.2.15) kernel using the `.config` file you remembered to copy over from your last kernel (2.2.14) version. It is even polite enough to recognize when there are new parameters available which weren't in the older kernel source tree, and will ask for clarification if it encounters an unanswered question.

2. Do a `make menuconfig`

This is the classic character interface. It will work with any well-behaved ssh, xterm or telnet session. You can navigate forwards and backwards through the options and call help functions if you need further explanation. The first screen of `menuconfig` is shown in Figure 80 on page 247.

```

Linux Kernel v2.2.15 Configuration
Main Menu
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

Code maturity level options --->
Processor type and features --->
Loadable module support --->
General setup --->
$/390 block device drivers --->
$/390 Network device support --->
$/390 Terminal and Console options --->
--- Character devices
[*] Unix98 PTY support
(256) Maximum number of Unix98 PTYs in use (0-2048)
v(+)

<Select>  <Exit >  <Help >

```

Figure 80. Kernel configuration with `menuconfig`

You can use the up and down cursor keys to scroll through the various option lines. You can use `Enter` to select an option line (a line with “`-->`” implies that there is a further menu “behind” this option line).

When you get to a question, you can answer it with “space” or one of the letters as described on the top four lines of the menu. Use the Tab key to navigate between the body of the application and the bottom line (for example, to “exit” a submenu and return to the higher level question).

3. Do a `make xconfig`

Linux also provides an X application to configure the kernel. It is very “point and shoot”. It does, of course, presume that you already have a kernel up and running well enough to be using X-windows over TCP/IP.

Decide which tool you want to use and configure the kernel options with:

```
make menuconfig
```

When finished, take the exit option. It will ask you if you want to save this kernel configuration. If you are not satisfied (or if you just wanted to browse the settings and touched one of them by accident), don’t panic. Just exit, and answer no to the question `save the kernel configuration?`

Next, the dependencies for the kernel you have just defined need to be generated:

```
make dep
```

The kernel build itself will write a lot of messages. If you want to keep a record of this for later, call the `script` command to capture the terminal messages:

```
script /home/sjm/kernel.make
```

Compile the kernel:

```
make image
```

Use the `exit` command to close the script process. You can review all messages of the kernel compile in `/home/sjm/kernel.make`.

The new kernel image is stored in the `/usr/src/linux-2.2.15/arch/s390/boot` directory:

```
[root@linux6 boot]# ls -l
total 1520
-rw-r--r--  1 root    root          1011 May 17 18:24 Makefile
-rwxr-xr-x  1 root    root       1507976 May 17 19:51 image
-rw-r--r--  1 root    root          5589 May 17 18:24 ipldump.S
-rwxr-xr-x  1 root    root          1024 May 17 19:51 ipldump.boot
-rw-r--r--  1 root    root          8953 May 17 18:24 ipleckd.S
-rwxr-xr-x  1 root    root          2048 May 17 19:51 ipleckd.boot
```

```
-rw-r--r-- 1 sujoma sujoma 4866 Jan 4 13:12 iplfba.S
-rwxr-xr-x 1 root root 1024 May 17 19:51 iplfba.boot
```

If an option M (for module) was selected during kernel configuration, create the modules:

```
make modules
```

Copy them into `/lib/modules/linux-2.2.15`:

```
make modules_install
```

If you find this hard to remember, you can do it on two lines:

```
make menuconfig
make clean dep modules modules_install image
```

11.6.6 Install object code only (OCO) modules

If you require an OCO module, you now have to make sure that it is placed in the correct directory (for example: `/lib/modules/2.2.15/net/lcs.o`).

Get the `/etc/rc.d/rc.sysinit` script from the Marist 2.2.15 file system. Note that it has changed a lot, especially in the module loading section. In 2.2.14, it did a `insmod /lib/modules/2.2.14/lcs.o`. Now it looks for the modules with the `depmod` command:

```
[root@linux6 /]# depmod -a -v
/lib/modules/2.2.15/net/lcs-2.2.15.o
/lib/modules/2.2.15/block/xpram.o
```

This generates `/lib/modules/2.2.15/modules.dep` for kernel 2.2.15, which is used to load the modules.

Note: If you receive messages like the following, it probably means that you have deposited other non-kernel things, such as LICENSE README or TROUBLESHOOTING) in the `/lib/modules/2.2.15 net` directory:

```
modprobe: not an ELF file
modprobe: not an ELF file
modprobe: not an ELF file
```

To see the file generated by the `depmod` command:

```
[root@linux6 2.2.15]# cat modules.dep
/lib/modules/2.2.15/net/lcs-2.2.15.o:
/lib/modules/2.2.15/block/xpram.o:
```

11.6.7 Activate the new kernel

Use one of the bootable devices to activate the kernel. We decided to use the `boot.alternate` directory for this. Change into the kernel directory:

```
cd /usr/src/linux-2.2.15/arch/s390/boot
```

Copy the files needed for silo to the boot directory. In this case, we were booting from an ECKD device:

```
cp -a ipleckd.boot /appl/boot.alternate
cp -a image /appl/boot.alternate/
```

Create a new boot record on the device that will reflect the new kernel:

```
silo -f image -d /dev/dasdc -p parm.line -b ipleckd.boot
```

Now activate kernel 2.2.15. Halt the Linux system and boot it from the device address reflecting `/dev/das<n>` node specified in the `-d` option of `silo`. This example uses `/dev/dasdc`.

The booted system will run under kernel 2.2.15:

```
[root@linux6 /root]# uname -a
Linux linux6 2.2.15 #4 SMP Tue May 17 19:45:19 EDT 2000 s390 unknown
```

11.6.8 Post-installation steps

xpram driver

Linux 2.2.15 supports the new `xpram` driver. You might need to create the device nodes for this new block device with `mknod`:

```
mknod /dev/xpram0 b 35 0
```

Create some `xpram` nodes just in case you need more than one. The following list shows the first 5 device nodes for `xpram`:

```
[root@linux6 /dev]# ls -l xpr*
brw-r--r-- 1 root root 35, 0 May 18 13:59 xpram0
brw-r--r-- 1 root root 35, 1 May 18 14:00 xpram1
brw-r--r-- 1 root root 35, 2 May 18 14:00 xpram2
brw-r--r-- 1 root root 35, 3 May 18 14:00 xpram3
brw-r--r-- 1 root root 35, 4 May 18 14:00 xpram4
```

update binary directories

Note that the `s390-ibm-linux` kernel provides two kernel-dependent utilities (binaries) `dasdfmt` and `silo`, which are created during the kernel compilation phase, and now need to be moved into your `/sbin` path.

Because these utilities are kernel-dependant, and because they might well change between kernel versions (sometimes referred to as *code drops*), we recommend copying them to /sbin with a name that reflects the kernel-tree which they came from (such as dasdfmt-2.2.15 or silo-2.2.15), and using a soft link to address them, as follows.

```
mv /sbin/silo /sbin/silo-2.2.14
mv /sbin/dasdfmt /sbin/dasdfmt-2.2.14
cp /usr/src/linux/arch/s390/tools/silo /sbin/silo-2.2.15
cp /usr/src/linux/arch/s390/tools/dasdfmt /sbin/dasdfmt-2.2.15
ln -s /sbin/silo-2.2.15 /sbin/silo
ln -s /sbin/dasdfmt-2.2.15 /sbin/dasdfmt
```

If you then chose to revert to an older kernel version, you would need to change the links back to the appropriate version of the utility.

Recommendation

Many new functions were integrated in the new Marist file system. If you did not have a lot of modifications done in the file system, we recommend you install the entire file system available at the Marist Internet page.

Chapter 12. Changing your root device

In this chapter we show you not only how to go from Marist-2.2.14 to Marist-2.2.15, but also what you might need to know when upgrading operating systems or going from one distribution level to another. Assuming that when you finish reading the main part of this redbook, you are comfortable with UNIX/Linux and equally fluent in the mainframe disciplines, you should be able to get through an upgrade without any major headaches.

12.1 Upgrading from Marist-2.2.xx to Marist-2.2.yy

Assume the following constellation:

- You have Marist-2.2.xx up and running.
- Now there's a Marist-2.2.yy.
- You don't want to lose your data, but you do want to be at the latest level.
- You decide to install the new system into its own (new) root partition, and later reuse the older disk for other things.

If you have VM, this task is considerably eased by the fact that you would probably have a Linux service machine defined where the new disks can be attached as needed and tailored to your specifications.

If you are running LPAR (or native), or if you are Linux root, but do not have VM->MAINT authority, then you can also change your root device by following these steps, adapting them to your needs.

12.2 Preparing a new volume

Assume the following machine:

```
G5->LPAR->VM/ESA->L390
the CMS "PROFILE EXEC" links the CTCAs and IPLs 0222 clear
Following DASD are defined.
DASD 0200 3390 LINUX2 R/W          200 CYL  DASD used as swap
DASD 0222 3390 LINUX4 R/W          2000 CYL  DASD current root /
DASD 0224 3390 LINUX2 R/W          1000 CYL  CMS
DASD 0235 3390 LINUX7 R/W          3000 CYL  DASD used as /data
The parameter line in the boot record on 0200 is as follows:
mdisk= dasd=200,222,235 root=/dev/dasdb1 ro noinitrd
```

Disk 224 was the CMS disk we used to hold the Marist data, kernels, initrd and so on. When the Marist-2.2.15 came out, we decided to use it as the new

root device. This meant that when we ran `dasdfmt` on it, to use it with the DASH driver, we would be changing the order of “visible” disks when Linux boots. It would become `/dev/dasdc` and move device 0235 down to `/dev/dasdd`. This was the theory; in practice it worked somewhat differently than expected.

If you want to think of this in PC terms, consider what happens to a PC with one SCSI controller and three disks at SCSI id(0,2,4). When you add a new disk at SCSI id(3), what used to be the third disk is now the fourth.

Linux does not yet have a method for adding new (disk) devices at run-time. This is being worked on in the 2.3.x series.

Edit the parameter line (in `/boot/parmfile`, on our system) to reflect the new device. Use `silo` to set up the boot record and reboot, expecting the init script `/etc/rc.sysinit` (on the Marist distribution) to fail when it hits the unformatted disk.

It does fail, as shown:

```
INIT: version 2.74 booting
Checking root filesystem
/dev/dasdb1: clean, 51634/180224 files, 232285/359997 blocks
[ OK ]
Remounting root filesystem in read-write mode [ OK ]
Finding module dependencies [ OK ]
Checking filesystems
(null):
The superblock could not be read or does not describe a correct ext2
filesystem. If the device is valid and it really contains an ext2
filesystem (and not swap or ufs or something else), then the superblock
is corrupt, and you might try running e2fsck with an alternate
superblock:
    e2fsck -b 8193 <device>

fsck.ext2: Bad magic number in super-block while trying to open
/dev/dasdc1
[FAILED]

*** An error occurred during the file system check.
*** Dropping you to a shell; the system will reboot
*** when you leave the shell.
Give root password for maintenance
```

We are now in “single user mode”, run level 1, with no network and only the root disk mounted, running through a teletype. This is exactly the environment

needed for changing things like `/etc/fstab` and mount points—if you can avoid it, do not have your system fully up, with all services and users.

```
[root@linux3 /root]# df
df
Filesystem          1k-blocks      Used Available Use% Mounted on
/dev/dasdb1         1417324        906476   438852   67% /
```

Verify that the new disk has been seen correctly, as follows:

```
[root@linux3 /root]# cat /proc/dasd/dev*
dev# MAJ minor node      Format
0200  94     0 /dev/dasda  4096
0222  94     4 /dev/dasdb  4096
0224  94     8 /dev/dasdc  4096
0235  94    12 /dev/dasdd  4096
```

The new disk at 224 used to be CMS, but we now want it to become DASD, so we format it with `dasdfmt -b 4096 -n 224` and `mke2fs -b 4096 /dev/dasdc1` to put a file system on it.

Now we fix `/etc/fstab` to move device 235, the old “third disk”, to “fourth disk”. We only have a teletype, so we’ll use `sed`, the stream editor.

Here’s the old `fstab` file:

```
cat fstab

/dev/dasda1 none swap sw 0 0
/dev/dasdb1 / ext2 defaults,errors=remount-ro 0 1
/dev/dasdc1 /data ext2 defaults 0 2
none /proc proc defaults 0 0
```

We substitute all `dasdc1` with `dasdd1`:

```
sed "s/dasdc1/dasdd1/g" fstab > fstab.new
```

Here is the result:

```
cat fstab.new
/dev/dasda1 none swap sw 0 0
/dev/dasdb1 / ext2 defaults,errors=remount-ro 0 1
/dev/dasdd1 /data ext2 defaults 0 2
none /proc proc defaults 0 0
```

It looks good, so we add a line for the new `dasdc1` to be mounted at `/newroot`. The sort is just a way of moving it from `/etc/fstab.new` to `/etc/fstab` and making it more readable:

```
echo '/dev/dasdc1 /newroot ext2 defaults 0 3' >>/etc/fstab.new
```

```
sort /etc/fstab.new -o /etc/fstab
```

After verifying that /etc/fstab looks OK, we add a new top-level directory mount point for the newroot file system when it comes back up.

```
mkdir /newroot
sync
shutdown -r now
```

After the system comes back, and we verify that the disks are mounted correctly, we need to untar the Marist big file system onto the new root disk. We mount the transfer directory from a laptop via NFS and untar the Marist initfs_big.tgz. (This route was chosen because the target disk was too small for both the tar file itself and the payload it carried.)

```
root@linux3:/ $ df
Filesystem            1k-blocks      Used Available Use% Mounted on
/dev/dasdb1            1417324      906544   438784   67% /
/dev/dasdc1            479468         20    454700    0% /newroot
/dev/dasdd1           2126008     1129624   888388   56% /data
glurp:/home/higson/xfer
                    3809728    2746232   869964   76% /glurp
$ (cd /newroot ; tar xvzBf /glurp/dist/MARIST-5-19-00/initfs_big.tgz)
```

Now we need to save some of the *personality* of the old root disk on the new one. You can type in a small loop to do the copying, as opposed to doing each copy on a separate line:

```
$ for i in hosts profile passwd passwd- resolv.conf syslog.conf
> do
> echo copying $i
> cp /etc/$i /newroot/etc
> done
copying hosts
copying profile
copying passwd
copying passwd-
copying resolv.conf
copying syslog.conf
root@linux3:/newroot $
```

You can pick up some other files, as well:

```
cp /usr/local/bin/* /newroot/usr/local/bin
cp /etc/sysconfig/network /newroot/etc/sysconfig
cp /etc/sysconfig/network-scripts/ifcfg-ctc0 \
/newroot/etc/sysconfig/network-scripts
```

The new root disk will also need the mount points for the other file systems:

```
mkdir /newroot/data /newroot/oldroot
```

Now we need to edit `/newroot/etc/fstab` (which will become `/etc/fstab` when we boot from there). Note that many distributions use an `fstab`, with very nice spacing and all the arguments aligned with tabs or blanks to make it look pretty. This doesn't get you much on the S/390, because you might need to edit it with `echo`, `sed`, and other "teletypisms".

```
/dev/dasda1 none swap sw 0 0
/dev/dasdc1 / ext2 defaults,errors=remount-ro 0 1
/dev/dasdb1 /oldroot ext2 defaults 0 2
/dev/dasdd1 /data ext2 defaults 0 3
none /proc proc defaults 0 0
```

After copying `/boot/*` to `/newroot/boot`, we ran `/newroot/sbin/silo` from `/newroot/boot`.

It looks OK, so we tried a shutdown:

```
shutdown -h now
```

After editing the `boot exec` a script to reflect the new boot device (224), we expected the system to come up smoothly, but instead it stopped with a CP message complaining about running S/370 code.

As a workaround, we reformatted the disk with `format 224 d` under CMS and rebooted from 222. We did not use the `dasdfmt` utility, but instead did a `mke2fs /dev/dasdc1`. We repopulated `/newroot` as previously described, pulled over the "personality files" (`/etc/passwd` et al), and did `silo` in `/newroot/boot`. We then rebooted to CMS, changed the `boot exec` over to 224 again and rebooted from 224. This time it worked.

12.3 Summation

So what can we conclude from this experience? Possibly the problem lies with the consolidated DASD driver and its view of disk and partition geometry. `silo` writes a boot block telling the `dasd-ipl` code where to find the kernel, parm card and possible `initrd`; these being given as absolute disk blocks relative to the start of the partition. The start of the partition will be an integer number of blocks from absolute zero (0).

Keep in mind that at the time of writing, Linux for S/390 had only been public for a few months—and it is surprisingly mature for a port of this age. However, there were major changes in the 2.2.14 and 2.2.15 DASD drivers, and the system used had also been used to develop 2.2.15 and 2.2.16_pre4 kernels,

so this was very definitely a “developers’ machine” and not set up for steady production.

Presumably, after Linux for S/390 distributions from SuSE, TurboLinux and Debian become available, the combination of kernel drivers, tool chains and libraries will have stabilized considerably.

Chapter 13. Hardware connectivity

This chapter discusses some of the hardware interfaces that can be used on S/390 in connecting a Linux for S/390 to the network.

13.1 OSA-2

OSA-2 provides SNA/APPN/HPR and TCP/IP applications with direct access to Ethernet, Token Ring, Fiber Distributed Data Interface (FDDI), and Asynchronous Transfer Mode (ATM) local area network clients; see Figure 81.

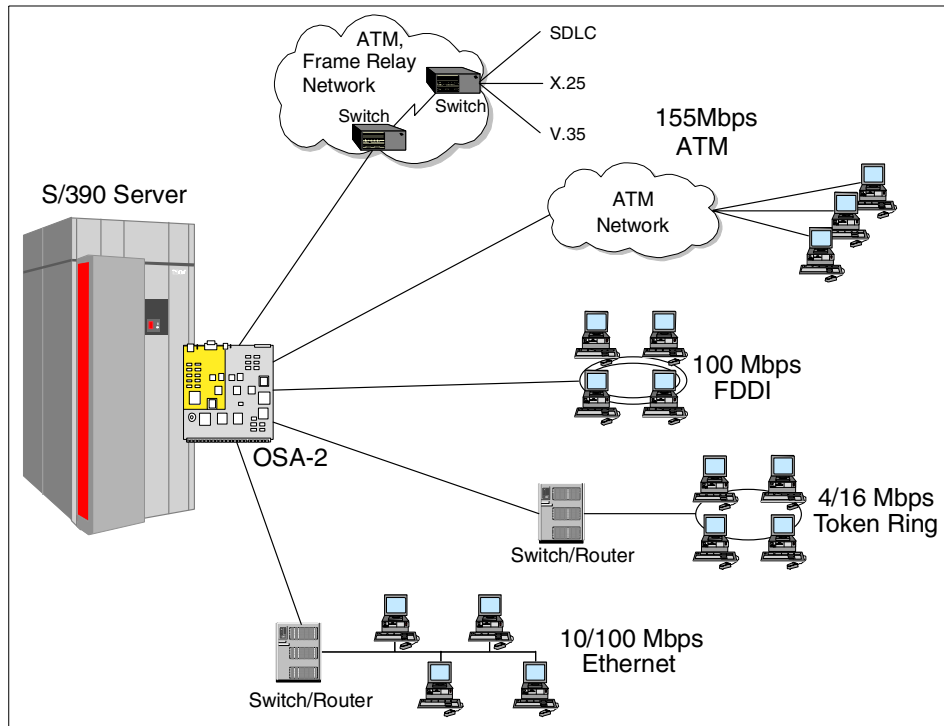


Figure 81. OSA-2 ENTR, FDDI, ATM, and FENET features

OSA-2 offers integrated, industry-standard LAN connectivity in a seamless manner, helping to reduce the total cost of computing and enhancing investments in local area networks (LANs). OSA-2 can use the current LAN infrastructure to provide connectivity to LAN backbones, other servers, high speed workstations, intelligent hubs, repeats, and routers in a heterogeneous, multivendor environment.

OSA-2 is an integrated hardware feature that has been implemented as a channel type on S/390 servers. It appears to application software as a channel-attached device. The small size of the hardware package allows the feature to be plugged into an I/O slot in a Central Processing Complex (CPC) or I/O expansion cage of the S/390 server.

OSA-2 is defined to the hardware as a new type of S/390 channel. As such, it is highly integrated into the hardware configuration to take advantage of the availability characteristics of S/390 servers.

The OSA-2 features are supported by the S/390 Open Systems Adapter Support Facility (OSA/SF) program product. OSA/SF allows you to customize the OSA-2 hardware to run in different modes. In some cases, the use of OSA/SF is optional, while in others it is required. It includes software that may be installed to run on a PC to communicate with OSA/SF on the host system and the OSA hardware feature. This product delivers a simple means to configure and manage OSA-2 and to download software updates (to the OSA-2 adapter) for supported applications.

When the S/390 server is running in LPAR (logically partitioned) mode, it is possible to configure OSA-2 so that any or all LPARs may share the same LAN connection. This is called *port sharing*. Port sharing support is provided by OSA/SF.

13.1.1 OSA-2 features

The OSA-2 Ethernet/Token Ring (ENTR) feature has two independent ports. These can be configured in either half duplex or full duplex as one of the following:

- Two 10Mbps Ethernets
- Two 4/16 Mbps Token Rings
- One 10 Mbps Ethernet and one 4/16 Mbps Token Ring

This provides maximum flexibility.

The OSA-2 FDDI feature has one 100 Mbps port that supports single ring or dual ring attachment, as well as attachment to an optical bypass switch.

The OSA-2 ATM feature has one 155 Mbps physical port and two logical ports that provide access to Ethernet and Token Ring LANs, and Wide Area Networks (WANs), attached to a high-speed ATM network. Each logical port is configured independently and supports TCP/IP, SNA/APPN, or both,

depending on the configured mode of operation. There is a single mode and a multimode fiber OSA-2 ATM feature.

The OSA-2 Fast Ethernet (FENET) feature has one port that can be attached to either a 100 Mbps or 10 Mbps Ethernet LAN, in either full duplex or half duplex mode. It uses auto negotiation to set the LAN speed and duplex mode of the port. The port LAN speed and duplex mode can also be set explicitly.

13.1.2 OSA-2 modes

With the introduction of OSA/SF for OS/390 V2R1, VM/ESA, and VSE/ESA, a graphical user interface (GUI) supporting Microsoft Windows 95 and Windows NT is now available, as well as the current OS/2 GUI, making it easier to find a nondedicated workstation to run the client application.

The OSA-2 can be configured, using OSA/SF, into several mode combinations depending on your system and network requirements:

- TCP/IP Passthru Mode (nonshared port)
In this mode, an OSA-2 port is capable of transferring TCP/IP LAN traffic to and from just one TCP/IP host or logical partition. This is the default mode of OSA-2 and does not require configuration using OSA/SF.
- TCP/IP Passthru Mode (shared port)
In this mode, an OSA-2 port is capable of transferring TCP/IP LAN traffic to and from more than one TCP/IP host within multiple logical partitions. The use of OSA/SF is required for this configuration.
- SNA Mode (nonshared port)
In this mode, an OSA-2 port is capable of transferring SNA LAN traffic to and from just one SNA host. The use of OSA/SF is required for this configuration.
- SNA Mode (shared port)
In this mode, an OSA-2 port is capable of transferring SNA LAN traffic to and from multiple SNA hosts in different logical partitions. The use of OSA/SF is required for this configuration.
- TCP/IP and SNA Mixed Mode (shared port)
In this mode, an OSA-2 port is capable of transferring TCP/IP and SNA LAN traffic to and from more than one TCP/IP and SNA logical partition. The use of OSA/SF is required for this configuration.
- TCP/IP and SNA Mixed Mode for OSA-2 ATM LAN Emulation (ATM LE)

In this mode, the one physical ATM port may be configured into two logical ports. Each logical port may then be configured to support TCP/IP, or SNA traffic, or both. Each logical port is capable of transferring TCP/IP and SNA LAN traffic to and from one or multiple TCP/IP and SNA logical partitions. The use of OSA/SF is required for this configuration.

- HPDT ATM Native Mode for APPN and TCP/IP (Classical IP RFC1577)

In this mode, the OSA-2 adapter supports Permanent Virtual Channels (PVC) and Switched Virtual Channels (SVC) for APPN and TCP/IP connections.

The ATM network media may be either multimode or single-mode fiber optic cables. An OSA-2 running in HPDT ATM Native mode cannot support any other mode at the same time. The use of OSA/SF is required for this configuration.

- ATM IP Forwarding (RFC 1483)

In this mode, the OSA-2 adapter supports direct connectivity to the Wide Area Network (WAN), allowing consolidation of WAN data traffic and carrying it on a single LAN backbone.

TCP/IP traffic is carried over ATM between the S/390 OSA-2 ATM card and the Ascend B-STDX 8000 and 9000 Multiservice WAN platforms. The Ascend switch then translates the protocol from ATM into a WAN protocol - Frame Relay or Switched Multimegabit Data service (SMDS). If the incoming or outgoing traffic is not Frame Relay, the switch encapsulates the data (for example: X.25/HDLC, SNA/SDLC). The switch can also convert protocols such as Point-to-Point Protocol (PPP) (for example: IP, Netware SPX/IPX) into Frame Relay. The use of OSA/SF is required for this configuration.

For further information on OSA-2, see *S/390 I/O Connectivity Handbook*, SG24-5444.

13.2 The 2216 hardware interface

The 2216 is another hardware interface you can use on S/390 to connect your Linux for S/390 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, one being an ESCON channel adapter and the other 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.

13.2.1 2216 ESCON channel adapter features

The ESCON 2216 channel adapter provides access to SNA and TCP/IP applications from LANs, WANs, and ATM over a duplex-to-duplex multimode fiber cable. The adapter features some of the following:

- Up to four ESCON adapters are supported in the 2216.
- Support of 64 subchannel addresses with software program MAS V3R2 PTF01 providing the 2216 with access to 16 hosts using LCS (LAN Channel Station) protocol.
- Connectivity provided for Ethernet, Token-Ring, FDDI, and ATM.

13.2.2 2216 ESCON channel protocols

The IBM 2216 ESCON channel adapter supports three types of channel protocols: LAN Channel Station (LCS), Link Services Architecture (LSA), and Multi-Path Channel (MPC). Each channel protocol supports several network protocols with LSA supporting SNA only and MPC+ also supporting both SNA and TCP/IP.

LCS is a channel protocol supported by TCP/IP host applications on the host. Each application defines a consecutive pair of subchannels with one for TCP/IP to read and the other for TCP/IP to write.

LSA is an interface to support SNA traffic over the channel. Each LSA path is a single bidirectional subchannel between the host application and the 2216 ESCON channel adapter. The host SNA application (VTAM) issues a READ command immediately following each WRITE command to receive data from the channel.

MPC+ is a Data Link Control interface on the channel. Each MPC+ path consists of one or more read subchannels and one or more write subchannels. These subchannels are bound together to form a transmission group. VTAM and the 2216 ESCON adapter exchange XIDs to identify the number and direction of subchannels at initialization, and then each frame has a header to indicate the sending/receiving applications.

In the 2216, all subchannels are connected to the base net handler, which in turn is connected to one or more virtual handlers. This differs from the 3172 ICP (Interconnect Controller Program), in which each subchannel is connected to one or more real LAN adapters. Each virtual net handler

supports one of the three channel protocols (LCS, LSA, MPC) and sends/receives the data to another net handler representing a network connection. There may or may not be any real LAN adapters connected to the 2216. As a gateway between the host and users, the 2216 creates an appearance of a LAN adapter so that the host application believes it's communicating with a real LAN.

For further information on 2216, see *IBM 2216 and Network Utility Host Channel Connections*, SG24-5303.

13.3 CTC

The channel-to-channel function simulates an S/390 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.

13.3.1 CTC support

The CTC is selected and responds in the same manner as any I/O device. It differs from other I/O devices in that it uses commands to open a path between the two channels it connects and then synchronizes the operations performed between the two channels.

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

In the Parallel channel environment, CTC connection is available via the 3088 MCCU device.

The Parallel channel can operate in either basic mode or extended mode. In basic mode, the channel provides basic communication functions. In extended mode, the channel provides communication functions in addition to those of basic mode and provides sense information to clarify errors and other unusual conditions.

The ESCON CTC is an IOCP configuration option of an ESCON-capable processor. The CTC option is specified in the IOCP configuration, which results in the CTC microcode being loaded into the ESCON channel hardware at power-on-reset (POR). ESCON channels that operate in CTC mode (also called Serial CTC or SCTC) support both extended mode and basic mode operations.

FICON channels allow faster and more efficient data transfer, while at the same time allowing customers to use their currently installed single-mode and multimode fiber optic cable. FICON provides all the strengths of ESCON while increasing the link rate from 20 MB/sec up to 100 MB/sec. The FICON implementation enables full duplex data transfer, so data travels in both directions simultaneously, rather than as with ESCON half-duplex data transfer. The FICON channel on a S/390 Generation 5 and 6 connects to an ESCON Director 9032 Model 5 containing a FICON bridge card. This card enables existing ESCON control units, without changes, to exploit the new FICON channel.

For further information on CTC, see *S/390 I/O Connectivity Handbook*, SG24-5444.

Chapter 14. Linux TCP/IP connectivity

This chapter covers some of the aspects of Linux connectivity using TCP/IP. We begin with a short explanation of the most common TCP/IP protocols used in Linux and continue with a discussion of IP addressing. We move on with short descriptions of the diverse configuration files, scripts, and daemons. These are followed by a description of a few of the troubleshooting tools that are available to you on your Linux machine. Finally, we close with how you can connect to your data and applications.

14.1 Assumptions

We assume that you are running the Marist Linux for S/390 big file system. Further, you should have the following files on your Linux for S/390 system:

- /etc/hosts
- /etc/services
- /etc/protocols
- /etc/HOSTNAME
- /etc/inetd.conf
- /etc/rc.d/init.d/network

We also assume that your network is up and running with such programs as `ping`, `netstat`, `ifconfig`, `route`, `telnet`, and `ftp` available for your use.

We will be giving you a guided tour through some of the more important corners of your Linux networking environment. A detailed discussion of the TCP/IP protocol itself is beyond the scope of this chapter. For coverage of this subject we refer you to G.3, "Other resources" on page 506.

14.1.1 Skills

You will need to be able to edit a text file and download from the Internet, and have a sense of adventure. We also assume that you know how to navigate through the Linux for S/390 file system.

14.2 TCP/IP protocols

We give you a list and brief description of the standard protocols used in IP networking. These protocols are from the *Transport Layer*, which rests on top of the *Network Layer* in which IP operates. IP is a *connectionless* and

unreliable protocol that just sends the datagrams it is given over the net. The Transport Layer protocol adds reliability to the Network Layer (IP). To give you a reference point as to where we are in the OSI Networking Model, we show the lower four layers in Table 26. Note that we give just a few examples of what belongs in which layer.

Table 26. OSI model

OSI Model
4. Transport Layer (TCP, UDP, ICMP)
3. Network Layer (IP)
2. Datalink Layer (Token Ring, Ethernet)
1. Physical Layer (Twisted Pair, Coax)

14.2.1 Transmission Control Protocol (TCP)

Transmission Control Protocol was developed on top of IP to provide the reliability mechanism that is absent from IP. It manages the disassembly of a mass of data into a stream of *datagrams* and passes these to the Network Layer, which uses IP to send them to the receiving machine. On the receiving machine TCP receives the datagrams from the Network Layer and reassembles them into a stream of data. This is a *connection-oriented* protocol that uses acknowledgments to ensure that each of the sent datagrams has arrived. If a datagram is missing from the delivered stream, TCP ensures that it is resent.

14.2.2 User Datagram Protocol (UDP)

User Datagram Protocol uses a single datagram to send a message from one machine to another. This protocol is usually used for networking housekeeping tasks. It is a *connectionless* protocol that sends its datagrams out onto the net and does not check whether they are delivered to the given address.

14.2.3 Internet Control Message Protocol (ICMP)

The *Internet Control Message Protocol* is used to carry control messages across the Internet. Internet hosts communicate with each via these messages. In most cases, applications do not use this protocol. Examples of applications that use ICMP are `ping` and `traceroute`, which are discussed later in this chapter.

14.3 IP address types

There are two ways of getting a machine connected to the network using IP addresses: with a static address or with a dynamic one.

14.3.1 Static IP addresses

Static IP addressing is done by your network administrator assigning TCP/IP addresses to identify the clients in your network. These static TCP/IP addresses were defined as part of the installation of your Linux system. See Chapter 5, “Native S/390 installation and operation of Linux” on page 45 and Chapter 6, “VM installation and operation of Linux for S/390” on page 85. You have a permanent address that will only change if your network administrator assigns your Linux for S/390 machine a new one. This should not happen very often.

14.3.2 Dynamic IP addresses

Dynamic IP addressing differs from static IP addressing in that your addresses are obtained from a Dynamic Host Control Protocol (DHCP) server, which is responsible for answering any DHCP *requests* from clients who want an address on the network.

A DHCP client would request an IP address (at boot time, for example) from a DHCP server, which returns an IP address. If the server name is not known beforehand, then the client *broadcasts* its request out onto the net. The address comes with a *lease*, which basically gives the client a limited period during which the address is valid. When a client using DHCP logs on again, it first checks its DHCP lease database. If the client has a valid lease, the client tries to access the network with the IP address that is associated with that lease. If it fails, then the DHCP server is asked once again to provide a new IP address, which in turn will have a new lease. If the lease database no longer contains a lease that is valid, then the client just asks the DHCP server for another IP address and the process starts at the beginning.

14.4 Configuration files

The files we describe here hold information that networking programs use as they run. They can all be found in the `/etc/` directory.

14.4.1 The hosts file

The hosts file holds a text database consisting of the host names and IP addresses of the hosts your Linux for S/390 system will need before being

able to contact its Domain Name Server (DNS). For more on DNS see Chapter 18, “Domain Name Service (DNS)” on page 349. The database is read to resolve these host names into their IP addresses. The lines have the following syntax, with one line for each host entry:

```
numerical.IP.address internet.hostname [nickname]
```

For example, the entry in hosts for the mailhost machine on the example network could be:

```
9.9.9.9 mailhost.example.com mailhost
```

Note that it is possible to give a host more than one nickname. This form of host resolving is not very scalable and will only remain useful for small intranets, because with more than four or five hosts in a network the administration overhead gets out of hand. The solution to this is to move to DNS.

14.4.2 The services file

The services file names the networking services that your Linux for S/390 machine provides to other hosts on the network, identifying the port and host-to-host protocol that the service uses. The entries in this file have the following syntax, with one entry for each service offered:

```
service_name IP_port / udp | tcp
```

For example, the entry in services for the `telnet` service could be:

```
telnet 23/tcp
telnet 23/udp
```

This shows that `telnet` uses port 23 and can use either TCP or UDP as the transport protocol.

14.4.3 The protocols file

The protocols file names the TCP/IP protocols that your Linux for S/390 system recognizes and assigns a number to each. You will most likely never need to edit this file. An example of what we found on our version of the Marist Linux for S/390 big file system follows:

```
# /etc/protocols:
# $Id: protocols,v 1.1 1995/02/24 01:09:41 imurdock Exp $
#
# Internet (IP) protocols
#
#   from: @(#)protocols5.1 (Berkeley) 4/17/89
#
```

```
# Updated for NetBSD based on RFC 1340, Assigned Numbers (July 1992).
```

```
ip      0      IP      # internet protocol, pseudo protocol number
icmp    1      ICMP    # internet control message protocol
igmp    2      IGMP    # Internet Group Management
ggp     3      GGP     # gateway-gateway protocol
ipencap 4      IP-ENCAP # IP encapsulated in IP (officially ``IP'')
st      5      ST      # ST datagram mode
tcp     6      TCP     # transmission control protocol
egp     8      EGP     # exterior gateway protocol
pup     12     PUP     # PARC universal packet protocol
udp     17     UDP     # user datagram protocol
hmp     20     HMP     # host monitoring protocol
xns-idp 22     XNS-IDP # Xerox NS IDP
rdp     27     RDP     # "reliable datagram" protocol
iso-tp4 29     ISO-TP4 # ISO Transport Protocol class 4
xtp     36     XTP     # Xpress Transfer Protocol
ddp     37     DDP     # Datagram Delivery Protocol
idpr-cmt 39    IDPR-CMTP # IDPR Control Message Transport
rspf    73     RSPF    #Radio Shortest Path First.
vmtp    81     VMTP    # Versatile Message Transport
ospf    89     OSPFIGP # Open Shortest Path First IGP
ipip    94     IPIP    # Yet Another IP encapsulation
encap   98     ENCAP   # Yet Another IP encapsulation
```

14.4.4 The HOSTNAME file

This file holds your Linux for S/390 machine's name. For example, if you named your machine `heaven.linux.com`, then the contents of your `HOSTNAME` will read:

```
heaven
```

You will have done this during the installation of your Linux for S/390 machine, and if you ever want to rename your Linux for S/390 machine, then you would have to, among other things, edit this file. This can be done by hand or with the `hostname` command. Refer to its man page for further details.

14.4.5 The `inetd.conf` file

This configuration file is read by `inetd`, the TCP/IP super server, which is covered in 14.6.1, "Overview of `inetd`" on page 272.

Each line in this file describes how `inetd` handles one service. The syntax is as follows, with the entry fields separated by a tab or space:

```
service name
```

```
socket type
protocol
wait/nowait[.max]
user[.group]
server program
server program arguments
```

Simply, each line gives `inetd` the ports to which it should listen and the server that it should activate to handle traffic on a given port.

For example, to specify the standard `telnet` service, the entry could look like the following line taken from our Linux for S/390 machine:

```
telnet stream tcp nowait root /usr/bin/tcpd in.telnetd
```

This tells `inetd` that when a connection request occurs on the `telnet` port, it should activate the `telnetd` daemon by running `/usr/bin/tcpd` with the argument `in.telnetd`. For more information, see the `inetd.conf` and `tcpd` man pages.

14.5 The network script

This script is found in the `/etc/rc.d/init.d/` directory and is called from the main system boot script, `rc.sysinit`. The `network` script ensures that commands are executed to activate your TCP/IP networking resources, which include connections, interfaces, and daemons.

The location of this script will vary from distribution to distribution. We describe only the Marist Linux big file system networking script here. For more detailed information, see the documentation that should come with your distribution.

14.6 Network daemons

This section discusses several daemons that are related to networking.

14.6.1 Overview of `inetd`

Often the design of servers in a client/server environment is the same: a process daemon listens on a well-known TCP/IP or UDP/IP port and when a request for a service comes in, another process daemon is started, or forked, to communicate with the client. Rather than having the same code duplicated many times in many servers, a *generic listener* or *super-server* named InterNET services daemon (`inetd`) is used. It is a server designed to listen on many well-known ports for incoming connection requests. When a

connection request is received, the socket is accepted and then `inetd` forks and executes the appropriate server. Using a generic listener reduces system load by only running applications when they are needed.

Linux for S/390 uses `inetd` to start the following services and invoke the corresponding servers:

Service	Server command invoked
ftp	<code>/usr/sbin/tcpd in.ftpd -l -a</code>
telnet	<code>/usr/sbin/tcpd in.telnetd</code>
shell	<code>/usr/sbin/tcpd in.rshd</code>
login	<code>/usr/sbin/tcpd in.rlogind</code>
talk	<code>/usr/sbin/tcpd in.talkd</code>
ntalk	<code>/usr/sbin/tcpd in.ntalkd</code>
finger	<code>/usr/sbin/tcpd in.fingerd</code>
auth	<code>/usr/sbin/in.identd in.identd -l -e -o</code>

Refer to 24.7, “Use a tcp wrapper (`tcpd`)” on page 438 for more information about the `tcpd` program. These servers are specified in the configuration file `/etc/inetd.conf`, which we discussed in 14.4.5, “The `inetd.conf` file” on page 271. The ports that `inetd` knows are specified in the file `/etc/services` and the ports listened to are specified in `inetd.conf`.

Because the service name is the link between the two files, the `grep` command can be used to determine the well-known port numbers and which server is started. For example, the following `grep` command shows that the `ftp` well-known ports are 20 and 21, and that `in.ftpd` is started by `inetd`:

```
[mikem@itsolinux1 mikem]$ grep ^ftp /etc/inetd.conf /etc/services
/etc/inetd.conf:ftp stream tcp nowait root /usr/sbin/tcpd in.ftpd -l -a
/etc/services:ftp-data 20/tcp
/etc/services:ftp      21/tcp
```

14.6.2 Telnetd

Telnetd is the server daemon that handles `telnet` connections. It is activated by `inetd` whenever a datagram arrives at the telnet port 23 and if the service is activated in `inetd.conf`. Telnetd then negotiates the parameters of the connection between the sender to the port and the receiver (itself). When it finishes with the connection, it returns control of the port to `inetd`.

Telnetd operates by allocating a *pseudo-terminal* for a client, then creating a login process that runs the slave side of the pseudo-terminal. Telnetd manipulates the master side of the pseudo-terminal and when it starts up, it sends options to the client for setup. Refer to the `telnetd` man page for a full listing of the options.

14.6.3 Ftpd

The File Transfer Protocol daemon, `ftpd`, is covered in 17.6.2, “The FTP daemon” on page 338 and mentioned in 24.6, “Use `scp` instead of FTP” on page 437.

14.6.4 Syslogd

It is worth mentioning here that you can have the logging done by `syslogd` to trace and track any problems you might be having in your Linux network. The messages that are generated and logged by this daemon are described in 9.9, “System logs” on page 203. The configuration file `syslog.conf` is also covered in that section.

14.7 Troubleshooting

In this section we discuss some helpful tools to track down problems you may have with your network connections. Some are reporting tools to track the status of your network. Others can be used to manipulate networking definitions that are related to your networking environment.

14.7.1 The ping command

The `ping` command is used to send out an echo request to determine if a host is accessible. This command is useful for troubleshooting problems in your network and for determining which resources are available. The command syntax is as follows:

```
ping [options] host
```

For a complete listing and description of the options, see the `ping` man page. The `host` option may be either an IP address or the Internet domain name. The following is an example from a `ping` command using, first, `www.ibm.com`:

```
$ PING www.ibm.com (198.133.16.99): 56 data bytes
64 bytes from 198.133.16.99: icmp_seq=0 ttl=255 time=0.0 ms
64 bytes from 198.133.16.99: icmp_seq=1 ttl=255 time=0.0 ms
64 bytes from 198.133.16.99: icmp_seq=2 ttl=255 time=0.0 ms
```

and then using an IP address:

```
$ PING 9.12.9.180 (9.12.9.180): 56 data bytes
64 bytes from 9.12.9.180: icmp_seq=0 ttl=255 time=0.0 ms
64 bytes from 9.12.9.180: icmp_seq=1 ttl=255 time=0.0 ms
64 bytes from 9.12.9.180: icmp_seq=2 ttl=255 time=0.0 ms
```

This tool is also very handy for determining the quality and speed of your network connections as you can see by the output.

14.7.2 The netstat command

The `netstat` command displays the status of the network. Information shown consists of TCP/IP connections, gateways, network clients, and routing information. The command syntax is as follows:

```
netstat [options]
```

One of the most common options is `-r`, which shows your kernel routing table. The following is a sample output (with addresses changed to protect the innocent):

```
$ netstat -r
Kernel IP routing table
Destination Gateway Genmask          Flags  MSS Window  irtt  Iface
9.9.9.13    *        255.255.255.255  UH     0  0      0     ctc0
9.9.9.14    *        255.255.255.0   U      0  0      0     ctc0
127.0.0.0   *        255.0.0.0       U      0  0      0     lo
default    9.9.9.13  0.0.0.0          UG     0  0      0     ctc0
```

For a complete listing and description of the options, see the `netstat` man page.

14.7.3 The ifconfig command

This command is used to *activate* or *shut down* an interface. You can use it for such interfaces as channel-to-channel, Inter User Communication Vehicle, Token Ring, Ethernet, PPP and loopback devices.

The command syntax has two options:

- `ifconfig [interface]`
- `ifconfig interface [atype] options | address ...`

The first line is a syntax example for using `ifconfig` as a reporting tool. If you leave the interface option blank, you get a complete listing of the active interfaces on your Linux for S/390 machine. For example, when `ifconfig` is executed on our Linux for S/390 under VM/ESA machine, the following output is generated:

```
$ /sbin/ifconfig
ctc0      Link encap:Serial Line IP
          inet addr:9.12.9.180 P-t-P:9.12.9.174 Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MTU:1500 Metric:1
          RX packets:3092 errors:0 dropped:0 overruns:0 frame:0
```

```
TX packets:2120 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:168 errors:0 dropped:0 overruns:0 frame:0
        TX packets:168 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
```

Note that the command is executed from the `/sbin` directory. This is only to show how a non-root user on your system can also execute this command.

In the output you see that there are two interfaces defined, a *channel-to-channel* device and a *loopback* device. For more information, see the `ifconfig` man page.

The second line of the syntax shows you the options for actually setting up an interface. The elements of the syntax are as follows:

```
interface - this names the actual interface (tr0, eth0, ctc0...).
aftype (address families) - the name of a supported address family.
options - see man page for more details.
address - the IP address to be assigned to this interface.
```

We would like to caution you about attempting to set up an interface before you are comfortable with this command. For more information, we refer you to the `ifconfig` man page.

14.7.4 The `route` command

The `route` command—or, to be more precise, the *root* user using the `route` command—is used to manipulate the Linux kernel's routing table. The kernel uses its routing table to determine which interface has access to which hosts. This process is managed by the router daemon; we refer you to the man page for more information.

Because this command manipulates the network routing table of the Linux kernel, it should only be used when you know what you are doing. We suggest that you use the `netstat -rn` command instead for reporting. For more, consult your network administrator, and if that is you, we suggest some serious research into this subject before attempting to manipulate the kernel's routing tables with `route`. You have been warned!

14.8 Access to data and applications

In this section we describe a few applications that give you various ways to connect to your Linux for S/390 data and applications. We describe both secure and not-as-secure connection methods.

14.8.1 The telnet command

The `telnet` program allows you to connect to a remote machine and log on just as if it were your local machine.

`telnet` can also be used to connect to remote machines and use programs like `ftp`. For example, from your own Linux3 machine you access a remote machine called Linux1 with `telnet` and then connect to another machine, Linux2, with the `ftp` program. You are now connected from your own Linux 3 machine to the Linux1 machine via `telnet`, which is then connected to Linux2 via the `ftp` program, and you are remotely copying data from Linux2 to Linux1. The possibilities are endless!

Telnet supports both full-screen and line-mode emulation using the ASCII character set. The `telnet` program can be invoked with either an IP address or the name of the host you are trying to connect to. Following is an example of the `telnet` command accessing a host by IP address:

```
$ telnet 192.145.122.11
```

After entering this command, you will then be asked to log onto the machine as if you were sitting in front of it.

14.8.1.1 The X-Windows system

We offer a short overview of the *X-Windows* system and show you a nice way to get around the fact that at this point in time, the X-Windows server is not available on your Linux for S/390 machine.

X-Windows uses the concept of a *client* and a *server*, but not as you might think. The *server* in this model drives the hardware to actually draw the graphical image (i.e., window) on the screen. The *client* is a program that requests a service from the X-Window server, namely that the server provide the actual drawing on the display to which the X-Window server is attached. When using this system over a network, the X-Window server resides on the machine where the graphics card and monitor are attached. The client is the program that can be running anywhere and only makes use of the X-Windows server to display its graphical interface.

We now provide an example of these techniques by running a program on your Linux for S/390 machine and using the X-Windows server on your local Linux machine (PC). This is an often-used trick to get around the fact that `telnet` has no X-Windows capabilities. We will be accessing our remote Linux1, which is a Linux for S/390 under VM/ESA machine, using `telnet` from our Linux desktop PC and exporting the display to our local X-Windows server on our Linux desktop PC. The steps are as follows (in our example the Linux desktop PC is called Home and the remote Linux for S/390 is called Linux1):

1. On Home you allow remote machines access to your X-server by issuing the following command (you can also name the remote host by using its IP address). This is nothing more than adding a host to the X-servers access control list.

```
[root@Home]$ xhost +Linux1
```

2. Next you log into the remote machine:

```
[PCuser@Home]$ telnet Linux1
Trying 9.9.9.9...
Connected to Linux1.
Escape character is '^]'
```

```
Linux 2.2.14 (linux1) (tty0)
linux1 login:
```

3. You then log on as the user you want to be, using the user's password. Next you would need to export Linux1's display to the Home machine:

```
[user@linux1 linux]$ export DISPLAY=HOME:0.0
```

Note that you must enter the IP address of your own machine instead of HOME.

4. Next you may try out the command `xclock`:

```
[user@linux1 linux]$ xclock &
```

This should give you a nice clock on your local machine's display, but it is running on the Linux1 machine! (The "&" character allows you to get the command line back, if you do not do this then as long as `xclock` is running you will not have access to the command line of Linux1.) Check out what time it is on the remote host!

There are other nice X-programs, such as `xload`, `xbill`, `xman` and many more!

14.8.2 ftp

The File Transport Protocol is covered in Chapter 17, “File Transfer Protocol (FTP)” on page 335.

14.8.3 rlogin, rsh and rcp

These commands are mentioned here to let you know that they exist and to warn you about using them. It is generally accepted that they are too much of a security risk to be used on modern day systems. They are:

```
rlogin - log in to a remote host
rsh - execute a shell command on a remote host
rcp - copy a file to or from a remote host
```

The problem with these commands is that a host can declare that another host is equivalent to itself, thus bypassing the password security. Another problem is that the data being sent back and forth, including any user identification and passwords, is unencrypted. This is the main reason why you are advised to use either *telnet* (ensuring that you know who is logging on) or *openssh*, which is described in 14.8.4, “ssh” on page 279.

If you still want to configure these commands we refer you to the man pages.

14.8.4 ssh

To prevent any sort of security leaks, such as the sending of clear text passwords that you get in the *r** commands described above, we recommend that you set up an *ssh* (Secure SHell) environment.

Today this is called *openssh* due to the fact that the original *ssh* program was being released with increasingly restrictive licensing. Some very nice people then took an older (less restricted) version of *ssh* and developed an Open Source version called *openssh*. For more information, see <http://www.openssh.com/history.html>.

The packages you will need can be downloaded from the S/390 rpm database (see Appendix G.4, “Referenced Web sites” on page 507):

```
openssh-1.2.2-2.s390.rpm
openssh-clients-1.2.2-2.s390.rpm
openssh-server-1.2.2-2.s390.rpm
```

Install with the following command (enter the correct *ssh* filename in place of [filename.s390.rpm]):

```
rpm -ivh [filename.s390.rpm]
```

For the setup and configuration of your `ssh`, refer to the following books:

- *Computer Networks*
- *The Linux Network*

The command syntax is:

```
ssh [options] host
```

The most used option would be `-l user`, which means log in as this user name. The `host` should be replaced with the name or IP address of the remote machine you are connecting to. Note that this form of syntax has been reserved for remote shell compatibility.

A second, more user-friendly, command syntax is:

```
ssh USER@HOST
```

Enter your user name in place of `USER` and the machine name or IP address in place of `HOST`, to start the connection process.

What basically happens when you set up a connection using `ssh` to a remote machine is as follows (assuming you have installed the necessary `ssh` software packages):

You log into the remote host with the following command (insert a valid user name for the remote machine, which we refer to here as `Linux1`):

```
$ ssh -l user Linux1
```

You will be prompted to enter the given user's password to complete the login:

```
user@Linux1's password:
```

The session will be authenticated between your machine and the remote host using encryption. Once this is done you will have your own encrypted transport between you and the remote host, meaning that everything you send over that connection will be encrypted from end to end.

Also worth mentioning is that the `ssh` provides compression of all data that is encrypted. This means that you're not only getting confidentiality on your data transfers, but much faster transportation of your data due to the compression.

By using `ssh` to connect to a remote machine and running programs that need X-Windows (see 14.8.1.1, "The X-Windows system" on page 277), you also have another benefit: You not only get the above mentioned encryption and compression, you can open the programs locally on your local display. We saw earlier that you needed to export your display (`export`

`DISPLAY=IP_ADDRESS:0.0`), but with an `ssh` connection this is often done automatically. You just start the remote program and it is displayed on your local machine!

Chapter 15. Linux for S/390 connectivity to VM, OS/390, VSE

This chapter describes the possible options and configurations for connecting a Linux for S/390 system to existing VM/ESA, OS/390, and VSE/ESA.

We cover TCP/IP connectivity and access to data from a Linux for S/390 system to VM/ESA, OS/390 and VSE/ESA, and vice versa.

15.1 Configuring the network

There are several ways to connect a Linux for S/390 virtual machine to a physical network or to other Linux for S/390 virtual machines.

For a description of the communications hardware that you can use with Linux for S/390, refer to Chapter 13, "Hardware connectivity" on page 259.

Three network drivers are provided with Linux for S/390:

- LAN Channel Station (LCS)
- Channel-to-channel (CTC)
- Inter-User Communications Vehicle (IUCV)

The first two drivers can be used to drive a physical network connection. The CTC driver can also be used to communicate through a gateway IP host such as a VM TCP/IP service machine. The IUCV driver can be used in a similar fashion. It makes connections between memory buffers in the VM/ESA Control Program to act as a high-speed communications pipe between virtual machines. Table 27 summarizes the options.

Table 27. Linux for S/390 network driver choices

Linux network driver	Physical network interface	Virtual network interface	Network routing via VM TCP/IP stack ^a	Connection to another Linux for S/390 LPAR or virtual machine
LCS	✓			LPAR
CTC	✓	✓	✓	✓
IUCV		✓	✓	Virtual machine

a. Or another guest operating system TCP/IP stack

A fundamental decision must be made about *ownership* of physical network interfaces. The following can be the owners:

- Linux for S/390 running in either an LPAR or a virtual machine
- The VM TCP/IP stack
- Another operating system on the same S/390 processor

If Linux for S/390 is running in a virtual machine and the VM TCP/IP stack owns the network interface, then point-to-point links must be established from each Linux for S/390 virtual machine to the VM TCP/IP stack in order to access the physical network.

If Linux for S/390 owns the network interface, a point-to-point link must be established from the TCP/IP stack in OS/390, VM/ESA or VSE/ESA so that users of those operating systems can access the physical network.

If the TCP/IP stack of a VSE/ESA or OS/390 guest of VM owns the physical network interface, Linux for S/390 virtual machines can access the network through point-to-point links to the VSE/ESA or OS/390 guest's TCP/IP stack.

Figure 82 on page 285 shows the IP addressing and network environment we used for this project.

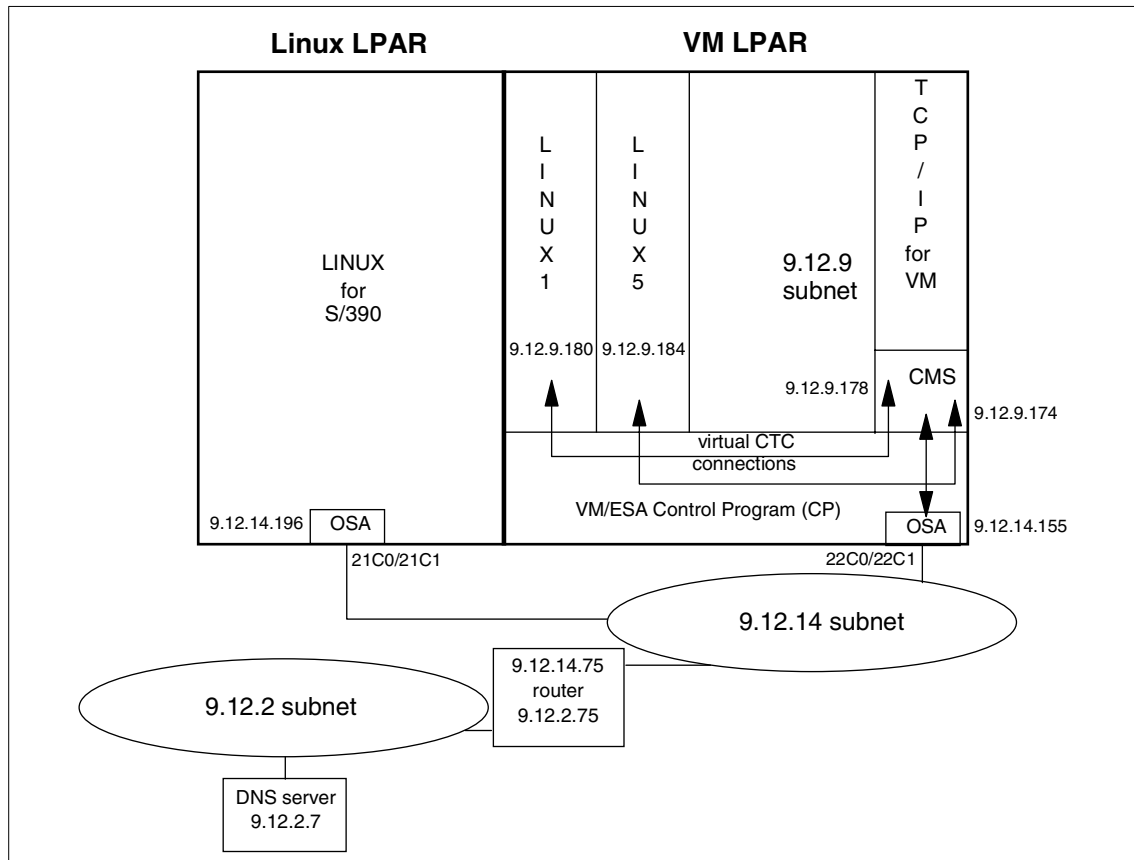


Figure 82. IP network topology

15.2 Logical partition

For a Linux for S/390 system running on an LPAR, the following options to connect to a TCP/IP-based network are available:

- Via a CTC connection to another TCP/IP running on an LPAR or under VM/ESA
- Via an Open Systems Adapter (OSA) directly connected to the network
- Via an (old) 3172 network adapter (which behaves like an OSA adapter)

We only describe the use of an OSA adapter as we did not use the other options.

15.2.1 OSA-2 in LPAR

Our LPAR running Linux for S/390 was connected to the network using an OSA-2 adapter as shown in Figure 82 on page 285.

The address of the OSA adapter was entered during the very first `netsetup` script. In our case this was 21C0. The setup and the following actions, mainly copying the definitions and removing symbolic links, are described in 5.7.5, “Customizing Linux for S/390 configuration files” on page 75. To get the OSA adapter activated from Linux for S/390, you have to enter the OSA device address. The settings are in `/etc/conf.modules`:

```
[root@linuxx /etc]# cat conf.modules
alias block-major-35 xpram
alias tr0 lcs
options lcs devno_portno_pairs=0x21C0,0
[root@linuxx /etc]#
```

This address should be reported by Linux for S/390 during boot. You can find the address information in `/var/log/dmesg`:

```
[root@linuxx log]# grep -i 21c0 dmesg
SenseID : device 21C0 reports: Dev Type/Mod = 3088/60
[root@linuxx log]#
```

The settings that reflect your hardware can be checked using the `ifconfig` command. As we are interested in the settings for the Token Ring interface only, we restrict the listing to the `tr0` settings.

```
[root@linuxx /root]# ifconfig tr0
tr0      Link encap:16/4 Mbps TR  HWaddr 40:00:09:FF:71:C0 ①
         inet addr:9.12.14.196 Bcast:255.255.255.0 Mask:255.255.255.0
         UP BROADCAST RUNNING MULTICAST MTU:2000 Metric:1
         RX packets:429653 errors:0 dropped:0 overruns:0 frame:0
         TX packets:513 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:100
```

`HWaddr ①` gives the universally administered address of the OSA card. If you see this address, it means the OSA card was successfully initialized by Linux for S/390.

15.3 Linux for S/390 running in a virtual machine

The following sections describe the networking options and definitions when running Linux for S/390 in a virtual machine.

For detailed information on network definitions for the VM TCP/IP stack, refer to *VM/ESA V2R4.0: TCP/IP Function Level 320 Planning and Customization*, SC24-5847. Another good reference is *TCP/IP Solutions for VM/ESA*, SG24-5459.

15.3.1 Networking definitions

If you are using VM TCP/IP as a network gateway, Figure 83 shows where to match up the Linux for S/390 network prompts with the network definitions for the VM TCP/IP stack. Besides the prompts generated by the `net.setup` script, Linux for S/390 network definitions can also be entered using Linux for S/390 commands.

When using the VM TCP/IP stack to connect to the physical network, you need to match the responses to the initial Linux for S/390 network prompts with the relevant entries in the TCP/IP configuration file.

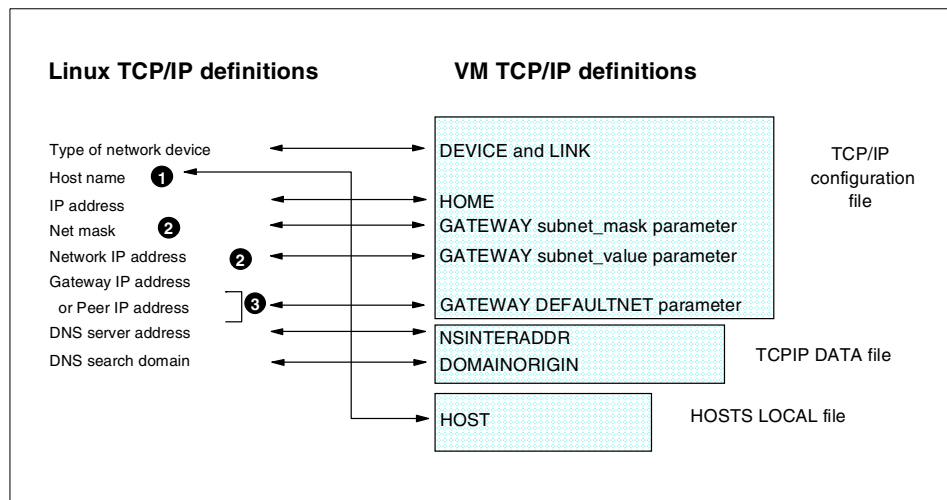


Figure 83. Where to enter network definitions in Linux for S/390 and VM TCP/IP

- ❶ The HOSTS LOCAL file is optional if the VM TCP/IP stack is using an external Domain Name Server (DNS).
- ❷ For both Linux for S/390 and VM TCP/IP, a subnet mask and subnet value are coded when you are using a network router. For a point-to-point link you enter the parameter HOST in place of subnet mask and subnet value on the GATEWAY statement in the TCP/IP configuration file.

- ③ If defining a CTC link, the Linux for S/390 kernel will prompt for a peer IP address instead of a gateway IP address.

15.3.2 LAN Channel Station (LCS)

The LCS driver can be used to drive an Open Systems Adapter-2 (OSA-2) and a number of other communications devices, such as 3172, 2216 and LAN adapter cards installed in a P/390 or R/390. It is designed to manage any device that behaves as an LCS.

Figure 84 illustrates how Linux for S/390 virtual machines can drive an OSA-2 network interface.

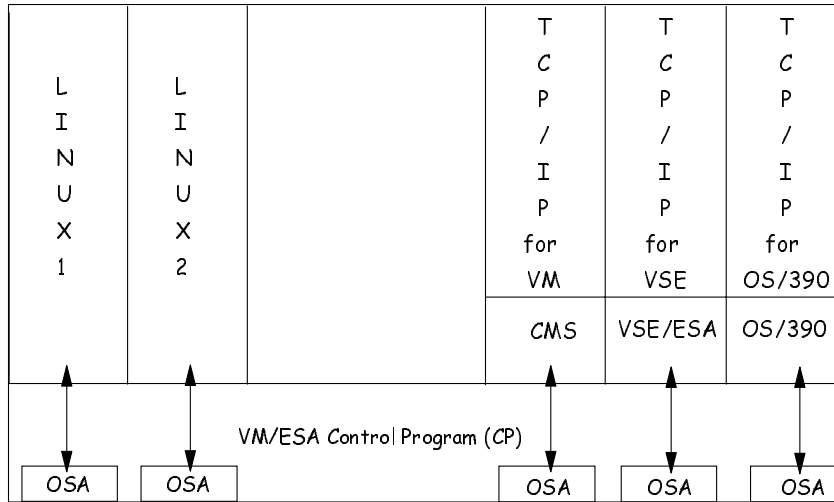


Figure 84. OSA-2 connections for Linux for S/390 running in a virtual machine

For a Linux for S/390 virtual machine to use an OSA-2, a pair of real device numbers associated with the OSA-2 must be dedicated to it. Normally this is done through DEDICATE statements in the CP directory entry for the virtual machine.

For example, if the real OSA-2 device number pair is 21C0/21C1 and you use the same virtual device numbers, the control statements would be:

```
DEDICATE 21C0 21C0
DEDICATE 21C1 21C1
```

Linux for S/390 definitions for an OSA-2 network link managed by a Linux for S/390 virtual machine are the same as for an LPAR. See 15.2.1, “OSA-2 in LPAR” on page 286.

For the VM TCP/IP stack to use an OSA-2, a pair of OSA-2 device numbers must be dedicated to the TCP/IP service machine. Here is an example of coding OSA-2 definitions in the VM TCP/IP configuration file:

```
DEVICE VMTOSA LCS 21C0 ❶  
LINK WTSCVMT IBMTR 0 VMTOSA ❷  
HOME  
...  
9.12.14.155 WTSCVMT ❸  
GATEWAY  
...  
9 = WTSCVMT 2000 0.255.255.0 0.12.14.0 ❹
```

where:

- ❶ The even device number of the OSA-2 device number pair is 21C0. The network device name is VMTOSA. The OSA-2 is driven as an LCS by TCP/IP.
- ❷ Network link WTSCVMT is associated with network device VMTOSA. The OSA-2 LAN interface is a Token Ring adapter.
- ❸ The IP address of this network link is 9.12.14.155.
- ❹ All traffic for the 9.0.0.0 network that is not otherwise routed is sent on the WTSCVMT link.

The subnet to which the OSA-2 interface is connected is 9.12.14. The subnet mask is 255.255.255.0. The convention for coding these values in the GATEWAY statement may seem curious to a Linux networking specialist.

You should enter zero for each octet of the IP address that represents the class of the IP network. Our example shows a class A IP network.

15.3.2.1 CTC

The CTC driver can be used over a virtual channel-to-channel link to connect to the TCP/IP stack on VM, the TCP/IP stack of another S/390 operating system running as guest of VM/ESA, or other Linux for S/390 virtual machines.

For virtual channel-to-channel connections, a pair of devices must be defined both for the Linux for S/390 virtual machine and the target virtual machine.

The CTC driver can also drive a real channel-to-channel link directly. In this case, a real CTC device number pair must be dedicated to the Linux for S/390 virtual machine that will manage the physical CTC interface.

To dedicate a pair of device numbers to a virtual machine, use DEDICATE statements in the CP directory entry for that machine.

For example, if the real CTC device number pair is 808/809 and you use the same virtual device numbers, the control statements would be:

```
DEDICATE 808 808
DEDICATE 809 809
```

CP couple commands are used to connect the CTC device number pairs. Care must be taken to connect the sending device number to the receiving device number and vice versa. See Figure 10 on page 103.

Linux for S/390 definitions for a CTC network link managed by a Linux for S/390 virtual machine are the same as for an LPAR.

Possible CTC connections are illustrated in Figure 85.

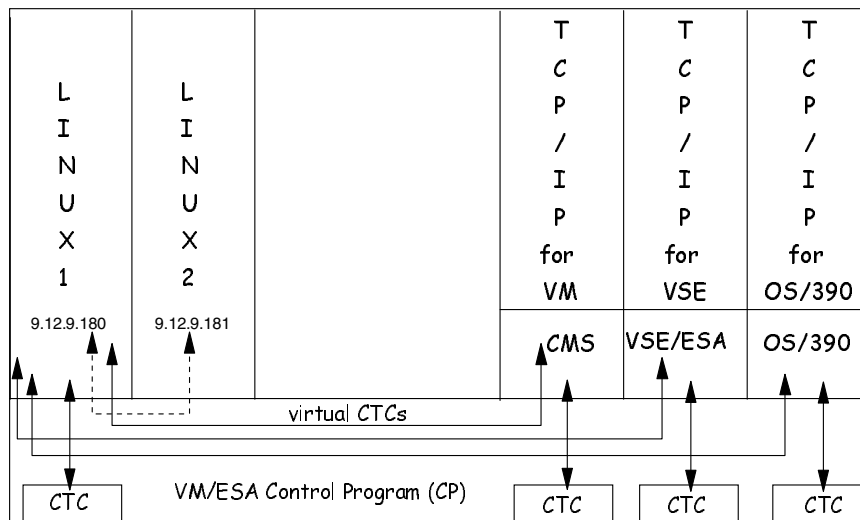


Figure 85. CTC connections for Linux for S/390 running in a virtual machine

CTC connections between Linux for S/390 virtual machines are shown with dotted lines. This indicates that point-to-point connections between pairs of Linux for S/390 virtual machines are not essential if routing to the physical network is through the VM TCP/IP stack.

Here is an example of the VM TCP/IP configuration file definitions needed:

```
DEVICE LINUX5 CTC 808 ❶  
LINK LINUX5VM CTC 0 LINUX5 ❷  
  
HOME  
  
...  
9.12.9.178 LINUX5VM ❸  
  
GATEWAY  
  
...  
9.12.9.184 = LINUX5VM 1500 HOST ❹
```

where:

- ❶ The even device number of the CTC device pair is 808. The network device name is LINUX5.
- ❷ Network link LINUX5VM is associated with network device LINUX5.
- ❸ The IP address of this network link is 9.12.9.178.
- ❹ Traffic destined for IP address 9.12.9.184, the Linux for S/390 CTC network interface, is sent over the LINUX5VM link.

15.3.3 IUCV

The IUCV driver does not manage a physical network interface, but instead provides a high-speed pipe for communications between Linux for S/390 virtual machines and the VM TCP/IP stack.

IUCV connections can also be established between pairs of Linux for S/390 virtual machines on the same VM system, or even on different VM systems. IUCV connections yield highest bandwidth and shortest latency between two Linux for S/390 virtual machines.

To use the IUCV driver for communications, a virtual machine must be authorized. The IUCV control statement in the CP user directory permits a virtual machine to create an IUCV communication path with another virtual machine.

The ALLOW parameter specifies that any other virtual machine can establish a communication path with this virtual machine. No further authorization is required in the virtual machine that initiates the communication.

The ANY parameter is a general authorization indicating that a communications path can be established with any other virtual machine.

The following control statement in the CP directory entry for the TCP/IP service machine would be sufficient to allow an IUCV link to be established by Linux for S/390 to VM TCP/IP:

```
IUCV ALLOW
```

For completeness the appropriate directory entry for the Linux for S/390 userid to connect to any userid (including TCP/IP service machine):

```
IUCV ANY
```

There is an IUCV driver in the VM/ESA TCP/IP stack. This enables TCP/IP communications to be established between Linux for S/390 virtual machines and VM/ESA TCP/IP stacks using high-speed IUCV links.

Figure 86 on page 293 shows how IUCV connections can be used by Linux for S/390 virtual machines to communicate with one another or with a VM TCP/IP service machine.

IUCV connections between Linux for S/390 virtual machines are shown with dotted lines. This indicates that point-to-point connections between pairs of Linux for S/390 virtual machines are not essential if routing to the physical network is through the VM TCP/IP stack.

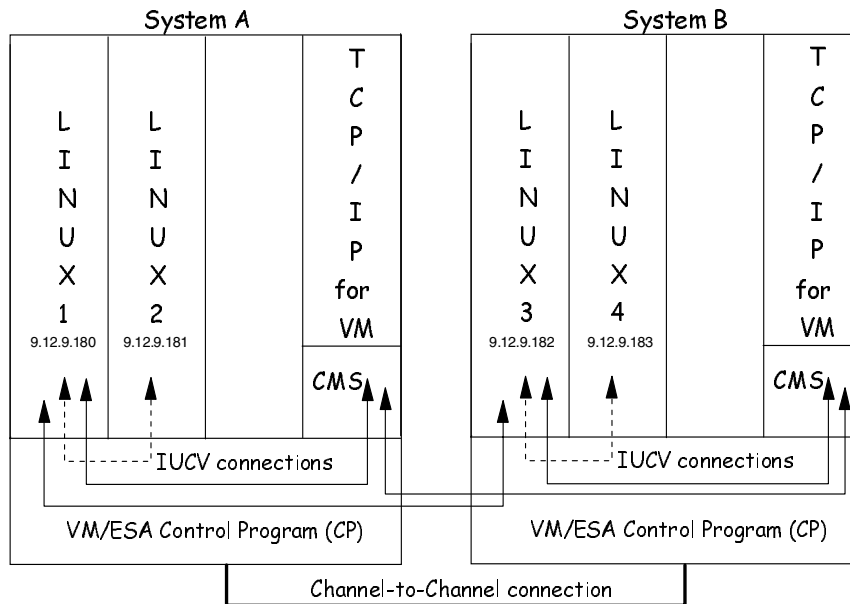


Figure 86. IUCV connections for Linux for S/390 running in a virtual machine

When you define an IUCV link, either in Linux for S/390 or the VM TCP/IP service machine, one IUCV connection is created for reading and writing.

IUCV connections from a Linux for S/390 virtual machine cannot be defined through the network prompts when the Linux for S/390 kernel boots.¹

The boot parameter file must contain an IUCV parameter specifying the user IDs of other virtual machines with which you wish to communicate. The syntax of that parameter is:

```
iucv=userid1,userid2,...
```

One of these user IDs could be the TCP/IP service machine.

Linux for S/390 always uses the same method to name devices. This means the IUCV device names are of the form `iucv0`, `iucv1`, and so on. If the IUCV statement in the kernel parameter file looked like this:

```
iucv=tcPIP,linux4
```

¹ The Linux for S/390 netsetup script is being enhanced to prompt for IUCV network links.

then the IUCV connection to the `tcpip` virtual machine would be on device `iucv0` and the IUCV connection to the `linux4` virtual machine would be on device `iucv1`.

Once the kernel has finished booting, an IUCV link can be established with another virtual machine by entering the following Linux for S/390 command:

```
ifconfig iucvn your_IP_address pointopoint target_IP_address
```

The first IP address is associated uniquely with the Linux for S/390 IUCV network device. In accordance with the way Linux for S/390 identifies devices, you need to increment *n* for each successive connection you define. The first IUCV network device created by Linux for S/390 will be device `iucv0`.

When connecting to the VM TCP/IP stack, appropriate definitions for IUCV links will also need to be made for the TCP/IP service machine. Here is an example of the VM TCP/IP configuration file definitions needed:

```
DEVICE VMTTCP IUCV 0 0 LINUX5 A ❶  
LINK VMT1 IUCV 0 VMTTCP ❷  
HOME  
...  
9.12.9.173 VMT1 ❸  
GATEWAY  
...  
9.12.9.186 = VMT1 1500 HOST ❹
```

where:

- ❶ The network interface device type is IUCV. Its name is VMTTCP.
- ❷ Network link VMT1 is associated with network device VMTTCP.
- ❸ The IP address associated with this network link is 9.12.14.173.
- ❹ Traffic destined for IP address 9.12.9.186, the Linux for S/390 IUCV network interface, is sent over the VMT1 link.

The MTU size specified here is 1500, which was probably chosen to match the MTU size on the physical LAN interface, thereby avoiding the potential for IP packet disassembly and reassembly. The default MTU size used by Linux for S/390 for an IUCV link is 4092. You can override this by specifying a different MTU size as an option on the `ifconfig` command.

For example, we could have used the following statement in the VM TCP/IP configuration file

```
9.12.9.186 = VMT1 8192 HOST
```

and matched it with this Linux for S/390 `ifconfig` command:

```
ifconfig iucv0 9.12.9.186 pointopoint 9.12.9.173 mtu 8192
```

If the VM TCP/IP stack is used to route IP traffic from Linux for S/390 to the physical network, then you need to use a Linux for S/390 `route` command to define a default route:

```
route add -net default iucvn
```

where *n* is the sequence number for the particular network device.

Here is an example from our project of starting an IUCV link from Linux for S/390:

```
ifconfig iucv0 9.12.9.186 pointopoint 9.12.9.173 ❶
ifconfig
ctc0      Link encap:Serial Line IP
          inet addr:9.12.9.184 P-t-P:9.12.9.178 Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MTU:1500 Metric:1
          RX packets:1 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
iucv0     Link encap:Serial Line IP ❷
          inet addr:9.12.9.186 P-t-P:9.12.9.173 Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MTU:4092 Metric:1 Outfill:4092
Keepalive:4092
          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
route add default gw 9.12.9.173 ❸
```

where:

❶ This Linux for S/390 command establishes a point-to-point connection between the Linux for S/390 IUCV network interface (9.12.9.186) and the VM TCP/IP stack's IUCV network interface (9.12.9.173).

② Details of all Linux for S/390 network connections are displayed with the Linux for S/390 `ifconfig` command. 4092 is the default message size coded in the IUCV driver. It can be changed by specifying the MTU option on the `ifconfig` command.

③ The Linux for S/390 `route` command forces TCP/IP traffic to be routed over the IUCV network interface by default.

15.3.3.1 Making IUCV definition permanent

If Linux for S/390 is using IUCV as a communications transport, you may wish to make these network definitions permanent. This eliminates the need to enter `ifconfig` and `route` commands after booting Linux for S/390.

This can be achieved by treating the IUCV link as a CTC link in the `netsetup` prompts and then copying and editing the resulting Linux network definition files.

For example, the `netsetup` script creates the file `/etc/sysconfig/network-scripts/ifcfg-ctc0`.

Copy this file as `/etc/sysconfig/network-scripts/ifcfg-iucv0`. Edit the file `etc/sysconfig/network-scripts/ifcfg-iucv0` and change `DEVICE=ctc0` to `DEVICE=iucv0`. Also edit the file `/etc/sysconfig/network` to change `GATEWAYDEV=ctc0` to `GATEWAYDEV=iucv0`.

Remember to add the appropriate `iucv=` statement to the kernel parameter file that you are using.

15.3.3.2 Distributed IUCV

IUCV communications can even span VM/ESA system images transparently.

In Figure 84 on page 288, we show two VM/ESA images connected physically by a CTC link. These two systems form a Communications Services (CS) collection using the Inter-System Facility for Communications (ISFC) component of the VM/ESA Control Program (CP).

When a virtual machine attempts to make an IUCV connection to another virtual machine, the CP on the local system first tries to locate the target user ID locally. If it fails to find it there, it tries to locate the target user ID on another VM system in the CS collection. When the target user ID is found, an IUCV connection is established to it. The cross-system communications are transparent to the application.

By exploiting distributed IUCV, it becomes possible to extend the scope of a VM-based Linux for S/390 virtual server farm to multiple system images.

To implement distributed IUCV communications, you must first enable this capability in the VM system configuration file (SYSTEM CONFIG) on each system participating in the CS collection. Add the following statements:

```
SYSTEM ID model cpuid system_id domain_name
DISTRIBUTE IUCV YES
```

Although the same system identifier can be specified for multiple VM systems, the domain name must be unique for each system in the CS collection. For further information on the VM system configuration file, consult *VM/ESA V2R4.0 Planning and Administration*, SC24-5750.

When both VM systems are initialized, the CS collection is established by entering this CP command on each system:

```
ACTIVATE ISLINK rdev
```

The device number is the CTC device number. With just one CTC device number, you will get only half-duplex communications. For full-duplex communications, you need an even/odd pair of device numbers. However, only one of the device numbers needs to be specified on the CP ACTIVATE ISLINK command.

Unique user IDs should be enforced on each system in the CS collection to avoid confusion or ambiguity.

We experimented with establishing an IUCV link from a Linux for S/390 virtual machine to a Linux for S/390 virtual machine on another system. We did not succeed, but that may have been due to problems in the IUCV device driver itself rather than with IUCV or the CS collection.

15.3.4 CTC or IUCV

Linux for S/390 in a virtual machine can use either the CTC or IUCV device driver for communications with other virtual machines.

Both drivers provide high-speed communications. Your choice should take the following characteristics of each driver into account:

- The CTC driver can be used with both virtual and real CTC devices. The `netsetup` script prompts for CTC networking definitions.
- IUCV is even faster than CTC and works between VM system images. It does not need virtual devices to be defined. However, it does require

definition in the kernel parameter file. The Linux for S/390 `ifconfig` and `route` commands are needed to establish TCP/IP connections over an IUCV link.

In selecting IP addresses for Linux to use in your network, bear in mind that VM TCP/IP does not support proxy Address Resolution Protocol (ARP). Therefore, a point-to-point link connected to VM TCP/IP cannot use an IP address in a subnetwork LAN connected to the same VM TCP/IP. Assigning IP addresses can become unnecessarily difficult and can lead to configuration errors that are difficult to explain. Proxy ARP support will be developed for VM TCP/IP to facilitate using virtual point-to-point links to connect other VM guest virtual machines such as Linux for S/390 to VM TCP/IP. This support will be delivered through APAR PQ37902.

15.3.5 Linux for S/390 configuration files

Linux for S/390 network definitions are stored in more than one file. General definitions can be found in `/etc/sysconfig/network` and device-specific definitions can be found in `/etc/sysconfig/network-scripts/ifcfg-<network device name>`.

For example, here are the contents of the files created by the network prompts when we installed Linux for S/390 in a virtual machine:

```
[root@linux5 /]# cat /etc/sysconfig/network
NETWORKING=yes
FORWARD_IPV4=no
HOSTNAME=linux5
GATEWAYDEV=ctc0
GATEWAY=9.12.9.178
[root@linux5 /]# cat /etc/sysconfig/network-scripts/ifcfg-ctc0
DEVICE=ctc0
USERCTL=no
ONBOOT=yes
BOOTPROTO=none
REMIP=9.12.9.178
NETWORK=9.12.9.0
NETMASK=255.255.255.0
IPADDR=9.12.9.184
```

15.3.6 VM TCP/IP configuration files

This chapter assumes that the reader has some background in the installation and configuration of TCP/IP for VM. A detailed discussion of this topic is beyond the scope of this book.

If you need an introduction to TCP/IP for VM, refer to *VM/ESA V2R4.0: TCP/IP Function Level 320 Planning and Customization*, SC24-5847, and *TCP/IP Solutions for VM/ESA*, SG24-5459.

Specific VM TCP/IP definitions required for connecting to a Linux for S/390 guest are described elsewhere in this chapter and in Chapter 6.3.3, “Networking definitions” on page 103.

Here are the Linux-related extracts from the VM TCP/IP configuration file that we used on this project.

OSA-2 network interface:

```
DEVICE VMTOSA LCS      21C0
LINK WTSCVMT IBMTR  0 VMTOSA
```

CTC connection from WTSCVMT to LINUX1:

```
DEVICE LINUX1  CTC      800
LINK LINUX1V  CTC  0 LINUX1
```

CTC connection from WTSCVMT to LINUX2:

```
DEVICE LINUX2  CTC      802
LINK LINUX2V  CTC  0 LINUX2
```

CTC connection from WTSCVMT to LINUX3:

```
DEVICE LINUX3  CTC      804
LINK LINUX3V  CTC  0 LINUX3
```

CTC connection from WTSCVMT to LINUX4:

```
DEVICE LINUX4  CTC      806
LINK LINUX4V  CTC  0 LINUX4
```

CTC connection from WTSCVMT to LINUX5:

```
DEVICE LINUX5  CTC      808
LINK LINUX5V  CTC  0 LINUX5
```

IUCV connection from WTSCVMT to LINUX5:

```
DEVICE VMTTCP  IUCV 0 0 LINUX5 A
LINK VMT1 IUCV  0 VMTTCP
```

IP addresses for each network link:

```
HOME
9.12.14.155 WTSCVMT
9.12.9.174  LINUX1V
```

```
9.12.9.175 LINUX2V
9.12.9.176 LINUX3V
9.12.9.177 LINUX4V
9.12.9.178 LINUX5V
9.12.9.173 VMT1
```

Gateway statements:

```
GATEWAY
9          =          WTSCVMT  2000      0.255.255.0 0.12.14.0
9.12.9.186 =          VMT1     1500      HOST
9.12.9.180 =          LINUX1V  1500      HOST
9.12.9.181 =          LINUX2V  1500      HOST
9.12.9.182 =          LINUX3V  1500      HOST
9.12.9.183 =          LINUX4V  1500      HOST
9.12.9.184 =          LINUX5V  1500      HOST
DEFAULTNET 9.12.14.75 WTSCVMT  2000      0
```

Start each network interface:

```
START VMTOSA
START LINUX1
START LINUX2
START LINUX3
START LINUX4
START LINUX5
START VMITCP
```

For this project we used only static routing in VM TCP/IP. For dynamic routing you would need to implement the route daemon (routed service virtual machine) and code BSDROUTINGPARMS statements in the TCP/IP configuration file.

15.4 TCP/IP for OS/390 connectivity

The TCP/IP stack for OS/390 begins with Version 2.5, it is part of the Communication Server for OS/390. Both now share various service routines such as buffer management, line driver, etc. Though TCP/IP still lives in the OS/390 environment -- for example, it is started as a *started task* (STC) -- it serves *both* the conventional and the Unix System Services environment. This may have certain impacts on the configuration that we discuss here briefly.

TCP/IP on OS/390 can be started more than once! That is, you can use completely isolated IP stacks with different IP addresses. The primary application for such a multistack environment is to provide the same or similar

TCP/IP functionality on their *well-known port*. Telnet is such an example. With the availability of UNIX System Services there were two ways to `telnet` into OS/390: via the long-existing Telnet for 3270 (`tn3270`), and to `telnet` into the UNIX System Services environment, which is a character-oriented dialog. Both use port 23. With two stacks you can easily keep the port number for both types of Telnet the same. In other words, starting more than one TCP/IP stack on OS/390 gives the impression of connecting to different hosts.

Our system had two stacks, one with address 9.12.2.17 (WTSC52OE), the other with address 9.12.14.208 (WTCS52); see Figure 87.

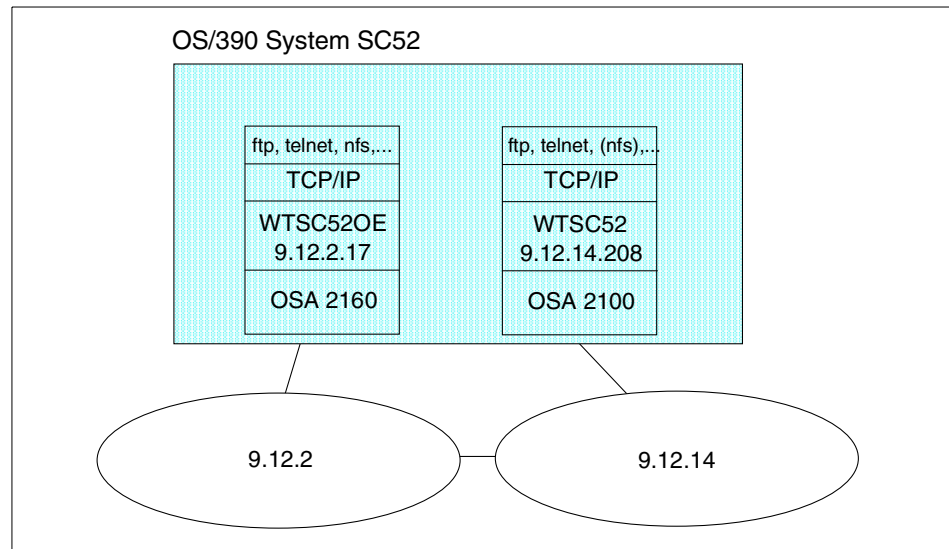


Figure 87. Two-TCP/IP-stack configuration for OS/390

The parameters for this configuration are stored in the PROFILE data set. The statements that define the basic IP parameters are the `DEVICE`, `LINK`, `HOME`, and `GATEWAY` statements:

```

; Hardware definitions: OS/390 TCP/IP stack
; WTSC52
;
;
DEVICE OSA2100 LCS 2100
LINK OSAL2100 IBMTR 0 OSA2100
;
; HOME internet (IP) addresses of each link in the host.
;
; NOTE:

```

```

;
; The IP addresses for the links of an Offload box are specified in
; the LINK statements themselves, and should not be in the HOME list.
;
HOME
    9.12.14.208    OSAL2100
;
GATEWAY
;
; Direct Routes - Routes that are directly connected to my interfaces.
;
; Network First Hop Link Name Packet Size Subnet Mask Subnet Value
9          =      OSAL2100    4096    0.255.255.0    0.12.14.0

; Hardware definitions: USS TCP/IP stack
; WTSC52OE
;
;
DEVICE OSA2160 LCS          2160
LINK   OSAL2160 IBMTR      0    OSA2160
;
; HOME internet (IP) addresses of each link in the host.
;
; NOTE:
;
; The IP addresses for the links of an Offload box are specified in
; the LINK statements themselves, and should not be in the HOME list.
;
HOME
    9.12.2.17      OSAL2160
;
GATEWAY
;
; Direct Routes - Routes that are directly connected to my interfaces.
;
; Network First Hop Link Name Packet Size Subnet Mask Subnet Value
9          =      OSAL2160    4096    0.255.255.0    0.12.2.0

```

15.4.1 Where to find daemons or services

The major question with this kind of configuration is: where do I find important TCP/IP services like FTP, NFS, etc.? There are two places to look, either the port statements and startup procedures, or scripts of the corresponding service. The port statements in the TCP/IP *profile data set* define to which side of OS/390 the ports belong:

```

PORT
  20 TCP OMVS          ; OE FTP Server
      DELAYACKS        ; Delay transmission acknowledgements
  21 TCP OMVS          ; OE FTPD control port
  23 TCP INTCLIEN      ; MVS Telnet Server
  80 TCP OMVS          ; OE Web Server
  111 TCP PORTMAP      ; Portmap Server
  111 UDP PORTMAP      ; Portmap Server
  135 UDP LLBD         ; NCS Location Broker

```

Here port 23 (Telnet) belongs to the OS/390 side. This is the definition for the tn3270 client. Port 80 belongs to OMVS (the acronym for UNIX System Services), which indicates that the Web server is running on the UNIX part of OS/390.

Consequently, you will recognize all settings for the IP stack that serves the UNIX System Services side of OS/390:

```

PORT
  20 TCP OMVS          ; OE FTP Server
      DELAYACKS        ; Delay transmission acknowledgements
  21 TCP OMVS          ; OE FTPD control port
  23 TCP OMVS          ; OE Telnet Server
  80 TCP OMVS          ; OE Web Server
  111 UDP OMVS         ; OE Portmapper Server
  111 TCP OMVS         ; OE Portmapper Server
  443 TCP OMVS         ; OE Web Server SSL Port
  512 TCP OMVS         ; OE Remote Execution Server
  513 TCP OMVS         ; OE Rlogin Server
  514 TCP OMVS         ; OE Remote Shell Server
  514 UDP OMVS         ; OE SyslogD Server
  515 TCP OMVS         ;
; 623 TCP INTCLIEN    ; Telnet Server

```

All ports are assigned to the USS side. The line that is commented out shows that on this stack tn3270 was assigned to use port 623 instead of the well-known port 23.

The second parameter you may have to look for is the environment variable `_BPXK_SETIBMOPT_TRANSPORT`. This variable makes it possible to control to which stack an application binds in case more than one stack is active. Normally, TCP/IP applications use an `unspecific bind()` request, which connects to a default stack. The following example shows the startup JCL for the FTP server using the OS/390 stack `TCPIPOMVS`:

```

//FTPMVS PROC MODULE='FTPD',PARMS='',
//          P1='ENVAR("_BPXK_SETIBMOPT_TRANSPORT=TCPIPOMVS")'

```

```

//FTPD MVS EXEC PGM=&MODULE,REGION=0K,TIME=NOLIMIT,
//          PARM=' POSIX (ON) ALL31 (ON) &P1/&PARMS '
//*STEPLIB DD DSN=TCPIP.SEZALINK,DISP=SHR
//*          DD DSN=CEE.SCEERUN,DISP=SHR
//CEEDUMP DD SYSOUT=*
//SYSFTPD DD DISP=SHR,DSN=TCPIP.MVS.&SYSNAME..FTP.DATA
//SYSTCPD DD DISP=SHR,DSN=TCPIP.MVS.&SYSNAME..TCPIP.DATA
//SYSFTSX DD DISP=SHR,DSN=TCPIP.MVS.STANDARD.TCPXLBIN
//TCPXLBIN DD DISP=SHR,DSN=TCPIP.MVS.STANDARD.TCPXLBIN

```

The FTP startup procedure shown in the next example connects to the TCP/IP stack serving the USS environment:

```

//FTPDOE PROC MODULE='FTPD',PARMS=' ',
//          P1='ENVAR("_BPXK_SETIEMOPT_TRANSPORT=TCPIPOE")'
//FTPD OE EXEC PGM=&MODULE,REGION=0K,TIME=NOLIMIT,
//          PARM=' POSIX (ON) ALL31 (ON) &P1/&PARMS '
//*STEPLIB DD DSN=TCPIP.SEZALINK,DISP=SHR
//*          DD DSN=CEE.SCEERUN,DISP=SHR
//CEEDUMP DD SYSOUT=*
//SYSFTPD DD DISP=SHR,DSN=TCPIPOE.&SYSNAME..FTP.DATA
//SYSTCPD DD DISP=SHR,DSN=TCPIPOE.&SYSNAME..TCPIP.DATA
//*YSFTSX DD DISP=SHR,DSN=TCPIP.STANDARD.TCPXLBIN

```

And, yes, you can start these services in OS/390!

For more detailed information, refer to *Accessing OpenEdition MVS from the Internet*², SG24-4721.

15.4.2 Troubleshooting OS/390 TCP/IP to Linux for S/390

We assume that the OS/390 system you are going to access already provides TCP/IP-based services at a production level.

Confusion can be caused by the fact that your OS/390 server has two TCP/IP stacks running. This is normally the case to preserve well-known ports. That is, Telnet runs on port 23, but there are two Telnets available: Telnet 3270 (tn3270) to emulate a 3270 session, and character mode Telnet to interact with Unix System Services on OS/390. Both use port 23. To preserve their well-known ports, a second TCP/IP instance may run on OS/390. Both instances will run with their *own* parameter sets.

The following brief summary will help you locate the IP address or port number a service uses.

² OpenEdition was the former name of OS/390 UNIX System Services, MVS was the predecessor of OS/390. This redbook contains information that is still valid and helpful.

- Telnet** If your OS/390 system has a two-stack implementation, you have to ask which IP address serves the OS/390 tn3270 and which serves the UNIX System Services. In case there is only one stack started, ask for the port number the service in question uses. You will typically use a port number other than 23 because this is already in use for tn3270. For a Telnet client this implies that you can change the default port number 23 to the one required by your target system.
- FTP** Primarily ask for the IP address. The FTP server on OS/390 can serve both types of file systems.
- NFS** NFS can only be started once. Ask for the IP stack and the address to which NFS is bound.

As in other systems, the most helpful service in the OS/390 environment is `ping` (1). We suggest that you use it from the Unix System Services shell, because you will try to access services there. The following shows an example:

```
HDM § :/u/hdm/xedit::>ping
EZZ3112I Host name or address not entered.
HDM § :/u/hdm/xedit::>ping linuxx (1)
CS V2R8: Pinging host linuxx (9.12.14.196)
Ping #1 response took 0.036 seconds.
HDM § :/u/hdm/xedit::>nslookup (2)
Default Server: itsodns.itso.ibm.com
Address: 9.12.2.7

> linuxx
Server: itsodns.itso.ibm.com
Address: 9.12.2.7

Name: linuxx.itso.ibm.com
Address: 9.12.14.196

>exit
HDM § :/u/hdm/xedit::>host linuxx (3)
EZZ8321I linuxx.itso.ibm.com has addresses 9.12.14.196
```

Another service that may help to resolve problems when connecting both systems is `nslookup` (2). Using this you can check whether your Linux for S/390 system can be named and addressed using the DNS of your domain.

Finally, the `host` (3) command can be of help, too. Its output is very similar to that of `nslookup`. In case `host` or `nslookup` cannot be found:

```
HDM § :/u/hdm/xedit::>host
host: FSUM7351 not found
```

Check whether the symbolic links have been set correctly:

```
HDM $ :/u/hdm/xedit::>ls -al /bin/host
lrwxrwxrwx  1 AAAAAAA OMVSGRP      25 Sep 17  1999 /bin/host ->
../usr/lpp/tcpip/bin/host
HDM $ :/u/hdm/xedit::>ls -al /bin/nslookup
lrwxrwxrwx  1 AAAAAAA OMVSGRP      30 Sep 17  1999 /bin/nslookup ->
../usr/lpp/tcpip/bin/onslookup
```

15.4.3 Inetd daemon in OS/390

With UNIX System Services there is an explicit inetd available. It will be started at IPL time when USS is initialized. It is customized very much like any inetd daemon running on UNIX or Linux. The following is a configuration file from our OS/390 test system:

```
###
# SCCSID(@(#)inetd.conf1.24.1.6AIX)/* Modified: 19:38:52 9/23/91 */
# Internet server configuration database
#
# (C) COPYRIGHT International Business Machines Corp. 1985, 1989
# All Rights Reserved
# Licensed Materials - Property of IBM
#
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#
# /etc/inetd.conf
#
#           Internet server configuration database
#
# Services can be added and deleted by deleting or inserting a
# comment character (ie. #) at the beginning of a line
#
#=====
# service | socket | protocol | wait/ | user | server | server program
# name    | type   |          | nowait|      | program | arguments
#=====
#
otelnet stream tcp nowait OMVSKERN /usr/sbin/otelnetd otelnetd -l -D all
-m -t
shell   stream tcp nowait OMVSKERN /usr/sbin/orshd orshd -LV
login   stream tcp nowait OMVSKERN /usr/sbin/rlogind rlogind -m
exec    stream tcp nowait OMVSKERN /usr/sbin/orexecd orexecd -LV
#finger stream tcp nowait OMVSKERN /usr/sbin/fingerd fingerd

# Let inetd listen for #incoming smtp(sendmail) connections. We do
# not need to run sendmail as a daemon (-bd) if this line is uncommented.
```

```

#smtp      stream tcp nowait OMVSKERN /usr/lib/sendmail sendmail -bn
#talk      dgram  udp  wait   OMVSKERN /usr/sbin/talkd talkd

#
# Time, discard, chargen, echo services
#
echo       stream tcp nowait STC internal
discard    stream tcp nowait STC internal
chargen    stream tcp nowait STC internal
daytime    stream tcp nowait STC internal
time       stream tcp nowait STC internal
echo       dgram  udp  nowait STC internal
discard    dgram  udp  nowait STC internal
chargen    dgram  udp  nowait STC internal
daytime    dgram  udp  nowait STC internal
time       dgram  udp  nowait STC internal

```

15.5 Access to data and applications

If you are already running OS/390, VM/ESA, or VSE/ESA, or any combination of these three systems, you may want to access data and applications on these platforms from Linux for S/390, or vice versa.

15.5.1 OS/390

Regarding data exchange between Linux for S/390 and OS/390, you have to consider that OS/390, like VM/ESA and VSE/ESA, stores text-oriented data predominantly in EBCDIC, whereas Linux for S/390 stores all such data in ASCII. Further, you may observe the use of different EBCDIC code pages. This is mainly due to the need to support diacritical characters. For example, in Germany you will find text and program sources stored typically in EBCDIC code page 273. In the UNIX System Services environment you find a further code page used, IBM-1047. When you are going to exchange data between these systems in either direction, proper conversion tables must be used. But, to avoid confusion, there is *no* restriction on storing data in other code pages. You have to apply the proper translation when you access the data.

When accessing data on OS/390, you have to consider where this data is stored: in the *conventional* file system of OS/390 or the *hierarchical* file system of UNIX System Services. The latter behaves like every UNIX file system regarding the structure in which data is stored: It uses directories, there is a current working directory, file names can have long names and are case sensitive, etc.

The conventional file system of OS/390 has several properties that need to be discussed in more detail. Files in this file system are called *data sets*.

- Data sets have names that are a maximum of 44 characters in length.
- The naming structure uses qualifiers, each 8 bytes maximum in length, separated by a “.” (period). Not all characters can be used in the name. The first character of a qualifier must always be an alpha character, or one of \$, #, or @.
- The high-level qualifier typically indicates the owner, the lowest level qualifier represents the data type: JCL or CNTL for Job Control Language data, PL1 for source code written in the PL/1 programming language.
- There are two types of data sets, *sequential* and *partitioned*. The first type can be compared with flat files on a UNIX system, while the latter looks like a flat file, too, but has an internal directory that makes it possible to store different members separately. A partitioned data set can be compared with an archive file (.a). When accessing a partitioned data set through NFS or FTP, it behaves like a directory on a UNIX or DOS-like system. Each member behaves like a sequential data set.
- Unlike files on other operating systems, data sets on OS/390 have an external structure that consists of a *record format*, a *record length*, and a *block size*. For text data you should set the new line delimiter, which is *carriage return line feed* (crlf) by default.
- A data set on OS/390 has to be *allocated* before it can be used. Allocation is done in space units, which can be tracks, cylinders, or size (in bytes). Allocation is split into two parts, *primary* and *secondary* allocation. The first gives the size a data set initially has, the latter defines the amount of space by which the size may increase dynamically.

Regarding binary data representation, you should be aware of the following: S/390 is a big endian system. That means the high-value byte is at the lower address. Intel-based platforms are called little endian systems. When you exchange data with such a system, the application that will use the data *must* act accordingly to make sure that the data will be in a useful format.

Exchange between S/390 systems and RISC-based systems should work immediately. Care has to be taken with floating-point data. With 9672 generations 5 and 6, S/390 supports IEEE hardware floating-point arithmetic in addition to hexadecimal floating point. Linux for S/390 supports only IEEE floating-point data. So data conversion has to take place when exchanging hexadecimal data with OS/390 and Linux for S/390.

The following methods can be used from Linux for S/390 to access data and applications on OS/390.

15.5.1.1 Network File System (NFS)

The Marist Linux for S/390 distribution contains a compiled NFS client. This can be used to mount the following kinds of OS/390 data:

- Sequential files
- VSAM files
- USS Hierarchical File System (HFS) files
- Partitioned Data Sets (PDS and PDSE)

The NFS server function in OS/390 is delivered through the OS/390 NFS component. There is also an NFS client in OS/390 that allows you to mount all or part of a Linux for S/390 file system from OS/390.

For information on setting up an NFS server on OS/390, refer to *OS/390 V2R6.0 NFS Customization and Operation*, SC26-7253.

15.5.1.2 Telnet

Telnet is a *interactive* access to another system (OS/390, Linux, UNIX, VM/ESA, VSE/ESA). For OS/390, Telnet comes in two flavors, one to access UNIX System Services, another to access traditional 3270-based applications. As both normally work on port 23, special care has to be taken to separate these two access functions. This is described in 15.4, “TCP/IP for OS/390 connectivity” on page 300.

15.5.1.3 File Transfer Program (FTP)

FTP provides a means to import and export data between Linux for S/390 and OS/390. Data can be transferred either from OS/390 to Linux for S/390 or vice versa. In addition to this base functionality, FTP on OS/390 makes it possible to access data stored in the DB2 Database (for more information see *Accessing OpenEdition from the Internet*, SG24-4721). Furthermore, you can use FTP to *submit* (start) OS/390 jobs in the background. As an example, it is possible to store the jobs described in Figure 10 on page 52 or Figure 11 on page 53 on your Linux for S/390 system and start them through this FTP function. Certainly you need an OS/390 userid with the appropriate authorization rights.

FTP is capable of serving both types of files (conventional and HFS) on OS/390 in the same session. The differentiation is done with the *change directory* (`cd`) command. A `/` as the first character indicates files in the HFS, whereas a qualifier type `cd` routes to the conventional file system, which is shown in the following sequence:

```

[root@linuxx /root]# ftp wtsc52
Connected to wtsc52.itso.ibm.com.
220-FTPDMSV1 IBM FTP CS V2R8 at wtsc52oe.itso.ibm.com, 22:37:25 on
2000-05-28.
220 Connection will close if idle for more than 5 minutes.
Name (wtsc52:root): hdm
331 Send password please.
Password:
230 HDM is logged on. Working directory is "HDM.". (1)
Remote system type is MVS.
ftp> cd main (2)
250 "HDM.MAIN." is the working directory name prefix.
ftp> ls
200 Port request OK.
125 List started OK
Volume Unit    Referred Ext Used Recfm Lrecl BlkSz Dsorg Dsname
TOTTS7 3390    1999/10/07 1  30  FB      80 27920 PO  C
TOTTS3 3390    2000/05/24 1  30  FB      80 27920 PO  CNTL
.
.
TOTTSO 3390    2000/05/08 4   5  FB      80  3120 PS  SCRXTM
TOTTS2 3390    2000/05/03 1  30  FB      80  6160 PO  SOURCE
250 List completed successfully.
ftp> cd /u/hdm (3)
250 HFS directory /u/hdm is the current working directory
ftp> ls
200 Port request OK.
125 List started OK
total 39480
-rw-r--r--  1 AAAAAAA  SYS1          821 Dec 20  1997 README.bin.os390
-rwx-----  1 AAAAAAA  SYS1          111 Dec 10  1998 Vsmregm.c

```

The default setting for this server points to the conventional file system (1). Using the `cd` (2) command we navigate within this file system and display some entries there. At (3) we switch into the hierarchical file system of USS. The file system initially selected is defined through the start parameter `startdir` of `ftp.data`. This configuration file can be a data set of a file in the HFS in `/etc`.

To get a status of the current settings from the FTP configuration data set (`ftp.data`), issue the command `rstatus`. For more information on using the FTP server on OS/390, see *OS/390 TCP/IP OpenEdition User's Guide*, GC31-8305.

15.5.1.4 Remote Execution (REXEC) and Remote Shell (RSH) access

The Linux for S/390 `rexec` and `rsh` commands can be used to execute a command on OS/390 and receive the results back. To use `rsh` or `rexec`, you must have a REXEC daemon running on OS/390.

The RSH client passes the local user name, remote user name, and command to the RSH daemon. The remote user name may be in the form user/password when the RSH daemon is an OS/390 host. The daemon provides automatic logon and user authentication, depending on the parameters that you set.

The REXEC client passes the user name, password, and command to the REXEC daemon. The daemon provides automatic logon and user authentication, depending on the parameters that you set.

There are also a Unix System Services `orexec` client and `orexec` daemon available for OS/390.

If you want to execute UNIX commands on Unix System Services of OS/390 via `rexec` and `rsh`, the following settings must be set:

- Port numbers 512 and 514 must be assigned to OMVS in the appropriate profile data set of TCP/IP:

```
Port
512 TCP OMVS
514 TCP OMVS
```

In case you have two TCP/IP stacks running on OS/390, this must be done for the one working for the Unix System Services.

- The `inetd.conf` file must contain appropriate statements to handle the `rsh` and `rexec` commands. An example is given in 15.4.3, “Inetd daemon in OS/390” on page 306.

RSH and REXEC can be used to submit OS/390 batch jobs as well as TSO or Unix System Services commands.

Due to a potential security exposure, use of these TCP/IP applications should be very closely controlled. In a Linux for S/390 environment we would expect them to be used only over a point-to-point connection to a trusted OS/390 host running in another LPAR or virtual machine on the same system.

15.5.1.5 TCP/IP sockets

You can write your own unique sockets applications to connect Linux for S/390 applications to OS/390 applications and data. OS/390 provides both Open Network Computing remote procedure call (ONC/RPC) interfaces and

Distributed Computing Environment (DCE) RPCs. Your application can be built on these interfaces.

15.5.1.6 Special purpose middleware

To date, IBM has announced that it will provide the following middleware that will run on Linux for S/390:

- DB2 Universal Database for Linux for S/390
- WebSphere advanced edition with Java JDK

Connectors for:

- DB2
- MQSeries
- CICS
- IMS

Also, a Tivoli TSM Client statement of intent.

These middleware solutions will make it much easier to connect from Linux for S/390 to OS/390 applications and data.

15.5.2 VM/ESA

The following methods can be used from Linux for S/390 to access data and applications residing on VM.

15.5.2.1 Telnet

You can use the `telnet` command from Linux for S/390 to log on to a VM system in line mode. Generally, this has little value but might make some sense when using the VM/ESA OpenEdition shell.

Brief experimentation showed that you are likely to run into codepage translation problems.

From a CMS session you can also Telnet to Linux for S/390:

```
telnet linux5.itso.ibm.com
VM TCP/IP Telnet Level 310
Connecting to LINUX5.ITSO.IBM.COM 9.12.9.184, port TELNET (23)
```

```
Using Line Mode...
```

```
Notes on using Telnet when in Line Mode:
- To hide Password, Hit PF3 or PF15
```

```

- To enter Telnet Command, Hit PF4-12, or PF16-24
Linux 2.2.15 (linux5.itso.ibm.com) (tty0)

linux5 login: root
Password:
Last login: Tue May 30 00:00:29 from 9.12.9.178
root@linux5 /rootâ# ls /boot
System.map image ipldump.boot ipleckd.boot iplfba.boot kernel.h
parmline
root@linux5 /rootâ#

```

Note that there are some minor codepage issues to be dealt with.

15.5.2.2 Network File System (NFS)

The Marist College Linux for S/390 distribution contains a compiled NFS client. It can be used to mount the following kinds of VM/ESA data:

- CMS minidisk files
- Shared File System (SFS) files
- OpenEdition Byte File System (BFS) files

The NFS server function in VM/ESA is currently delivered through the NFS feature of VM/ESA.

For information on setting up an NFS server on VM/ESA, refer to *VM/ESA V2R4.0: TCP/IP Function Level 320 Planning and Customization*, SC24-5847. Another useful reference is the ITSO redbook *TCP/IP Solutions for VM/ESA*, SG24-5459.

There is further detailed discussion of NFS in Chapter 19, “Network File System (NFS)” on page 367.

15.5.2.3 File Transfer Program (FTP)

FTP provides a means to import and export data between Linux for S/390 and the VM/ESA Conversational Monitor System (CMS). Data can be transferred either from CMS to Linux for S/390 or vice versa.

FTP is discussed further in Chapter 17, “File Transfer Protocol (FTP)” on page 335.

15.5.2.4 Remote Execution Protocol (REXEC) and Remote Shell Protocol (RSH)

The Linux for S/390 `rexec` and `rsh` commands can be used to execute a command on VM/ESA and receive the results back.

To use `rsh` or `rexec`, you must have a REXEC daemon running on VM/ESA. The REXECD virtual machine implements the REXEC and RSH daemons.

RSH and REXEC can be used to submit VM/ESA batch jobs as well as CMS commands.

Use of these TCP/IP applications should be very closely controlled as there are potential security exposures.

15.5.2.5 TCP/IP sockets

You can write your own unique sockets applications to connect Linux for S/390 applications to CMS applications and data.

15.5.2.6 Special purpose middleware

IBM has announced DB2 Connect, which runs on Linux for S/390. This middleware solution makes it much easier to connect from Linux for S/390 to DB2 databases on VM/ESA.

These middleware solutions will also make it much easier to connect from Linux for S/390 to VM/ESA applications and data. See 15.5.1.6, “Special purpose middleware” on page 312 for a complete list of announced middleware.

15.5.3 VSE/ESA

The following methods can be used from Linux for S/390 to access data and applications residing on VSE/ESA running in another LPAR or as a guest of VM.

15.5.3.1 Network File System (NFS)

The Marist College Linux for S/390 distribution contains a compiled NFS client. This can be used to mount the following kinds of VSE/ESA data

- VSAM ESDS files
- VSE library members
- VSE/POWER queue entries

The NFS server function in VSE/ESA is currently delivered through the NFS Feature of IBM program product TCP/IP for VSE/ESA, 5686-A04.

There are some limitations for VSE NFS servers. Because NFS was primarily designed to link UNIX systems, it is dependent on having a UNIX file system available on the server. Therefore, the NFS feature of TCP/IP for VSE/ESA cannot process files that do not have an emulated directory structure.

These include ICCF, sequential (flat) files, and VSAM files defined individually in the file system.

VSAM ESDS and KSDS file types are supported for retrieval when accessed using a VSAM catalog entry in the file system. ESDS is the only VSAM file type that NFS supports for output on the VSE system. You can also access VSE libraries and the POWER RDR, PUN, and LST queues.

For information on setting up an NFS server on VSE/ESA, refer to the *TCP/IP for VSE/ESA User's Guide*, SC33-660. Another mine of excellent information is the redbook *Getting Started with TCP/IP for VSE/ESA 1.4*, SG24-5626.

15.5.3.2 File Transfer Program (FTP)

FTP provides a means to import and export data between Linux for S/390 and the VSE/ESA operating system. Data can be transferred either from VSE/ESA to Linux for S/390 or vice versa.

15.5.3.3 TCP/IP sockets

You can write your own unique sockets applications to connect Linux for S/390 applications to VSE/ESA applications and data.

15.5.3.4 Special purpose middleware

IBM has announced DB2 Connect, MQ Series, and CICS connectors that will run on Linux for S/390.

These middleware solutions will make it much easier to connect from Linux for S/390 to VSE/ESA applications and data. See 15.5.1.6, "Special purpose middleware" on page 312 for a complete list of announced middleware.

Chapter 16. Development tools

Here we describe some of the most important tools to compile and possibly debug software from source code. We assume, of course, that Linux is installed, configured and running. Porting issues are not treated in this chapter. There is plenty of software, both commercial and free, for these purposes. You might want to check the list at:

<http://sal.kachinatech.com/>

16.1 Archiving and compression tools

Most source code packages come in some kind of archive that has to be unpacked before compiling. In case you are already familiar with `gzip`, `bzip`, `tar`, and similar programs, you can skip this section.

16.1.1 The `gzip` command

`gzip` is used for file compression. The command:

```
gzip myfile.txt
```

compresses the file `myfile.txt` into `myfile.txt.gz`. The operation is in-place, i.e., the original file is removed after compression. The compressed file can be uncompressed (again in-place) by:

```
gzip -d myfile.txt.gz
```

The sanity of the same archive can be checked by:

```
gzip -t myfile.txt.gz
```

There is no output if the archive is ok; an error message is given if it is not. The content of a `gzip` archive can be listed by:

```
gzip -l myfile.txt.gz
```

The most interesting piece of information in the printed output is perhaps the size that the file will have when uncompressed. Another option that is sometimes useful is `-9`, which chooses maximum compression.

16.1.2 The `bzip2` command

The command `bzip2` is analogous to `gzip`, except that for some mysterious reason the `-l` option is not implemented. The compression rates achieved with `bzip2` are generally superior to `gzip`. Files compressed with `bzip2` usually have the name extension `.bz2`.

16.1.3 The compress command

`compress` is the dinosaur among UNIX compression tools. The name extension indicating an archive created by `compress` is `.Z`. These files are uncompressed with the `uncompress` command. The compression rate is lower than that of `gzip` or `bzip2`.

16.1.4 The tar command

`tar` is used to store multiple files into one archive file. For example, the command:

```
tar -cvf data.tar mydir/
```

creates a file that contains all files (including subdirectories) under the directory `mydir/`. The archived files are not removed. In the last example the option `-v` causes `tar` to operate verbosely—that is, to list all of the files being archived. The contents of the archive can be listed with:

```
tar -tvf data.tar
```

and restored with:

```
tar -xvf data.tar
```

So the three important flags to remember are `-c` for Create, `-x` for eXtract, and `-t` for list.

It is common to compress archives generated by `tar` to save space. The compressed archive (named `data.tar.gz`) obtained from the above by:

```
gzip data.tar
```

can be generated directly using the `-z` switch:

```
tar -cvzf data.tar.gz mydir/
```

Decompression while untarring an archive is also accomplished by the `-z` option. Compatibility between `gzip`-compressed files, `compress`-compressed archives can be handled transparently: use the `-Z` (capital z) option for that purpose. The double file extension `.tar.gz` is sometimes replaced by just `.tgz`. `bzip2` compression can be invoked by using `-I` (capital i) instead of `-z`:

```
tar -cvIf data.tar.bz2 mydir/
```

To unpack the archive in one step, use:

```
tar -xvIf data.tar.bz2
```

Occasionally, the option `-p` is needed, which tells `tar` to set the modes of the extracted files as recorded in the archive without having the `umask` setting.

Note that most, if not all, distributions of Linux allow `gzip` and `bzip2` compression in conjunction with the `tar` command. However, other UNIXes, such as AIX, do not recognize the `tar -z` flag. Also, other UNIXes have moved to the `pax` command, but Linux has stayed with the traditional `tar`.

16.1.5 The `zip` command

The `zip` command (and its counterpart `unzip`) is an archiving tool that comes from the DOS world. Listing the contents of a `zip` archive is done by:

```
unzip -l data.zip
```

and unpacking by:

```
unzip -La data.zip
```

It is not common to find `zip` files on Linux systems.

16.1.6 Other archiving tools

Plenty of other archive formats exist, among them `arj`, `arc`, `cpio`, `rar`, `lha`, and `zoo`. Free software to create/unpack these formats is included in Linux distributions or can easily be obtained from the Web. However, archives of these types are rarely encountered.

16.2 Compilers

Compilers (or interpreters) for virtually every programming language are available for Linux. Here we restrict our attention to the C/C++ compiler, `gcc`, and `perl`.

16.2.1 The `gcc` and `g++` compilers

The GNU C compiler, `gcc`, compiles and links C source code. The GNU C++ compiler `g++` compiles and links both C and C++ input.

```
gcc -Wall hello.c -o hello
```

compiles the file `hello.c` into the binary executable `hello`. Without the `-o` option, the binary name defaults to `a.out`. The switch `-Wall` activates a reasonable set of warnings. Always use `-Wall` with `gcc`.

The command:

```
gcc --help
```

lists the available options for `gcc`. More detailed documentation is given in the man page for `gcc`. Extensive information can be found in the info pages. If you are familiar with `emacs`, it will be convenient to read info pages from within `emacs`. Start `emacs` and use `ctrl-h i` to invoke the reader for info pages.

As most source packages come with makefiles, however, it is sufficient to just say (often only after `./configure` has been executed):

```
make
```

after unpacking the respective archive and changing to the newly created directory.

16.2.2 Perl

Perl stands for Practical Extraction and Report Language.

From the Perl man page:

```
Perl is a language optimized for scanning arbitrary text files,
extracting information from those text files, and printing reports
based on that information. It's also a good language for many system
management tasks. The language is intended to be practical (easy to
use, efficient, complete) rather than beautiful (tiny, elegant, minimal).
```

Perl script files typically have the extension `.pl`. Execute a Perl script by:

```
perl -w myscript.pl
```

Most perl scripts are executables using the same mechanism that makes shell scripts executable. Apart from having the executable bit in the file permission set the first line of the file reads:

```
#!/usr/bin/perl -w
```

The `-w` option causes Perl to report important warnings about possible errors in the script. Always use `-w`. The fact that Perl can be invoked without `-w` is considered a bug in the Perl man page. Perl scripts that do not have the `-w` option are likely to be quick hacks.

Almost everything about Perl can be found at:

```
http://www.perl.com/
```

16.2.3 Regina

This is REXX for UNIX systems. It is an attractive interpreter to anyone familiar with the REXX language. REXX was originally implemented on VM/ESA, but its popularity has spread and it is available today on a multitude of other architectures and platforms.

16.3 Editing Linux files

There have been innumerable flame wars on the subject of “the best UNIX editor”. There is no such animal. Choose whatever you are familiar with in the environment you are working in. I would not recommend becoming dependent on an editor that only works in X, because there will be times when X is not available. There will be times when you will need to know how to edit files without even having an editor available (for example, if your system comes up after a crash and won’t mount /usr/bin).

Your distribution will have chosen a few editors for you. Get to know some of them before you go out and start rolling your own.

There are probably more editors available for Linux than for most other platforms. A quick overview of some of the more common ones can be found at:

<http://www.linuxstart.com/applications/texteditorsreaders.html>

People coming from the S/390 environment will often look for a “prefix” type of editor similar to XEDIT or the TSO editors. The closest equivalent would be THE Hessling Editor, available from:

<http://www.lightlink.com/hessling/THE/index.html>

THE is intended to be similar to the VM/CMS System Product Editor, XEDIT, and to KEDIT from Mansfield Software.

16.3.1 The vi editor

The vi editor is ubiquitous. It will run on almost any kind of terminal worthy of that name. (This does not include the 3215, which is trying to emulate a WTX Teletype, not a terminal). vi does not need cursor keys, although it can use them if the environment is set up correctly. vi comes in various flavors, from plain vanilla up through elvis, vile and vim.

The vi editor is primitive. There are very few moving parts; it only has two gears, Text or Command. You cannot use it to surf the Web, or read your

news and mail. It is just a *Visual Interface* to whatever texts you will need to see, and possibly modify in UNIX. It's simple and it works, so give it a try.

The vi that ships with most Linux distributions is actually vim, which stands for *vi improved*. See:

<http://www.linuxdoc.org/HOWTO/Vim-HOWTO.html>.

16.3.2 emacs

emacs stands for either Escape+Meta+Alt+Control+Shift or Eight Megabytes Always Constantly Swapping.

emacs is not an editor, it is a way of life. Once you are through the basic navigation stage, and know how to use the help system, emacs can become your "character-driven front end" to the rest of the (UNIX) world and beyond. See:

<http://www.cs.cornell.edu/Info/People/raman/emacspeak/emacspeak.html>

For the purpose of this redbook it is sufficient to know that emacs is possibly *the* most productive environment for people that do enormous amounts of text-based work, including correspondence, coding and documentation.

emacs has a built-in tutorial that is started with `ctrl-h t`.

If you know emacs, you should install it. If you do not, install it later.

16.3.3 joe, jove, pico

joe, or Joe's Own Editor, is quite often used by those whose background includes CP/M or Wordstar. `Ctrl+k Ctrl+h` gets you to the help screen, if you do not use it often.

jove (Jonathan's Own Version of emacs) is a text editor based mainly on the original emacs editor written by Richard Stallman at M.I.T. Although jove is mostly emacs-compatible, there are several differences. For example, jove offers automatic indentation, multiple views of more than one file, and shorthands in the form of keys, words, lines and paragraphs.

pico is the editor used by the pine e-mail system, which is familiar to many North American students.

16.3.4 The sed editor

The name sed stands for Stream EDitor. sed is a non-interactive editor. It is useful if it is not possible to open an interactive editor. Its main use is replacing a piece of text in a file.

For example:

```
sed 's/foo/bar/g' myfile.txt > newfile.txt
```

will create the file newfile.txt with the same contents as myfile.txt with all occurrences of the text *foo* replaced by *bar*.

16.3.5 The pfe editor

The Programmers File Editor, pfe, from:

```
http://www.lancs.ac.uk/people/cpaap/pfe/
```

is not a Linux editor, but is included here because you will find it invaluable if you also have to work with W9x, NT, W2K or OS/2, and need an editor that respects the CR/LF/EOF or Newline conventions of both worlds. If you have a pfe icon on your desktop, you can just drag and drop text files from the Windows file manager onto it. This is particularly useful in conjunction with a Samba share of your Linux home directory, or your ~/public_html.

16.3.6 The THE editor

THE stands for The Hessling Editor. It will probably be the preferred editor for Linux users who are familiar with the XEDIT editor of the VM/ESA Conversational Monitor System (CMS).

16.4 Make tools

Make tools are used to automatically build software.

16.4.1 Make

The make man page summarizes make as follows:

The purpose of the make utility is to determine automatically which pieces of a large program need to be recompiled, and issue the commands to recompile them.

In order to do anything sensible with make, one must have a makefile (usually named *Makefile*) in the current directory. The makefile does not contain any kind of program but rather consists of rule sets. These consist of entries that roughly look like this:

```
target1: dep1 dep2
    do_this dep1 dep2 > tmp.txt
    do_that tmp.txt -o target1
```

The first line says that target1 depends on dep1 and dep2 (which might be built by another rule). Lines 2 and 3 are commands to be executed in order to build target1.

Both line 2 and line 3 must start with a tab character. If not, make will produce an error message. As an example, suppose the tab is missing in line 361; this results in the following error message:

```
makefile:361: *** missing separator. Stop.
```

This is not mentioned in the man page for make.

The info pages for (GNU-)make produce a pretty exhaustive reference for most aspects of this utility.

16.4.2 automake

The role of automake is to produce at least an autoconf[1]-compatible Makefile template for completion by autoconf. Typically, the person creating an automake/autoconf package will create an automake “template” file (usually called *Makefile.am*) that identifies which end products are to be built and/or installed.

16.4.3 autoconf

The role of autoconf is to create a configure shell script, which is capable of determining many facets of the machine/system configuration on which it runs, then propagate such settings into other files—typically one or more Makefiles or config.h C/C++ headers.

Figure 88 on page 325 shows the relationship between autoconf, the configure script, make, and the compiler.

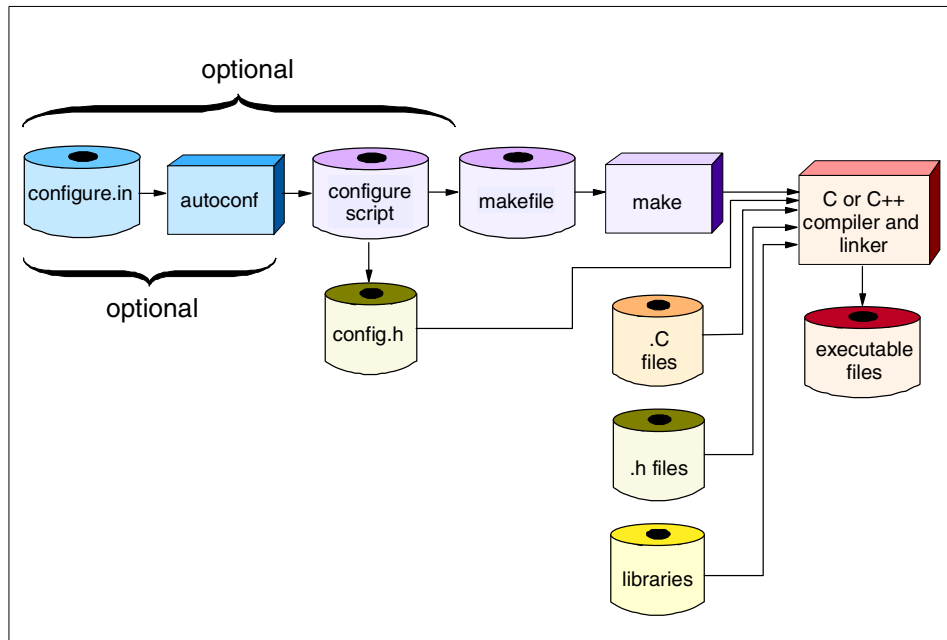


Figure 88. Package build process

16.5 Source code control tools

The following description of source code control systems is taken from:

<http://metalab.unc.edu/mdw/HOWTO/CVS-RCS-HOWTO-1.html>

Source code control system is a *must* to manage the changes occurring to a software project during development. Developer needs a complete history of changes to backtrack to previous versions in case of any problems. Since source code is the most vital component of any software project and software development takes a huge amount of time and money, it is very important to spend some time in safeguarding the source code by using the source code control systems like CVS and RCS.

16.5.1 RCS and CVS

Revision Control System (RCS) and Concurrent Version System (CVS) are version control systems. CVS is built on top of RCS and offers extended functionality like distributed development, often operated as Internet CVS repository. An important feature of these tools is the ability to create an old

version of the source code, which often helps immensely to locate at which point a bug entered into the source tree.

For a nice introduction, see the info pages of CVS or visit:

<http://metalab.unc.edu/mdw/HOWTO/ CVS-RCS-HOWTO.html>

16.5.2 Kdevelop

Kdevelop is an integrated development environment (IDE) running under X11. It offers a documentation browser, a source code editor with syntax highlighting, and a graphical user interface (GUI) for the compiler. Especially if you plan to write programs for X11, the Qt-lib, or KDE, you should consider using it.

Kdevelop comes with detailed documentation. Its Web site is:

<http://www.kdevelop.org/>

16.6 Code analyzers

This section addresses source code analysis tools.

16.6.1 LCLint

The following description of LCLint is taken from:

<http://lclint.cs.virginia.edu/index.html>

LCLint is a tool for statically checking C programs. With minimal effort, LCLint can be used as a better lint. If additional effort is invested adding annotations to programs, LCLint can perform stronger checks than can be done by any standard lint.

Note that LCLint does not work with C++ code. The tool comes without a man page or info page. For more usage information, use the command:

```
lclint -h
```

or visit the LCLint site at:

<http://lclint.cs.virginia.edu/>

16.6.2 Compiler code analyzer features

Always, always, always use the abilities of the compiler to analyze your code. For `gcc` the most interesting single switch is `-Wall`, which does not really enable all possible warnings but a reasonable subset. Study the section of

the man page about “warning options” to choose which warning options not included in `-Wall` you will enable.

In some cases it can be instructive to compile C as C++ just to see which additional warnings are issued by the compiler. To do this, use the `g++` command.

Perl has the `-w` switch; you should never use Perl without it.

16.7 Debugging facilities

SIMPLE is an acronym for Sheer Idiot's Monopurpose Programming Language Environment. This language, developed at the Hanover College for Technological Misfits, was designed to make it impossible to write code with errors in it. The statements are, therefore, confined to BEGIN, END and STOP. No matter how you arrange the statements, you can't make a syntax error. Programs written in SIMPLE do nothing useful. Thus they achieve the results of programs written in other languages without the tedious, frustrating process of testing and debugging.

This quote is from an unknown source, to be found at:

<http://www.dcs.port.ac.uk/~lesterc/humour/G-newpls.html>

Unless you are programming in SIMPLE, you will have to debug your code to achieve the desired results.

16.7.1 The gdb debugger

`gdb`, the GNU debugger, allows you to examine executables (compiled from C or C++ source) in two ways: “post mortem” or “in action”. Assume that the program `myprog` was aborted (probably by a segfault) and the core dump went into a file named `core`. The situation can then be analyzed with:

```
gdb myprog core
```

An interactive debugging session is started. Among the different `gdb` commands, the one most likely to be used in this situation is `bt` (backtrace), which returns a calling trace to the point of the fault.

Observing a program while running is possible after starting a `gdb` session with:

```
gdb myprog
```

After possibly setting breakpoints (`break func_name`) the program is started with `run` (plus possible arguments). After the program has stopped at a breakpoint, one can list the current piece of code (`list`), examine the value of variables (`print <varname>`) or move up (`up`) or down (`down`) in the stack frame.

Detailed information about gdb and its usage is given in the gdb info pages.

16.7.2 The Data Display Debugger, ddd

ddd, the Data Display Debugger, is a graphical front end for gdb. When X11 is available, it offers an intuitive interface to gdb. Additional functionality of ddd, such as the graphic display of data structures, makes debugging very comfortable. The homepage of ddd is at:

<http://www.gnu.org/software/ddd/>

16.7.3 The MALLOC_CHECK_ environment variable

The following is from the information page Libc - Memory Allocation - Unconstrained Allocation - Heap Consistency Checking:

Another possibility to check for and guard against bugs in the use of `malloc`, `realloc` and `free` is to set the environment variable `MALLOC_CHECK_`. When `MALLOC_CHECK_` is set, a special implementation is used which is designed to be tolerant against simple errors, such as double calls of `free` with the same argument, or overruns of a single byte (off-by-one bugs). Not all such errors can be protected against, however, and memory leaks can result. If `MALLOC_CHECK_` is set to `0`, any detected heap corruption is silently ignored; if set to `1`, a diagnostic is printed on `stderr`; if set to `2`, `abort` is called immediately. This can be useful because otherwise a crash may happen much later, and the true cause for the problem is then very hard to track down.

Using the `MALLOC_CHECK_` on programs that come binary-only can be quite interesting and helpful for arguments with the respective software vendor.

16.7.4 nana

Among the various utilities we only mention nana, a library for “improved support for assertion checking and logging in C and C++.” Consult the pertinent page for detailed information about its concepts and usage.

16.8 The strace and ltrace tools

strace and ltrace are tools to trace the system and library calls, respectively. In order to trace the actions of a command just enter it with all options, but preceded by the trace statement.

```
ltrace test -d /usr
```

results in the following output showing all library calls invoked:

```
__libc_start_main(0x0804aee0, 3, 0xbffff3e4, 0x08048754, 0x0804b38c <unfinished ...>
__register_frame_info(0x0804d268, 0x0804d3b0, 0xbffff388, 0x4003a098, 0x400f6618) = 0x400f71c0
setlocale(6, "") = NULL
bindtextdomain("sh-utils", "/usr/share/locale") = "/usr/share/locale"
textdomain("sh-utils") = "sh-utils"
strchr("abcdefghijklmnopstuwxyz0Gnz", 'd') = "defgkLhprsStuwxyz0Gnz"
__xstat64(3, 0xbffff589, 0xbffff2cc, 0x0ab39b0e, 0xbffff2e4) = 0
exit(0) = <void>
__deregister_frame_info(0x0804d268, 4096, 0, 4096, 8) = 0x0804d3b0
+++ exited (status 0) +++
```

The command:

```
strace test -d /usr
```

lists the system calls; that is, those functions called that require kernel action:

```
execve("/usr/bin/test", ["test", "-d", "/usr"], [/* 61 vars */]) = 0
brk(0) = 0x804d508
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=15947, ...}) = 0
mmap(NULL, 15947, PROT_READ, MAP_PRIVATE, 3, 0) = 0x40014000
close(3) = 0
open("/lib/libc.so.6", O_RDONLY) = 3
fstat(3, {st_mode=S_IFREG|0755, st_size=4061504, ...}) = 0
read(3, "\177ELF\1\1\0\0\0\0\0\0\0\0\3\0\0\1\0\0\0\340\213"... , 4096) = 4096
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x40018000
mmap(NULL, 924892, PROT_READ|PROT_EXEC, MAP_PRIVATE, 3, 0) = 0x40019000
mprotect(0x400f3000, 31964, PROT_NONE) = 0
mmap(0x400f3000, 20480, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED, 3, 0xd9000) = 0x400f3000
mmap(0x400f8000, 11484, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x400f8000
close(3) = 0
munmap(0x40014000, 15947) = 0
personality(PER_LINUX) = 0
getpid() = 2261
brk(0) = 0x804d508
brk(0x804d540) = 0x804d540
brk(0x804e000) = 0x804e000
open("/usr/share/locale/locale.alias", O_RDONLY) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=2265, ...}) = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x40014000
read(3, "# Locale name alias data base.\n#"..., 4096) = 2265
read(3, "", 4096) = 0
close(3) = 0
munmap(0x40014000, 4096) = 0
open("/usr/share/i18n/locale.alias", O_RDONLY) = -1 ENOENT (No such file or directory)
open("/usr/share/locale/english/LC_MESSAGES", O_RDONLY) = -1 ENOENT (No such file or directory)
open("/usr/share/i18n/english/LC_MESSAGES", O_RDONLY) = -1 ENOENT (No such file or directory)
```

```
stat("/usr", {st_mode=S_IFDIR|0755, st_size=4096, ...}) = 0
_exit(0) = ?
```

Besides being hugely helpful for debugging purposes, the output of the `trace` commands gives lists of library and system calls whose man pages you will want to study before doing your own UNIX programming.

16.9 bash's -v and -x option

If you are unlucky enough to have to debug a bash script, you will find the options `-v` and `-x` pretty handy (cite from the man page):

```
-v Print shell commands as they are read.
-x Print shell commands and their arguments as they are executed.
```

Try the following on the command line:

```
alias ll='ls -l'
set -v
set -x
ll *
```

These commands should create the following output:

```
ll * 1
+ ls -l file1.txt file2.tar 2
-rw-r--r--  1 root    root      1276 May  4  1996 file1.txt
-rw-r--r--  1 root    root     87210 May  4  1996 file2.tar
```

where:

- 1** - The effect of the bash `-v` flag
- 2** - The effect of the bash `-x` flag; a `+` is put in front

Both switches can be reset with the shell command:

```
set -
```

Placed in strategic positions in shell scripts, `set -v` and `set -x` can be quite a help in debugging.

16.10 Performance analysis with gprof

`gprof`, the GNU profiler, is a tool for the performance analysis of a program. It generates a call graph that shows how often and from where each function was called in the program run. Moreover, the individual and cumulative times spent inside the routines are obtained. In order to profile an executable, it

must be compiled with the option that causes profiling output to be generated during the program run. With `gcc` that option is `-pg`:

```
gcc -Wall -pg -O myprog.c -o myprog
```

The `-O` switch invokes compiler optimizations for speed. During a program run, the file `gmon.out`, which contains the collected profiling data, is created.

The command:

```
gprof myprog gmon.out > mytiming.txt
```

creates a table of human-readable profiling data; the first few lines of a real-world example are:

```
Flat profile:
Each sample counts as 0.01 seconds.
 % cumulative self self total
time seconds seconds calls us/call us/call name
58.06 0.18 0.18 206 873.79 873.79 fht(double *, long)
32.26 0.28 0.10 323 309.60 309.60 fht0(double *, long)
6.45 0.30 0.02 206 97.09 97.09 carry_thru(double *, long, int)
3.23 0.31 0.01 117 85.47 85.47 fht_convolution(double *, double *, long)
0.00 0.31 0.00 1596 0.00 0.00 hfloat::normalized(void) const
0.00 0.31 0.00 208 0.00 0.00 fxtmult::max_dfxflen(void)
0.00 0.31 0.00 206 0.00 97.09 carry(double *, long, int)
0.00 0.31 0.00 206 0.00 0.00 void scramble<double>(double *, long)
0.00 0.31 0.00 206 0.00 873.79 dit_fht(double *, long)
...
```

This is followed by many more lines and the call graph, one of whose entries is:

```
hfdata::radix(int, char *, int) [8]
0.01 0.01 756/9171 get_radix_format(int) [35]
0.05 0.02 2863/9171 hfdata::dump(char *, unsigned long) const [11]
0.10 0.05 5552/9171 fmod(hfloat const &, hfloat const &, hfloat &) [14]
0.16 0.08 9171+366 hfdata::radix(int, char *, int) [8]
0.0 6 0.00 4024/9411 void diff<double>(double *, unsigned long, unsigned long) [17]
```

Though not entirely obvious, `gprof` is useful as a debugging aid, too. In some situations one has a certain expectation about how often a function should be called. The callgraph produced by `gprof` supplies a convenient way to check that. Excessively long or short times spent in parts of a program also often give good hints about where something might be wrong.

16.11 `lex` and `yacc`

GNU's `lex` is `flex` (fast lexical analyzer generator). Its man page says:

```
flex is a tool for generating scanners: programs which recognized
lexical patterns in text. flex reads the given input files, or its
standard input if no file names are given, for a description of a
```

scanner to generate. The description is in the form of pairs of regular expressions and C code, called rules. flex generates as output a C source file, lex.yy.c, which defines a routine yylex(). This file is compiled and linked with the -lfl library to produce an executable. When the executable is run, it analyzes its input for occurrences of the regular expressions. Whenever it finds one, it executes the corresponding C code.

GNU's yacc (Yet Another Compiler Compiler) is bison. The man page of yacc explains:

Yacc reads the grammar specification in the file filename and generates an LR(1) parser for it. The parsers consist of a set of LALR(1) parsing tables and a driver routine written in the C programming language. Yacc normally writes the parse tables and the driver routine to the file y.tab.c.

... and bison claims to be compatible:

Bison is a parser generator in the style of yacc(1). It should be upwardly compatible with input files designed for yacc.

If you do not know what all this is about you probably do not plan to write your own compiler within the next few weeks. If you do, check out the "Compiler Construction Kits " for pointers to Eli, and others, at:

<http://www.first.gmd.de/cogent/catalog/kits.html>

16.12 A simple example

As a demonstration for the usage of make and gdb we compile and debug a small C program. With our favorite editor we create the file hello.c, which contains the following lines:

```
#include <stdio.h>
void hello()
{
    char h[] = "hello world!";
    printf("%s\n", h);
}

int main()
{
    hello();
    return 0;
}
```


In order to automate compilation, a Makefile, which is input to the `make` command, is used:

```
BIN=hello

all:
    gcc -Wall -O -g hello.c -o $(BIN)
    ./$(BIN)

clean:
    rm -f $(BIN)
```

Note that the name of the binary (`hello`) is put into a variable, so that a change of that name only requires a change in one place in the makefile.

The first job (called `all`) compiles and runs the program. The job named `clean` deletes files that are created at compile time: in real-world situations these are likely a bunch of object files. In our case, it's simply the binary. In order to launch the first job, we enter the command:

```
make
```

We get:

```
gcc -Wall -O -g hello.c -o hello
./hello
hello world!
```

The program compiled and ran successfully, and a binary executable named `hello` can be found in the current directory.

Now, let's try the debugger:

```
gdb hello
```

We get copyright information and some information on the started debugger, and a prompt where commands can be entered:

```
GNU gdb 4.18
Copyright 1998 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-suse-linux"...
(gdb)
```

We set a breakpoint at the beginning of the function `hello()`:

```
(gdb) br hello
Breakpoint 1 at 0x8048425: file hello.c, line 7.
```

We could instead also enter the line number where the program should be stopped. By entering `run`, we start the program. At the breakpoint, the debugger stops the execution.

```
(gdb) run
Starting program: /home/jj/t/hello

Breakpoint 1, hello () at hello.c:7
7          printf("%s\n", h);
```

With `bt` (for BackTrace), the stack frame can be examined. The value of an initialized variable can be checked with the `print` command. In more complex situations with a deeper stack frame, one could move up and down in the stack frame by using the commands `up` and `down`.

```
(gdb) bt
#0  hello () at hello.c:7
#1  0x804844b in main () at hello.c:12
(gdb) print h
$1 = "hello world!"
```

We continue the execution and observe that the program works as expected:

```
(gdb) cont
Continuing.
hello world!
```

```
Program exited normally.
```

A final command ends the debugger session and brings us back to the shell prompt:

```
(gdb) quit
```

Chapter 17. File Transfer Protocol (FTP)

This chapter addresses FTP, anonymous FTP, and briefly discusses TFTP.

17.1 Overview of FTP

FTP is a client/server application that uses TCP/IP to transfer files between computers. After http, it is probably the most commonly used Internet protocol.

The client command is normally named `ftp`, which the user invokes to open an interactive session with a server on another host. Initially, the user's identity is authenticated by the remote server and then files can either be sent (`put`) or received (`get`). See 17.7, "Client notes" on page 344 for a list of commonly used `ftp` client subcommands.

17.2 TFTP

There are actually two flavors of FTP. Besides the conventional FTP protocol, there is the Trivial FTP protocol (TFTP), which uses the less reliable but faster UDP/IP network layer. It is often used for booting network devices.

On S/390 Linux, the TFTP service is commented out in the `inetd.conf` file with the following recommendation:

```
# Tftp service is provided primarily for booting. Most sites
# run this only on machines acting as "boot servers." Do not uncomment
# this unless you *need* it.
#
#tftp  dgram  udp      wait    root    /usr/sbin/tcpd  in.tftpd
```

Therefore, TFTP is available on Linux for S/390, but is not started by default.

17.3 Anonymous FTP

The FTP server can be configured to be an anonymous FTP server on Linux for S/390. This allows anyone to access files in the anonymous FTP directory by logging in with a user ID of *anonymous* and a password of (by convention) the e-mail address.

On some Linux distributions, an anonymous FTP server is a standard feature of the install. With Linux for S/390, anonymous FTP is partially set up, but it must be configured after the install.

To do this, the following is suggested:

1. If you are using the large file system from Marist, the FTP user ID should exist. You can verify this by checking the `/etc/passwd` file. If it does not exist, it can be added with the command:

```
$ /usr/sbin/adduser -c "FTP User" -s "" -u 14 ftp
```

Then edit the `/etc/passwd` file and replace the password “!” with “x”. The password is the second colon-delimited field. The value “x” means the FTP user ID cannot log on interactively.

2. Get the `anonftp-2.8-2.s390.rpm` from the Thinking Objects download site:

```
http://linux.s390.org/download/ftp/RPMS/s390/
```

This rpm contains the files that have to exist in the `/home/ftp` directory. Install the rpm as root with the following command:

```
[root@linux390 rpm]# rpm --install anonftp-2.8-2.s390.rpm
```

You should now be able to use anonymous FTP. The files that will be accessible to anonymous FTP users are in `/home/ftp`. Typically the `/pub` (for public) directory is used to make files available to anyone with network connectivity to the server.

17.4 Controlling access

Access to the Linux for S/390 FTP server can be controlled using either the traditional or anonymous FTP security models. Often, both conventional and anonymous FTP servers are used in tandem through the administration of a single `ftpd` server process.

17.4.1 Traditional FTP security

The traditional FTP server forces the user to authenticate himself with a valid user ID/password pair. As such, each person accessing the FTP server must have a valid login name and password defined in the `/etc/passwd` file.

17.4.2 Anonymous FTP security

Properly securing an anonymous FTP server is very important as it is often a means to gain control of the system by those with malicious intent. Using the anonymous FTP files described in 17.3, “Anonymous FTP” on page 335 may have most of the proper securities in place, but to thoroughly secure your anonymous FTP server, it is suggested you read the respective Computer Emergency Response Team (CERT) reports.

They are on the Internet at:

`ftp://ftp.cert.org/pub/tech_tips/anonymous_ftp_config`

and:

`ftp://ftp.cert.org/pub/tech_tips/anonymous_ftp_abuses`

17.5 Converting files

As with other Linux systems, text on Linux for S/390 is in ASCII. Therefore, there is no need to be concerned with conversion to EBCDIC when communicating with other ASCII-based systems. As usual, you must first issue the `binary` subcommand before moving files that do not contain text.

When FTPing to other S/390 operating systems such as OS/390 or VM, data conversion follows the same rules as when communicating with any other ASCII system.

17.6 Administrative tools

There are some commands and configuration files associated with the FTP server. The commands are:

<code>in.ftpd</code>	The FTP server
<code>ftpwho</code>	Show current process information for each FTP user
<code>ftpcount</code>	Show the current number of FTP users
<code>ftpshut</code>	Close down the FTP servers at a given time

There are five initialization files in the `/etc/` directory associated with FTP server administration:

<code>ftpassess</code>	Configure the operation of <code>in.ftpd</code>
<code>ftpusers</code>	The list of users who <i>cannot</i> use FTP to this server
<code>ftpgroup</code>	The list of groups who <i>cannot</i> use FTP to this server
<code>ftphosts</code>	The individual user host access file
<code>ftpconversions</code>	The FTP conversions database

Each command and configuration file is documented in the following sections.

17.6.1 The `tcpd` command

The FTP server process, `in.ftpd`, is typically invoked via `inetd` (see 14.6.1, “Overview of `inetd`” on page 272).

On Linux for S/390, this can be verified by grepping through the `inetd.conf` file:

```
[root@linmike mndd]# grep ^ftp /etc/inetd.conf
ftp      stream  tcp      nowait  root    /usr/sbin/tcpd  in.ftpd -l -a
```

Here we see that the executable that is spawned to service incoming FTP requests is actually `tcpd`. This is an access control facility for Internet services. Think of it as a front door for Internet servers. The head of the `tcpd` man page describes it as follows:

```
The tcpd program can be set up to monitor incoming requests for telnet,
finger, ftp, exec, rsh, rlogin, tftp, talk, comsat and other services
that have a one-to-one mapping onto executable files.
```

```
The program supports both 4.3BSD-style sockets and System V.4-style
TLI. Functionality may be limited when the protocol underneath TLI is
not an internet protocol.
```

```
Operation is as follows: whenever a request for service arrives, the
inetd daemon is tricked into running the tcpd program instead of the
desired server. tcpd logs the request and does some additional checks.
When all is well, tcpd runs the appropriate server program and goes
away.
```

```
Optional features are: pattern-based access control, client username
lookups with the RFC 931 etc. protocol, protection against hosts that
pretend to have someone elses host name, and protection against hosts
that pretend to have someone elses network address.
```

17.6.2 The FTP daemon

The FTP daemon is invoked, as needed, by `inetd`. For a brief description of `inetd`, see 14.6.1, “Overview of `inetd`” on page 272.

The flags with which the FTP daemon is invoked can be modified by the system administrator. By default, `in.ftpd` is invoked with the `-a` and `-l` flags from settings in the `inetd.conf` file. The `-l` option specifies that each FTP session is logged in the `syslog`. Following is an example of a logged FTP session, in this case from the file `/var/log/messages`:

```
May 22 17:41:30 linux390 ftpd[616]: ANONYMOUS FTP LOGIN FROM 9.12.0.115
[9.12.0.115], mike@foo.com
```

The `-d` or `-v` options specify that debugging information is written to `syslog`. If the `-i` option is specified, files *received by* `in.ftpd` will be logged to the `xferlog`. If the `-o` option is specified, files *sent by* `ftpd` will be logged to the `xferlog`. The FTP server will timeout an inactive session after 15 minutes. If the `-t` option is

specified, the inactivity timeout period will be set to timeout seconds. A client may also request a different timeout period; the maximum period allowed may be set to timeout seconds with the -T option. The default limit is 2 hours.

Authentication of users is performed by ftpd as follows (from the ftpd man page):

1. The user name must be in the password data base, /etc/passwd, or whatever is appropriate for the operating system, and the password must not be null. In this case a password must be provided by the client before any file operations may be performed.
2. The user name must not appear in the file /etc/ftpusers.
3. The user must have a standard shell returned by getusershell(3).
4. If the user name is ``anonymous'' or ``ftp'', an anonymous ftp account must be present in the password file (user ``ftp''). In this case the user is allowed to log in by specifying any password (by convention this is given as the client host's name).

17.6.3 The ftpaccess file

The ftpaccess file is the main configuration file of the FTP daemon; for more information, you can refer to the ftpaccess man page (which is quite lengthy). Apparently, this file is somewhat complex and at times limited in how the ftp daemon can be configured. One append to a news group on this topic suggested using ProFTPD, because its ftpaccess file is much more similar to the Apache Web server's configuration file. See 17.8, "A different FTP server - ProFTPD" on page 345.

The default ftpaccess file shipped with S/390 Linux is as follows:

```
[root@itsolinux1 /etc]# cat ftpaccess
class all real,guest,anonymous *

email root@localhost

loginfails 5

readme README* login
readme README* cwd=*

message /welcome.msg login
message .message cwd=*

compress yes all
tar yes all
chmod no guest,anonymous
delete no guest,anonymous
```

```
overwrite      no          guest,anonymous
rename         no          guest,anonymous
```

```
log transfers  anonymous,real inbound,outbound
```

```
shutdown /etc/shutmsg
```

```
passwd-check  rfc822 warn
```

To test this file, it was modified in order to change the message file from the root directory to the /etc directory with a more useful file name. The line:

```
message /welcome.msg      login
```

was replaced with:

```
message /etc/ftp.msg      login
```

After this change, two files were created: /etc/ftp.msg and /home/ftp/etc/ftp.msg, to which appropriate messages were added. Pointing a Web browser to this FTP server then shows the greeting shown in Figure 89.

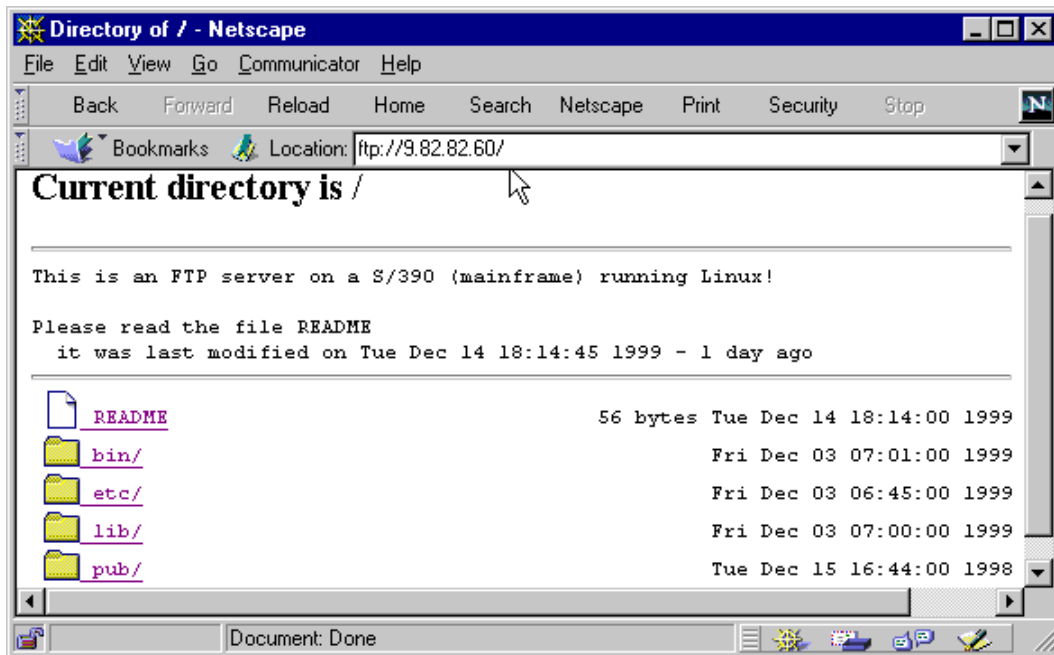


Figure 89. Browser's view of Anonymous FTP server

17.6.4 The ftpusers file

The users listed in the file `/etc/ftpusers` *cannot* be authenticated to the FTP server. By default, the following users are included in the S/390 Linux ftpusers file, and therefore excluded from logging in to the ftp server.

```
[root@linmike /etc]# cat ftpusers
bin
daemon
adm
lp
sync
shutdown
halt
mail
news
uucp
operator
games
nobody
```

17.6.5 The ftpgroups file

When accessing the ftp server from a client, the user can issue the `site group` and `site gpass` subcommands. Doing so will specify an enhanced access group and associated password. If the credentials are valid, the user becomes a member of the group specified in the group access file `/etc/ftpgroups`. The format of the group access file is as follows:

```
access_group_name:encrypted_password:real_group_name
```

where `access_group_name` is a string, `encrypted_password` is the password encrypted via the `crypt` command (exactly like in `/etc/passwd`), and `real_group_name` is the name of a valid group listed in `/etc/group`.

17.6.6 The ftphosts file

The `ftphosts` file is used to allow or deny access to certain accounts for various users or from various hosts. The description in the man page is as follows:

Access Capabilities

```
allow <username> <addrglob> [<addrglob> ...]
```

Only allow host(s) matching `<addrglob>` to log in as `<username>`.

```
deny <username> <addrglob> [<addrglob> ...]
```

Always deny host(s) matching `<addrglob>` to log in as `<username>`.

A username of `anonymous` or `ftp` specifies the anonymous user.

The <addrglob> may be also be specified as address/cidr or address:netmask.
For example: 10.0.0.0/8 or 10.0.0.0:255.0.0.0

As an example, you may want to deny the user mikem to log in. If you add the following line to the ftphosts :

```
deny 9.12.0.115
```

Then, when any user from the machine with an IP address of 9.12.0.115 tries to login, he simply gets denied with no clear error message issued.

```
Connected to linux390.itso.ibm.com.
220 linux390.itso.ibm.com FTP server (Version wu-2.4.2-VR17(1) Thu May
18 03:18:
13 EDT 2000) ready.
User (linux390.itso.ibm.com:(none)): mikem
331 Password required for mikem.
Password:
530 Login incorrect.
Login failed.
```

17.6.7 The ftpconversions file

The ftpconversions file describes how the ftp server should process files; it is used in conjunction with the ftpaccess file. Usually it describes how to decompress files.

The following is taken from the ftpconversions man page:

```
The conversions known by ftpd(8) and their attributes are stored in an
ASCII file that is structured as below. Each line in the file provides
a description for a single conversion. Fields are separated by colons
(:).
%s:%s:%s:%s:%s:%s:%s:%s
1 2 3 4 5 6 7 8
Field      Description
1          strip prefix
2          strip postfix
3          addon prefix
4          addon postfix
5          external command
6          types
7          options
8          description
```

Though the default ftpconversions file shipped with the large file system is similar, one appender to a news group suggests using the following ftpconversions file:

```
:.Z: : :/bin/compress -d -c %s:T_REG|T_ASCII:O_UNCOMPRESS:UNCOMPRESS
:-z: : :/bin/compress -d -c %s:T_REG|T_ASCII:O_UNCOMPRESS:UNCOMPRESS
.gz: : :/bin/gzip -cd %s:T_REG|T_ASCII:O_UNCOMPRESS:GZIP
.z: : :/bin/gzip -cd %s:T_REG|T_ASCII:O_UNCOMPRESS:GZIP
: : :-lst:/bin/ZIP-LST %s:T_REG|T_ASCII:O_UNCOMPRESS:UNCOMPRESS
: : :.Z:/bin/compress -c %s:T_REG:O_COMPRESS:COMPRESS
: : :.gz:/bin/gzip -c %s:T_REG:O_COMPRESS:GZIP
: : :.z:/bin/gzip -c %s:T_REG:O_COMPRESS:GZIP
: : :.tar:/bin/tar -c -f - %s:T_REG|T_DIR:O_TAR:TAR
: : :.tar.Z:/bin/tar -c -Z -f
      - %s:T_REG|T_DIR:O_COMPRESS|O_TAR:TAR+COMPRESS
: : :.tar.z:/bin/tar -c -z -f - %s:T_REG|T_DIR:O_COMPRESS|O_TAR:TAR+GZIP
: : :.tar.gz:/bin/tar -c -z -f
      - %s:T_REG|T_DIR:O_COMPRESS|O_TAR:TAR+GZIP
: : :.zoo:/bin/ZOO %s:T_REG|T_DIR:O_COMPRESS|O_TAR:ZOO
: : :.zip:/bin/ZIP %s:T_REG|T_DIR:O_COMPRESS|O_TAR:ZIP
```

17.6.8 The ftpcount command

The ftpcount command shows the current number of FTP users. Following is the man page synopsis:

Syntax

```
ftpcount
```

Description

The command shows the current number of users (and the limit) for each class defined in the ftpaccess file.

The following example shows that two users are currently connected and that there is no maximum number of users.

```
[root@linux390 /etc]# ftpcount
Service class all - 2 users (no maximum)
```

17.6.9 The ftpshut command

The ftpshut command is used to close down the FTP server at a given time. Following is the man page synopsis:

Syntax

```
ftpshut [ -l min] [ -d min] time [ warning-message ... ]
```

Description

The ftpshut command provides an automated shutdown procedure that a superuser can use to notify ftp users when the ftp server is shutting down.

Following is an example:

```
[root@linux390 /]# ftpshut now
```

If clients are connected to the FTP server when it is shut down, they will just see the following message the next time they try to perform any operation:

```
221-System shutdown at Wed May 24 15:41:00 2000
221 Server shutting down. Goodbye.
```

Therefore, you may want to check if any users are connected first via the `ftpcount` command.

Once this command is run, the file `/etc/shutmsg` is created (file name is set in the file `/etc/ftppass` on the line beginning with 'shutdown'). The file `/etc/shutdown` is queried by the FTP server to see if it is shut down; if found, it will continue to refuse connections until this file is removed. When the FTP server is shut down and you try to connect from an FTP client, you will see the error:

```
500 linux390.itso.ibm.com FTP server shut down -- please try again later.
```

When you remove the file, FTP clients can connect as usual.

17.6.10 The `ftpwho` command

The `ftpwho` command shows current process information for each FTP user. Following is an example:

```
[root@linux390 /]# ftpwho
Service class all:
  1 ?      S      0:00 init [3]
  1 ?      S      0:00 init [3]
- 2 users (no maximum)
```

17.7 Client notes

Traditionally, the FTP client uses a command line interface. On Windows desktops, there are now many GUI-based clients that give a graphical view of the files on the FTP server. We will focus on the traditional command line client. Once you are in an FTP session, there are many subcommands that can be issued. The more common commands are as follows:

<code>open/user</code>	Subcommands for establishing and authenticating FTP sessions
<code>cd</code>	The subcommand for traversing file systems
<code>mkdir/rmdir/rename/del</code>	Subcommands for manipulating the file system

bin/ascii	Subcommands for toggling between text and binary transfer type
get/put	Subcommands for moving a single file
mget/mput/prompt	Subcommands for moving sets of files
site	The subcommand to send a specific instruction to the server
hash/tick/beep	Subcommands for giving more feedback of transfer status
bye/quit	Subcommands for ending FTP session

The FTP client does extra processing if a file named `.netrc` exists in the user's home directory. The `.netrc` file contains login and initialization information used by the auto-login process. The following tokens are recognized:

machine name	Identify a remote machine name.
default	This is the same as machine name except that default matches any name. There can be only one default token, and it must be after all machine tokens.
login name	Identify a user on the remote machine.
password string	Supply a password. If this token is present, the auto-login process will supply the specified string if the remote server requires a password as part of the login process.

Following is an example of a short `.netrc` file that is useful if the systems to which you FTP share the same user IDs and passwords. When this file exists in your home directory, the FTP client automatically sends the user ID and password.

```
[mikem@itsolinux1 mikem]$ cat .netrc
default login mikem password nunyablz
```

17.8 A different FTP server - ProFTPD

The FTP server included with the big file system is `wu-ftpd` (wu for Winconsin University). There is a feeling among some Linux users that ProFTPD is a better choice. If you plan to seriously use a Linux for S/390 anonymous FTP server, using the ProFTPD code might be worth the extra effort.

A description of this FTP server can be found on the ProFTPD home page at:

<http://www.proftpd.net/>

17.8.1 Obtaining ProFTPD

The latest ProFTPD release at the time of writing was 1.2.0 pre 10. It can be found on the Web starting at:

<http://www.proftpd.net/download.html>

However, it is easier to get rpms than to build packages from scratch. One was found in the file `proftpd-1.2.0pre10-2.s390.rpm` on the Thinking Object's Web site at:

<http://linux.s390.org/download/ftp/RPMS/s390/>

Also, a free user's guide by Mark Lowes was found on the Web at:

http://hamster.wibble.org/proftpd/proftpd_userguide.html

To install ProFTPD, the current FTP server should be stopped via the `ftpshtut` command. Then the existing FTP daemon must be deleted via the `rpm --erase` command. Finally the `proftpd` rpm can be installed; for example:

```
[root@linux390 ProFTPd]# /usr/sbin/ftpshtut now
[root@linux390 ProFTPd]# rpm --erase anonftp-2.8-2
[root@linux390 ProFTPd]# rpm --erase wu-ftp
[root@linux390 ProFTPd]# rpm --install proftpd-1_2_0pre10-2_s390.rpm
error reading information on service proftpd: No such file or directory
/var/tmp/rpm-tmp.33909: /etc/rc.d/init.d/proftpd: No such file or directory
execution of script failed
```

This install script did not run perfectly, however, the binaries and configuration file (`/etc/proftpd.conf`) were created. Therefore, we were able to proceed. Also, the `in.ftpd` line in `/etc/inetd.conf` was successfully commented out by the install.

While the `wu-ftp` daemon is typically started via `inetd`, the ProFTPD daemon is typically run *standalone*. That means the `ftp` daemon is always running and waiting for incoming connections on port 21. Perhaps the only important error is due to the lack of a start script which varies among distributions. For the purpose of installing ProFTPD, we modified an existing script, and created the file `/etc/rc.d/init.d/proftpd` as follows:

```
#!/bin/sh
#
# chkconfig: - 55 55
# description: The proftpd daemon gets started here - it does NOT start
from inetd

# Source function library.
. /etc/rc.d/init.d/functions
```

```

# Get config.
. /etc/sysconfig/network

# Check that networking is up.
if [ ${NETWORKING} = "no" ]
then
    exit 0
fi

# See how we were called.
case "$1" in
start)
    echo -n "Starting proftpd"
    case $SILENT in true|yes) silent=-q ;; *) silent= ;; esac
    daemon proftpd $silent
    ;;
stop)
    echo -n "Stopping proftpd services: "
    killproc proftpd
    ;;
status)
    status proftpd
;;
restart|reload)
    $0 stop
    $0 start
    ;;
*)
    echo "Usage: proftpd {start|stop|status|restart|reload}"
    exit 1
esac

exit 0

```

We then made symbolic links from `/etc/rc.d/rc3.d/K55proftpd` and `S55proftpd` to `/etc/rc.d/init.d/proftpd` so the daemon will be started and stopped at run level 3. ProFTPd successfully came up on the next reboot, and anonymous FTP services were also available.

Chapter 18. Domain Name Service (DNS)

Domain Name Service (DNS) is a service that you can use to translate domain names into IP addresses within the network.

18.1 Introduction to DNS

We will begin with a discussion of DNS solutions on S/390. We will then provide a very basic description about how DNS works and describe the configuration files involved. We will conclude with a guided tour through a simple caching of DNS.

We would like to note here that DNS concepts are complicated, and the exact implementation is beyond the scope of this chapter. We intend only to get you started; therefore we will be covering only the simple concepts involved and trying to provide you with a working knowledge of DNS. For complete coverage of DNS we refer you to the book *DNS and BIND* and to the Linux man pages.

18.1.1 Assumptions

We are assuming that you have successfully installed the Marist Linux big file system, either on an LPAR or in a Virtual Machine. We also assume that your Linux is configured so that you can `telnet` to and from your machine and that you can connect to the internet (i.e. routing is set up correctly).

Further, you should have the following files on your Linux machine:

- `/etc/nsswitch.conf`
- `/etc/host.conf`
- `/etc/resolv.conf`
- `/etc/hosts`

Finally, we assume that you are able to issue `telnet 127.0.0.1` from your Linux console to log into your own machine.

18.1.2 Skills

You will need to be able to edit a text file, use `telnet`, and have a basic understanding of how to navigate through the Linux file system.

18.2 How it works, in theory

The domain name system is a global network of servers that can translate host names like `www.marist.edu` into their numerical IP (Internet Protocol) addresses, in this case `148.100.1.21`. This system is a distributed database, providing data for name to IP address resolution at different levels within the name space and providing access to the data across the entire network through a client-server scheme. Performance is achieved through replication and caching at various levels in the domain name space.

Name servers contain information about their domains and about where other name servers can be located. They make this information available to their clients, called *resolvers*. A resolver can be almost any application or machine that needs to access a name on the net, therefore it will be trying to *resolve* a domain name to an IP address. There are three possible solutions you can use when implementing DNS:

1. You implement a local name server that acts as a *primary* name server.

A primary name server maintains all data locally. This name server has authority over the name data belonging to the zone for which it is acting. Secondary name servers in the network request transfers from the primary name server.

2. You configure a local name server that acts as a *secondary* name server.

This configuration performs well for all queries. A secondary name server requests a transfer of all name data from a remote name server during initialization and stores the data in its local database. Then it responds to local queries. Once a secondary name server receives the zone data, it has authority for that zone.

At regular intervals, it will request a new transfer, refreshing the local copy of the remote name server data based on the value defined in the Start of Authority (SOA) record for the zone.

3. You configure a local name server that acts as a *caching-only* name server. This configuration performs well for frequently used queries. A caching-only name server maintains a local cache with the most recent responses to queries. If the name server is able to serve a query from the cache, it will do so. If not, the name server will forward the query to another name server and cache the response on return.

The first thing you need to understand is the domain name space. This is the structure of the DNS databases just discussed and it is how the different domains on the Internet are represented. It is partially illustrated in Figure 90 on page 351.

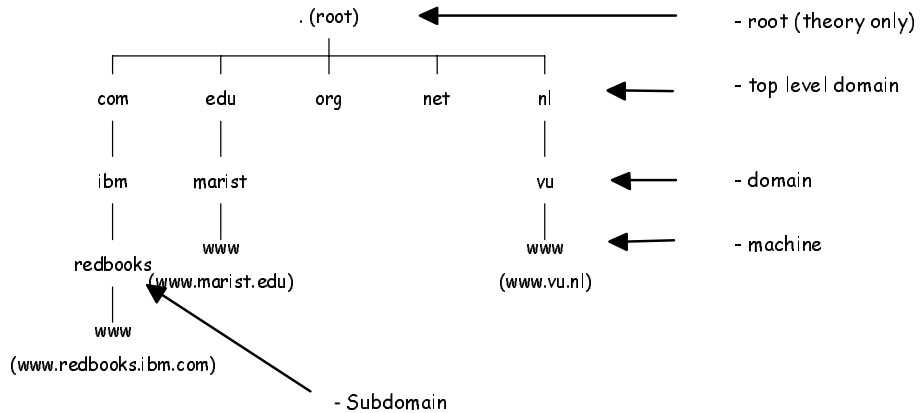


Figure 90. Internet domain name space (partial representation)

The organization of the DNS database is represented by an inverted tree, with the root at the top. This root is only a theoretical domain level and does not represent a physical machine or location. In DNS, the root's name is the null label and is written as a single dot "." in text. If you wanted to represent the root node in a domain name then you would have a name such as `www.ibm.com.` with the root as the final dot. It is customary to leave the root out of domain names. Each node in the database represents a partition, or a *domain* in the domain name system. Everything below a node falls into that domain such that one domain can be part of another domain. For example, the machine `.redbooks` is part of the `.ibm` domain as well as the `.com` domain.

18.2.1 In action

A DNS server is a computer running DNS software. As most servers are UNIX or UNIX variants, the most popular program is BIND (Berkeley Internet Name Domain). Linux for S/390 has this available in the form of two packages that can be downloaded from the S/390 rpm database, (see Appendix G.4, "Referenced Web sites" on page 507). We used these on our Linux for S/390 under VM/ESA and they were installed as described in 18.4, "Hardware and software setup" on page 353.

When you enter a fully-qualified domain name such as `www.vu.nl`, your client machine sends a request to a known name server. This is, in most cases, the name server that was entered into the configuration file during the Linux network setup.

If the queried name server has ever fielded such a request (within a set time limit, to prevent a server from passing old information) for this host name, it will find that information in its cache and return it to the requesting client. When a name server can answer a query without asking another name server, it's known as an *authoritative server*.

If the name server cannot resolve the domain name itself, then it will attempt to *solve* the problem by using one of several methods. A description of these methods is outside the scope of this chapter and we refer you to the book *DNS and BIND*.

Once the information is located, it's passed back to the requesting client machine and you can continue on your way. This usually occurs very quickly, but occasionally it can take much longer and your client machine will time-out, saying that the domain name does not exist (but you know that it does!). This can be caused by heavy network traffic somewhere and the name servers may still be busy trying to get an answer back to your client machine. In such a case, if you try again it will most likely work because the information will then have had enough time to get back to your requesting client machine.

18.3 DNS solutions on S/390

Linux for S/390 provides an alternative environment for a DNS server on S/390. If you implement a primary or secondary DNS server on OS/390 or VM/ESA, the data repository is a DB2 database. On Linux for S/390, it is a flat file.

18.3.1 Using OS/390

With OS/390 you may use the Domain Name System in one of four different ways (these are covered in more detail in the next section):

1. You implement a local name server that acts as a *primary* name server, which stores the zone data in DB2 tables.
2. You configure a local name server that acts as a *secondary* name server, which also stores the zone data in DB2 tables.
3. You configure a local name server that acts as a *caching-only* name server.
4. You do not configure a name server on your MVS system, but point to another host in the network that runs a name server. This is done by placing an NSINTERADDR statement in the hlq.TCPIP.DATA data set, with an IP address of the remote host running the name server.

18.3.1.1 Using VM/ESA

With VM/ESA, you have a similar set of options. The TCP/IP domain name server can be configured to run as a primary, secondary, or caching-only server. Both primary and secondary name servers store the zone data in DB2 tables. The caching-only name server uses a CMS file to define the host names and internet addresses for the remote name servers.

18.3.1.2 Using VSE/ESA

VSE/ESA does not support a domain name server.

18.4 Hardware and software setup

We ran the Marist Linux big file system installed on S/390 under VM. We accessed Linux through a telnet session from a SuSE 6.4 distribution running on a desktop PC.

18.4.1 Software not included

The *bind8* (*Berkeley Internet Name Domain version 8*) packages are not a part of the Marist Linux big file system and need to be downloaded from the S/390 rpm database (see Appendix G.4, “Referenced Web sites” on page 507). The first step is to obtain the two packages and place them together in one directory. We did this while logged in as `root` by typing the following commands:

```
$ cd
$ mkdir bind8
```

This creates a directory in the root directory called *bind8*, into which we placed the two downloaded packages:

```
bind-8.2.2_P3-1.s390.rpm
bind-utils-8.2.2_P3-1.s390.rpm
```

We typed the following commands to install the *bind8* packages:

```
$ rpm -ivh bind-8.2.2_P3-1.s390.rpm
$ rpm -ivh bind-utils-8.2.2_P3-1.s390.rpm
```

This command installs the *bind8* components and documentation as listed in Table 28.

Table 28. Executables and documentation installed with *bind8*

Directory	File
/usr/sbin/	dnskeygen
	irpd
	named
	named-bootconf
	named-xfer
	ndc
/usr/bin	dig
	dnsquery
	host
	mkserfdb
	nslookup
	nsupdate
/usr/doc/bind-8.2.2_P3	documentation

We removed the two Bind rpms by typing the following command:

```
$ rmdir -rf ~/bind8
```

Now we were ready to begin the configuration of our caching name server.

18.5 Caching-only name server

We begin by working with a caching-only name server, which will find the answers to name queries and then remember the answers. This means that the next time they are needed, the response time will be significantly better since the DNS caching server you set up will not have to send the request out onto the network.

The first thing we will explain is the concept of using *forwarding* to allow domain name lookups from within a protected network (i.e. from behind a firewall). We demonstrate this since it is the first step on your way into the

complex world of DNS servers. We will also show you the various configuration files that you need and give you a basic idea of how they work.

Before expanding on this caching-only name server, we recommend that you become thoroughly familiar with DNS as it is complex and extremely important to the domains it supports.

18.5.1 Forwarding out of a protected network

The first thing that you will encounter is the problem of not being connected directly to the network, as most companies employ some sort of firewall concept. This means you will be unable to query the DNS servers outside your protected network directly: you will have to forward your requests through a DNS server that is speaking to the outside world for you.

If you are connecting directly to the network (for example, if you are setting up your DNS server to be just outside of the protected network) then you should use a *hints* file, which is explained in the following section. When the query cannot be resolved locally, this provides the name daemon with a starting point by listing the addresses of the root-level domain name servers. How and where to get the most up-to-date *hints* file, in case you want to do this sort of setup, is also explained later on.

18.5.2 Configuration files

We start with the creation of a few configuration files so that *named*, which is BIND's daemon program, knows what it should do and in which directories it can find things. Unless stated otherwise, all mentioned configuration files can be located in the */etc/* directory.

18.5.2.1 Named.conf

The first file is *named.conf*, which you need to create in the directory */etc*. You do this by opening your favorite Linux editor and creating a file that looks like "" on page 356.

```

// Config file for caching only name server Linux on
S/390!

options {
    directory "/var/named";
    // This is a forwarding option to allow for the
fact
    // that you are behind a firewall, thus needing
to
    // get your lookup done by the DNS server that
deals
    // with the outside world for your network.

    forward first;
    forwarders {
        9.9.9.8; // This is a fake address.
        9.9.9.9; // Backup (secondary) fake address.
    };
};

zone "." {
    type hint;
    file "root.hints";
};

```

Figure 91. The *named.conf* file

These are the basics you need to start your name daemon. The line `directory` tells `named` where to look for files. For example, when the `named.root` file is mentioned, `named` will look for it in `/var/named/named.root` directory. It does not matter where you put this directory, as long as you have enough space.

The commented lines explain the `forward` option that is listed under it (you fill in the IP address or addresses of DNS servers you use from within your protected network; fake addresses are listed in this example). The only option you will need right now is `forward`, so that you can point to a DNS server or servers that are outside of you firewall. You can add many more options as you become more familiar with the DNS concepts, but these fall outside of the scope of this chapter.

Root.hints

The `root.hints` file is a configuration file containing the locations of all top-level domain servers. It has to be obtained from the ftp server at

RS.INTERNIC.NET, the maintainers of this information. As mentioned earlier, this will only be important to you if you are not using the forward option, and therefore are connecting to the network directly.

The procedure for this is as follows:

1. Obtain the wget package (wget-1.5.3-5.s390.rpm at this time) from a S/390 rpm database. This is a file retrieval utility that allows you to obtain the named.root file from an ftp server using your Linux for S/390 machine.
2. Install it as user root with `rpm -ivh wget-1.5.3-5.s390.rpm`
3. Enter the following (with your proxy machine's address in place of proxy.company.tld):

```
export ftp_proxy=http://proxy.company.tld:8080/  
wget ftp://rs.internic.net/domain/named.root
```

This will get you the named.root file, which describes the root name servers in the world, in the directory where you executed the above commands. Move this to the /var/named/ directory and rename it to root.hints with the command:

```
mv named.root /var/named/root.hints
```

Now we are ready to move on to the next configuration file.

18.5.2.2 Zone file for 127.0.0

The last section of our named.conf file contains a zone file for 0.0.127.in-addr.arpa, which defines our zone (the one for which we are the cached name server). It says that we are the master server for it and that it is stored in a file called /var/named/db/127.0.0.

You will also create this file in the following steps:

```
cd /var/named/  
mkdir db
```

Now, in your favorite Linux for S/390 editor, create a file containing the data shown in Figure 92 on page 358.

This file contains three Resource Records: a Start of Authority (SOA), a Name Server (NS), and a Pointer (PTR). The SOA line starts with an @, which is the short notation of the *origin*, which in this case is the domain column for the file 127.0.0.in-addr.arpa.

The SOA record is the preamble to all zone files, and there should be exactly one in each zone file. It describes the zone and where it comes from (a

machine called localhost). Next it describes who is responsible for its contents in the form of an e-mail address with a . in place of @, in this case `root@localhost` (you could place your system administrator's e-mail address here). Finally, it describes what version of the zone file this is. In this example we chose the form `YYYYMMDD##`, giving you up to 100 versions per day. For the rest of the fields (refresh, retry, expire, minimum), refer to the man page.

In the second line there is no @, since it is implicit from the first line. This line tells DNS which machine is the name server of the domain `0.0.127.in-addr.arpa`, in this case the localhost.

Finally, the PTR record says that the host at address 1 in the subnet `0.0.127.in-addr.arpa` (aka `127.0.0.1`) is named localhost.

```
@      SOA      localhost. root.localhost. (
                2000051501 ; serial
                3H      ; refresh
                15M     ; retry
                1W      ; expire
                1D )    ; minimum

      NS      localhost.
1     PTR     localhost.
```

Figure 92. The `127.0.0` file

18.5.2.3 `Resolv.conf`

The next configuration you need is the file `resolv.conf`, which should already exist since it was a part of the standard installation. You should see at least two lines, for example:

```
search subdomain.your-domain.com your-domain.com
nameserver 127.0.0.1
```

(Replace *subdomain* and *your-domain* with your own real domain names.)

The *search* line specifies what domains should be searched for any host names you want to connect to (for example, should you be in the Netherlands, then you would have `subdomain.your-domain.nl your-domain.nl`). The *nameserver* line specifies the address of your nameserver, in this case your own machine since that is where your `named` runs. Should you want to list other name servers, add a *nameserver* line for each one. Note that `named` never reads this file; only the resolver that uses `named` reads it.

The next step depends on your libc version, which determines whether you will be adjusting `nsswitch.conf` or `host.conf`. To avoid complications we will just do both.

18.5.2.4 Nsswitch.conf

The `nsswitch.conf` file is a long one, and you just need to be sure that there is a line containing the following:

```
hosts: files dns
```

It might include other things on the same line, but it must have `files` and `dns` mixed in there somewhere. This line says that programs should first look in the `hosts` file, then check DNS according to the `resolv.conf`.

18.5.2.5 Host.conf

The `host.conf` file must have the following line:

```
order hosts,bind
```

If this line is missing, you should add it. This line tells the host name resolving routines to first look in `hosts`, and then ask the name server (which in `resolv.conf` you said is at `127.0.0.1`).

18.5.3 Starting named

Now that everything is configured, the first step is to stop the name daemon if it is running and then restart it; otherwise, you just start it. To reset the name daemon, which will put the changes you made into effect, type the following command:

```
$ /usr/sbin/ndc stop
```

This gives you the following output:

```
Shutdown initiated.
```

Otherwise it was not running and will give you an error message. The next step is to start it up again with the following:

```
$ ndc start
```

This gives you a message telling you the new process ID number (pid). You will be able to verify that all has gone well by checking the system message log with the following command and output:

```
$ tail -30 /var/log/messages
```

In the output look for the following line:

May 19 00:33:50 linux1 named[1213]: Ready to answer queries.

It verifies that you are now ready to give your caching name server a try!

18.5.4 Testing with nslookup

To see if this all works we use a tool called `nslookup`, which gives a fairly clear view with its answers to your queries. We will do this using the tool's *interactive mode*, which means we will enter each command by hand, one at a time. To start with, we will start `nslookup` and examine our work:

```
$ nslookup
Default Server: localhost
Address: 127.0.0.1

>
```

This is what you get if it is working correctly—if not, go back and check everything we covered up to this point (no typos?). Remember that each time you change `named.conf` you will have to restart `named` using the command:

```
$ ndc restart
```

If all went well, we need to try out a query such as the IBM Web site:

```
> www.ibm.com
Server: localhost
Address: 127.0.0.1

Name: www.ibm.com
Address: 198.133.16.99
```

`nslookup` asked your `named` to look for the machine called `www.ibm.com`, then used a machine from your `root.hints` file and asked for `www.ibm.com` from that machine. If things take a while, it could mean, for example, that it may have to search through all the domains you listed in your `resolv.conf` before finding `www.ibm.com`.

Something worth noting is that if you ask for the same lookup again, `www.ibm.com`, you will get a somewhat different answer:

```
> www.ibm.com
Server: localhost
Address: 127.0.0.1

Non-authoritative answer:
Name: www.ibm.com
Address: 198.133.16.99
```

If you see the answer is *non-authoritative*, this means that the answer is coming from your own cache! It did not go out onto the network to find the answer, but retrieved it from its own cache. This change to non-authoritative is meant only to serve as a warning that you are dealing with a cached answer and that there is a (very slight) possibility that the answer is stale (not up-to-date).

When `nslookup` gives this answer, you know that your DNS caching-only name server is working. You can exit `nslookup` by typing the command `exit`. Congratulations and take a break, you deserve it!

18.6 Tools

The various tools that were installed with your BIND packages are explained here. This will only be a brief overview; we refer you to previously mentioned literature for a more detailed discussion, (see Appendix G, “Related publications” on page 505).

The installed tools are:

- `dig`
- `dnsquery`
- `host`
- `nslookup`
- `nsupdate`

18.6.1 dig

Domain Information Groper, or `dig` for short, is a flexible command-line tool that can be used to gather information from Domain Name System servers. It has two modes: simple interactive mode for a single query, and batch mode to execute a list of several query lines. The `dig` tool will send domain name query packets to name servers. The most commonly used command syntax is:

```
dig [@server] domain
```

The `@server` option is used when you are behind the firewall, referring to the DNS server that queries the network for you. The domain is the address you want to look up. For a sample use of `dig` to query the IBM Web site address (where names and addresses have been changed), see Figure 93 on page 362.

```

[root@linux1 /root]# dig @9.9.9.9 www.ibm.com

; <<>> DiG 8.2 <<>> @9.9.9.9 www.ibm.com
; (1 server found)
;; res options: init recurs defnam dnsrch
;; got answer:
;; ->HEADER<<- opcode: QUERY, status: NOERROR, id: 6
;; flags: qr rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 2, ADDITIONAL: 2
;; QUERY SECTION:
;;      www.ibm.com, type = A, class = IN

;; ANSWER SECTION:
www.ibm.com.      27m54s IN A      9.9.9.10
www.ibm.com.      27m54s IN A      9.9.9.11
www.ibm.com.      27m54s IN A      9.9.9.12
www.ibm.com.      27m54s IN A      9.9.9.13

;; AUTHORITY SECTION:
www.ibm.com.      11h42m50s IN NS     some.where.ibm.com
www.ibm.com.      11h42m50s IN NS     some.where2.ibm.com.

;; ADDITIONAL SECTION:
some.where.ibm.com.      11h42m50s IN A      9.9.9.14
some.where2.ibm.com. .  11h42m50s IN A      9.9.9.15

;; Total query time: 10 msec
;; FROM: linux1 to SERVER: 9.9.9.9
;; WHEN: Thu May 18 21:34:31 2000
;; MSG SIZE  sent: 29  rcvd: 171

```

Figure 93. A dig sample

For an overview of the output see the dig HOWTO. For a complete listing of options see the dig man page.

18.6.2 The dnsquery program

The dnsquery program is a general interface to name servers via BIND resolver library calls. This program is intended to be an alternative to programs like nstest, nsquery and nslookup. It should be noted here that nstest and nsquery are not available on the Marist Linux big file system.

```
dnsquery [-n nameserver] host
```

All arguments except for host and nameserver are treated without case-sensitivity. See Figure 94 on page 363 for an example of a query to an IBM Web site address.

```
[root@linux1 /root]# dnsquery -n 9.9.9.9 www.ibm.com

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 14770
;; flags: qr rd ra; QUERY: 1, ANSWER: 6, AUTHORITY: 2, ADDITIONAL: 2
;;      www.ibm.com, type = ANY, class = IN
www.ibm.com.      10h45m28s IN NS   some.where.ibm.com.
www.ibm.com.      10h45m28s IN NS   some.where2.ibm.com.
www.ibm.com.      14m54s IN A     9.9.9.10
www.ibm.com.      14m54s IN A     9.9.9.11
www.ibm.com.      14m54s IN A     9.9.9.12
www.ibm.com.      14m54s IN A     9.9.9.13
www.ibm.com.      10h45m28s IN NS   some.where.ibm.com.
www.ibm.com.      10h45m28s IN NS   some.where2.ibm.com.
some.where.ibm.com. 11h19m27s IN A     9.9.9.14
some.where2.ibm.com. 11h19m27s IN A     9.9.9.15
```

Figure 94. A dnsquery sample

For an overview of the output, see the dnsquery HOWTO. For a complete listing of options, see the dnsquery man page.

18.6.3 The host program

The host program is used to look up host names using a DNS server. By default, it simply converts between host names and IP addresses.

```
host [-t querytype] [-a] host [server]
```

The complete set of options can be found in the host man page, but we explain a few here:

-a is equivalent to verbose, and reports all types of information available.

-t querytype looks for a specific type of information.

The host entered in your command line should be the Internet address of the machine you are trying to translate into an IP address. The server parameter is optional, but if you are behind a firewall it will be necessary to fill this in with the address of your DNS server that can talk to the outside network for you.

See Figure 95 on page 364 for an example of a query to an IBM Web site address.

host example

```
[root@linux1 /root]# host www.ibm.com 9.9.9.9

Using domain server 9.9.9.9:
www.ibm.com has address 9.9.9.10
www.ibm.com has address 9.9.9.11
www.ibm.com has address 9.9.9.12
www.ibm.com has address 9.9.9.13
```

Figure 95. A host example

We suggest trying the above command with your company's own Internet address and DNS server, but by using the `-a` option you will see a bit more information. Once again, for a more detailed analysis we refer you to the `host` man page.

18.6.4 The `nslookup` tool

This tool was already mentioned when we described testing your caching name server using the interactive mode. It is a tool to query Internet domain name servers and has a non-interactive mode of operation. Non-interactive mode is used to print just the name and requested information for a host or domain. The command syntax is as follows:

```
nslookup [-option ...] [host-to-find | name server]
```

As seen earlier, interactive mode is entered in the following cases:

- When no arguments are given, the default name server will be used.
- When the first argument is a hyphen (-) and the second argument is the host name or Internet address of a name server.

Non-interactive mode is used when the name or Internet address of the host to be looked up is given as the first argument. The optional second argument specifies the host name or address of a name server.

Once again, for a more complete discussion of the options and uses of `nslookup`, refer to the man page.

18.6.5 The `nsupdate` command

The `nsupdate` command is used to update Internet domain name servers interactively. It uses the DNS resolver library to pass messages to a DNS server requesting the addition or deletion of DNS resource records. This tool

is not used when keeping a caching-only name server. Therefore, we refer you to the documentation and man page for more information.

18.7 Summary

Domain Name Service (DNS) is a service that you can use to translate domain names into IP addresses within the network. We began with a discussion of DNS solutions on S/390, followed by how DNS works in theory. We then explained a very simple example of DNS at work and want to remind you that we left out a lot of detail on purpose. Next we guided you through installing the necessary packages and listed the parts that would be installed.

Then we took you through a setup to make your own caching name server, which we want to remind you is not a Primary or Secondary DNS server. A Primary or Secondary DNS server setup should only be attempted after you are completely knowledgeable about DNS theory. Finally, we discussed a few of the tools that are available to help you with setting up, using and maintaining your caching-only domain name server.

There is documentation in HTML format in directory `/usr/doc/bind-8.2.2_P3/html/` which gives excellent tips and help.

Chapter 19. Network File System (NFS)

Network File System (NFS) is a means of accessing file systems that reside on remote computers over the net. The program that makes the files on the remote machine available is called the NFS server. The program that accesses files over NFS is referred to as an NFS client. The necessary steps before accessing files via NFS are the installation (and configuration) of an NFS server and on the client side mounting the respective file system across the network. Once this is done the files are accessed just as if they were mounted locally.

19.1 Installation of the server

If you are not using a Linux distribution that supplies the server software including start and stop scripts for the required processes, then you should consult the NFS-HOWTO.

The NFS server consists of several different processes that have to run as daemons (background processes):

- portmap** A process that converts RPC program numbers into DARPA protocol numbers.
- rpc.kmountd** The process that handles the export of NFS filesystems.
- nfsd** The user-level part of the actual NFS server, which delivers data to the clients. Multiple instances of nfsd may be running.
- rpc.kstatd:** Implementation of the Network Status Monitor (NSM) RPC protocol. Used to implement lock recovery when the NFS server machine crashes and restarts.

To check that all of these processes are running, use:

```
ps auxw | grep portmap
```

On the client side the kernel must support NFS, either compiled in or using a kernel module. This functionality is probably already included in the kernel that you are using. If not, configure and compile a kernel that supports NFS.

19.2 Customizing

Linux can act both as an NFS Server and as an NFS Client. We describe the necessary steps to get it running as such. We describe in detail the various scenarios.

19.2.1 NFS Server

Access control for NFS is configured in the file `/etc/exports`. This file consists of lines of the form:

```
/some/dir clientname(option1,option2,...)
```

and comment lines that start with a `#`. Client name may be an IP address (195.27.208.189), or a hostname fully qualified with the domain, or just the hostname itself. Groups of clients can be given as address/netmask pairs or with wildcard characters (`?` or `*`) in hostnames. Among the available options, the most important are:

<code>ro</code>	Only permit read-only access
<code>rw</code>	Permit read and write access
<code>root_squash</code>	Accesses from the client's root are mapped to the anonymous user (nobody by default) on the server

Some examples of valid lines for `/etc/exports` follow.

```
/usr/doc host23(ro)
```

The directory `/usr/doc` can be mounted read-only by `host23`. Multiple clients, each with their own set of options, can be named in one line:

```
/work *.mydomain.com(ro) 195.27.208.189(rw)
```

This makes the directory `/work` accessible read-only for hosts in the domain `mydomain.com` (not including hosts in subdomains like `host.sub.mydomain.com`) and grants read/write access to the host with IP address 195.27.208.189.

```
/usr/man *doc.mydom.com(ro)
```

This line grants read-only access for hosts with names like `doc.mydom.com`, `aadoc.mydom.com`, and `xyzdoc.mydom.com`, but not for `abc.aadoc.mydom.com`.

See man pages `man exports` for further details.

19.2.2 NFS Client

The options available for mounting an NFS file system are explained in the man page `nfs(5)`. The most important are `rsize`, `wsize` and `intr`. The following definitions are from the man page:

<code>rsize=n</code>	The number of bytes NFS uses when reading files from an NFS server. The default value is dependent on the kernel, currently
----------------------	---

	1024 bytes. (However, throughput is improved greatly by asking for <code>rsize=8192</code> .)
<code>wsiz=n</code>	The number of bytes NFS uses when writing files to an NFS server. The default value is dependent on the kernel, currently 1024 bytes. (However, throughput is improved greatly by asking for <code>wsiz=8192</code> .)
<code>intr</code>	If an NFS file operation has a major timeout and it is hard mounted, then allow signals to interrupt the file operation and cause it to return <code>EINTR</code> to the calling program. The default is to not allow file operations to be interrupted.

A typical statement for an NFS mount might look like:

```
mount -t nfs srvname:/expdir/ /mnt/nfs -o intr,rsize=8192,wsiz=8192
```

The corresponding line in `/etc/fstab` would be:

```
srvname:/expdir/ /mnt/nfs nfs intr,rsize=8192,wsiz=8192
```

19.3 Operation

We used the following sequence to *export* directories from Linux for S/390 so they could be accessed from NFS clients on other platforms.

First we define a configuration file `/etc/exports`.

```
[root@linux6 /etc]# cat exports
/etc tot62(rw)
/etc 10.112.34.208(rw)
/home/httpd wtsc52oe.itso.ibm.com(rw)
/home/httpd hdntp(rw)
/etc wtsc47(rw)
/etc erpriscl(rw)
[root@linux6 /etc]#
```

We then used `exportfs -a` to activate these exports.

```
[root@linux6 /etc]# exportfs -av
exporting hdntp:/home/httpd
exporting wtsc52oe.itso.ibm.com:/home/httpd
exporting hdntp:/etc
exporting erpriscl:/etc
exporting wtsc47.itso.ibm.com:/etc
exporting 10.112.34.208:/etc
exporting tot62:/etc
```

These are the settings we used for scenarios where Linux for S/390 runs the NFS server.

19.3.1 As a server for AIX

Our first attempt to access files on Linux for S/390 from AIX was based on the following `exports` list:

```
[root@linux6 /etc]# cat exports
/etc tot62(rw)
/etc 10.112.34.208(rw)
/home/httpd wtsc52oe.itso.ibm.com(rw)
/home/httpd hdmtprw)
/etc wtsc47(rw)
/etc erprisc1(rw)
[root@linux6 /etc]#
```

On the AIX system we used `smit` to mount the remote file system on Linux. The first attempt failed with the following messages:

```
May 18 17:21:25 linux6 mountd[367]: authenticated mount request from
erprisc1:723 for /etc (/etc)
May 18 17:21:25 linux6 kernel: nfsd: request from insecure port
(090c0049:45161) !
```

The reason for this is that AIX used a port below 1024 and Linux assumes this is *insecure* because all ports below 1024 are *well known* and cannot be reused for other IP services.

Therefore, we changed the `exports` file as follows:

```
[root@linux6 /etc]# cat exports
/etc tot62(rw)
/etc 10.112.34.208(rw)
/home/httpd wtsc52oe.itso.ibm.com(rw)
/home/httpd hdmtprw)
/etc wtsc47(rw)
/etc erprisc1(rw,insecure)
```

With this change we could mount the `/etc` directory successfully.

19.3.2 As a server to OS/2

Although OS/2 does not have the kind of client security required for a secure network, we implemented OS/2 as an NFS client of Linux for S/390. Therefore, we needed to have either the old NFS Client from the TCP/IP package or the Hummingbird NFS client.

The description here refers to an OS/2 installation based on OS/2 Warp version 4.0 with the TCP/IP Extended Networking Kit installed.

Aside from local NFS support, you do not need any of the support functions (*mvslogin*, *mvslogout*, *showattr*) provided with OS/390.

The details are shown in 19.3.4, “As a client to OS/390” on page 374. At ❶ in Figure 96 we used the command *showexp*, which gives an overview of what is exported by our NFS server on LINUX6.

```
[F:\]showexp linux6 ❶
export list for linux6:
/etc      erprisc1 wtsc47.itso.ibm.com hdmtip 9.12.14.208 tot62
/home/httpd      wtsc52oe.itso.ibm.com

[F:\]mount z: linux6:/etc ❷
mount: linux6:/etc
UID: 0
GID: 0

NFS Drive 'z:' was attached successfully.

[F:\]dir z:h* ❸

The volume label in drive Z is NFS.
The Volume Serial Number is 0000:0030.
Directory of Z:\

12.07.94  20.28      26      0  host.conf.rpmorig
29.07.95  1.26      347     0  hosts.deny.rpmorig
29.07.95  1.26      161     0  hosts.allow.rpmorig
12.07.94  20.28      26      0  host.conf
29.07.95  1.26      161     0  hosts.allow
29.07.95  1.26      347     0  hosts.deny
20.02.00  17.00     <DIR>    0  httpd
17.05.00  19.22      7       0  HOSTNAME
16.05.00  23.29      70      0  hosts
          9 file(s)      5241 bytes used
                               505970688 bytes free
```

Figure 96. Mounting file system on Linux for S/390 from OS/2 client

With step ❷ we issue the *mount* command. It requests a *uid* and *gid*. The *directory* command at ❸ then displays several files located in the */etc* directory on LINUX6.

19.3.3 As a server for OS/390

Linux as an NFS server for OS/390 can have several interesting uses. Since NFS-mounted file systems on a remote server can be accessed from programs running on the client side, data access can be achieved *without* copying it. On OS/390 it is even possible to access these files from traditional programs written in COBOL or PL/1. This requires that the remote data can be structured with a so-called *dummy dcb*.

Another application could be to use the ADSM client running on OS/390 UNIX System Services to back up data from the Linux server.

Note: this is a circumvention only as long as there is no ADSM client available for Linux for S/390.

The setup on Linux is pretty much the same as in the case for the AIX client. The exports file has to have an entry for an insecure client:

```
root@linux6 /etc]# cat exports
/etc tot62(rw)
/etc 9.12.14.208(rw)
/home/httpd wtsc59oe.itso.ibm.com(rw,insecure)
/home/httpd wtsc52oe.itso.ibm.com(rw,insecure)
/home/httpd hdmtip(rw)
/etc wtsc47(rw)
/etc erpriscl(rw,insecure)
```

These definitions are activated using the `export -a` command. On the OS/390 side we used a 2.9 release, which provides a `mount` command within the shell. The option `-v` gives some further information about the `mount` command's progress.

The steps to connect, use and unmount a file system are shown in Figure 97 on page 373. First the mount has to be done ❶. You then can use the remote file system on Linux ❷.


```

HDM@SC59 /u/hdm:>mount -v -t nfs -o "LINUX6:/home/httpd" -f LINUX6
/mnt/linux6 ❶
FOMF0501I Async mount proceeding for LINUX6
HDM@SC59 /u/hdm:>ls -al /mnt/linux6 ❷
total 272
drwxr-xr-x  5 STC      SYS1      4096 Feb 20 17:00 .
drwxr-xr-x  3 STC      OMVSGRP   8192 Mai 19 16:12 ..
drwxr-xr-x  2 STC      SYS1      4096 Feb 20 08:31 cgi-bin
drwxr-xr-x  3 STC      SYS1      4096 Feb 20 17:00 html
drwxr-xr-x  3 STC      SYS1      4096 Feb 20 17:00 icons
HDM@SC59 /u/hdm:>showmount linux6 ❸
wtsc52oe.itso.ibm.com
wtsc59oe.itso.ibm.com
9.12.0.73
HDM@SC59 /u/hdm:>umount -o immediate /mnt/linux6 ❹
HDM@SC59 /u/hdm:>showmount linux6 ❺
wtsc52oe.itso.ibm.com
9.12.0.73
HDM@SC59 /u/hdm:>

```

Figure 97. Mount/unmount scenario Linux as a NFS Server for OS/390

Steps ❸, ❹, and ❺ show the status using `showmount` and an `umount` of the remote file system.

The following should be considered if you have problems getting the `mount` done:

- The active NFS client log data set may contain further information. You have to flush the in-storage buffers first using the command `nfstat -d f` from the OS/390 UNIX System Services shell.
- Further, to use `mount` you need either superuser authority or READ access to the profile SUPERUSER.FILESYS.MOUNT resource found in the UNIXPRIV.

Now, since we had the files on Linux6 mounted, we tried to do an incremental backup. The backup is initiated with a current working directory of `/mnt/linux6/html/manual`. The command we issued was `inc -subdir=yes *.*`. The excerpt from the log is shown in Figure 98 on page 374.

```

Normal File--> 2,649
/mnt/linux6/html/manual/vhosts/index.html
[Sent]
Normal File--> 5,877
/mnt/linux6/html/manual/vhosts/ip-based.html
[Sent]
Normal File--> 14,831
/mnt/linux6/html/manual/vhosts/mass.html
[Sent]
Normal File--> 6,706
/mnt/linux6/html/manual/vhosts/name-based.html
[Sent]
Normal File--> 16,087
/mnt/linux6/html/manual/vhosts/vhosts-in-depth.
html [Sent]
Normal File--> 8,271 /mnt/linux6/html/manual/vhosts/virtual-host.html
[Sent]
Successful incremental backup of '*.*'

Total number of objects inspected:      135
Total number of objects backed up:      134
Total number of objects updated:        0
Total number of objects rebound:        0
Total number of objects deleted:        0
Total number of objects failed:         0
Total number of bytes transferred:      1.29 MB
Data transfer time:                     0.47 sec
Network data transfer rate:             2,754.18 KB/sec
Aggregate data transfer rate:           14.60 KB/sec
Objects compressed by:                  0%
Elapsed processing time:                 00:01:30
dsmc>
===>

```

Figure 98. Incremental backup of files on Linux from OS/390

This may be a way to back up files on Linux as long as there is no native ADSM client available.

19.3.4 As a client to OS/390

Since OS/390 provides two kinds of file systems (see 15.5, “Access to data and applications” on page 307), the NFS server has to be told which one is to be mounted. This is done through a prefix on the `mount` command.

```
mount -t nfs wtsc52:/hfs/etc /mnt/wtsc52
```

This mounts the /etc directory on host WTSC52, whereas the following command mounts the conventional file system with a mount point at a high level qualifier of HDM:

```
mount -t nfs wtsc52:HDM /mnt/wtsc52
```

It is the /hfs prefix that selects the UNIX file system on OS/390.

It may be that your installation has chosen to use a different prefix. You can either ask your system programmer or use the `showattr` command, which is also provided in the package. The following shows the output for our test system WTSC52:

```
[root@linux6 nfs-mvs]# showattr wtsc52
```

```
OS/390 Network File System Server Data Set Creation Attributes:
```

lrecl(80)	recfm(fb)	blksize(0)
space(100,10)	blks	dsorg(ps)
dir(25)	unit()	volume()
recordsize(512,4K)	keys(64,0)	nonspanned
shareoptions(3,3)		
mgmtclas()	dsntype(pds)	norlse
dataclas()	storclas()	

```
OS/390 Network File System Server Processing Attributes:
```

text	crlf	blankstrip
nofastfilesize	retrieve	maplower
mapleaddot	executebitoff	setownerroot
attrtimeout(120)	readtimeout(90)	writetimeout(30)
sync	nofileextmap	xlat(oemvs311)
sidefile(os390nfs.sc52mvs.nfs.mapping)		

```
OS/390 Network File System Server Site Attributes (not modifiable):
```

mintimeout(1)	nomaxtimeout	logout(604800)
nfstasks(8,16,8)	restimeout(48,0)	hfs (/hfs)
bufhigh(32M)	readaheadmax(16K)	cachewindow(112)
percentsteal(20)	maxrdfsleft(32)	logicalcache(16M)
smf(none)	pcnfsd	security(saf,saf,saf)
leadswitch	sfmax(20K)	nochecklist
fn_delimiter(,)	readdirtimeout(30)	
public()		

If you are going to access the root directory, don't forget the /:

```
[root@linux6 2]# mount -t nfs wtsc52:/hfs /mnt/wtsc52
```

```

mount: wtsc52:/hfs failed, reason given by server: No such file or
directory
[root@linux6 2]# mount -t nfs wtsc52:/hfs/ /mnt/wtsc52
[root@linux6 2]#

```

The root directory is *really* a directory whose name is `/`.

19.3.5 As a client to VM/ESA

From Linux for S/390 you can mount files from any of the three types of CMS file systems. These are the minidisk file system, the Shared File System, and the Byte File System.

```

[root@linux5 /]# mkdir /mnt/nfs
[root@linux5 /]# mount
wtscvmt:linux5.191,userid=linux5,password=penguin,trans=yes,lines=nl
/mnt/nfs
[root@linux5 /]# ls -l
total 3107
-r-xr-xr-x  1 root    daemon      31 Apr  3 19:56 $$nt$$userdata
-r-xr-xr-x  1 root    daemon      256 May  5 19:21 charlot.exec
-r-xr-xr-x  1 root    bin        108768 May 17 02:51 cons1.drvt13
-r-xr-xr-x  1 root    daemon     180 May 17 22:51 cpsend.exec
-r-xr-xr-x  1 root    daemon    54960 May  9 00:16 ctc.c
...
[root@linux5 /]# cat /mnt/nfs/marist.exec
/* REXX EXEC to boot Linux for S/390 from VM reader */
/* using RAM disk as the intial root file system */
'CP CLOSE RDR'
'CP PURGE RDR CLASS L'
'CP SPOOL PUN * RDR CLASS L'
'PUNCH READER MARIST A (NOH' /* Kernel image */
'PUNCH PARM MARIST A (NOH' /* Parameter file */
'PUNCH INITRD MARIST A (NOH' /* RAM disk root file system */
'CP SPOOL PUN * RDR CLASS A'
'CP SPOOL RDR KEEP CLASS L'
'CP IPL 00C CLEAR'
[root@linux5 /]# umount /mnt/nfs

```

In this example we have specified that EBCDIC-ASCII translation should occur and that new line characters are to be inserted at CMS record boundaries. Several of these options can be specified in the setup of the VM NFS server daemon so that they don't need to be entered on the mount command. For example, the treatment of a file as binary or needing codepage translation can be based on the CMS filetype of the file. This is specified with VMFILETYPE statements in the VMNFS configuration file.

Consult *VMESA V2R4.0: TCP/IP Function Level 320 Planning and Customization*, SC24-5847 and *VM/ESA V2R4.0 TCP/IP FL320 User's Guide*, SC24-5848 for further information.

19.3.6 As a client to VSE/ESA

We did not have the opportunity to test the mounting of VSE files from Linux for S/390. Details of the VSE/ESA NFS server capabilities can be studied in *Getting Started with TCP/IP for VSE/ESA 1.4*, SG24-5626.

19.3.7 Data representation considerations

Linux for S/390 is an ASCII-based implementation whereas OS/390, VM/ESA, and VSE/ESA store data predominantly in EBCDIC. This is true for both conventional file systems and the UNIX file system. You have to remember this specifically in the case where you have both types of data within the same directory. In the OS/390 case, you may need two mounts for the same directory, as shown in Figure 99.

```
[root@linux6 2]# mkdir /mnt/wtsc52/binary
[root@linux6 2]# mkdir /mnt/wtsc52/text
[root@linux6 2]# mount -t nfs wtsc52:/hfs/ /mnt/wtsc52/text
[root@linux6 2]# mount -t nfs wtsc52:/"hfs/,binary" /mnt/wtsc52/binary
```

Figure 99. Mount OS/390 data with a binary view

We will use these two subdirectories to differentiate between the binary and text views.

```
[root@linux6 2]# ls /mnt/wtsc52/binary
bin etc lib notesdata samples tmp usr was
dev krb5 mnt opt test-mdh u var web
[root@linux6 2]# ls /mnt/wtsc52/binary/test-mdh
hello.c helloa.c
[root@linux6 2]# cat /mnt/wtsc52/binary/test-mdh/helloa.c
#include <stdio.h>

int main ( int argc, char ** argv )
{
    printf ("\n Hello Ascii world \n" ) ;
}
[root@linux6 2]#
```

Figure 100. Mountings for text and binary view of data

The source file `helloa.c`, which we stored on OS/390 earlier, is encoded using an ASCII code page. So under the binary view, we can see it perfectly on Linux for S/390!

With the steps described here we have both a binary and a text view of the same data.

19.3.8 OS/390 access security

One major difference between UNIX-based systems like Linux and OS/390 is the way a user ID is defined. On UNIX systems there is a *login* name associated with a number, the *uid*. It is the *uid* that is the primary key to authorization on UNIX systems.

On OS/390, the character user ID is the key to authorization. Compared with UNIX, it is more like the login name.

The problem starts when a user, who is identified on a UNIX system, needs to have access to OS/390 resources such as data sets. UNIX propagates the *uid*, which is a number, but OS/390 needs a name. These cannot be matched directly. In this section we describe the security options and what tools are available to support them.

The NFS server on OS/390 supports the following security options when exporting data sets or HFS files:

- SAF This stands for Security Access Feature. It allows the NFS server, in cooperation with RACF, to fully control the access to files and data sets. For data sets this means we use profiles as they are defined to RACF. For UNIX files stored in the HFS, the *uid/gid* of the current OS/390 user ID is used.
- Export This creates a list of exported files and directories. The allowed access (*ro*, *rw*) to each of these is also defined.
- SAFEXP This a mixture of both of the above.

To use SAF an OS/390 user ID must be assigned to a mount connection. This is achieved through a small program provided by OS/390-NFS, *mvslogin*. There is also a logout service available, *mvslogout*. Both are provided in source. A third function is provided, *showattr*, to display the current setting of the NFS server on OS/390.

You can download these as a tarball from *prefix.NFSTARB*, which is available when the NFS Server on OS/390 is installed. The member *gfsawaix* contains the source as a tarball. We downloaded this package from our test system

WTSC52, as described in *OS/390 V2R6.0 NFS Customization and Operation*, SC26-7253. The package was installed using:

```
[root@linux6 2]#tar xvf gfsawaix.tar
```

You will see all the source files and a makefile exploded:

```
[root@linux6 3]# tar xvf ../gfsawaix.tar
./gfsawaxd.c
./gfsawclt.c
./gfsawlin.c
./gfsawlou.c
./gfsawmcl.c
./gfsawmou.c
./gfsawsha.c
./gfsawmnt.h
./gfsawsho.h
./makefile
tar: Only read 6160 bytes from archive ../gfsawaix.tar
tar: Error is not recoverable: exiting now
[root@linux6 3]#
```

Note: We ignored the error messages.

A makefile is provided so the executables can be generated. The first thing we had to change was the setting of the `cc` compiler command. Our Linux had no `cc` command.

```
[root@linux6 2]# make
cc -c -I. gfsawsha.c
make: cc: Command not found
make: *** [gfsawsha.o] Error 127
[root@linux6 2]#
```

We decided to put a symbolic link on `gcc`:

```
[root@linux6 bin]# ln -s gcc cc
[root@linux6 bin]# cc
cc: No input files
```

Since the `cc` command is also used to link, this is a very convenient way to proceed.

This makefile allows generation for several platforms. As a first test we used just `make`, which generates Sun executables according to the documentation *OS/390 V2R6.0 NFS Customization and Operation*, SC26-7253. There were several warnings issued, like the following:

```

.
.
cc -c -I. gfsawaxd.c
gfsawaxd.c: In function `xdr_rtnattr':
gfsawaxd.c:231: warning: passing arg 4 of `xdr_pointer' from
incompatible pointer type
.
.
.

```

There were no error messages, and we accepted these warnings for our first tests and tried the compiled commands. The results were promising:

```

[root@linux6 2]# showattr
GFSA972I usage: showattr [-t] [host] [/mountpoint]
[root@linux6 2]# mvslogin
GFSA956I usage: mvslogin [-pn] [-P password] [-g group] [-a account]
host [mvs_username]
[root@linux6 2]# mvslogout
GFSA959I usage: mvslogout host
[root@linux6 2]#

```

Logging in with a user ID on OS/390 which has a uid that does not match the one on Linux for S/390 produced the following:

```

[root@linux6 nfs-mvs]# mvslogin wtsc52 mdh
Password required
GFSA973A Enter MVS password:
GFSA978I MDH logged in ok.
Mismatch in uid/gid: OpenEdition uid is 99, gid is 0,
client uid is 0, gid is 0.
[root@linux6 nfs-mvs]#

```

In this case, user MDH was identified and authorized to access data sets and UNIX files. We show a mount of UNIX files in Figure 101 on page 381.


```

[root@linux6 2]# mount -t nfs wtsc52:/hfs/ /mnt/wtsc52
[root@linux6 2]# ls /mnt/wtsc52
bin etc lib notesdata samples u var web
dev krb5 mnt opt tmp usr was
[root@linux6 2]# ls -al /mnt/wtsc52
total 36868
drwxr-xr-x 17 root bin 8192 May 6 14:13 .
drwxr-xr-x 5 root root 4096 May 9 11:31 ..
dr-xr-xr-x 2 root bin 8192 Jun 22 1999 ...
-rw----- 1 root bin 2758 May 6 14:19 h_history
drwxr-xr-x 4 root bin 49152 Mar 18 11:59 bin
drwxr-xr-x 2 root bin 8192 May 5 16:45 dev
drwxr-xr-x 2 root root 8192 Jul 30 1999 etc
lrwxrwxrwx 1 root bin 16 Jul 30 1999 krb5 ->
etc/dce/var/krb5
drwxr-xr-x 2 root bin 8192 Sep 17 1999 lib
drwxr-xr-x 2 root bin 8192 Mar 22 04:28 mnt
drwxr-xr-x 2 root bin 8192 Mar 20 16:56 notesdata
drwxr-xr-x 2 root bin 8192 Jun 22 1999 opt
drwxr-xr-x 4 root bin 8192 Feb 4 11:23 samples
drwxrwxrwt 2 root bin 8192 Apr 19 08:20 tmp
drwxr-xr-x 2 root bin 8192 Apr 19 08:23 u
drwxr-xr-x 10 root bin 8192 Jun 22 1999 usr
drwxrwxrwx 2 root bin 8192 Jun 22 1999 var
drwxr-xr-x 3 root bin 8192 Mar 20 16:56 was
drwxr-xr-x 3 root bin 8192 Mar 20 16:56 web
[root@linux6 2]#

```

Figure 101. Directory listing of hfs as seen from Linux

A detailed view of the /bin directory of wtsc52 is shown in Figure 102 on page 382.

```

[root@linux6 2]# ls -al /mnt/wtsc52/bin
total 649216
drwxr-xr-x  4 root    bin          49152 Mar 18 11:59 .
drwxr-xr-x 17 root    bin          8192 May  6 14:13 ..
lrwxrwxrwx  1 root    bin           27 Jul 30 1999 EUVASRVR ->
../usr/lpp/dc
e/bin/EUVASRVR
drwxr-xr-x  2 root    bin          20480 Mar 18 11:59 IBM
drwxr-xr-x  2 root    bin          8192 Sep 17 1999 X11
lrwxrwxrwx  1 root    bin           27 Jul 30 1999 acl_edit ->
../usr/lpp/dc
e/bin/acl_edit
-rwxr-xr-x 17 root    bin          69632 Jul 12 1999 alias
-rwxr-xr-x  2 root    bin          163840 Jul 12 1999 ar
-rwxr-xr-x  2 root    bin          77824 Jul 12 1999 asa
-rwsr-xr-x  3 root    bin          176128 Mar 18 11:59 at
-rwxr-xr-x  2 root    bin          368640 Mar 11 17:21 awk
.
.
.
-rwxr-sr-x  2 root    111          122880 Jul 12 1999 write
-rwxr-xr-x  2 root    bin          167936 Dec 29 11:40 xargs
-rwxr-xr-x  2 root    bin          364544 Jul 12 1999 yacc
-rwxr-xr-x  3 root    bin          155648 Jul 12 1999 zcat
[root@linux6 2]#

```

Figure 102. Directory listing of /bin on hfs as seen from Linux

You may want to access files that are in a different UNIX file system. Typically for OS/390 UNIX System Services, this is the case for directories related to individual users. They have their own home directory in /u, (i.e. /u/mdh). Most installations use the auto-mount feature to manage mounting of individual file systems. Your mount on the client *has to span* this mountpoint as well.

```

[root@linux6 /root]# mount -t nfs wtsc52:/hfs/u/hdm /mnt/wtsc52/u/hdm
[root@linux6 /root]# mount -t nfs wtsc52:/hfs/u/mdh /mnt/wtsc52/u/mdh

```

With these commands we were able to mount OS/390 directories (/u/hdm, /u/mdh) that were in file systems other than the root (/) file system.

```
[root@linux6 /root]# ls /mnt/wtsc52/u/hdm
README.bin.os390  hello          make-3_76_1_os390_bin_tar.Z  xcalc
Vsmregn.c        hello.C        misc                          xcalc.tar
amadeus          hello.o        netprog                       xcalc_tar.Z
apache           host.c         new                            xclock
apache_136       host.o         nfstarb                       xedit
bin              hostname.c     precomp                       xntp
bpx1gps          hostname.o     sendmail                      xterm
cvs              info           stevens                       zeta
flex-2.5.3       itools        test
fsinuse          lh-able       unix
gzip-1_2_4.tar  lh-able.tar   wtsc52oe
```

Figure 103. Contents of /u/hdm directory on OS/390

```
[root@linux6 /root]# ls -al
total 32
drwxr-x---  4 root    root      4096 May  9 16:33 .
drwxr-xr-x 18 root    root      4096 May  4 17:07 ..
-rw-----  1 root    root     15286 May 11 09:11 .bash_history
-rw-r--r--  1 root    root         0 May  3 13:35 bootsect
-rw-r--r--  1 root    root         0 May  3 13:35 image
-rw-r--r--  1 root    root         0 May  3 13:35 ipldevice
drwxr-xr-x  4 root    root      4096 May 10 15:10 nfs-mvs
drwxr-xr-x  2 root    root      4096 May  9 15:51 nfs-test?[D
-rw-r--r--  1 root    root         0 May  3 13:35 parmfile
[root@linux6 /root]# ls -al > /mnt/wtsc52/u/mdh/test_from_linux6
[root@linux6 /root]#
```

Figure 104. Creating test data from the client

With the last command in Figure 104, we had some test data that could be seen immediately on OS/390, as the following output shows:

```
HDM @ :/u/mdh:~>ls -al
total 72
drwxr-xr-x  2 MDH     SYS1      8192 May 10 23:27 .
drwxr-xr-x 36 AAAAAAA SYS1      8192 May  4 19:02 ..
-rwxr-xr-x  1 AAAAAAA SYS1         24 May  5 20:26 .profile
-rwxrwxrwx  1 MDH     SYS1         20 May  5 20:16 .setup
-rw-----  1 MDH     SYS1         98 May  5 20:27 .sh_history
-rw-----  1 MDH     SYS1      1444 May 10 21:48 mdh
-rw-r--r--  1 MDH     SYS1         577 May 10 23:27 test_from_linux
HDM @ :/u/mdh:~>cat test_from_linux6
```

```

total 32
drwxr-x---  4 root    root      4096 May  9 16:33 .
drwxr-xr-x 18 root    root      4096 May  4 17:07 ..
-rw-----  1 root    root     15286 May 11 09:11 .bash_history
-rw-r--r--  1 root    root         0 May  3 13:35 bootsect
-rw-r--r--  1 root    root         0 May  3 13:35 image
-rw-r--r--  1 root    root         0 May  3 13:35 ipldevice
drwxr-xr-x  4 root    root      4096 May 10 15:10 nfs-mvs
drwxr-xr-x  2 root    root      4096 May  9 15:51 nfs-test-[D
-rw-r--r--  1 root    root         0 May  3 13:35 parmfile
HDM @ :/u/mdh:~>

```

It may be necessary to access the conventional file system also. After identifying yourself to OS/390 using `mvslogin`, mounting from the conventional file system is then very similar, as shown in Figure 105.

```

[root@linux6 2]# mvslogin wtsc52 mdh
Password required
GFSA973A Enter MVS password:
GFSA978I MDH logged in ok.
      Mismatch in uid/gid: OpenEdition uid is 99, gid is 0,
      client uid is 0, gid is 0.
[root@linux6 2]# mount wtsc52:'sys1.parmlib' /mnt/wtsc52
[root@linux6 2]# ls /mnt/wtsc52
adyset00 commnd43 dfhipcsp ieaic54 igdsmsbj jes3inmf prog1z progrn
adyset01 commnd47 dfm00    ieaic61 igdsmsbg jes3inpr prog54 progs0
adyset02 commnd48 diag00  ieaic71 igdsmsmt jes3inr4 prog55 progs1
.
.

```

Figure 105. Mounting an OS/390 data set on Linux for S/390

We selected `cat commnd00` as an example; this is shown in Figure 106 on page 385.

```

[root@linux6 2]# cat /mnt/wtsc52/commnd00
COM='START VLF, SUB=MSTR, NN=01 '
COM='D T'
COM='CD SET, SDUMP=(CSA, GRSQ, LPA, LSQA, ALLNUC, ALLPSA, RGN, SQA, SWA) ,ADD'

COM='CD SET, SDUMP=(SUM, TRT, COUPLE, XESDATA) ,ADD'
COM='CD SET, SDUMP, TYPE=XMEME, MAXSPACE=1536M, Q=YES '
COM='CD SET, SYSDUMP=(ALL, ALLNUC) ,ADD'
COM='S RMF, , , (MEMBER (&RMFMON01)) '
COM='S IRRDPTAB'
COM='DD NAME=DUMP.D&MON.&DAY..H&HR..&SYSNAME..&JOBNAME..S&SEQ.'
.
.
.
.
COM='SET EXS=00'
COM='MN JOBNAMES,T'
COM='K M, AMRF=Y'
[root@linux6 2]#

```

Figure 106. Display of a mounted OS/390 data set

Attention

We followed the guidance given in the chapter “Porting the mvlogin, mvlogout, and showattr Commands” in *OS/390 V2R6.0 NFS Customization and Operation, SC26-7253* and we did not observe any problems in our tests. However, we recommend being very careful when following these examples.

19.4 VM/ESA access security

The MOUNT command can pose a security problem, because NFS client systems often store this command's argument values to respond to a later query asking for information about the mounts that are currently in effect. A password supplied in an argument to a MOUNT command could then be revealed in the query response to someone other than the user who executed the MOUNT command.

This exposure is evident in Linux. After mounting the LINUX5 virtual machine's 191 minidisk, the response to a Linux `mount` command was:

```
[root@linux5 /root]# mount
/dev/dasda1 on / type ext2 (rw,errors=remount-ro)
none on /proc type proc (rw)
wtscvmt:linux5.191,userid=linux5,password=penguin on /mnt/nfs type nfs
(rw,addr=
9.12.14.155)
[root@linux5 /root]#
```

The MOUNTPW command provides an alternative path for sending passwords, account, and user ID information to the VM NFS server. This information can then be omitted from the subsequent related MOUNT command, and therefore is not present in a display of currently mounted file systems.

A MOUNTPW command precedes the related MOUNT command, which must follow within 5 minutes. After 5 minutes, the VM NFS server discards the information it received in a MOUNTPW command. If a second MOUNTPW command is issued for the same CMS disk or directory before a MOUNT command for that object, the data from the first MOUNTPW command is discarded.

The MOUNTPW C source file is supplied on the VM NFS feature tape, as well as executable modules built for the IBM AIX and OS/2 environments. To use MOUNTPW with Linux for S/390 the MOUNTPW C file must be copied to Linux and compiled into an executable program.

Compile the MOUNTPW program using the Linux C compiler. The location of header files (.h) in the MOUNTPW source program will need to be changed to reflect their place in the Linux file system structure.

We did not attempt to compile MOUNTPW on Linux for S/390.

Another common technique is to use a PCNFS daemon to handle client user ID and password authentication prior to issuing the NFS mount request. The VM/ESA NFS feature has PCNFS daemon capability.

Once verified, the UID and GID by which that user is known at the host are returned to the client.

In VM, the POSIXINFO and POSIXGLIST CP directory entry statements define the UID and GIDs associated with a user ID.

When PCNFSD is used, there is no need to send user ID and logon password in the MOUNT or MOUNTPW. If user ID or password are not provided by MOUNTPW or MOUNT, the PCNFSD values are used by the NFS server, as

long as the MOUNT request is received by the NFS server immediately following the PCNFSD request.

If an NFS client is configured to call PCNFSD user ID authentication services and PCNFSD is available on your VM NFS server, using MOUNTPW is not necessary because user ID and password information are passed on PCNFSD requests. This applies to mounts for SFS directories, BFS directories, and minidisks protected by ESMs. A minidisk protected by link passwords must still provide the password on the MOUNT or MOUNTPW command.

We could find no evidence of the existence of a PCNFS client for Linux. Currently MOUNTPW appears to be the best way to avoid exposure of VM user IDs and passwords.

Chapter 20. Samba

Samba is an open source software package that enables a UNIX machine to become a Windows file and print server. This is a simple matter from an end user's point of view. With Samba up and file and print *shares* defined, a desktop that has SMB support (Windows, OS/2, others) can easily get access to the defined resources. No additional software needs to be installed on the desktop, unlike with NFS, for example.

The Samba package allows you to marry the many SMB clients in the world with reliable Linux/UNIX servers.

20.1 Installation

Samba is not included with the large file system from Marist. It can be installed on Linux for S/390 in the following ways:

- From an RPM binary
- From an RPM source
- From the source in the original package

20.1.1 Installing from an RPM binary or source

Samba RPMs can be obtained from the Thinking Objects Linux for S/390 Web site at:

```
http://www.linux.s390.org/download/
```

The following RPMs are available:

- samba-2.0.5a-12.s390.rpm
- samba-client-2.0.5a-12.s390.rpm
- samba-common-2.0.5a-12.s390.rpm

We installed the Samba server binaries via the following commands:

```
[root@linux390 samba]# rpm --install samba-common-2_0_5a-12_s390.rpm  
[root@linux390 samba]# rpm --install samba-2_0_5a-12_s390.rpm
```

These commands put the Samba executables in the directory `/usr/sbin` and the `smb.conf` file in the `/etc` directory.

20.1.2 Installing from source in the original package

An overview diagram of installing packages from source is shown in Figure 107 on page 390.

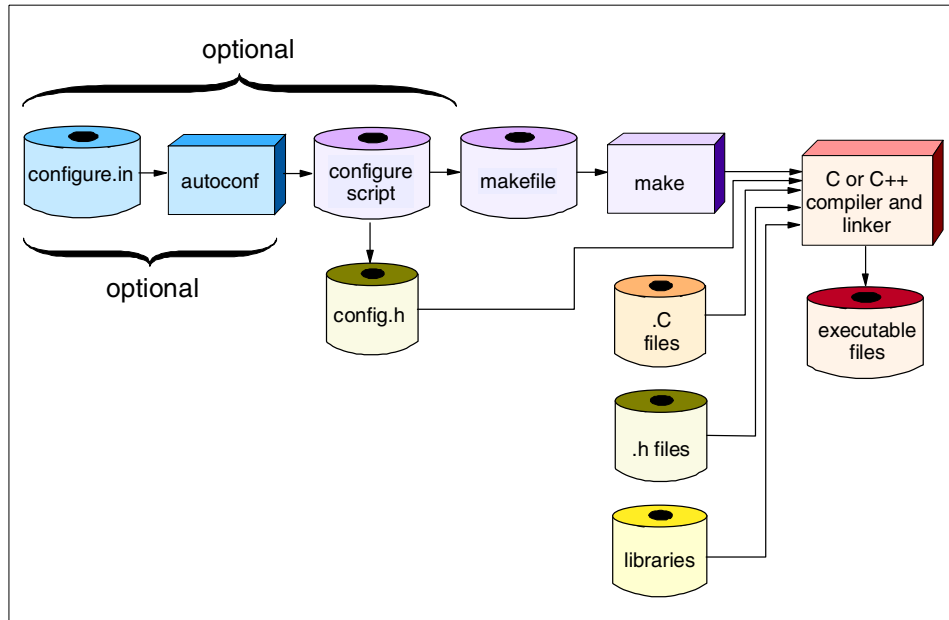


Figure 107. Building open source software packages

The Samba package can be obtained on the Web from:

<http://www.samba.org>

This Web site is where the owning organization makes Samba available to the world. We obtained `samba-2.0.7.tar.gz` (4276392 bytes) which, at the time, was the same file as `samba-current.tar.gz`.

The package was moved to Linux for S/390 in binary via FTP. It was unwound with `tar` (note that the `z` flag on `tar` avoids the need to use `gzip`) ¹ so as to leave the tarball compressed and thus save a little disk space. Then for convenience, a symbolic link was established from `samba` to `samba-2.0.7` ².

```

[mikem@linux390 samba]$ ls -l
total 4192
-rw-rw-r-- 1 mikem mikem 4276392 May 2 13:11 samba-2.0.7.tar.gz
[mikem@linux390 samba]$ tar xzf samba-2.0.7.tar.gz 1
[mikem@linux390 samba]$ ln -s samba-2.0.7 samba 2
[mikem@linux390 samba]$ ls -l
total 4196
lrwxrwxrwx 1 mikem mikem 11 May 2 13:13 samba -> samba-2.0.7/
-rw-rw-r-- 1 mikem mikem 4276392 May 2 13:11 samba-2.0.7.tar.gz
  
```

Then we changed to the samba directory. A quick inspection of the README file told us that the install instructions were not in the file INSTALL, but rather in docs/textdocs/UNIX_INSTALL.txt. That file told us that building Samba could be accomplished via the common 3-step approach:

```
./configure
make
make install
```

Invoking `./configure` revealed a shortcoming of Linux for S/390 that was not a surprise: the system type could not be determined.

```
[mikem@linux390 source]$ ./configure --prefix=/usr
creating cache ./config.cache
checking for gcc... gcc
...
checking host system type... configure: error: can not guess host type;
you must specify one
```

The problem is that the files `config.guess` and `config.sub` have not been updated in the Samba package to include the s390 architecture. These two files are used by the configure script to determine the architecture of the host system. Because Linux running on S/390 is a new architecture, most copies of `config.guess` and `config.sub` will not recognize it.

A proper version of these files was found in `/usr/lib/rpm`. The original pair were saved and the new pair were copied with the following commands:

```
[mikem@linux390 source]$ mv config.guess good.config.guess
[mikem@linux390 source]$ mv config.sub good.config.sub
[mikem@linux390 source]$ cp /usr/lib/rpm/config.guess .
[mikem@linux390 source]$ cp /usr/lib/rpm/config.sub .command locate
```

The configure script now ran successfully, with the difference being that the host system type could now be determined:

```
checking host system type... s390-ibm-linux-gnu
```

Next the command `make` was invoked to build the executables. It ran flawlessly! It took about 5 minutes on a 144 BogoMIP machine.

Next the command `make install` was invoked to install the executables. Here we got an error:

```
[mikem@linux390 source]$ make install
...
mkdir: cannot make directory `/usr/local/samba': Permission denied
```

This happened because we were running with a UID of 500 **1** and the directory /usr/local is owned by root -- and only the owner has write permission **2**.

```
[mikem@linux390 source]$ id
uid=500(mikem) gid=500(mikem) groups=500(mikem) 1
[mikem@linux390 source]$ ls -ld /usr/local
drwxr-xr-x 11 root root 4096 Aug 2 1998 /usr/local/ 2
```

To remedy this, we invoked the command `su` with no arguments to change our effective UID to 0 (root). Now the command `make install` ran quickly and flawlessly.

Samba should now be installed. The executables are probably in the directory /usr/local/samba/bin, though it is possible that `make install` will put them in /usr/sbin. It should be noted that in most Linux distributions, the Samba executables are located in the /usr/sbin directory.

You now need a copy of the smb.conf file.

20.1.2.1 Copying a sample smb.conf file

The main configuration file for Samba is named smb.conf. You need a simple one to get started. There is an example of one in the Samba example directories. It can be copied to the install lib directory via the following commands:

```
[root@linux390 samba]# cd samba/examples/simple
[root@linux390 simple]# cp smb.conf /usr/local/samba/lib
```

20.1.2.2 Starting Samba

Now you might want to test bringing the Samba daemons up. `nmbd` provides a NetBIOS browse list, and `smbd` is the main provider of SMB shares.

```
[root@linux390 simple]# cd /usr/local/samba/bin
[root@linux390 bin]# ./nmbd -D
[root@linux390 bin]# ./smbd -D
```

The `ps` command piped to `grep` should show that the two Samba daemons are up. Logs are written to the file log.nmb and log.smb in the var directory:

```
[root@linux390 bin]# ps -ef | grep mbd
root      465      1  0 14:13 ?          00:00:00 ./nmbd -D
root      471      1  0 14:13 ?          00:00:00 ./smbd -D
root      473     412  0 14:13 tty0      00:00:00 grep mbd
[root@linux390 bin]# ls ../var
locks log.nmb log.smb
```

Samba should be ready to go from the server side.

20.2 Customization

This section describes the steps taken to customize Samba on Linux for S/390.

20.2.1 Starting Samba automatically

For our purposes we stored Samba executables in `/usr/bin`. The documentation that comes with this distribution proposes, for BSD-style UNIX systems, to add code into `rc.local`, which can be found in the `/etc` or `/etc/rc.d` directories. In our installation of Linux this shell script is in `/etc/rc.d`. We modified it to our needs as follows:

```
# added by HDM, 5/4/2000 according to SWAT doc, chapter 2.05
# we have smbd/nmbd stored in /usr/bin therefor we kick them
# from there
if [ -x /usr/bin/smbd ]; then
    echo "Starting smbd now..."
    /usr/bin/smbd -D -d4
    echo "Starting nmbd now ..."
    /usr/bin/nmbd -D
```

With this addition to `rc.local`, Samba started at the next boot of our Linux.

20.2.2 Starting SWAT automatically

The Samba Web Administration Tool (SWAT) enables you to maintain Samba via a Web browser. See 20.3.1, "SWAT" on page 399 for more details.

It is common for SWAT to be launched via `inetd` (see 14.6.1, "Overview of `inetd`" on page 272 for a discussion of `inetd`). To add SWAT to `inetd`, you need to edit your `/etc/inetd.conf` and `/etc/services` files. The Samba documentation suggests a port number of 901, since it is not commonly used and is below 1024 (some `inetd` daemons have a security hole when using ports above 1024).

In the file `/etc/services` add the line:

```
swat 901/tcp
```

In the file `/etc/inetd.conf` add the line:

```
swat      stream  tcp  nowait  root    /usr/bin/swat swat
```

Once you have edited `/etc/services` and `/etc/inetd.conf`, you need to tell `inetd` to reread the configuration file. This is done by sending the process a `SIGHUP` signal via the `kill` command. For example:

```
[mikem@linux390 /etc]$ ps -ef | grep inetd
root      333      1  0 May02 ?        00:00:00 inetd
[root@linux390 /etc]# kill -SIGHUP 333
```

20.2.3 Using SWAT to customize Samba

Now that the `inet.conf` file has been reread by `inetd`, SWAT should be listening on TCP/IP port 901. It can be accessed via a browser using the server name and a “:901” suffix in the URL. For example, we accessed SWAT via the URL:

```
http://linux390.itso.ibm.com:901/
```

You will first be prompted for a user ID and password and then shown the main SWAT screen (Figure 108 on page 395).

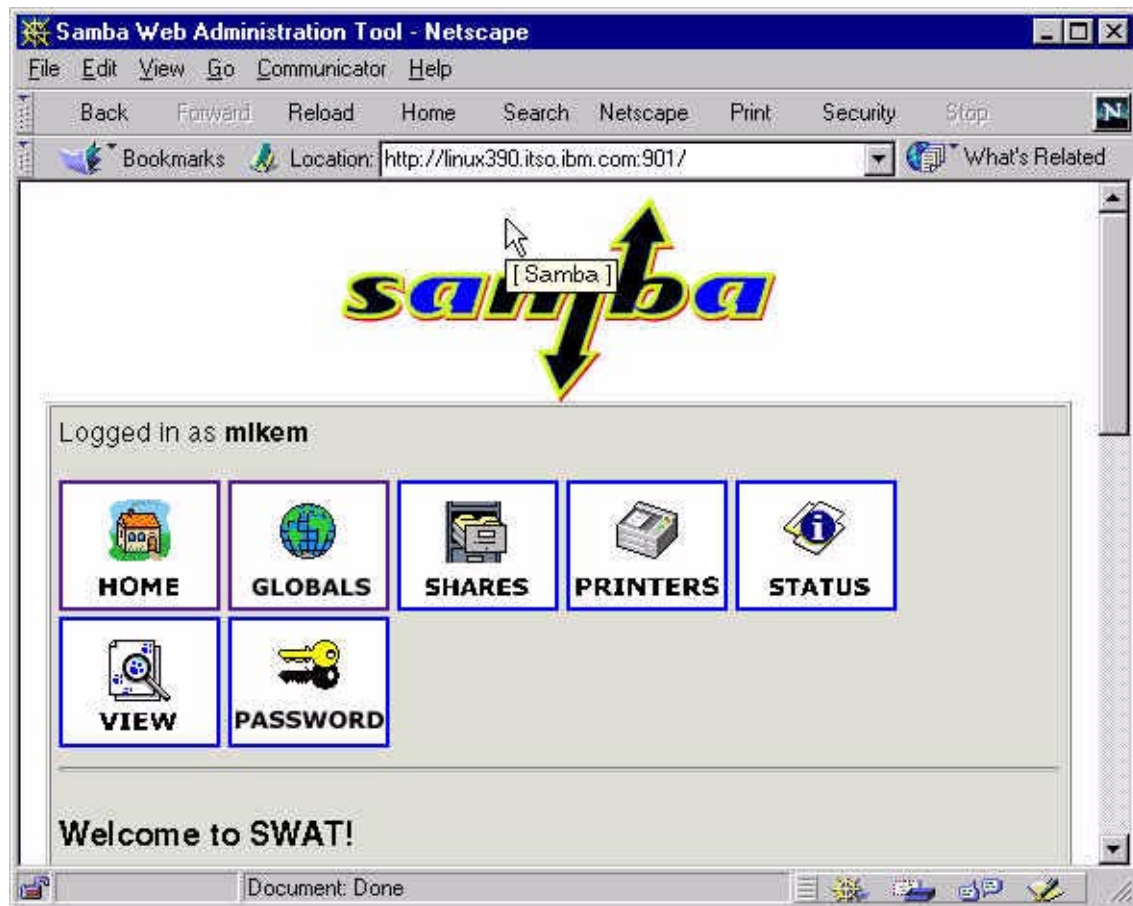


Figure 108. SWAT main window

SWAT can be used to administer all aspects of Samba.

20.2.4 Additional resources

The following are some good books on Samba:

- *Samba: Integrating UNIX and Windows*
- *Using Samba*

The latter is available online via SWAT! From the main window of SWAT, click **Using Samba**, near the bottom of the Web page.

20.2.5 Client-side operation

Because Samba is a client/server application, both sides must be working properly in order to share file and print resources. Client-side access to a Samba server can be tricky. Overall, in order to get a Samba *share*, there are two main hurdles to overcome:

- Finding the server
- Supplying the proper credentials

If you are using Linux or another UNIX-based system as a client, you will probably want to use NFS because that is the standard network file system. However, you can also install the “smb client.”

Because Microsoft Windows is such a common desktop operating system, we will focus on it as a client. Let us divide the Windows client world: Windows NT, which now includes Windows 2000, and Windows 9x, which includes Windows 95 and 98 and will probably include the forthcoming Windows Millennium.

We will address access from both a command line and a GUI point of view.

20.2.6 Finding a Samba server from Windows

The format of the Universal Naming Convention (UNC) is `\\server\share`. Originally, it was only a NetBIOS name that could be included as the server.

Finding a Samba server from NT is easier than from 9x because you can specify either a DNS name or a dotted decimal address in the server portion of the UNC. Therefore, if the server’s DNS name is `linux390.itso.ibm.com` and its IP address is `9.12.14.158`, then from NT you can either specify `\\linux390.itso.ibm.com\share` or `\\9.12.14.158\share` as the UNC. If the client were set up to resolve properly, you could use either `linux390` or `linux390.itso.ibm.com` synonymously because the resolver would assume the domain part for you. The use of IP numbers in a UNC or URL is a giveaway for sloppy DNS practices, and usually implies a poor DNS configuration.

From Windows 9x, however, you can only use the server’s NetBIOS name. There are two common ways of resolving the NetBIOS name:

- Via the NetBIOS *browse list* (aka Windows Network Neighborhood)

The daemon `nmbd` announces the NetBIOS name of the Samba server. It should be propagated to the NetBIOS browse list (this may take some time initially), added to the NetBIOS name list, and become accessible.

Another way of finding a NetBIOS machine is via the Find Computer dialog (**Start -> Find -> Computer**).

- Via the LMHOSTS file

This file allows you to specify an IP address to NetBIOS name mapping. On Windows 9x, a sample LMHOSTS file is shipped in C:\WINDOWS\LMHOSTS.SAM. It can be copied and edited with the following commands:

```
C:\WINDOWS\>copy lmhosts.sam lmhosts
C:\WINDOWS\>notepad lmhosts
```

Then, a one-line entry can be added:

```
9.12.14.158 linux390
```

When Windows tries to resolve the NetBIOS name, it actually checks this mapping file first.

20.2.7 Supplying the proper credentials from Windows

Both a user ID and password must be supplied to the Samba server in order to get a shared resource.

20.2.7.1 Supplying the user ID

Windows NT is again easier than 9x because it allows you to supply the user ID. From the Map Network Drive dialog, there is an additional text field, "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. So 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 "%" character delimiter. For example, accessing the share `\\linux390\myShare%mikem` will allow the share `myShare` to be accessed with the user ID `mikem`.

20.2.7.2 Supplying the password

To supply the password, you need to know whether the password going over the network is encrypted. Windows 95 OEM2 and later, and Windows NT, Service Pack 3 and later all encrypt the password going over the wire. 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.

A good background discussion and an explanation of the security and other trade-offs of using password encryption are provided in the file ENCRYPTION.txt in the Samba directory docs/textdocs.

20.2.7.3 Turning off password encryption on clients

To turn off password encryption on Windows clients, read the files Win95.txt and WinNT.txt in the docs/textdocs directory for complete details. A brief description is given here.

For NT or Windows 2000, bring up the registry editor with the command `regedt32`. Go to the following NT hive:

```
HKEY_LOCAL_MACHINE\system\CurrentControlSet\Services\Rdr\Parameters\.
```

For Windows 2000, go to:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\LanmanWorkStation\Parameters
```

For Windows 9x, go to:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\VxD\VNETSUP
```

For any of these operating systems, add the following value:

```
EnablePlainTextPassword:REG_DWORD=1
```

Reboot Windows and passwords will no longer be encrypted over the wire.

20.2.7.4 Accepting encrypted passwords

If turning off password encryption is not feasible or possible, you can also instruct Samba to accept encrypted passwords. To do this, use SWAT.

Bring up SWAT and click **Passwords** at the top. You should see two sections labeled “Server password management” and “Client/server password management.”

20.2.7.5 Using Windows NT to validate the credentials

If you are using “security = server,” the validation of credentials can be done by an NT server. See one of the Samba manuals previously referenced for a complete description of this feature.

20.2.8 Accessing a Samba share from a DOS prompt

From DOS, you can use the `net use` command to access a Samba share. Type `net use /?` for the exact syntax, as it differs on various flavors of Windows.

20.2.9 Accessing a Samba share from a GUI

Windows NT, Windows 9x and now Windows 2000 all have slightly different GUI interfaces. From the various types of Explorer interfaces, you can bring up the Map Network Drive panel. On some systems this is accessed by clicking **Tools** and then **Map Network Drive**. However, on Windows NT, you can only get to this panel via the toolbar. Click **View** then **Toolbar**, then look for the Map Drive icon.

20.3 Tools

As mentioned, SWAT is the premier tool for administering Samba.

There are several small tools available to administer Samba from the Linux shell. These are:

- `smbstatus`
- `smbpasswd`
- `testparm`
- `testprns`

Also, the client tool `smbclient` is discussed.

20.3.1 SWAT

We think it is no longer necessary to use `smbpasswd`, `testparm` and `testprns` because SWAT gives you a very convenient and safe way to manage settings for Samba. Settings done through SWAT can be assumed to be correct.

There may be an exception from this for `smbstatus`, which may be suitable for Samba status over time. If you add `smbstatus` to the list of processes to be controlled by `cron`, you can easily get an ongoing overview of Samba. The following shows an output from `smbstatus`. To get this you have to redirect both `stderr` and `stdout` to the same file.

```
[root@linux6 /home]# smbstatus &> smbst.lst
[root@linux6 /home]# cat smbst.lst
doing parameter log file = /var/log/samba/log.%m
doing parameter max log size = 50
doing parameter socket options = TCP_NODELAY SO_RCVBUF=8192
SO_SNDBUF=8192
doing parameter dns proxy = No
Processing section "[homes]"
doing parameter comment = Home Directories
doing parameter read only = No
```

```

doing parameter browseable = No
Processing section "[printers]"
doing parameter comment = All Printers
doing parameter path = /var/spool/samba
doing parameter print ok = Yes
doing parameter browseable = No
Processing section "[tot62]"
doing parameter comment = Share for tot62
doing parameter path = /home/tot62
doing parameter read only = No
pm_process() returned Yes
Trying sysv shm open of size 1048576

```

Samba version 2.0.5a

```

Service      uid      gid      pid      machine
-----
tot62        tot62    tot62    9639     tot62    (9.12.2.124) Mon May 16
14:16:57 2000

```

No locked files

Share mode memory usage (bytes):

```

1048464 (99%) free + 56 (0%) used + 56 (0%) overhead = 1048576 (100%)
total

```

All items that need to be administered to run Samba can easily be set using SWAT. We show this with the following example.

Figure 109 on page 401 shows the SWAT Web page with password window selected. It is divided into two parts, *Server Password Management* and *Client/Server Password Management*, respectively. The first handles the password in `/etc/smbpasswd`, whereas the second allows managing a password on another SMB server.

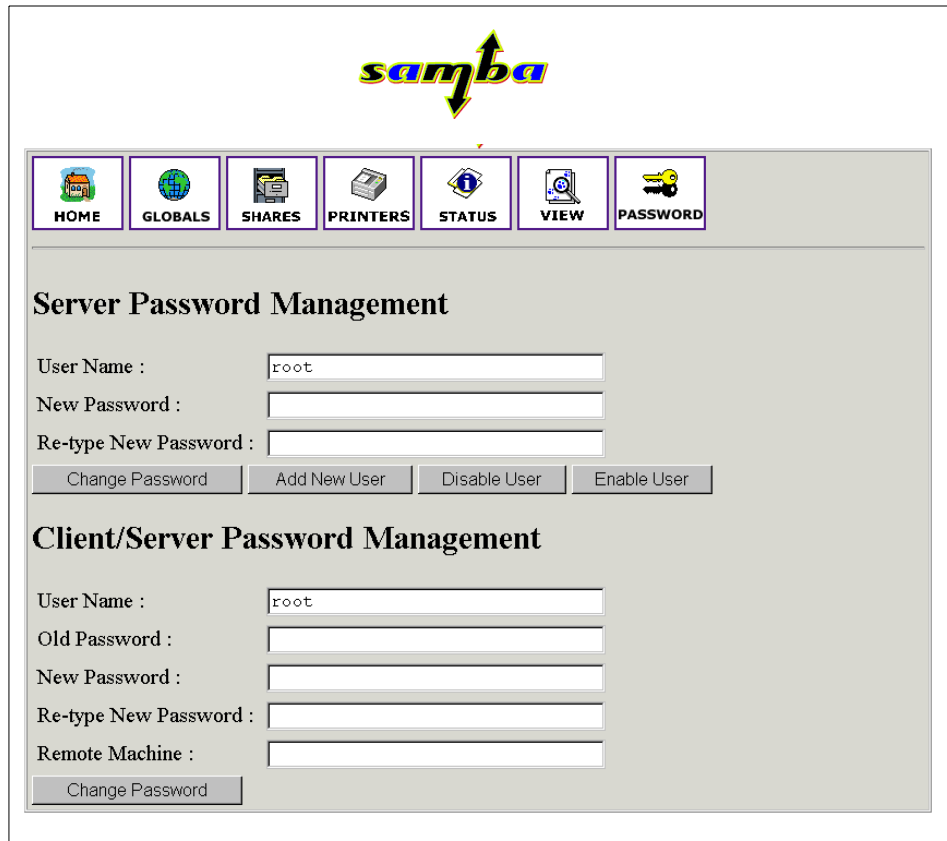


Figure 109. SWAT password management interface

There is a small caveat: You would probably like to synchronize the Samba password and the UNIX passwords. In that case you have to set the following manually in `/etc/smb.conf`:

```
[global]
    unix password sync = yes
```

Security Options

Help	security	USER	Set Default
Help	encrypt passwords	Yes	Set Default
Help	update encrypted	No	Set Default
Help	guest account	nobody	Set Default
Help	hosts allow		Set Default
Help	hosts deny		Set Default

Figure 110. SWAT password management without UNIX synchronization

After we applied this change, SWAT showed this setting (Figure 111):

Security Options

Help	security	USER	Set Default
Help	encrypt passwords	Yes	Set Default
Help	update encrypted	No	Set Default
Help	passwd program	/usr/bin/passwd %u	Set Default
Help	unix password sync	Yes	Set Default
Help	guest account	nobody	Set Default
Help	hosts allow		Set Default
Help	hosts deny		Set Default

Figure 111. SWAT with UNIX password sync setting

The same is true for the password change program. We recommend that you use a Samba configuration file that contains *all* [global] settings. You will then be able to modify any setting through SWAT.

20.3.2 smbclient

The tool `smbclient` is useful. It was originally designed as a testing tool, but ended up being used interactively to access SMB file shares in a fashion similar to that of an FTP client. To use the `smbclient` command, you must supply the share name as the first parameter. The Universal Naming Convention (UNC) format for an SMB share is `\\server\share`. However, the backslash is the escape character in UNIX, so every backslash has to be

doubled. An example of accessing an SMB server via the `smbclient` command follows:

```
[root@linux390 bin]# ./smbclient \\\itsont2\\e$ -U mikem
added interface ip=9.12.14.158 bcast=9.12.14.255 nmask=255.255.255.0
Password:
Domain=[WORKGROUP] OS=[Windows NT 4.0] Server=[NT LAN Manager 4.0]
smb: \> dir
 5600                               DA          0 Thu Jan 13 10:02:51 2000
 5609                               DA          0 Fri Apr 28 09:18:39 2000
 5665                               D           0 Thu Mar 30 12:30:35 2000
 5681                               D           0 Wed May 31 19:24:38 2000
...
    49131 blocks of size 131072. 13293 blocks available
```

Chapter 21. The Apache Web server

The Apache Web server has proven to be one of the most popular Web servers on the Internet; some estimates state that more than 50% of the Web servers in the world are using Apache. The Apache project was started by people running the original National Center for Super Computing Applications (NCSA) Web server. When the primary developer for that server left the NCSA, people using the NCSA server began exchanging patches. Before long the group realized they needed a forum to manage the patches, and the Apache project was born. Apache is a full-function Web server that performs quite well.

The Apache Web server supports:

- HTTP/1.1 protocol
- File-based configuration
- Common Gateway Interface (CGI) support
- Virtual Host support
- HTTP (or basic) authentication
- Integrated Perl (the defacto Apache CGI script language)
- Operation as a caching proxy server
- Customizable logging
- Server Side Includes (SSI)
- SSL (through a set of patches called Apache-SSL)
- User session tracking capability
- FastCGI
- Java Servlets (through the Jserv module)

The Apache Web server is built in a highly modular fashion. It consists of a small core of code, and a large number of configurable modules that supply the majority of function. Using the configuration file and configure script it is possible to build a custom Apache server that includes only the function you need. In addition, the Apache Web server supports a server API. This API makes it possible to create your own modules extending the function of the server

The Apache Web server runs on a wide variety of hardware platforms. It runs on most if not all of the various types of UNIX systems, as well as on

Windows 95/98/NT and many other desktop and server-class operating systems. It was even ported to OpenEdition VM/ESA.

Many people contribute to the project, as well as vendors such as IBM. It continues to evolve toward meeting today's e-business needs at a rapid pace. For more information see:

<http://www.apache.org>.

21.1 Installation

The Marist large file system comes with the Apache Web server already installed. In order to use this Web server you must first update the `ServerName` statement in the `HTTPD CONF` file. This file is located in the `/etc/httpd` directory. This statement sets the host name of the server. You should update the statement to include the fully qualified domain name of your host (not the short name), for example:

```
ServerName webserver.mycompany.com
```

21.1.1 Obtaining a later Apache level

The level of Apache installed in the large Marist file system is 1.3.5. However, more current versions of the server are available from:

<http://www.apache.org>

We decided to obtain the source for a more recent version of Apache and install it into the large Marist file system.

The Apache source comes in a UNIX composite file format known as a Tape ARchive or tar file. In addition, this tar file is compressed, so it will have an extension of `.Z`. The level we obtained from <http://www.apache.org> is 1.3.12. The name of the tar file we downloaded is `apache_1.3.12.tar.Z`.

21.1.2 Exploding the Apache tar file

If you use the Apache tar file, download the source file to your Linux for S/390 system. Then you need to extract all of the individual files from the composite tar file. When you invoke the `tar` command, it will create a subdirectory within your current working directory named `apache_1.3.12` and place all of the files and subdirectories include in the tar file in that directory. If you want this directory created under your root, then you should change directory (`cd`) to `/` before invoking the `tar` command.

If you want to place the new directory tree in some other location, then `cd` to that location now. Enter the following command:

```
tar -xzvf apache_1.3.12.tar.Z
```

If the Apache tar file is not located in the directory from which you are issuing the tar command, you will need to preface the file name with the appropriate directory path to locate the `apache_1.3.12.tar.Z` file. For example, if the tar file was located in `/download`, then the command you would issue is:

```
tar -xzvf /download/apache_1.3.12.tar.Z
```

21.1.3 Building Apache

Before you begin to build Apache, make sure that you are in the `apache_1.3.12` directory. After you are in that directory, you are ready to begin building Apache from the source tree. The normal sequence of commands to follow is:

Configure

Make

Make install

Configure is an open source tool used to build makefiles. This tool takes into account software and hardware platform dependencies; for more information on how configure works, see Figure 107 on page 390. However, in order to build Apache, nothing will need to be modified for S/390.

If you want to build a custom Apache server, you must copy the file `./src/Configuration.tmpl` to a file named `./src/Configuration` and edit the `Configuration.tmpl` file to change the `AddModule` statements. The defaults provided in the configuration template are usually sufficient for a typical installation of Apache. (In our installation, we did not modify this file.)

At this point, you can run the command:

```
./configure
```

This command builds the Makefile used to compile and link all of the components included in the Apache Web server.

Note the following sample:

```
./configure
```

```
Configuring for Apache, Version 1.3.12
```

- + Warning: Configuring Apache with default settings.
- + This is probably not what you really want.
- + Please read the README.configure and INSTALL files
- + first or at least run './configure --help' for
- + a compact summary of available options. ❶
- + using installation path layout: Apache (config.layout) ❷

```
Creating Makefile
```

```
Creating Configuration.apaci in src
```

```
Creating Makefile in src
```

- + configured for Linux platform
- + setting C compiler to gcc
- + setting C pre-processor to gcc -E
- + checking for system header files
- + adding selected modules
- + checking sizeof various data types
- + doing sanity check on compiler and options

```
Creating Makefile in src/support
```

```
Creating Makefile in src/regex
```

```
Creating Makefile in src/os/unix
```

```
Creating Makefile in src/ap
```

```
Creating Makefile in src/main
```

Creating Makefile in src/lib/expat-lite

Creating Makefile in src/modules/standard

❶ This warning is issued because we did not specify a `--prefix` on the `configure` command. The `--prefix` specifies the path under which Apache will copy files during the `make install` operation. The default location for copying files is `/usr/local/apache`. This was fine for our installation and worked quite well since it did not overlay any of the original Apache installation in the large Marist file system.

❷ The default layout is being used, since we did not specify a `--with-layout=` operand. The default layout is an Apache conforming subdirectory layout. If you would rather have a GNU conforming layout, you would specify `--with-layout=GNU` operand on the `configure` command. The subdirectory structure of various layouts can be seen by using the `--show-layout` operand on the `configure` command.

Enter the following command to compile and link Apache:

```
make
```

Note: This command will take some time to run. The length of time is dependent on the type of S/390 processor you are using, and the current execution load. You should not see any errors during the processing of the `make` command. Once this command completes, all of the programs needed to run the Apache server will have been created. All that is needed now is to copy all of the runtime components to the destination directory tree. Since we used the default during `configure`, a directory tree will be created under `/usr/local/apache`.

Enter the following command to create the Apache runtime environment:

```
make install
```

While the `make install` command progresses, messages will be written to the console indicating the operation being performed, and the files involved.

When the command completes, you will receive the following banner:

```
+-----+
| You now have successfully built and installed the
| Apache 1.3 HTTP server. To verify that Apache actually
| works correctly you now should first check the
| (initially created or preserved) configuration files
|
| /usr/local/apache/conf/httpd.conf
|
| and then you should be able to immediately fire up
| Apache the first time by running:
|
| /usr/local/apache/bin/apachectl start
| Thanks for using Apache.           The Apache Group
|                                     http://www.apache.org/
+-----+
```

The *httpd.conf* file that was copied to */usr/local/apache* during the *make install* contains a comment for the *ServerName* statement. This means that you will need to remove the comment character (*#*) from the *ServerName* statement before trying to start the Apache server. In addition, remember to include the fully qualified domain name for your server in the statement.

21.1.4 Customization

Now that your Apache server is installed, you can begin using it. The default *make install* process created a directory named */usr/local/apache/htdocs* for you. This directory is named on the *DocumentRoot* statement in the *httpd.conf* file, and it is the directory you should put HTML files into for the Web server to deliver. By default, the *make install* process created several *index.html.xx* files for various international languages. If you start your Web server at this point, you will see the following page returned to your browser:

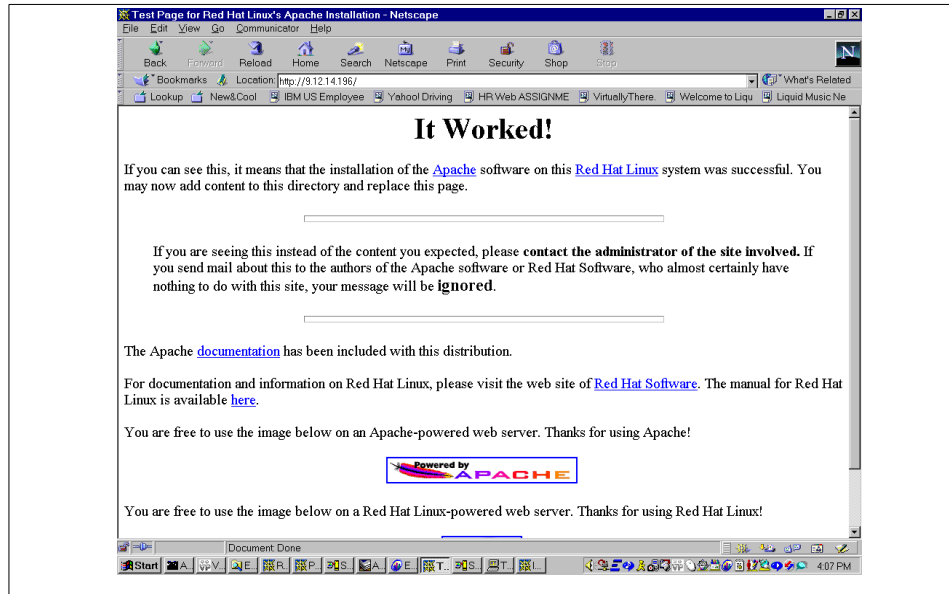


Figure 112. Apache webserver testpage

This page provides a link for you to view the Apache documentation via your Web browser.

21.1.5 Server configuration settings

Earlier releases of Apache used three configuration files to tailor the operation of the Web server. These files were:

httpd.conf

srm.conf

access.conf

These files are still provided with Apache_1.3.12, and are created in the /usr/local/apache/conf directory. However, the current recommendation is to consolidate all configuration information into httpd.conf. Consequently, the sample files created in the configuration file directory are empty.

Based upon the default settings in the httpd.conf file, your Apache server is currently capable of serving static HTML pages from port 80. In addition, the server will not restrict any clients from viewing pages in the document root directory or any subdirectories of the document root. If you would like to

change the port that Apache is listening on, you must change the *Port* directive in the `httpd.conf` file. If you would like to add restrictions on who can visit the site, you can modify the *Directory* stanzas, or create `.htaccess` files in the document directory or subdirectories.

21.1.6 Virtual hosting

Apache is capable of running multiple Web sites from a single server. The default `httpd.conf` file only defines a single main site. However, it is possible to define additional Web sites through statements in the `httpd.conf` file, with each site identified by a unique domain name.

In order for clients to access the virtual Web sites, updates must be made to Domain Name Service (DNS) servers within your network. The DNS servers need to be updated to include the new DNS names and resolve those names to the IP address of the Linux for S/390 system.

Virtual sites are defined to the Apache server using the `VirtualHost` container in the `httpd.conf` file. An example of a virtual host definition follows:

```
<VirtualHost 9.130.31.15>
DocumentRoot /usr/local/apache/newsite/htdocs
ServerName www.newsite.com
</VirtualHost>
```

In this example, the IP address on the `VirtualHost` statement is used by Apache to match an incoming request to this `VirtualHost` definition. If the IP interface over which a request is received matches `9.130.31.15`, then the `VirtualHost` definition is used to process the request. This particular `VirtualHost` definition inherits all of the configuration attributes of the main server with the exception of the `DocumentRoot` and the `ServerName`.

The `DocumentRoot` identified in the `VirtualHost` statement will be used to satisfy the client request, and the `ServerName` from the `VirtualHost` will be returned to the client.

21.1.7 Operation

The following sections discuss various aspects of Apache Web server operation.

21.1.7.1 Starting and stopping Apache

To initially start Apache and test the server, enter the following commands:

```
cd /usr/local/apache/bin
./apachectl start
```

You should see a message indicating that Apache started when the `apachectl` command completes.

You can verify that Apache is running by entering the following command:

```
ps -ef
```

Check to make sure you have multiple instances of `/usr/local/apache/bin/httpd` running.

You can also verify that Apache has a listen socket open by issuing the command:

```
netstat -vlp
```

In the response from this command, look for a line listing `httpd`. The local address column will identify the socket that `httpd` is listening on. Since the display lists well known sockets by service name rather than number, you will see `*.www` if Apache is listening on Port 80. You can verify this is the port associated with the `www` service by entering:

```
grep www /etc/services
```

To stop the Apache server enter the following commands:

```
cd /usr/local/apache/bin
./apachectl stop
```

When the `apachectl` command completes, you should see a message stating the `httpd` stopped.

21.1.7.2 Automating startup and shutdown

Now that you have tested your Apache server, you will want to make sure that the proper initialization script runs to start this new level of Apache when your Linux for S/390 system is IPLed. In addition, you will want to make sure that the proper kill script is executed when you shutdown your Linux for S/390 system.

Enter the following commands to create the proper initialization script:

```

cd /etc/rc.d/init.d ❶

cp /usr/local/apache/bin/apachectl . ❷

cd ../rc3.d ❸

rm S85httpd ❹

ln -s ../init.d/apachectl S85httpd ❺

cd ../rc0.d ❻

rm K15httpd ❼

ln -s ../init.d/apachectl K15httpd ❽

```

❶ This directory contains the application specific startup scripts called during boot and shutdown

❷ Copy the Apache startup and shutdown script to the init.d directory.

❸ Change to the directory containing scripts executed from run level 3.

❹ Remove the link to the startup script for the old level of Apache (`../init.d/httpd`).

❺ Create a new link to the `apachectl` script. (The S85 prefix indicates this is a startup script, and it should be executed after scripts numbered lower than 85, and before scripts numbered higher than 85.)

❻ Change to the directory containing scripts executed from run level 0 (halt).

❼ Remove the current kill script for the old level of Apache.

❽ Create a new link to the `apachectl` script. (The K15 prefix indicates this is a shutdown or kill script, and it should be executed after scripts numbered lower than 15, and before scripts numbered higher than 15.)

21.1.7.3 Logging

Apache has an extensive logging facility that provides information about who is accessing your site. Customizing the logging characteristics of your server can be accomplished by modifying the appropriate directives in the `httpd.conf` file. The default `httpd.conf` file specifies that an error log be produced in `/usr/local/apache/logs/error_log`. In addition, an access logfile file will be produced in `/usr/local/apache/logs/access_log` common.

21.1.8 CGI and SSI

The default server configuration includes the support modules necessary to run Common Gateway Interface (CGI) scripts, and contains *httpd.conf* entries

enabling CGI scripts. The following configuration steps were performed to enable execution of CGI scripts:

- Tell Apache where your CGI scripts are stored.
- Set up CGI handlers for specific file extensions.
- Indicate which file extensions should be considered CGI programs.

It is a good idea to create a directory that is outside of your htdocs directory tree where you can store CGI scripts. This prevents Web clients from seeing an index listing of the CGI scripts, and provides as little information as possible about your CGI scripts.

The make install process creates a directory for you for CGI scripts that is outside the htdocs directory tree. The directory is `/usr/local/apache/cgi-bin`. In addition, two sample CGI scripts are installed in the `cgi-bin` directory.

21.1.8.1 Telling Apache where CGI scripts are stored

The `ScriptAlias` directive tells Apache where in the Linux file system to locate CGI scripts when a particular URL path is specified. The format of the `ScriptAlias` directive is:

```
ScriptAlias url_path "/file/system/path/"
```

The default `ScriptAlias` included in the default `httpd.conf` file is:

```
ScriptAlias /cgi-bin/ "/usr/local/apache/cgi-bin/"
```

The following URL would execute a CGI script in this directory:

```
http://some.com/cgi-bin/aprogram
```

The default `httpd.conf` file also contains a `directory` statement that allows access to execute CGI scripts in this directory to any incoming client.

21.1.8.2 Set up CGI handlers for specific file extensions

With the default configuration described in 21.1.8.1, "Telling Apache where CGI scripts are stored" on page 415, any file referenced in the `/usr/local/apache/cgi-bin` directory will be assumed to be executable. If you would rather restrict executables to certain file extensions, then you should perform the following configuration steps:

- Remove `ScriptAlias` directives from the `httpd.conf` file.
- Create `Alias` directives to associate a URL path with the location of CGI Scripts:

```
Alias /cgi-bin/ "/usr/local/apache/cgi-bin/"
```

- Create a *Directory* container for the CGI directory:

```
<Directory /usr/local/apache/cgi-bin>
Options ExecCGI
AllowOverride None
Order allow,deny
Allow from all
AddHandler cgi-script .cgi .pl
</Directory>
```

21.1.8.3 Define file extensions that are CGI scripts

Mapping a file extension to an executable file is accomplished through changes to `mime.types`. The `AddType` directive allows you to make changes to `mime.types` without actually editing that file. For example, to specify that the extension `newcgi` is an executable file type, you would add the following statement to `httpd.conf`, or to a `.htaccess` file in a directory with option `ExecCGI`:

```
AddType application/x-httpd-cgi .newcgi
```

If the statement is added to `httpd.conf`, then it will apply to every directory that contains the `ExecCGI` option. If the statement is added to a directory-specific `.htaccess` file, then it will apply only to that directory.

21.1.8.4 Server Side Includes (SSI)

Server Side Includes are HTML pages that include embedded commands for the Web server. SSI provides a mechanism to create dynamic HTML pages without the need to write CGI scripts. However, an SSI-enabled Web page requires that the Web server parse the complete content of the page looking for the embedded commands. This can introduce performance problems

The default `httpd.conf` we used is not configured to enable SSI HTML pages. In order to enable the use of SSI pages you must perform the following configuration steps:

1. Enter `/usr/local/apache/bin/httpd -l`. (Look at the resulting list of modules included in the Apache server to ensure that `mod_include` is present.)
2. Add a new handler for SSI/IXSSI HTML pages. (This is typically done in the context of a `Directory` container.)
3. Add a new file extension for SSI/IXSSI HTML pages. (This is also typically done in the context of a `Directory` container.)
4. Enable SSI parsing for a directory. (This will contain the parsing operation to a particular portion of the site directory tree.)

```
<Directory /usr/local/apache/htdocs/myssi>
AddHandler server-parsed .shtml
AddType text/html .shtml
Options +Include
</Directory>
```

In this example, the directory myssi has been defined with the Include option to enable SSI parsing. In addition, the file extension .shtml within this directory is to be considered of mime type text/html, and processed by the mod_include module.

21.1.9 SSL

The Apache Web server we used does not have SSL function included. If you're interested in transforming your apache_1.3.12 Web server into a secure Web server that implements SSL, consult the following Web site:

<http://www.apache-ssl.org>

This Web site describes Apache-SSL, a secure Web server based upon Apache and SSLeay/OpenSSL. This code is licensed under a BSD-style license, and is available without charge.

From this site (or one of the mirrors identified) you can download the required patches to the Apache 1.3.12 source, as well as SSLeay version 0.5.1b+ and OpenSSL. Once the patches are applied to Apache source, you will need to link the object modules with either SSLeay or OpenSSL:

- SSLeay is an implementation of Netscape's Secure Socket Layer protocol. It is available without charge, and provides support for SSL V2, SSL V3, and TLS V1.
- OpenSSL is a separate project based upon the SSLeay library previously described.

Since SSL incorporates cryptographic routines, it is important to understand all of the legal issues associated with cryptographic routines. These rules apply to uses of these products outside the United States.

During this project, we did not acquire the patches or SSL implementations to create an Apache-SSL server.

21.1.10 Web Server security considerations

Once you have your Web server installed and configured, it is useful to spend some time considering how you can make it secure.

Outside access to the content provided by your Web server can be controlled by the access control statements in `httpd.conf`. In previous releases of Apache, these directives were typically placed in a file named `access.conf`. However, later versions of Apache have consolidated these statements into `httpd.conf`, leaving the `access.conf` file as a pointer to the moved statements.

The typical security scheme is to begin by defining a very restrictive set of permissions for the `/` directory. Following this, additional statements are provided that allow particular feature and access authorizations. The reader is directed to the documentation that accompanies Apache for the complete syntax of the set of statements that authorize access and features.

Within the `httpd.conf` file, there are several directives that you will need to be careful with, as described in the following sections.

21.1.10.1 ExecCGI

This option specifies that CGI programs can be executed within the directory hierarchy associated with this option. Unless the CGI programs are carefully controlled, it is possible that security holes could be opened due to poor coding practices. If CGI programs are needed, the best practice is to restrict CGI programs to a specific directory that is outside the hierarchy of your `httpd` root file system. If this directory is access controlled, then only the Web Master, or another authorized person, may add or change CGI programs. The `ScriptAlias` directive is used to define the particular directory you want to use for CGI programs.

21.1.10.2 FollowSymLinks

Symbolic links are small files within a Linux file system that point to the location of another file. When a symbolic link is accessed, it behaves as though the user accessed the real, referenced file. Often a symbolic link is created as a shorthand reference to a file that includes a long path specification.

The Apache `FollowSymLinks` option allows remote users to follow symbolic links in the directory they are referencing by clicking the associated hyperlinks. The security exposure here occurs when someone inadvertently links to an important internal system file. This would allow a remote user to violate the barrier that separates the Web hierarchy from the system file hierarchy.

21.1.10.3 The Includes option

Server Side Includes (SSI) allow for the inclusion of dynamic information in otherwise static HTML documents. This is a convenient method of creating

dynamic documents without writing CGI programs. Most of the SSI directives do not introduce any serious security exposures.

However, one in particular should be avoided: the EXEC directive. This directive allows you to specify system commands within your source HTML such as:

```
<!--#exec cmd=" ls -l /"-->
```

This will produce a directory listing in the HTML output stream.

If the HTML page also has a form that takes user input, an attacker could download the HTML source, insert malicious exec commands, and then submit the form. Your server would process the form and unwittingly execute the commands specified with the exec directive.

To avoid this situation, it is recommended that you not allow the exec cmd directive, if SSI commands are allowed. Specify the following to accomplish this:

```
Options IncludesNOEXEC
```

21.1.10.4 Directory indexing

We recommend that you *do not enable* the directory indexing option because this option causes Apache to send a directory listing (similar to an `ls -l` command) if a default HTML page is not found.

Typically the default Web page is defined to be `index.html` or something similar. If a non-specific URL is received (that is, a URL that only identifies a directory path), and Apache cannot find the default HTML page, a directory listing will be returned instead.

This potentially allows a Web user to browse the list of files in a particular directory, and navigate between directories. If you have included files in the directory that are not usually referenced by any HTML page, the directory listing will expose the files to Web users.

Chapter 22. Firewall configuration

In Chapter 14, “Linux TCP/IP connectivity” on page 267, and Chapter 15, “Linux for S/390 connectivity to VM, OS/390, VSE” on page 283, various methods for connecting a Linux for S/390 system to a network were described. However, if you dedicate a real communication adapter to your Linux for S/390 system, you’ll need to protect your system if that adapter attaches to an external network such as the Internet.

Linux for S/390 can function as a packet-filtering firewall. This means that you will be able to use your Linux for S/390 system to prevent packets from leaving your system and going to certain destinations, as well as prevent packets from entering your system from sites on the external network.

A packet-filtering firewall typically operates at the IP network and transport protocol layers. In addition it is usually implemented within an operating system, as in the case of Linux. The basic way in which this type of firewall protects a system is by making routing decisions after filtering packets using information in the IP packet header. The firewall allows you to either discard a packet and notify the sender of this action, or simply discard a packet without notification.

Since a packet-filtering firewall operates at the IP network and transport layers, it is important to understand the limitations of such a firewall. The packet-filtering firewall makes decisions based upon the network interface and host IP address over which a packet was received, the source and destination IP addresses, the TCP or UDP ports, the TCP connection flags, the ICMP message types and whether the packet is incoming or outgoing.

However, this type of firewall does not have the ability to verify that the sender is who they claim to be. The only identifying information is the source IP address, which can be spoofed. In addition, these layers cannot verify that the application data is correct. The higher level protocols need to perform these types of checks. Still, a packet-filtering firewall does provide a high degree of control over direct port access, and what packets can actually be passed to higher level protocols on the system. Thus it is a very valuable component in the overall security scheme you will implement.

The packet-filtering firewall on your Linux for S/390 system is known as *ipchains*. This name relates well to how the firewall operates. The firewall performs input and output filtering by applying lists of rules to individual packets. The lists of rules are defined as chains because a packet is matched against each rule in the list until a match is found or the list ends. For a

particular network interface, both an input chain and an output chain are maintained.

The Linux ipchains package is based upon the Linux IPv4 firewall code, and represents a rewrite of the older ipfwadm. The minimum kernel version supported by ipchains is 2.1.102. It was at this point that the firewall code was integrated into the mainstream Linux kernel.

In addition to the code in the kernel, you also need a tool named ipchains that communicates with the kernel to define the packet filtering rules (or chains). This tool can be obtained from the following Web site in rpm format and installed on a Marist College-based system:

<http://linux.s390.org>

22.1 Installation

Since the packet filtering code is integrated into a 2.1.102 and above kernel, you do not need to “install” the firewall code. However, most kernels (including the Marist Linux for S/390 binary kernel) are not built with the firewall code activated. You can easily determine if your code has the ipchains firewall code activated by entering the following command:

```
ls /proc/net/ip_fwchains
```

If the response to this command indicates the file exists, then your kernel was built with the firewall code activated. If this file does not exist, then you will need to rebuild your kernel, specifying the following three options when you perform the make config:

```
CONFIG_FIREWALL=y  
CONFIG_IP_FIREWALL=y  
CONFIG_IP_ROUTER=y
```

The third configuration statement listed above is not absolutely necessary for running the firewall code or ipchains. However, if you are configuring your Linux for S/390 system to act as a firewall, you will presumably also want it to function as an IP router. This kernel option adds the high performance router function.

Rebuild your kernel following the steps outlined in 11.6, “Build and customize the kernel” on page 238.

Once your kernel has been built with the firewall code active, you also need the ipchains tool in order to define filtering rules.

The current version available in source form is 1.3.8. Source can be downloaded from the following URL:

<http://www.adelaide.net.au/~rustcorp/ipfwchains/ipfwchains.html>

A Linux for S/390 binary rpm is available from the following URL:

<http://linux.s390.org>

The level of ipchains available from this site is 1.3.9-3. For this residency, we obtained the s390 binary from linux.s390.org and installed the rpm package:

```
rpm -ivh ipchains-1.3.9-3_s390.rpm
ipchains                               #####
#####
rpm -qlp ipchains-1.3.9-3_s390.rpm
/sbin/ipchains
/sbin/ipchains-restore
/sbin/ipchains-save
/sbin/ipfwadm
/sbin/ipfwadm-wrapper
/usr/doc/ipchains-1.3.9
/usr/doc/ipchains-1.3.9/COPYING
/usr/doc/ipchains-1.3.9/HOWTO-1.html
/usr/doc/ipchains-1.3.9/HOWTO-10.html
/usr/doc/ipchains-1.3.9/HOWTO-2.html
/usr/doc/ipchains-1.3.9/HOWTO-3.html
/usr/doc/ipchains-1.3.9/HOWTO-4.html
/usr/doc/ipchains-1.3.9/HOWTO-5.html
/usr/doc/ipchains-1.3.9/HOWTO-6.html
/usr/doc/ipchains-1.3.9/HOWTO-7.html
/usr/doc/ipchains-1.3.9/HOWTO-8.html
/usr/doc/ipchains-1.3.9/HOWTO-9.html
/usr/doc/ipchains-1.3.9/HOWTO.html
/usr/doc/ipchains-1.3.9/HOWTO.shtml
/usr/doc/ipchains-1.3.9/HOWTO.txt
/usr/doc/ipchains-1.3.9/Makefile
/usr/doc/ipchains-1.3.9/README
/usr/doc/ipchains-1.3.9/ipchains-quickref.ps
/usr/man/man4/ipfw.4
/usr/man/man8/ipchains-restore.8
/usr/man/man8/ipchains-save.8
/usr/man/man8/ipchains.8
/usr/man/man8/ipfwadm-wrapper.8
```

As you can see from the rpm -qlp command, the binary package consists of several modules that are placed into the /sbin directory, and a number of documentation files. The /sbin/ipfwadm, and /sbin/ipfwadm-wrapper programs

are provided for compatibility with the old Linux IP Firewall code. The ipchains tools replace the ipfwadm tool, and thus should be used instead.

The /sbin/ipfwadm-wrapper tool can be used as a quick means of upgrading a system which was using ipfwadm for firewall rule specification.

Even though you created a new kernel specifying CONFIG_IP_ROUTER, you must still update the network script to permanently enable IP forwarding. This can be done by editing the file /etc/sysconfig/network:

```
Change FORWARD_IPV4=no to FORWARD_IPV4=yes
```

This change will take effect the next time you boot your system. If you want the change to take effect immediately, you must also execute the following command:

```
echo "1" > /proc/sys/net/ipv4/ip_forward
```

22.2 Customization

Now that your system is configured to use ipchains, it is time to define the rules that will be used by the firewall code. There are two basic approaches that you can take when defining filtering rules:

- Deny all incoming and outgoing packets, then specifically enable packets associated with services you want to allow.
- Allow all incoming and outgoing packets, then specifically deny packets associated with services and locations you want to restrict.

The first approach is generally considered to be the most effective method for protecting a system. This method does not leave your system unprotected if you overlook some service nuance that would allow packets to flow that should be restricted. Rather, you may encounter the situation where a particular service does not work that should. You can at this point define the particular rules that remedy the situation.

The firewall code within the kernel works in the following manner:

- When a packet is received on a particular interface, the rules in the input chain will be checked against the information in the IP header of the packet.
 - If a matching rule is not found, then the default policy for the chain is applied to the packet. If we choose the “deny all” approach, then the packet will be silently discarded (in the case of DENY), or discarded and an ICMP message returned to the sender (in the case of REJECT).

- If the packet passes the input check, it is then examined to see if it is destined for the local machine, or needs to be routed to another machine.
 - If the packet needs to be routed, then it is compared against the rules in the forward chain. Again, if the default policy is “deny all”, then the information in the packet header must match one of the rules in the forward chain in order to proceed.
 - If a match is found, then the kernel will compare the packet header information with the rules in the output chain. Packets generated from the local machine would be compared against the rules in the forward chain, and the rules in the output chain.

Rules are defined to the kernel using the `ipchains` utility. These rules are dynamically defined, and persist only for the life of a kernel boot. In order to make these rules “permanent”, an initialization script must be created that will redefine the rule set each time you boot the system. Utility programs are included in the `ipchains` package that allow you to save the set of chains currently in use by the kernel to a file, and restore the saved chains from a file to the kernel.

By default, the kernel maintains three chains. These chains are named `input`, `output` and `forward`. It is possible to clear all of the rules from these chains, but it is not possible to delete these built-in chains. In addition to these built-in chains, the `ipchains` utility allows you to create your own chains with names you choose. These user-defined chains can then be associated with a particular built-in chain for use by the kernel. In effect this provides a convenient mechanism for grouping sets of rules, and assigning a name to them.

When rules are defined for various chains, it is also possible to specify whether or not to log packets that match the particular rule. While this might produce an unmanageable amount of output for routine packets, it might be highly desirable to log packets that match rules designed to catch unusual attempts to break into your system.

Since the logging is done by the kernel, the output is typically captured by `klogd` and transferred to `syslogd` for writing. The `syslogd` facility is customized using the `/etc/syslog.conf` file. Within this file, stanzas define where and how to log entries from particular facilities and levels. In the case of `ipchains`, the facility involved is the `kernel`, and the level associated with `ipchains` is `info`.

On a standard Marist system, the `syslog.conf` file specifies that kernel messages should be logged to the console (`/dev/console`), and `info` messages

should be logged to /var/log/messages. Thus all ipchains logging would be displayed both on the Linux console, and in the /var/log/messages file.

The ipchains utility allows you to perform operations on an entire chain, or on a single rule. Some of the common operations you can perform on an entire chain are:

- Create a new chain (-N)
- Delete an empty chain (-X)
- Change the default policy for a built-in chain (-P)
- List the rules in a chain (-L)
- Flush the rules out of a chain (-F)

For example, you might want to list the rules on your built-in chains before you begin defining additional rules. To do this, you would enter:

```
ipchains -L input
ipchains -L output
ipchains -L forward
```

Since you have not yet defined any rules, the output of this command will simply list the default policy for the particular chain (which begins as ACCEPT). Since the default policy is ACCEPT for all chains, this means that currently your firewall is not blocking any incoming or outgoing packets.

You can change the policy for any of these chains by using the -P option of ipchains. For example, to implement the “deny all” policy for input, you would enter:

```
ipchains -P input DENY
```

At this point, all incoming packets will be silently discarded. Processes trying to send packets to your machine will simply time out, not receiving any other error indication. Since presumably you will want to allow some set of packets into your machine, you will now need to add some additional rules to the input chain to specify the particular packets you want to accept. The ipchains command allows you to manipulate the rules associated with a chain in the following ways:

- Append a new rule to the chain (at the end of the chain) (-A)
- Insert a new rule at some position in the chain (-I)
- Replace a rule at some position in the chain (-R)
- Delete a rule (either the first one that matches or at a particular position within the chain) (-D)

When you delete a rule, you can identify the rule to be deleted by either specifying its numeric position within the chain (beginning with 1), or by specifying the entire text of the rule to be matched.

For example, suppose you had appended a rule to the input chain as follows:

```
ipchains -A input -s 127.0.0.1 -p icmp -j DENY
```

You could delete the rule by simply changing the `-A` to a `-D`. This would cause a scan of the rules on the input chain for a matching rule. The first one found would be deleted.

Both the insert and replace commands require that you identify the rule by a numeric rule number. In the case of insert, the number specifies the number of the new rule within the list. If you already had a chain of rules and then you specified `ipchains -I input 1 ...`, this would cause the new rule to be placed at the beginning of the list.

It is possible to find the numeric position of a rule within the chain by using the list command with the `--line-numbers` option:

```
ipchains -L input --line-numbers
```

In a rule specification, you will tell the kernel the set of conditions the packet must meet, and what to do if the packet meets the specifications. The set of conditions you specify covers most of the significant fields in an IP packet header. You can specify:

- Source and Destination IP addresses (`-s` or `-d`)
- Protocol number (`-p`)
- UDP or TCP port numbers
- ICMP message type and code (`-s` and `-d`)
- Interface name (for input or output) (`-i`)
- TCP packets with only the SYN flag set (`-y`)

Telling the kernel what to do with a packet that matches a rule is done by specifying a target for the rule. Targets are specified using the `-j` parameter (as in jump to the target specification). The following targets can be specified with the `-j` parameter:

- ACCEPT
- DENY
- REJECT
- REDIRECT
- RETURN

We have already discussed the difference between DENY and REJECT. ACCEPT is specified when you want to allow the packet to continue through IP processing. REDIRECT is an uncommon target specification that applies only to packets using the TCP or UDP protocol. REDIRECT instructs the kernel to send the packet to a local port instead of the destination specified in the packet. RETURN causes an immediate end to comparison with rules in the chain, imposing the default policy on the packet.

As an example, the following rules would resume packet flow over the loopback interface (since we earlier set a default policy of DENY for both input and output chains):

```
ipchains -A input -i lo -j ACCEPT
ipchains -A output -i lo -j ACCEPT
```

You activate logging for a particular rule by include the -l flag in the rule. This will cause log entries to be generated when a packet matches the particular rule.

If you create rules that use the -s option to specify a source IP address or range of addresses, you need to be aware that it is possible to spoof IP addresses and thus defeat the rules. The Linux kernel offers some protection against address spoofing through the activation of source address verification. The following shell script routine will enable source address verification for all interfaces on the Linux system:

```
for f in /proc/sys/net/ipv4/conf/*/rp_filter; do
    echo 1 > $f
done
```

Since this chapter is not intended to be a comprehensive work on firewall design, you are encouraged to look at the many publications that deal with this subject in depth. Creating the rules for the various chains supported by the Linux firewall code involves analyzing all of the TCP/IP applications and services offered on the firewall machine and determining the packet characteristics associated with each service and application. You also need to consider the access you want to allow both into and out of your firewall machine.

One subject which is beyond the scope of this chapter but integral to the firewall code and ipchains, is the capability of the firewall code and ipchains to support Network Address Translation or IP-Masquerade. This capability allows you to utilize local IP addresses in your network behind the firewall machine, and have a registered IP address used for communication between hosts on the local network and applications on the Internet.

In addition to the IP-Masquerade capability, you will also want to investigate the use of proxy services as an integral part of your firewall strategy.

22.3 Operation

As mentioned earlier, the `ipchains` command makes dynamic changes to the chains maintained by the kernel. Since this is the case, you must take steps to insure that the firewall policy you want to implement is captured in shell scripts invoked when the system boots, and perhaps when an interface is started or stopped.

Your shell script should begin by flushing any existing rules, so that it can be executed at any time:

```
ipchains -F
```

Next, you should implement the default policy for each chain:

```
ipchains -P input DENY
ipchains -P output REJECT
ipchains -P forward REJECT
```

At this point, all traffic is blocked. You could then proceed to allow unlimited traffic on the loopback interface (since this is completely local to the firewall machine). Following this, you should ensure that `tcp_syncookies` support in the kernel is enabled:

```
echo 1 > /proc/sys/net/ipv4/tcp_syncookies
```

You will also at this point want to enable source address verification for each interface:

```
for f in /proc/sys/net/ipv4/conf/*/rp_filter; do
    echo 1 > $f
done
```

You may then proceed to specify rules that allow certain packets to be **ACCEPTED** for the services and activities you want to support.

The firewall script you build can be easily integrated into the system boot process by adding a line at the end of the `/etc/rc.d/rc.local` file:

```
sh /etc/rc.d/rc.firewall
```

This statement presumes that the name of your shell script to define the `ipchains` rules is `rc.firewall`, and that the script resides in `/etc/rc.d`

If you are building your security policy interactively using the `ipchains` command. You can easily save the current state of your chains using the `ipchains-save` utility. To create a file containing the current rules enter:

```
ipchains-save > the_firewall
```

This will create a file named `the_firewall` in the current working directory containing `ipchains` commands to define all of the rules currently defined. If you flush the chains, or reboot the system, you can easily put these rules back using the `ipchains-restore` utility:

```
ipchains-restore < the_firewall
```

Chapter 23. Printing with Linux

In this chapter we describe how to print with Linux for S/390 by sharing a printer on a different system.

23.1 Devices

As mentioned, with Linux for S/390, our approach to printing was to share a printer (queue) on a remote system.

We first installed the package `lpr-0_48-1_s390.rpm`, which provides the main functions for using printers on Linux for S/390:

- `lpc` A management function to deal with printers and queues.
- `lpd` The line printer daemon; it works on queues and writes to a printer.
- `lpr` The print command to send a file for printing.

We downloaded the package from:

<http://www.linux.s390.org>

Downloading was done directly into a working directory on our Linux for S/390 server using Samba, as shown in Figure 113.

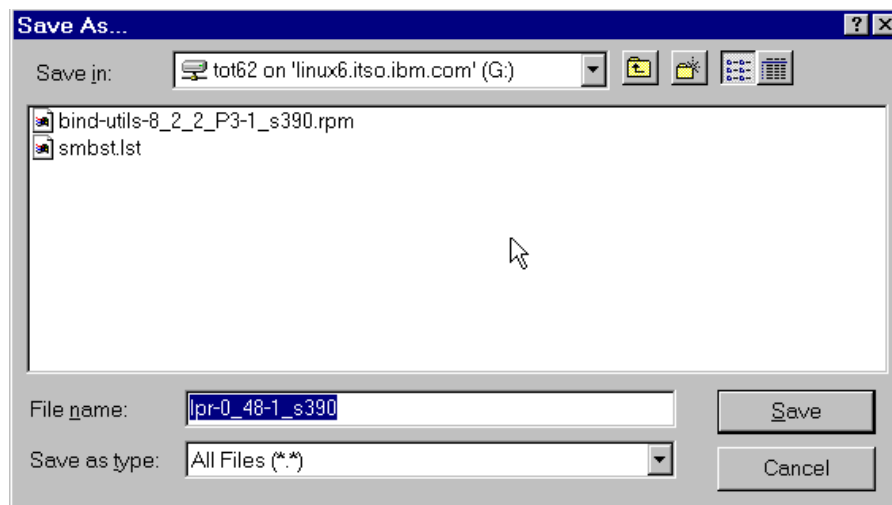


Figure 113. Downloading from the Internet onto the Linux server

These services were installed using the rpm package install function. No source was installed with the binaries.

23.2 Using VM resources

We tested printing from Linux for S/390 to a VM-managed IBM Network Printer 24.

One test used the `lpr` command on Linux for S/390 to print a flat file. Printer output was sent from Linux for S/390 to the TCP/IP LP daemon (LP SERVE) on VM/ESA, which routed it to an RSCS LPR link. The RSCS LPR driver sent the output to the LPD daemon inside the Network Printer 24. A second test used `lpr` to print a PostScript file to the same printer. In practice one would never take such a circuitous route, but it serves to demonstrate the flexibility of using VM as a centralized print server.

Figure 114 illustrates the routing.

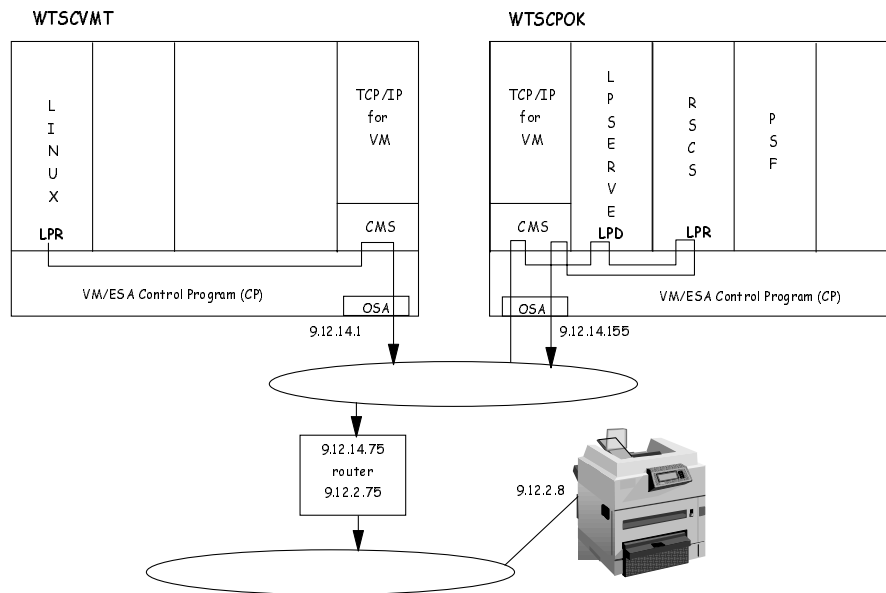


Figure 114. Printing from Linux to VM-managed printers

Interoperation between the VM TCP/IP printing services and RSCS printer drivers and queue managers allows almost any IP printer, SNA printer, AFP printer, or system printer to be accessed from Linux.

To enable remote printing from Linux for S/390 to printers managed by VM, we needed to make the following definitions in Linux:

In `/etc/printcap` we made an entry for the Network Printer 24:

```
np24:rm=wtscpok.itso.ibm.com:rp=poknp24:sd=/var/spool/np24
```

A label `np24` was assigned to the remote printer. The VM system from which we managed the printer was WTSCPOK. The name of the printer in the VM TCP/IP LPD definitions was `poknp24`. The spooled output from the `lpr` command was stored under the `/var/spool/np24` directory.

We printed the sample files as follows:

```
lpr -Pnp24 /etc/printcap
lpr -Pnp24 ttosvu.ps
```

Because the VM LPD and RSCS definitions are complex, we do not show them here. For information on configuring RSCS and IP printer networks, refer to *VM/RSCS V3R2.0 Planning and Installation*, SH24-5219 and *VM/ESA V2R4.0: TCP/IP Function Level 320 Planning and Customization*, SC24-5847 and *TCP/IP Solutions for VM/ESA*, SG24-5459.

We also conducted a successful test of printing a flat file to an IBM 3130 printer via the TCP/IP LP daemon on VM LPSERVE, which routed the data stream to be printed on the 3130 under the control of Print Services Facility/VM (PSF/VM).

23.3 Using OS/390 resources

On one of the OS/390 systems (WTSC43OE) we had IPPrintway up and running. To use the print resources on that host we added a definition to `/etc/printcap`:

```
# This file can be edited with the printtool in the control-panel.
# printer q poke via wtsc43oe.ibm.com (HDM)
poke:rm=wtsc43oe.itso.ibm.com:rp=poke:sd=/var/spool/poke
```

The UNIX printer daemon requires a spool directory to be defined according to the definition in `/etc/printcap`. We allocate this with a simple `mkdir /var/spool/poke` command.

The `lpd` was not running, so we started it by using `lpd start` in `/etc/rc.d/init.d/`.

Following is the print command we used:

```
lpr -Ppoke services
```

The data was sent immediately to the host. From the messages in the OS/390 syslog you can see the actions taken by the print management software:

```
AOPD00I BPXAS AOPLPD (JOB32651) spool data for: WTSCPLX9 ROOT
PS065836 SERVICES from: root@linux6
IAT7001 JOB ROOT (PS065836) IS ON WRITER PRTWAY( ),RECORDS=1
ANFM700I Data set: WTSCPLX9.ROOT.ROOT.JOB32651.D0000015.SERVICES The
data set has been acquired by PRTWAY
*IAT7005 WTR (JOB00219), ON PRTWAY ( ), WAITING FOR WORK.
IAT7005 WTR (JOB00219), ON PRTWAY ( ), WS=(CL), WC=J.
ANFU141I Job=65836 Owner=ROOT Printer=PRTWAY Data=JOB32651.D0
000015 At=WTSCPLX9 -- Sent for root@linux6
ANFM601I Data set: WTSCPLX9.ROOT.ROOT.JOB32651.D0000015.SERVICES The
388
data set was successfully transmitted to host and
queue: 9.12.2.4 afccu2
ANFM604I Data set: WTSCPLX9.ROOT.ROOT.JOB32651.D0000015.SERVICES The
data set is being released to JES
```

The file we used for this was a plain ASCII file. A further test with a postscript file worked immediately with the same printer, as the server detects postscript files automatically.

23.4 Using Linux as a print data hub

Linux provides many *filter* services to convert printer data streams, mainly into postscript. The availability of these filters can make Linux very attractive in environments where various printer types are used and printer consolidation is required.

For example, you may have diverse HP, Canon, Epson or Lexmark printers in your shop. These are useful as *personal* printers to print small amounts of data. However, when you're planning to print a larger number of copies, you can use Linux filter functions to convert between these data streams and postscript.

Chapter 24. Linux security issues

In this chapter we merely scratch the surface of a few security-related issues. The URLs referenced in other sections of this book point to more complete security information.

You should be aware that millions of Linux installations exist that are connected to the Internet, and among its operators there are inevitably some that have the knowledge and intention to exploit security holes on remote systems. Therefore, “security by obscurity” is no longer an option.

Some Linux distributions offer a selection of security profiles for certain situations such as “in trusted environment”, “DMZ” (for DeMilitarized Zone), or “paranoia setting”. Starting from the most appropriate of those for your environment, you can reach the required level of security with only a few additional steps specific for the actual environment.

Note that the Marist distribution offers virtually no security as it is intended to allow experimentation and exploration of this Linux for S/390 system.

24.1 Consider using remote logging

Important system messages and warnings are usually logged in `/var/log/messages` and `/var/log/warn`, respectively. An attacker who manages to gain unauthorized access to the system is likely to be able to remove his traces in these (and other) files.

However, the ability to send the messages to be logged to another machine helps with this problem. The following man page gives instructions and examples for the configuration of `syslogd` for remote logging:

```
man syslog.conf
```

24.2 Disable unnecessary services

Flaws in the code of programs that run with root privileges (like several daemons for networked services) have repeatedly given attackers ways to gain unauthorized access to UNIX systems. Though the number of such incidents may decrease, it is certain that such things will happen as long as computers offer networked services. The system administrator should therefore follow the online forums that report known software weaknesses and possible fixes.

The more services are running on a system, the greater the probability that some program failure will produce a security problem. The simple prescription is to disable all services that are not necessary for the operation of the system. Many daemons are started by the “super daemon” inetd: commenting out the respective entries in /etc/inetd.conf by putting an # in front of the line and restarting inetd by the following command does it:

```
killall -HUP inetd
```

Other daemons are running all the time (listening on their port); so you should simply not start unnecessary ones.

24.3 Files and file system security

A few minutes of preparation and planning ahead before putting your systems online can help to protect them and the data stored on them.

World-writable files, particularly system files, can be a security hole if a cracker gains access to your system and modifies them. Additionally, world-writable directories are dangerous, since they allow a cracker to add or delete files as he wishes. To locate all world-writable files on your system, use the following command:

```
root# find / -perm -2 ! -type l -ls
```

Be sure you know why those files are writable. In the normal course of operation, several files will be world-writable, including some from /dev, and symbolic links, thus the ! -type l which excludes these from the previous find command. Unowned files may also be an indication an intruder has accessed your system. You can locate files on your system that have no owner, or belong to no group with the command:

```
root# find / -nouser -o -nogroup -print
```

Finally, before changing permissions on any system files, make sure you understand what you are doing. Never change permissions on a file because it seems like the easy way to get things working. Always determine *why* the file has that permission before changing it.

For additional information, see the following articles at the referenced Web sites:

- Linux Security Administrator's Guide by Dave Wreski
<http://www.nic.com/~dave/SecurityAdminGuide/SecurityAdminGuide.html>
- Linux Administrator's Security Guide by Kurt Seifried
<http://www.securityportal.com/lasg/>

24.4 Disable remote login for root

If remote login for root is not allowed, the only way to become the super user is by logging on with the personal account and then switching to root by using the “su” command. Even if the root password should become known to some attacker, he will need to get hold of another account on which he can log in remotely.

The device names of the ttys from which root can log in are listed in `/etc/securetty`.

If an FTP server is running on your system, `/etc/ftpusers` lists users who are not allowed to connect via FTP. Root and some other users should definitely be in this file; to get more information, check out the man pages (`man ftpusers`).

If you are running Linux for S/390 in a virtual machine, you may decide to limit root logons to the Linux console. In other words, only by logging onto the Linux virtual machine or by being an authorized secondary user could you attempt to log on to Linux as root.

24.5 Use encrypted connections

If an attacker has access to the physical network segment (e.g., by having a login account on a computer directly attached to the network) *he can access all information that is transmitted on the segment*. By using simple scripts on top of a “network sniffer” program (like `tcpdump`), he can filter out login/password pairs transferred in clear text. The only protection against this is not to transfer any unencrypted data.

Therefore, you should disable telnet and rlogin (`/etc/inetd.conf`), and allow remote login only via ssh. Transfer files only with the `scp` command.

24.6 Use scp instead of FTP

As data transferred by FTP is not encrypted and implementations of the FTP service have shown vulnerabilities in the past, one might ask for an alternative method of file transfer. Luckily there is `scp` (secure copy), which allows you to copy files between machines and uses the authentication mechanism of the secure shell. The data is encrypted during transfer by `scp`.

`scp` understands a syntax that is very similar to that of the usual `cp` command:

```
scp myfile.txt joe@host.mydomain.com:somedir/
```

This means copy myfile.txt into joe's home directory under the directory somedir/.

24.7 Use a tcp wrapper (tcpd)

The tcpd wrapper monitors, logs, and controls the startup of daemons normally started directly by inetd. It does some checks for apparent attempts to mount an attack (like a host pretending to have the name of another) and if nothing suspicious is detected, starts the service.

Linux distributions will likely install tcpd by default. If your system doesn't have tcpd installed, you should install it.

The configuration of tcpd is nicely explained in `man tcpd`. Most likely you should make entries in `/etc/hosts.allow` and `/etc/hosts.deny`, where access for hosts can explicitly be allowed or denied (you can issue the command `man 5 hosts_access` for additional information).

24.8 Use shadow passwords

The user (and group) passwords are not stored as clear text anywhere in the system. Instead, the passwords are encrypted and authentication is done by encrypting the entered passwords and comparing them to the stored encrypted passwords.

The encrypted passwords used to be stored in `/etc/passwd`, which is (and has to be) world-readable. In this situation, any user can obtain the list of all logins and encrypted passwords on the system. By using certain programs (like cracklib) that use a dictionary of frequently used passwords (and variations of the login sometimes used as password by users not aware of the risks), it is feasible to obtain passwords and thereby unauthorized access.

With the shadow password suite installed, the passwords are stored in a file (`/etc/shadow`) readable only by root, thereby eliminating the possibility that ordinary users could use cracklib or other dictionary attacks.

All Linux distributions today come with the shadow suite.

24.9 X11 server access control

Working with X11, you should be aware that anyone who is allowed to connect to the X server can launch attacks that render the server useless (e.g. by modifying the key map) or start keyboard sniffer programs (that record all keystrokes on the machine where the server runs).

After logging on to a remote machine, you usually have to set the DISPLAY variable (on the remote machine) according to the X terminal you work with:

```
export DISPLAY=myhost.mydomain.com:0
```

Attention

Suppose you try to start an X application and receive a message like the following once too often:

```
Xlib: connection to "myhost.mydomain.com:0" refused by server
Xlib: Client is not authorized to connect to server
```

You might be tempted to just allow connections from all hosts by entering this (on the X terminal):

```
xhost +
```

instead of typing:

```
xhost +myhost.mydomain.com
```

for each individual host that will connect to the X server. However, you should refrain from doing so unless your X terminal has no connection to any untrusted host!

In any case, xhost (host-based access control) is really not the end of the story: the very readable man page Xsecurity will point you to xauth (client-based access control); see also `man xauth`.

24.10 Consult the security-related Internet sites regularly

Formerly unknown security problems are first described on the Internet. Often an appropriate fix is published at the same time. If you care about security, then visit the security sites regularly or subscribe to the mailing lists run by them.

The following list will give you some good starting points:

<http://www.securityfocus.com/>
<http://rootshell.com/>
<http://www.insecure.org/>
<http://www.cert.org/>
<http://lsap.org/>

Linux distributors often have Web sites with information specific to the software packaged for their distribution, as you can see in the following URL:

<http://www.suse.de/de/support/security/>

Chapter 25. Sources of help and information

This chapter points you to further sources of information about Linux on S/390.

25.1 man pages

Manual (known as *man*) pages are installed on every UNIX (Linux) system. They are read using the `man` command; you can enter the following to obtain information about how to use the `man` command:

```
man man
```

Man pages are divided into the following nine sections:

1. Executable programs or shell commands
2. System calls (functions provided by the kernel)
3. Library calls (functions within system libraries)
4. Special files (usually found in `/dev`)
5. File formats and conventions eg `/etc/passwd`
6. Games
7. Macro packages and conventions (e.g. `man(7)`, `groff(7)`).
8. System administration commands (usually only for root)
9. Kernel routines [Non-standard]

Usually there is only one man page for the name you enter. However, some names have man pages in two or more sections; for example, the following command displays all man pages for the keyword “write”:

```
man -a write
```

In this case there were two `man` pages: one in section 1, and one in section 2. The following command explicitly selects the man page from section 2:

```
man 2 write
```

Reference to a man page in a particular section may look like “*see man write(2)*”.

Apart from the classification into nine sections, the collection of man pages is relatively unorganized. The `apropos` command searches the manual page names and descriptions for a keyword.

For example, you might want to search for man pages related to the Samba daemon `smbd` as follows:

```
$ apropos smbd
smbd          (8) - server to provide SMB/CIFS services to clients
testprns     (1) - check printer name for validity with smbd
```

The results show that besides smbd, the testprns man page is related.

25.2 info

The structure of info is different from man in that it has a tree structure through which you may be able to navigate, depending on which program you use for browsing the info pages.

GNU textinfo is invoked with:

```
info
```

This produces a black-and-white text display. In contrast, the following is an info reader that adds color and lynx-style navigation (right arrow to follow link, left arrow to go to upper level), but does not have name completion:

```
pinfo
```

However, pinfo is not part of the Marist large file system, nor does it appear to be available on the Thinking Objects rpm Web site:

```
ftp://linux.s390.org/pub/ThinkBlue/RPMS/s390/
```

Emacs is pretty comfortable for reading info pages. If you use emacs, you can use it as your info browser by typing `ctrl-h i` inside emacs. If you prefer vi, you'll use plain info anyway. You can also read the html versions of the info pages with your favorite browser.

25.3 help

`help` is a shell *built-in* (that is, a command that is part of the shell) and gives help about the other functions that are shell built-ins. You invoke it by entering:

```
help
```

Among the different shells, only bash offers `help`; others usually say "help: not found."

25.4 howto

Howtos should reside in the directory `/usr/doc/howto`. They should exist in (gzip compressed, plain ASCII) text format and probably also as HTML. To read the compressed ASCII format, it is not necessary to uncompress them first because `less` does it automatically:

```
less /usr/doc/howto/Highly-Cryptic-And-Powerful-HOWTO.gz
```

Howtos vary somewhat in style, from collections of pointers to further information to tutorials, and can be used in the following manner: suppose you want to set up some service on your system that you never installed before. If you do not have a package from a Linux distribution that does the installation and configuration more or less automatically, then you can consult the appropriate howtos.

Howtos also often provide useful examples.

25.5 RFC

A Request For Comments (RFC) document is also a source of information. If you do not have the RFCs local on your machine, check the following site:

```
http://www.rfc-editor.org/
```

If you installed a complete Linux distribution, you should find in the directory `/usr/doc/rfc/` about 2700 files named `rfc*.txt.gz`. In this case, the asterisk (*) stands for a number from 1 to 2792 (as at the time of writing).

The file `rfc-index.txt.gz` contains an index (by number) of what is covered in each RFC. The entry 2696 reads as follows:

```
2696 LDAP Control Extension for Simple Paged Results Manipulation. C.
Weider, A. Herron, A. Anantha, T. Howes. September 1999. (Format:
TXT=12809 bytes) (Status: INFORMATIONAL)
```

An arbitrarily chosen paragraph therein reads:

```
If the page size is greater than or equal to the sizeLimit value, the
server should ignore the control as the request can be satisfied in a
single page. If the server does not support this control, the server
MUST return an error of unsupportedCriticalExtension if the client
requested it as critical, otherwise the server SHOULD ignore the
control. The remainder of this section assumes the server does not
ignore the client's pagedResultsControl.
```

This is complex information about a rather specific topic, so less experienced users may find using RFCs a bit daunting, but they can be useful. For example, if an RFC exists about a protocol you want to program for, it will probably give you all the information you need in a well-structured form.

25.6 The Internet

The Internet contains numerous sites that have the kind of information you are interested in. We recommend you bookmark a few favorites, and also check for newsgroups and forums in your interest area that will be helpful to you.

25.7 Books

There are plenty of books about Linux and related subjects, differing widely in style and quality. Luckily, choosing the right one requires no wizardry: by reading some online recommendations and reviews (online bookstores usually have them), you'll get the picture. As you're likely to discover, for certain topics there is usually a particularly authoritative book.

25.8 Finding a file

To get a list of all files (and directories) whose names contain a “blah” you could use the `find` command, as follows:

```
find / -name '*blah*' -print
```

This will go through your whole file system and find all the files specified:

```
/usr/X11R6/include/X11/pixmap/mini.blah.xpm
/usr/X11R6/include/X11/3dpixmap/normal/blah_3d.xpm
/usr/X11R6/include/X11/3dpixmap/small/small.blah_3d.xpm
```

There are, however, two reasons why you do not really want to use `find`: it causes a massive system load, and it takes a long time.

Instead, you can set up a database of filenames and use the `locate` command to access that database, as follows:

```
locate blah
```

The output is the same as above. Updating the database is usually done daily by a cron job that launches the respective process, `updatedb`. Note, however, that files created *after the last update* will not be found using `locate`.

25.9 Determining the type of command

Commands can be of different types: they can be executable files on disk, shell built-ins, aliases, functions, or keywords. To find out what type a command is (for example, `pwd`), you can enter:

```
$ type pwd
```

This will return the answer:

```
pwd is a shell builtin
```

But wasn't there an executable named `pwd` in `/bin`? If this is the case, then the following command format will clarify the matter:

```
$ type -a pwd
```

This will return the answer:

```
pwd is a shell builtin
pwd is /bin/pwd
```

So there are actually two `pwd`s. To execute the binary, use the full path:

```
/bin/pwd
```

To determine the type, you enter:

```
type type
```

This will return the answer, courtesy of `bash`:

```
type is a shell builtin
```

25.10 DAU

The Documentation Access Utility (DAU) provides a unified interface to all sorts of documentation. It allows the user to ask Linux a question about some topic and presents the relevant information in browsable form.

Questions such as the following are understood by DAU:

```
dau tcpip basics
dau whatis device
dau security books advanced or intermediate
dau C library reference
dau list software firewall
dau howto about framebuffer
dau whereis fractint
dau /etc/securetty examples
```

```
dau /etc/passwd format
```

The following question is also understood by DAU:

```
dau who says "SIOC_ADR: cannot activate abc123"
```

Almost any sensible question leads to an “answer” constructed from within Linux documentation or references to other available documents, like online resources or books. Alas, so far no one has begun to program DAU.

Chapter 26. Monitoring the system

In this chapter we list and describe some of the most important utilities for monitoring (and sometimes, changing) the state of a running Linux system.

26.1 Linux facilities and tools

You can use the following facilities and tools to see what is currently going on in the Linux for S/390 system.

26.1.1 log files

Linux log files are located in `/var/log`. System messages and warnings are usually logged into the files `messages` and `warn`, respectively. Errors in programs or the kernel often leave traces in the log files *before* the process in trouble dies. That's why UNIX administrators often issue the following command to see if something is wrong with the system:

```
tail -40 /var/log/messages
```

The boot messages are stored after bootup in `/var/log/boot.msg`. This file is especially useful as a reference list of the detected hardware configuration.

26.1.2 The proc file system

Almost every detectable aspect of the state of a Linux system is mapped into pseudo files under the directory `/proc`. These “files” appear to have zero byte size when listed with `ls`, but reading them produces some output; for example `cat /proc/devices` might produce the following:

Character devices:

```
1 mem
2 pty
3 tty
4 ttyS
5 console
10 misc
```

Block devices:

```
1 ramdisk
7 loop
94 dasd
95 mnd
```

Look around in `/proc`, cat some files; you'll love it. Examine `/usr/src/linux/Documentation/proc.txt` to see there's even more.

26.1.3 top

`top` shows a table of processes which is continually updated. The top CPU processes appear on the top of the screen. `top` lets you interactively kill (or send signals to) individual processes. Its behavior like update intervals and sorting order can be adjusted interactively or by startup options. To exit `top` press "q".

26.1.4 ps

`ps` lists the current processes once. The usage of `ps` is rather obscure: the same option may be there with and without a dash and may produce different results. Try:

```
ps -e
```

versus:

```
ps e
```

to see the difference. However, `ps` is indeed a very useful tool. To list all running processes use:

```
ps r
```

To list all *your* processes (in long format, option "u"), enter:

```
ps ux
```

while:

```
ps aux
```

lists *all* processes. To find a process named "myproc", use:

```
ps aux | grep myproc
```

You might want to take a look at the man page. You'll find that `ps` offers many nice options that determine what processes are displayed in which order and format. The man page even contains a few usage examples near the end.

26.1.5 pstree

`pstree` displays the process list as a tree. `pstree` may produce output such as the following:

```

init--atd
|-axnet
|-cardmgr
|-cron
|-gpm
|-httpd---httpd
|-in.identd---in.identd---2*[in.identd]
|-inetd
|-kflushd
|-khubd
|-klogd
|-kpiod
|-kswapd
|-kupdate
|-lockd---rpciod
|-login---bash---startx---xinit--X
|                                     ^-ctwm--xclock
|                                     |-xload
|                                     ^-xosview.bin
|-lpd
|-md_thread
|-5*[mingetty]
|-4*[nfsd]
|-nscd---nscd---5*[nscd]
|-portmap
|-rpc.kmountd
|-rpc.kstatd
|-sshd
|-syslogd
|-xterm---bash---pstree
^-xterm---bash--man---sh---less
  ^-man---sh---sh---less

```

An often useful option is `-p`, which also displays the process IDs (PIDs) of the listed processes. When given a user name, only the processes of that user are shown; for example:

```

pstree -p jj

```

produced (in reference to the previous example) the following:

```

bash(260)---startx(14416)---xinit(14425)--X(14426)
|                                     ^-ctwm(14429)--xclock(14437)
|                                     |-xload(14438)
|                                     ^-xosview.bin(14439)

bash(14444)---pstree(14627)

bash(14481)--man(14541)---sh(14545)---less(14547)
  ^-man(14552)---sh(14553)---sh(14555)---less(14556)

```

This should probably be named a `ps-forest`.

26.1.6 who and w

The `who` command lists which users are currently logged on to the system, from what tty, and how long they have been active. You can use the `-H` option to produce column headers and slightly improve the readability of the list.

The `w` command also lists who is logged on, but gives additional information about the processes the users are currently running.

26.1.7 xosview

`xosview` is an X application that displays CPU, memory, disk, and network use as horizontal bars. With `xosview` running, you can detect immediately that some program uses lots of CPU cycles or continuously allocates memory. This application is also useful for learning about system behavior in situations of normal operation.

26.1.8 xload

`xload` is an X application that shows a periodically updated histogram of the system load. The displayed graphics is useful for a load overview for the last, minutes, or hours. The visible range is controlled by the horizontal window size and the parameter following the option `-update` that gives the time interval between updates in seconds (the default is 1).

26.1.9 procmeter

`procmeter` is a runtime-configurable X application that can displays many aspects of the system load in histogram (or other) form.

26.1.10 uptime

The `uptime` man page provides the following description:

“uptime gives a one line display of the following information. The current time, how long the system has been running, how many users are currently logged on, and the system load averages for the past 1, 5, and 15 minutes.”

Nicely said; the output looks as follows:

```
11:18pm up 7:38, 2 users, load average: 0.06, 0.02, 0.00
```

26.1.11 free

`free` displays the amount of free and used memory in the system.

```
free
```

produces the following display:

	total	used	free	shared	buffers	cached
Mem:	62664	53972	8692	26628	20316	12596
-/+ buffers/cache:		21060	41604			
Swap:	136040	1864	134176			

The options `-k` and `-m` cause the quantities to be listed in units of kilobytes and megabytes, respectively.

26.1.12 `du` and `df`

The `df` (disk free) command reports file system disk space usage.

```
df
```

lists the following:

Filesystem	1k-blocks	Used	Available	Use%	Mounted on
/dev/dasdf1	1771660	446768	1234896	27%	/
/dev/dasdg1	104596	92280	6920	93%	/swap

The `-h` option is often useful because it causes the numbers be displayed in a more human-readable form (with suffixes like `M` for megabyte and `G` for gigabyte).

The `du` command reports the space occupied by the current directory and all directories within it. An alternative directory may be supplied as an argument. The `-s` switch is often useful because it makes `du` list only utilization summaries (the default behavior is to report utilization of all subdirectories, which often causes pretty lengthy output). `du` also understands the `-h` option.

26.1.13 `fuser`

The `fuser` command identifies processes that use a certain file, directory or socket:

```
fuser -v /root
```

(The `-v` option produces a much more readable output.)

	USER	PID	ACCESS	COMMAND
/root	root	24762	..c..	su
	root	24763	..c..	bash

You can even kill all processes that use a specific file, which comes in handy when a file system cannot be unmounted because some unknown process is accessing it. See the man page for details.

26.2 VM tools

Apart from native Linux tools, you can monitor the performance of a Linux virtual machine at the macro level using VM tools and facilities.

The CP `INDICATE USER` command displays performance-related information for a Linux virtual machine. This command can be issued from the console of a running Linux virtual machine by prefacing the command with the current terminal line end character (normally a # character); for example:

```
#CP INDICATE USER
```

Make sure that you have CP SET RUN ON for the Linux virtual machine so that its execution is not halted if it enters a CP READ state.

The INDICATE USER command can also be issued for a Linux virtual machine by another user with privilege class E.

You can use a performance monitoring product such as the Real Time Monitor VM/ESA (RTM/ESA), 5798-DWD, to track the real time performance of a Linux virtual machine.

When monitoring is enabled on a VM/ESA system, CP monitor data will be written for Linux virtual machines. These records can be analyzed later using a tool such as the VM Performance Reporting Facility (VMPRF), 5684-073.

You can find further information about the use of this product and RTM/ESA in the IBM Redbook *VM/ESA Performance Tools*, GG24-4152.

By using the Secondary Console Interface Facility (SCIF) of VM/ESA you can receive console output messages from multiple Linux virtual machines on the console of another virtual machine and respond to those messages. Used in conjunction with the Programmable Operator facility, you have a single point of control for the consoles of several Linux virtual machines.

This technique allows you to code a filter that alerts the VM system operator whenever a Linux virtual machine displays a certain message on its console. For more information see 6.8.3, “Secondary console interface” on page 126.

Appendix A. Intel architecture, S/390 architecture

Linux is usually considered a PC operating system. Linux was first developed for an Intel 80386 and the Intel architecture remains the most common platform for Linux.

In this appendix, we describe the fundamentals of the Intel architecture and that of S/390. The Intel symmetric multiprocessor (SMP) architecture is compared with S/390, with particular attention to reliability, availability and serviceability. We also give a brief introduction to S/390 virtual machines and VM/ESA. Using VM/ESA, you can potentially run hundreds or even thousands of Linux virtual servers on a single S/390 system.

A.1 Architecture description

This section provides a brief introduction to the basic architecture of the Intel 32-bit (IA32) and IBM S/390 processors. For further details refer to the *Intel Architecture Software Developer's Manual Volume 3*, Intel order number 243192, and the *ESA/390 Principles of Operation*, SA22-7021.

A.1.1 IA32

An IA32 processor offers different modes of operation, but some of them are provided only for compatibility reasons and today are not used very often. The so-called “protected mode” is the one used by today’s operating systems. It allows memory to be divided into different segments, isolated from one another, and supports paging to obtain a virtual memory larger than the physical one.

The IA32 processors contain different sets of registers:

- General purpose registers - these eight registers can be used to maintain operands and main memory pointers. Some of them are reserved for particular purposes; for example, the ESP register is used as a pointer to the stack.
- Segment registers - these registers are used to hold the descriptors of segments used while accessing code, data, or the stack.
- Status registers - these registers are used to encode the processor status and to specify its mode of operation.

A set of floating point registers is also provided to support floating-point operations. These registers are organized as a stack and are 80 bits each.

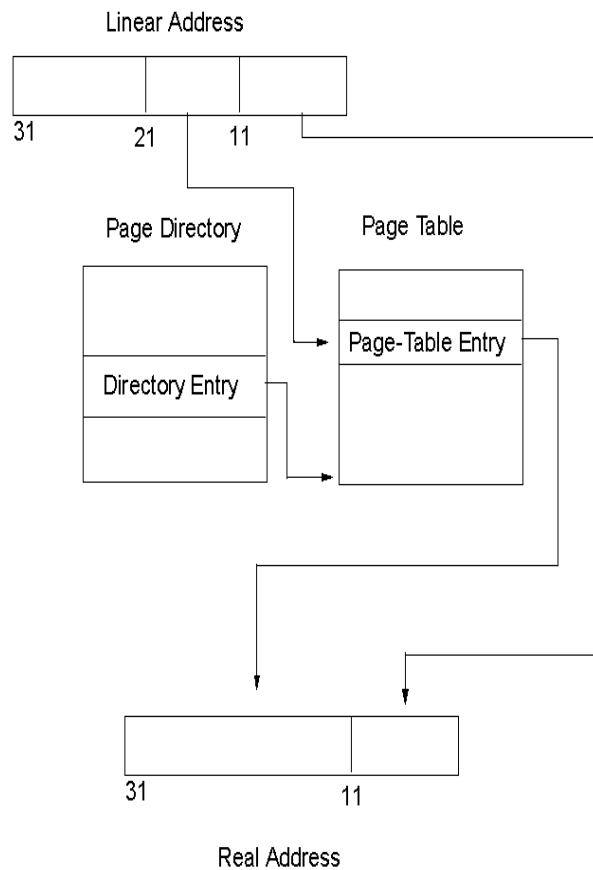


Figure 115. Address translation on IA32 processors

The memory on an IA32 processor operating in protected mode is divided into segments. Each segment has a maximum size of 4 GB (which has been increased to 64 GB on the latest Pentium processors). A segment is accessed by means of a segment selector contained in one of the segment registers. A segment selector is composed of three parts:

- A segment descriptor index.
- A table indicator that allows you to establish if the index is referred to the global descriptor table (GDT) or the local descriptor table (LDT).
- A requested privilege level, used for segment protection.

The LDT is local to each thread and is saved during thread switch. The GDT is global to the entire system. The GDT and LDT contain the segment descriptors that are used to maintain information about:

- The size of the segment.
- Its base address in virtual storage.
- The type of the segment and its associated privilege level.

The segment type is used to distinguish between segments containing data or code and segments containing system structures (such as task-related information, the LDT and so on).

When accessing memory, a segment register and an offset register are used: the address of the data in memory is obtained by adding the base address of the segment referenced by the segment register to the offset contained in the offset register. This linear address can be equal to the physical address of data in memory in case paging is not active. Otherwise another translation step is required to translate the linear address into the physical one (see Figure 115 on page 454). The pages most commonly used by IA32 processors have a fixed size of 4 KB, but pages as large as 2 MB or 4 MB are supported.

The information that the processor uses during the page translation process is contained in the following data structures:

- The *page directory* contains up to 1024 32-bit page-directory entries. Each of them contains the base address of a page table and information about the status of the page (for example if it is valid, or if it is in main memory or swapped to secondary storage).
- The *page table* - each entry contains the address of a page and some bits used to maintain information about the state of the page and the privileges needed to access it.

Figure 115 on page 454 shows the process used to translate a linear address to a physical one. A control register is used to maintain the base address of the page table. Because this register is saved during a task switch, each task can have its own mapping of linear addresses to real addresses.

The protection mechanism of IA32 processors allows for proper isolation of processes. For example, during a memory access, checks are performed on the size of the referenced segment to ensure that the reference is within the segment.

Also, the processor uses segment and page level protection mechanisms. Four levels of privilege are used: level 0 is the highest one and allows access

to each memory region. Also levels 0, 1, and 2 identify processes running in supervisor mode, while level 3 is usually reserved for applications. Protection checks are performed on each memory access and when control is transferred from one procedure to another using particular segments known as *call gates*.

On a memory access, the privilege level of the current code (contained in the segment descriptor of the current code segment) and the privilege level contained in the segment descriptor used for data access (contained in a segment register like DS) are checked against the privilege level contained in the segment descriptor of the segment accessed. Only if the first two privilege levels are greater than the last one is access granted; otherwise, an exception is raised. The privilege level is also used to restrict the operations that a process can perform; in fact, some operations are allowed only when executing at level 0.

A protection mechanism is used when accessing pages, too. Each entry in the page table contains a bit indicating whether the page can be accessed in user mode (privilege level 3), or if the accessing process must be in supervisor mode; also, another bit is used to indicate if the page is read-only.

A.1.2 S/390

A S/390 system is usually composed of one or more central processing units, main storage, expanded storage, and a channel subsystem that is connected to the I/O devices; see A.1.3, “I/O subsystem” on page 459.

Main storage contains code and data used by the CPUs and is divided into blocks of 4 KB. In contrast, expanded storage cannot be used to hold data currently used by the CPUs, and it is usually reserved for paging purposes, in that it allows you to transfer memory blocks faster than mass storage. It is divided into pages of 4 KB, each addressed by a 32-bit identifier, thus allowing up to 2^{32} pages. Only complete pages can be transferred to or from expanded storage to main memory. Expanded storage is a unique feature of the S/390 architecture; there is nothing similar in IA32.

Each CPU contains a set of registers that can be used by programs. They can be divided into the following:

- The *program status word* is used to maintain information related to the status of the CPU and to instruction sequencing.
- The *general registers*, which are sixteen 32-bit registers used for integer arithmetic and for addressing.

- The *floating-point registers* - depending on the configuration, 4 or 16 registers are available and can be used both for hexadecimal and binary floating-point computations.
- The *control registers* - 16 registers are available and are used to control various hardware facilities.
- The *access registers* contain segment-table designators used to access address spaces.

Depending on the mode of operation, the addresses generated by a program can be interpreted in three different ways:

- *Absolute addresses* are addresses given to the main storage locations.
- *Real addresses* are translated to absolute addresses by prefixing.
- *Virtual addresses* are translated to real addresses by means of dynamic address translation.

The S/390 absolute address is like the IA32 real address; both are used to designate a given location in the main storage. The virtual address can be compared with the IA32 linear address because both are used to support paging and to provide a virtual address space larger than the real main memory.

Prefixing is used to allow different CPUs sharing main storage to work independently. Prefixing remaps the first 4 KB of storage using the value contained in the prefix register of each CPU: if a memory reference is within the first 4 KB, the address is added to the prefix to obtain the new address.

For translating a virtual address to a real one, each CPU uses address translation tables. The translation table used depends on the address space used. The control program associates, with each address space, a unique number of 16 bits known as an address-space number (ASN). The ASN is maintained by the CPU in one of its registers and is used as a key to retrieve the address space control information through a process known as address space number translation (ASNT).

The ASNT consists of a two-table lookup process. The first 10 bits of the ASN are used as a key in the first table to obtain the address of the second table. Then the remaining 6 bits of the ASN are used as an index in the second table (refer to Figure 116 on page 458) that contains the address space control information. This information, among other things, includes the address of the authority table and the address of the segment table. The authority table is used to determine if the current process can use the address space, thus allowing the use of a given address space only by

authorized processes; 2^{16} authorization levels are possible, so this table contains up to 2^{16} entries.

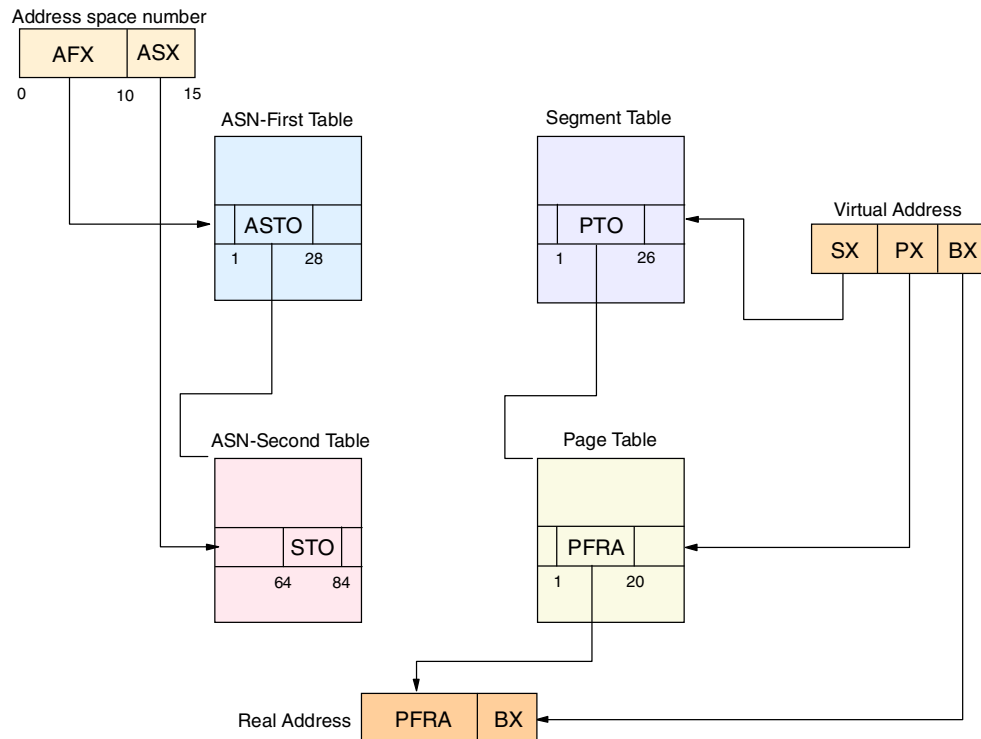


Figure 116. Translation of a virtual address to a real one

The segment table is used for supporting virtual address translation. Each virtual address is divided into three parts, as shown in Figure 116:

- The first 12 bits are used as an index in the segment table to obtain the address of a page table (PTO).
- The next 8 bits are used as an index in the page table to obtain the page frame real address (PFRA).
- The last 12 bits are simply used as an offset in the page.

Each entry in the segment and page table contains additional information that is related, for example, to the validity of the page and to its modifiability. Also, to speed up the page look-up process, a translation-lookaside buffer (TLB) is used to cache the PFRA of the most-recently-used pages.

This allows each CPU to access memory that is divided into address spaces of up to 2^{31} bytes each. Depending on the operation mode, each CPU can access up to 16 address spaces at the same time. Also, each address space is divided into 4 KB pages to allow paging and to provide a virtual memory larger than the real one. The paging process can be speeded up by using extended storage instead of mass storage to swap out unneeded pages.

A.1.3 I/O subsystem

The S/390 I/O subsystem is different from those of other architectures. In fact, S/390 defines a unified way to access all kind of devices through so-called *channels*.

All the I/O activity is managed by the channel subsystem, which is responsible for controlling the flow of information between the I/O devices and main storage.

Communication between the channel subsystem and devices happens through channel paths. Different kinds of channel paths are available, and each supports different data transfer speeds. A S/390 system can support, depending on the configuration, up to 256 channel paths.

Channel paths and devices are connected through *control units*. Control units are needed to adapt the standard form of control used by the channel subsystem to the particular needs of each kind of device.

Each I/O device is associated with a subchannel. Subchannels provide information about the device they are connected to and are the only means by which the channel subsystem can access the device. Each subchannel has an associated system-wide unique 16-bit ID that is used when requesting an I/O operation to indicate the device it is directed to. Given the size of a subchannel ID, a maximum of 65,536 subchannels (and I/O devices) can be connected to a channel subsystem.

While the subchannel associated with each device is unique, a device can be reached through different channel paths. In fact, a device can be connected to more than one control unit, and a control unit can be connected to more than one channel path. While communication to a device happens only through one path, the presence of multiple paths allows for better performance and higher reliability and availability. Indeed, when starting an I/O operation, the channel subsystem has the ability to perform path selection, choosing among all the available paths, and using a different path if one is busy.

Beyond the ability to connect a large number of devices, another advantage of this I/O architecture is its high efficiency while performing data transfers. Actually, the CPU is busy only during the first phase of the I/O operation, that is, passing the needed information and commands to the subchannel. After that, the CPU can continue its activity while the work of moving data is performed by the channel subsystem. When the I/O operation has ended, the CPU receives a notification through an asynchronous interrupt. If interrupts have been disabled, the pending interrupt is stored with information about its source; this way the CPU can interrogate the channel subsystem to know if a given channel ended its activity without being interrupted asynchronously during its work.

A.2 Symmetric multiprocessing

In a symmetric multiprocessor (SMP), multiple processors share the same resources such as main memory, the communication buses, the I/O subsystem and so on, allowing better distribution of the workload and simple and fast data sharing among processors.

A.2.1 Intel SMP

With the introduction of the Profusion chip set, Intel pushed the limit of its multiprocessing technology up to 8-way. In fact, the Profusion chip set supports up to 8 Pentium Xeon processors. The Xeon version is used because of its large on-chip second-level cache, which is needed to reduce the memory traffic that may be quite high due to the large number of processors.

The 8 processors are divided into two groups of four; the processors in each group share a common bus that connects them to the Profusion chip set (see Figure 117 on page 461).

The Profusion chip set creates a “fusion” of three Pentium III processor buses and two main memory subsystems. Two of the three processor buses are actually used to connect processors, while the other is used to connect I/O devices.

The Pentium Xeon processors have integrated L1 data and instruction caches of 16 KB each. The second-level cache is integrated in the same package and works at the same frequency as the memory.

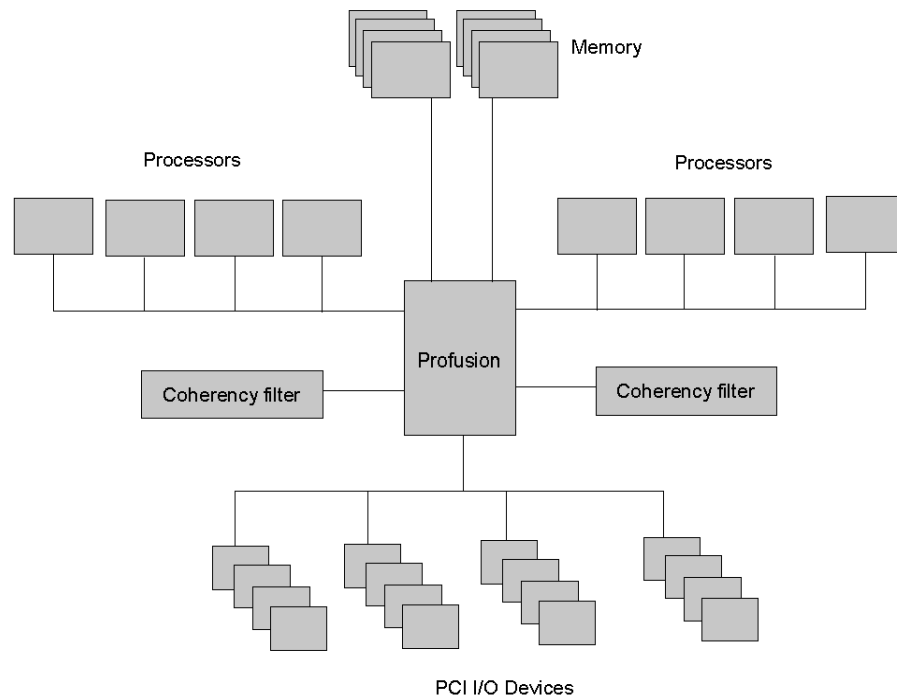


Figure 117. The Profusion chip set

The Profusion chip set coordinates the activity of the buses and connects them with main memory, and also allows for reduced coherency traffic thanks to the coherency filters. The chip set is divided into two chips (not shown in the figure): the Memory Access Controller (MAC) and the Data Interface Buffer (DIB).

The MAC contains the control functions for the system. It manages the memory contained in the coherency filters and controls coherency notifications, preventing unneeded coherency traffic from reaching processor buses. This is accomplished using the coherency filters, each of which contains information about a superset of the data cached by the processors in the related bus. Before propagating coherency information in a bus, the MAC verifies if it is needed by checking the information contained in the coherency filter.

The DIB contains the data paths needed to connect the buses and the main memory. It is controlled by the MAC and allows data movement between the

processor buses and the system memory. Also, the DIB generates error correcting codes (ECC) needed to recover from errors.

To improve performance, the memory is divided into two interleaved banks, each of which is connected to the profusion chip set through a dedicated port. In this way it is possible to double the maximum data transfer speed, to 1.6 GB/sec.

A.2.2 IBM SMP

The G5 and G6 generations of the S/390 multiprocessor are based on a design different from the one used for the previous generations (G3 and G4), and it offers higher performance and supports the load of a larger number of CPUs. The new design is based on the so-called *binodal cache* (see Figure 118 on page 463); the processors and cache memories are divided into two nodes connected to one another and to the memory cards.

Each node is composed of up to 6 processors (7 in the G6) with integrated L1 cache, an L2 cache, a system controller (SC), the memory cards, and the buses needed to connect them.

The L1 cache is unified (it contains both instructions and data) and has a size of 256 KB. It is of write-through type: if data is written to locations that are not in cache, that data is not transferred in cache. Instead, the data is immediately transferred to the L2 cache to be written to memory.

The L2 caches have a size of 4 MB each and operate at half the frequency of the processors, like all the other chips in the node. They contain all the data contained in the L1 caches of the node, thus simplifying the management of coherency. Also, L2 cache is store-in, meaning that if a store is made to a memory location that is not cached, the content of this memory location is first transferred to the cache and then updated. This way the next reference to this location will hit in cache.

The SC maintains information about the L2 cache, manages coherency of caches, controls the binodal architecture, and provides for communication with main memory. It is split into two chips to provide more pins and allow for larger L2 caches; each chip is part of one node.

The SC is responsible for maintaining cache coherence; this is done using a modified MESI (modified/exclusive/shared/invalid) protocol. The modification is needed to account for the possibility of sharing inside the node (locally) or outside between the two nodes (globally). This distinction allows faster

operations when a processor requires exclusive access to some data and the data are shared only locally.

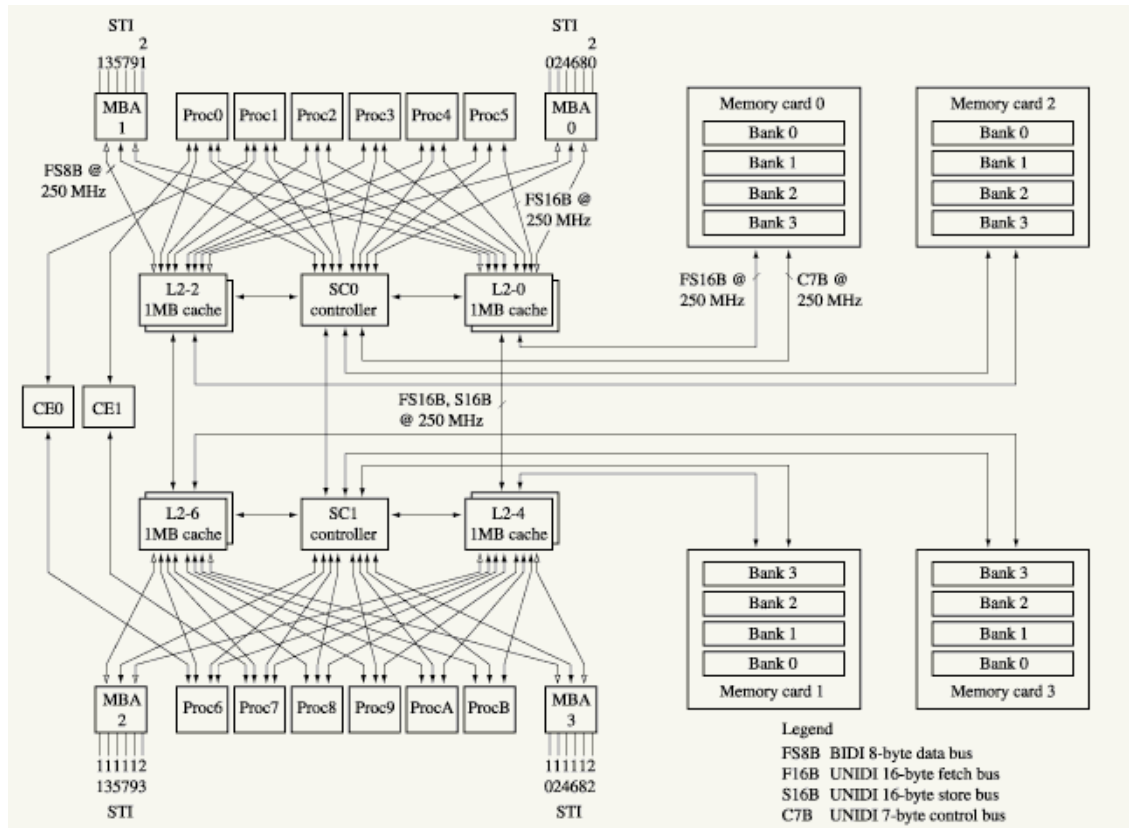


Figure 118. The binodal cache

Also, if an access to a node L2 cache misses, the SC tries to search the data in the L2 cache of the other node. In this way it is possible to improve performance because retrieving data from the other node cache is faster than retrieving it from memory.

The memory is organized into 4 cards, each containing 4 banks. This way 16 banks are available and up to 16 fetch and store operations to main memory can be served simultaneously.

Also, the design of the G5 and G6 has been optimized to efficiently perform the rich set of data movement operations. A series of hardware devices have

been added to allow only minimal processor involvement during the movement of data. For example, the hardware-assisted move engine allows you to move blocks of data from one area of main memory to another, optimizing the process to avoid conflicts with the read and store operations that are required by the processors.

The design of the S/390 server allows the high efficiency in memory access and data transfer that is needed to support a large number of processors and the large amount of data managed by an enterprise server. This requires a good balance between computational power and I/O capabilities.

A.3 RAS considerations

Important factors in evaluating high-end servers are reliability, availability and serviceability (RAS).

A.3.1 Intel Profusion chip set RAS considerations

The Intel chip set implements a number of techniques to obtain high RAS. The system, memory, and I/O buses are all protected by error-correcting codes (ECC) to allow discovery and correction of transmission errors.

Also, the system is able to continue to work even if some hardware failures occur:

- If a processor or processor bus fails, the system continues to work using only the other bus.
- If one of the memory ports fails, the system can work using only one port.
- If any of the coherency filters fails, it is disabled and the system remains operational, but with lower performance.
- If an I/O device fails, it is possible to isolate and disable it. Also, I/O devices are hot-pluggable, meaning that you can change them without interrupting server activity.

To allow for better serviceability, the chip set is able to perform error logging. This way it is possible to identify error sources in less time and promptly correct the system behavior.

A.3.2 S/390 RAS considerations

The S/390 system offers a very rich set of RAS capabilities and it was designed from the start with RAS in mind.

The processors contain duplicated instruction units (IU), fixed-point units (FXU), and floating-point units (FPU). On every operation, the results of the two copies are compared in order to discover possible errors. The check is performed by the register unit (RU) before committing the result of each operation. When the check succeeds, the current state is saved as a checkpoint to allow restart of the next operation in case of failure.

The RU, like the L1 cache, is not duplicated, but is protected through using ECC in general and using parity checking when the information is replicated elsewhere in the processor. This way the processor is able to discover and eventually recover from a large series of possible faults.

When a processor fails due to some permanent internal fault, it is isolated and removed from operation, while system operation continues unaffected for the other processors, thus maintaining a high degree of availability. Also, if a spare processor is configured, it is automatically activated and the state of the failing processor is transferred to it, allowing the resumption of normal activity without user intervention and without any impact on the application or the operating system. In some cases, depending on the type of fault, it is not necessary to completely stop the processor; instead, by disabling the faulty device, operations can continue, although with reduced performance.

The L2 cache and the buses connecting it with main memory and with processors are protected using ECC. When erroneous data is detected by an ECC station, data propagation is blocked to avoid its propagation inside the system. Also, when an L2 cache line is discovered to be invalid, it is purged to avoid use of the invalid data by any of the processors. The next time the data is referenced, it will be reloaded from main memory. Because of low failure rates, the purge of cache lines does not have a negative impact on system performance.

The L2 cache contains spare cache lines that are automatically used to substitute failing lines. In this way it is possible to recover from errors in a cache chip without the need to mark the whole chip as faulty.

Main memory is organized so that each DRAM module contributes only one bit to a given ECC check box. This way, because ECCs are able to correct single bit errors, it is possible to recover from all partial module failures and all complete module failures. Also, to avoid the accumulation of errors that can result in the inability to correct them, memory is continuously analyzed and errors are corrected as soon as they are discovered.

When the number of errors discovered (and corrected) on a given memory module becomes greater than a given threshold, the module is replaced with

a spare one. The contents of the failing module are copied into the new one, while all the intervening stores are propagated to both the modules. This way, when the copy is finished, the old module is completely replaced by the new one. The use of spare DRAM memory modules allows substitution of a failing module without stopping the machine, increasing the total availability of the S/390.

S/390 provides a large number of mechanisms to produce very high availability and integrity of data. In many cases, if an error is discovered, the hardware is able to recover from the error condition, isolating the faulty device and activating spare ones. Also, only the failing hardware is isolated from the system, thus allowing the largest possible number of devices to keep working and resulting in the least possible adverse effect on the performance of the system.

A.4 Comparing the IA32 and S/390 architectures

Both the IA32 and the S/390 are so-called Complex Instruction Set Computer (CISC) architectures, meaning that they offer a very rich set of instructions. From the micro-architectural point of view, the IA32 and the S/390 CPUs are very different. The IA32 transforms complex instructions into simpler ones that are executed in parallel and out of order, using a complex scheduling mechanism. So the IA32 core is optimized to execute simple instructions in parallel.

On the other hand, the S/390 G5 and G6 CPUs are able to execute only one operation per cycle, but are optimized to reduce the time needed to complete the execution of complex, long-running instructions that are often used in programs. Also, only one integer operation is executed per cycle even if, for example, two integer pipelines are available, because both of them execute the same operation to allow checking the result. The reliability of the processor is preferred to raw speed. In general, the S/390 CPU offers higher RAS and allows it to discover and recover from a large number of errors, usually disabling only the faulty devices.

Also, the S/390 instruction set offers a rich set of instructions to move memory blocks, supported by specialized hardware so as to obtain higher performance. For example, the move engine can be used to quickly copy pages from main storage to expanded storage, thus speeding up the paging process.

Expanded storage is a unique feature of the S/390. It allows access to a large and fast memory that can be used to swap out unused memory pages faster

than mass storage. Also, expanded storage can be used as a cache for minidisks, improving the performance of application I/O.

Both the IA/32 and the S/390 support virtual addressing, but they offer slightly different mechanisms. The IA32 allows each task to maintain information about its mapping of virtual addresses versus real ones; usually, each thread is mapped on a different task and all the threads of a single process share the same page mapping.

S/390 implements a different page mapping for each address space and allows access to up to 16 address spaces at the same time. Also, the S/390 uses the concept of a *segment* to group pages; during the address translation process, in fact, the first part of the virtual address is used to index the segment table and obtain the address of a page table.

In contrast, the IA32 uses segments to protect storage areas. There is no direct relation between pages and segments, but a segment is used to define the access rights to a given virtual memory region. This way, when accessing virtual memory, each process can only read or write what is contained in the segment defined by the descriptor it is using.

From an SMP architecture point of view, the S/390 supports a higher number of CPUs while maintaining a quite linear performance increase. While the Intel multiprocessor allows up to 8 CPUs, the G6 supports 12 processors plus two other processors configured for I/O or as spares.

The availability of spare processors allows the substitution of a faulty processor “on the fly”, without stopping the system and without intervention by the operating system or application.

The design of the binodal cache allows the disabling of a single faulty processor, while, for example, the Intel multiprocessor needs to disable 4 processors if its bus fails.

The use of spare devices is not limited to just processors. For example, the availability of spare memory modules allows automatic substitution for faulty ones while continuing to work, thus producing very high availability and avoiding the need for user intervention.

While the IA32 offers higher computational power with faster processors that are able to execute more than one simple instruction per clock cycle, the S/390 offers a higher degree of reliability and a better balance between raw computational power and I/O bandwidth, supporting a large number of devices and a very optimized I/O mechanism. The S/390 offers higher RAS, which is a very important factor when evaluating high-end servers.

Appendix B. VM/ESA virtual machines

In this project we used VM/ESA extensively to provide multiple Linux for S/390 environments on the same S/390 system.

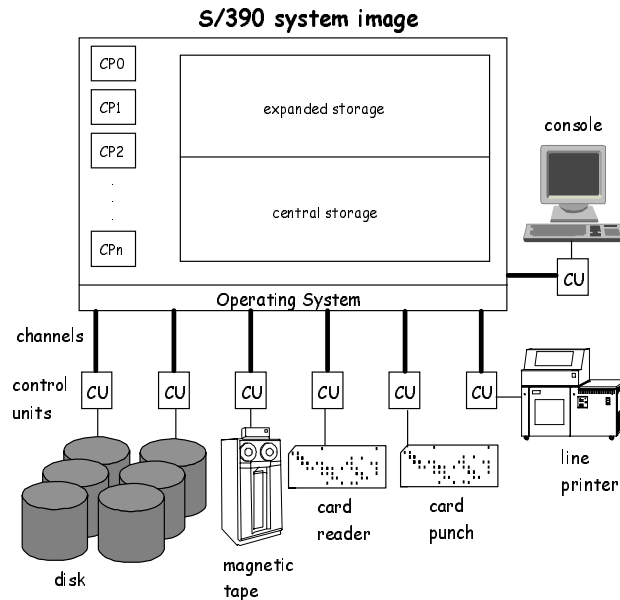


Figure 119. S/390 system resources

Figure 119 shows a highly simplified and stylized view of a S/390 processor and its physical components or resources - processor units, memory (real and expanded storage), channels, control units, and devices. All of these are managed by an operating system, typically OS/390, VM/ESA, or VSE/ESA.

VM/ESA has special abilities, however. Its two most important components are the control program (CP) and the Conversational Monitor System (CMS).

CP is able to *virtualize* hardware resources. It does this either by partitioning or sharing real hardware resources, or by emulating their behavior programmatically.

CP implements virtual S/390 machines. On a single piece of hardware, you can run several copies of the same or different operating systems under the control of CP. We refer to CP as a *hypervisor*. The users of each virtual S/390 machine are unaware that a hypervisor is providing the S/390 environment in which their application is running.

Each virtual machine has its own virtual memory, virtual devices, virtual processors, and so on. Within each virtual machine, a S/390 operating system is IPLed. You can even IPL VM/ESA within a virtual machine.

Each operating system running in its own virtual S/390 environment communicates with virtual devices. The mapping of virtual to real devices and resources is handled transparently by CP.

The result is that by running VM on the processor shown in Figure 119 on page 469, we can replicate the S/390 environment many times over. This is illustrated in Figure 120.

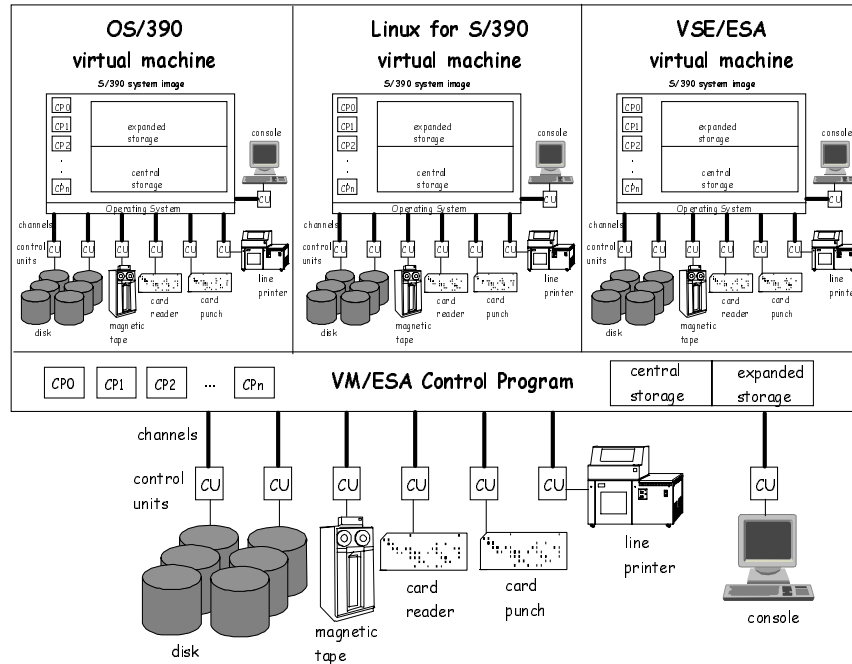


Figure 120. Virtual machines running under the control of a hypervisor

We shall look briefly at how CP implements certain device types. Because the S/390 virtual machine concept dates back to the late 1960s and early 1970s, some of the standard devices such as a card reader and card punch may seem antiquated, but they are still extremely useful when virtualized.

B.0.1 The CP directory

The definition of a virtual machine is stored in a master directory called the *CP directory*. In source form an entry might look like this:

```

USER LINUX5 XXXXXXXX 128M 256M G ❶
IPL 200 ❷
MACHINE ESA 4 ❸
CONSOLE 0009 3215 ❹
SPOOL 000C 3505 A ❺
SPOOL 000D 3525 A ❺
SPOOL 000E 1403 A ❺
LINK MAINT 0190 0190 RR ❻
LINK MAINT 019E 019E RR ❻
LINK MAINT 019F 019F RR ❻
LINK MAINT 019D 019D RR ❻
MDISK 0201 3390 0001 1000 LINUX5 MR ❼
MDISK 0202 3390 1001 1000 LINUX5 MR ❼
MDISK 0203 3390 2001 200 LINUX5 MR ❼
MDISK 0200 3390 3221 20 LINUX2 MR ❼
MDISK 0191 3390 555 50 VMZU1A MR ❼
SPECIAL 808 CTCA ❸
SPECIAL 809 CTCA ❸

```

Most of the statements in this entry define virtual resources or devices.

❶ The user ID that identifies this virtual machine is LINUX5. The virtual machine is defined with a default storage of 128 megabytes, but this can be redefined up to a maximum of 256 megabytes.

❷ When you log on to the virtual machine, an IPL will occur from device number 200.

❸ This statement describes the processor architecture of the virtual machine. The maximum number of processors that can be defined for this virtual machine is four. The default is one.

❹ The CONSOLE statement defines the operating console for the virtual machine.

❺ The next three statements define a reader, punch, and printer for the virtual machine.

❻ These are read-only links to minidisks owned by other virtual machines.

❼ These statements define four minidisks owned by this virtual machine.

Device numbers 201-203 are partitions of a real volume with volume identifier (void) LINUX5.

They occupy the following cylinder ranges on that device:

```
201 1-1000
202 1001-2000
203 2001-2200
```

Device number 200 occupies cylinder range 3221-3240 on the real disk with valid LINUX2. Device number 191 occupies cylinder range 555-604 on the real disk with valid VMZU1A.

These are all read/write minidisks.

③ These statements define a pair of device numbers for a virtual channel-to-channel device.

B.0.2 Processors

A virtual machine can have up to 64 virtual processors defined. If the operating system running in the virtual machine is multi-processor (MP)-capable, it will dispatch work on its virtual processors as if it were running without a hypervisor (on the real hardware). CP will handle the dispatching of virtual processors on the real processors available to that virtual machine. A real processor can be either dedicated to a virtual machine or shared among virtual machines. Dedicated processors can only be used by one virtual machine.

B.0.3 Storage

Each virtual machine has its own defined virtual storage or memory. CP manages the residency of virtual machines' pages in real storage with a sophisticated paging mechanism. Pages that have not been referenced can be moved out of real storage either into expanded storage or onto a paging device. When a virtual machine touches a page that is no longer in real storage, a page fault occurs and CP will bring the missing virtual page back into real storage.

The memory addresses in a virtual machine are virtual addresses. They have no meaning outside the virtual machine in which they are generated and used. Whenever required, these virtual addresses are translated to real addresses by access register translation (ART) and dynamic address translation (DAT) for the address space referenced by the user.

Using ART and DAT, CP keeps these address spaces absolutely separate from one another. It is impossible for one user to access an address space of another user unless the owner allows the other user to do so.

A portion of the real storage on a S/390 system can be dedicated to a virtual machine. In this case the storage of the virtual machine is real and the operating system running in the virtual machine can perform its own memory management without intervention by CP. Expanded storage can also be dedicated to a virtual machine.

CP also has facilities that allow the sharing of virtual pages by a number of virtual machines. A shared virtual page requires just one page of real storage no matter how many virtual machines are sharing it, thereby economizing on real storage requirements.

B.0.4 Minidisks

VM minidisks are virtual disk devices. They are implemented by partitioning a real S/390 volume into cylinder ranges that appear as separate disk volumes to the virtual machine. A minidisk can span a whole real disk volume. A real disk can also be dedicated to a virtual machine.

Minidisks can be shared or non-shared. If authorized, one virtual machine can link to a minidisk belonging to another virtual machine to access the data on it. Links can either read-only or read-write. When a minidisk is write-shared, some software is needed to manage access to the data.

CP is able to cache the contents of minidisks in real or expanded storage to improve application response times.

B.0.4.1 Temporary minidisks

Temporary minidisks are allocated from a defined pool of real disk storage, either when the virtual machine is logged on or by explicit definition. They last for the life of the virtual machine. When the virtual machine is logged off or the minidisk is detached, the temporary minidisk is destroyed.

B.0.4.2 Virtual minidisks

Virtual minidisks have similarities to temporary minidisks. Instead of being mapped to cylinders of real disk volumes, they are mapped into real storage by CP. This means that they avoid the need for disk I/O. The associated pages are managed by CP as part of its overall real memory management.

B.0.5 Reader, punch, printer

These devices are not associated with real devices, but are implemented through the CP spooling subsystem. The spool files created or read on these devices can be transferred between virtual machines very easily. They can

also be read from a real card reader, punched on a real card punch, or printed on a real line printer.

B.0.6 The console

The console is an important device for the virtual machine as it is the primary user interface. When you log on to a virtual machine from a real 3270 terminal or 3270 emulator, the virtual console becomes associated with the real 3270. This allows you to enter CP commands and to IPL an operating system.

When an operating system is IPLed in a virtual machine it will look for a device to be its system console. This device is often the one defined by the `CONSOLE` statement in the CP directory.

Once the operating system is IPLed in a virtual machine, the console device is sometimes no longer required and it may be disconnected. The virtual machine can continue to function with a disconnected console.

B.0.7 Channel-to-channel device

A virtual channel-to-channel device is implemented by CP entirely through software. Virtual I/O instructions are intercepted by CP and the data moved between memory buffers. This enables very high-speed communications between virtual machines.

B.0.8 Virtual I/O

An operating system such as OS/390 or VSE/ESA running in a virtual machine will issue normal S/390 I/O instructions to perform I/O. The operating system builds a string of Channel Command Words (CCWs) and issues a Start Subchannel (SSCH) instruction.

If the I/O is to a minidisk device, the virtual device number must be converted to a real device number and the virtual cylinder number must be converted to a real cylinder number. Also the address of the data to be read or written must be converted to a real address.

This process, called CCW Translation, is carried out transparently by CP.

By dedicating real storage and devices to a virtual machine, the processor-related overhead of CCW translation can be bypassed. A “preferred guest” can perform disk I/O without any intervention by CP.

B.0.9 CMS

The Conversational Monitor System (CMS) is a unique S/390 operating system. It is IPLed in a virtual machine in the normal way, but is often a single user environment. You can think of it as the original time-sharing PC!

The user interface to CMS is through the virtual machine console described in “The console” on page 474.

CMS is dependent on CP for some of its functions. For instance, I/O to CMS formatted minidisks is initiated through a Diagnose X'250' instruction. This is only meaningful to CP, which will handle the physical I/O for the real disk device.

CMS provides a rich application development and execution environment. It has powerful commands and tools that are extremely useful to developers and end users alike.

The REXX interpreter makes it easy to write command scripts and even whole applications. The XEDIT editor is a sophisticated editor with a large subcommand set. CMS Pipelines extends the concept of UNIX pipes to a new dimension. Through a wealth of standard filters and stages, complex applications and utilities can be built in a few lines, whereas using traditional procedural programming methods, hundreds of lines of code would be needed.

Besides acting as a single-user system, server applications can also be written that run on CMS. Many VM/ESA products and tools are implemented in this way. The TCP/IP service machine and related daemon virtual machines are a good example.

Appendix C. Linux for S/390 I/O implementation

The following section was copied from the Documentation/390 directory of the Linux distribution. It was written by Ingo Adlung and is copyright IBM 1999, under the GNU Public License.

The following paragraphs describe the I/O related interface routines that the Linux for S/390 common device support (CDS) provides to allow for device specific driver implementations on the IBM S/390 hardware platform.

These interfaces are intended to provide the functionality required by every device driver implementation that supports a specific hardware device on the S/390 platform. Some of the interface routines are specific to Linux for S/390, and some of them can be found on other Linux platforms' implementations.

In contrast to other hardware platforms, the ESA/390 architecture does not define interrupt lines managed by a specific interrupt controller, or bus systems that may or may not allow for shared interrupts, DMA processing, and so on.

Instead, the ESA/390 architecture defines a channel subsystem, which provides a unified view of the devices physically attached to the system. Though the S/390 hardware platform supports a huge variety of different peripheral attachments like disk devices (also known as DASD), tapes and communication controllers, they can all be accessed by a well defined method and they all present I/O completion a unified way: I/O interruptions.

Every single device is uniquely identified to the system by a subchannel. The ESA/390 architecture allows 64k devices to be attached.

Linux, however was first built on the Intel PC architecture, with its two cascaded 8259 programmable interrupt controllers (PICs), that allow for a maximum of 15 different interrupt lines. All devices attached to such a system share those 15 interrupt levels. Devices attached to the ISA bus system must not share interrupt levels (also known as IRQs), as the ISA bus operates on edge triggered interrupts.

MCA, EISA, PCI and other bus systems operate on level triggered interrupts, and thus allow for shared IRQs. However, if multiple devices present their hardware status using the same (shared) IRQ, the operating system has to call all device drivers registered on this IRQ in order to determine which one owns the device that raised the interrupt.

In order not to introduce a new I/O concept to the common Linux code, Linux for S/390 preserves the IRQ concept and semantically maps the ESA/390 subchannels to Linux as IRQs. This allows Linux for S/390 to support up to 64k different IRQs, each representing a unique device.

During its startup the Linux for S/390 kernel checks for peripheral devices. A subchannel uniquely defines each of those devices to the S/390 channel subsystem. While the subchannel numbers are system generated, each subchannel also takes a user-defined attribute, the S/390 device number. Both the subchannel number and the device number can not exceed 65535.

The `init_IRQ()` routine gathers the information about control unit type and device types that imply specific I/O commands (channel command words or CCWs) are needed to operate the device. Device drivers can retrieve this set of hardware information during their initialization step to recognize the devices they support using `get_dev_info_by_IRQ()` or `get_dev_info_by_devno()`.

This approach implies that Linux/390 does not need to probe for free (not armed) interrupt request lines (IRQs) on which to drive its devices. Where applicable, the device drivers can use the `read_dev_chars()` routine to retrieve device characteristics. This can be done without first having to request device ownership.

When a device driver has recognized a device for which it wants to claim ownership, it calls `request_IRQ()` with the device's subchannel id as the pseudo IRQ line. One of the required parameters is `dev_id`, defining a device status block. The CDS layer will use this to notify the device driver's interrupt handler about interrupt information observed. It is the responsibility of the device driver to handle those interrupts properly.

To allow for easy I/O initiation, the CDS layer provides a `do_IO()` interface. This takes a device-specific channel program (one or more CCWs) as input, sets up the required architecture-specific control blocks and initiates an I/O request on behalf of the device driver.

The `do_IO()` routine allows for both synchronous and asynchronous I/O methods. It can specify whether it expects the CDS layer to notify the device driver for every interrupt it observes or for final status only. It also provides a scheme to allow for overlapped I/O processing.

A device driver must never issue ESA/390 I/O commands itself, but must use the Linux/390 CDS interfaces instead.

To cancel a long-running I/O request, the CDS layer provides the `halt_IO()` function. Some devices must first issue a `HALT SUBCHANNEL (HSCH)`

command without having pending I/O requests. This function is also needed by `halt_IO()`.

When finished with a device, the device driver calls `free_IRQ()` to release its ownership of the device. During `free_IRQ()` processing the CDS layer also disables the device from presenting further interrupts.

The device driver does not need to take care of this. The device will be re-enabled for interrupts with the next call to `request_IRQ()`.

The common device support layer comprises the I/O support routines defined below.

Some of them implement common Linux device driver interfaces, while others are ESA/390-platform-specific.

`get_dev_info_by_IRQ()` / `get_dev_info_by_devno()`

allow a device driver to determine the devices attached (visible) to the system and their current status.

`get_IRQ_by_devno()` / `get_devno_by_IRQ()`

get IRQ (subchannel) from device number and vice versa.

`read_dev_chars()`

read device characteristics

`request_IRQ()`

obtain ownership for a specific device.

`free_IRQ()`

release ownership for a specific device.

`disable_IRQ()`

disable a device from presenting interrupts.

`enable_IRQ()`

enable a device, allowing for I/O interrupts.

do_IO()

initiate an I/O request.

halt_IO()

terminate the current I/O request processed on the device.

do_IRQ()

generic interrupt routine. This function is called by the interrupt entry routine whenever an I/O interrupt is presented to the system. The do_IRQ() routine determines the interrupt status and calls the device specific interrupt handler according to the rules (flags) defined during I/O request initiation with do_IO().

Appendix D. The parameter file

In regard to the parameter file, note the following:

- Linux for S/390 has a general read routine that reads in chunks of 1024 bytes. This routine detects short blocks. When reading the parmline file, this will be normally the case.
- The current kernel 2.2.15 handles parmline 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.

D.1 DASD

The DASD driver is configured by the *dasd=* kernel parameter in the parameter file

D.1.1 Syntax

The syntax for this parameter is as follows:

```
dasd=range[,...] | autodetect | probeonly dasd_force_diag=range[,...]
```

where:

Range is in the form *from*(device number) - *to* (device number) or explicitly specifying 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.

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 used. 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.

D.1.2 Example

```
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.

D.2 Mdisk

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.

D.2.1 Syntax

The syntax for this parameter is as follows:

```
mdisk=vdev
```

where:

vdev is the virtual device number specified in hex without the leading 0x. Multiple *mdisk=* statements are allowed. The minor numbers of the device will be assigned in the order that they appear in the parmline file.

D.2.2 Example

```
mdisk=193,194
```

This reserves minor numbers 0 and 4 for device numbers 193 and 194, respectively.

D.3 Root

This parameter tells Linux where to find the root filesystem.

D.3.1 Syntax

The syntax for this parameter is as follows:

```
root=path [ro] [noinitrd]
```

where:

Path points to the root filesystem.

Ro sets the filesystem to read-only in the event that an error occurs.

Noinitrd does not use the initial RAMdisk.

D.3.2 Example

```
root=/dev/ram0 ro
```

This tells Linux where to IPL from. 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.

D.4 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>`.

D.4.1 Syntax

The syntax for this parameter is as follows:

```
xpram_parts=<number_of_partitions>[,size[,...]]*
```

Number_of_partitions can contain a number from 1 to 32; the default is 1.

The size specification 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.

D.4.2 Example

```
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).

D.5 Ctc/Escon

Normally the CTC driver selects the channels in order (automatic channel selection). If you need to use the channels in a different order 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.

D.5.1 Syntax

The syntax for this parameter is as follows:

```
ctc=0,0xrxxx,0xwww,ddd
```

where:

- `rrrr` is the read channel address
- `www` is the write channel address
- `ddd` 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.

```
ctc=noauto
```

D.5.2 Example

For one network device:


```
ctc=0,0x600,0x601,ctc0
```

Or for two network devices:

```
ctc=0,0x601,0x600,ctc0 ctc=0,0x605,0x608,escon3
```

D.6 IUCV

The IUCV driver needs to know the user IDs of the target virtual machines with which it can communicate.

D.6.1 Syntax

The syntax for this parameter is as follows:

```
iucv=useridx,useridy,...
```

D.6.2 Example

```
iucv=tcPIP,linux3
```

In this example the Linux virtual machine can establish IUCV communication with user ID TCP/IP (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.

D.7 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. 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 applications such as vi.

D.7.1 Syntax

The syntax for this parameter is as follows:

```
condev=cuu
```

where cuu is the unit address/device number in hexadecimal.

D.7.2 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).

Appendix E. Troubleshooting and avoiding pitfalls

This appendix provides troubleshooting tips and describes how to avoid common pitfalls (also known as *gotchas*).

E.1 Cannot boot big file system on VM - /etc/fstab not modified

When installing a new Linux system under VM, we first booted from the RAM disk and answered the network questions successfully. When the network was up, we ftp'd the large file system to a minidisk (address 301).

It unwound successfully and we tried to boot from it by changing the Linux parameter file (the new parm line is `mdisk=301 root=/dev/mnda ro noinitrd`). However, the system would not boot and we got the following error:

```
Checking root filesystem
(null):
The superblock could not be read or does not describe a correct ext2
filesystem.  If the device is valid and it really contains an ext2
filesystem (and not swap or ufs or something else), then the superblock
is corrupt, and you might try running e2fsck with an alternate
superblock:
    e2fsck -b 8193 <device>

ext2fs_check_if_mount: No such file or directory while determining
whether /dev/dasda1 is mounted.
fsck.ext2: Invalid argument while trying to open /dev/dasda1
```

The problem was that the `/etc/fstab` file had to be modified. By default, this file in the large filesystem contains the following:

```
/dev/dasda1      /                ext2  defaults,errors=remount-ro 0 1
none            /proc           proc  defaults      0   0
```

It needs to be modified to mount a minidisk, not a DASD:

```
/dev/mnda       /                ext2  defaults,errors=remount-ro 0 1
none            /proc           proc  defaults      0   0
```

E.2 Editing /etc/fstab with vi - be sure the last line has a newline

The file `/etc/fstab` was edited with `vi`, and the next reboot of the system resulted in the minimalistic shell (run level 1).

The following messages were issued:

```
Warning... fsck.ext2 for device /dev/dasda1 exited with signal 10.
[FAILED]

*** An error occurred during the file system check.
*** Dropping you to a shell; the system will reboot
*** when you leave the shell.
Give root password for maintenance
```

The problem was that the newline was missing in last line of /etc/fstab. The `vi` editor will not insert this automatically. To avoid this problem, add a blank line at the end of the `fstab` file.

E.3 Irritating RPM messages/RPM update with RPM

Using the `rpm` that is present in the Marist big file system displays irritating messages such as `Macro %_:`

```
[root@linux6 rpms]# rpm -i --test THE-3_0-1_s390.rpm
Macro %__cat has empty body
Macro %__chgrp has empty body
Macro %__chmod has empty body
...
```

To solve this problem, first install the `bzip2` package named `bzip2-0_9_5c-1_s390.rpm`. Then replace `RPM` itself with the `RPM` package `rpm-3_0_3-3_s390.rpm`.

Note: Doing this in reverse order will lead to a missing shared library, resulting in an `rpm` that will no longer work.

E.4 Native Linux `silos` command - use the proper flags

You may encounter errors using the `silos` command to boot Linux for S/390 from DASD.

To avoid this, first make sure that the `-b` flag points to the `ipleckd.boot` file in the boot directory on the Linux boot device. Then be sure the `-t2` flag was included:

```
[root@linuxxx sbin]# silos -f image -d /dev/dasdc -p parmline -b ipleckd.boot
o->image set to image
...
```

```
IPL device is: '/dev/dasdc'
bootsector is: 'ipleckd.boot'...ok...
bootmap is set to: './boot.2jpTy4'...ok...
Kernel image is: 'image'...ok...
original parameterfile is: 'parmline'...ok...final parameterfile is:
'parmline'.
..ok...
WARNING: silo does not modify your volume. Use -t2 to change IPL records
...
```

E.5 Linux under VM won't boot - forgot to ftp files in FB80

When first bringing up a Linux system under VM, we got the Linux kernel and the initial RAMdisk over to VM via FTP. We tried to boot with a small exec (LINUXIPL) but failed, as follows:

```
LINUXIPL
0000002 FILES PURGED
Record exceeds allowable maximum
RDR FILE 0185 SENT FROM VMLINUX PUN WAS 0185 RECS 0001 CPY 001 A NOHOLD
NOKEEP
Record exceeds allowable maximum
0000001 FILE CHANGED
IPL UNIT ERROR; IRB 01004417 00000010 00207E61 00800000
CP entered; disabled wait PSW 000E0000 00000232
```

The problem was that we remembered to FTP in binary, but forgot to set the record format to fixed 80 (via the FTP subcommand `quote site fix 80`). Without this command, the file gets a record format of V 8192, which prevents Linux from being able to boot.

E.6 Linux under VM won't boot after improper shutdown

You may get the following message while booting Linux for S/390 under VM:

```
In swapper task - not syncing
HCPGIR450W CP entered; disabled wait PSW 000A0000 800355F8
```

It is possible this was the result of Linux not being shut down properly. To work around this, boot with the command:

```
ipl clear
```

Thanks to Dr. Holger Smolinski for his post about this problem to the LINUX-VM list server.

E.7 Use the `-c` flag with the `ping` command

From the HMC, you might want to use the `ping` command after the `netsetup` script has been executed, because using the usual `ping host` command will cause it to run forever as `Ctrl-C` cannot be entered. Use `ping -c count host` where `count` is a number of packets to send. Typically this will be 3 or 5.

The same consideration applies using the `ping` command from the Linux console when running Linux in a virtual machine, since it is not always easy to remap the keyboard to produce a `^c` key sequence.

E.8 Linux under VM - can't find the vertical bar on the keyboard

You may need the vertical bar character in order to use it in a CMS Pipeline. This can be accomplished with the following CMS commands:

```
SET INPUT | 7C
SET OUTPUT 7C |
```

If the vertical bar is not present on your keyboard, or you are unable to map it, then you can set another key to generate the correct hexadecimal representation.

E.9 Rerun `silob` after changing the kernel parameter file

If you changed the kernel parameter line and rebooted, you may still come up with the old kernel parameter file definitions. You must rerun the `silob` command in order for these changes to take effect.

E.10 Linux under VM - reserve the minidisk

During a boot of Linux under VM, you may get this message from the VM minidisk driver:

```
mnd: device_number is not reserved
```

To avoid this, reserve the minidisk via the CMS `RESERVE` command.

E.11 Linux under VM - format the minidisk

During a boot of Linux under VM, you may also get the following messages:

```
mnd: Cannot read label of device_number - is it formatted?
mndn: register device at major 5F with 0 blocks 512 blksize
```

```
/boot/ipleckd.boot is not on device (94/0) but on (1/0)
```

To avoid this, format the minidisk via the CMS `FORMAT` command.

E.12 Linux under VM - IPL hangs

If your IPL of Linux under VM hangs, it may be because the kernel boot parameter file does not correctly reflect the Linux virtual machine's disk configuration. Adding or detaching a disk device or redefining a disk device number can cause this error.

If you get this symptom, carefully check your kernel parameter file entries against the VM configuration and work out which device names Linux will associate with S/390 device numbers.

E.13 Device not registered by the kernel

The following message may appear during boot:

```
mmd: Cannot acquire I/O irq of virtual_device_number for paranoia reasons, skipping
```

If this occurs, check that you have not defined the same device `cua` to both the DASD and the minidisk driver at the same time (which can cause either the `cannot acquire` message, or a message that the device is producing unsolicited interrupts, depending on who got what first).

E.14 Cannot mount file system - block sizes not the same

If you are trying to mount a file system, you may get the following error:

```
root@linux5 /]# mount /dev/dasda1 /mnt/dasda
mount /dev/dasda1 /mnt/dasda
mount: wrong fs type, bad option, bad superblock on /dev/dasda1, or too many mounted file systems
```

This may be because the device was low-level formatted at 4096 bytes, but `mke2fs` was used with a different block size. The default for `mke2fs` is to use 1024 byte blocks.

The solution is to use the `-b 4096` flag with the `mke2fs` command.

E.15 Disk device ranges in kernel parameter file

Devices managed by the DASD device driver may be specified by device number range. However, devices managed by the VM minidisk driver must be itemized individually and separated by commas.

The following example shows the differences in the kernel parameter file:

```
dasd=200-203 mdisk=300,301,302
```

E.16 RAM disk full

When you have the kernel booted with the RAM disk as the root file system, you may find that you can't create a directory or write a file on the RAM disk. This is because the RAM disk, as delivered, is very full. You may need to erase one or two files to make sufficient space to complete the next step of your install.

To avoid the potential problem, once the tarball is installed and exploded you can reboot with the kernel parameter changed to use that as the root file system.

E.17 The Virtual CTC connection does not start

When a CTC link to the VM TCP/IP stack is initialized, it can take a few seconds to establish the connection.

If the link fails to start, check the following:

- The CTC pairs are coupled (use the #CP Q V CTC command from the Linux virtual machine)
- The coupling order is correct - see Table 10 on page 103.

If for any reason the TCP/IP service machine is bounced, this will break the CTC coupling to the Linux virtual machine. You will need to enter CP COUPLE commands prior to restarting the link with a Linux `ifconfig` command; refer to the following example:

```
ifconfig ctc0 down
#CP COUPLE 808 TCPIP 809
#CP COUPLE 809 TCPIP 808
ifconfig ctc0 9.12.9.184 pointopoint 9.12.9.178 netmask 255.255.255.0
mtu 8192
```

E.18 Bad superblock

When you boot Linux, you may get the following type of error when it is mounting file systems:

```
Checking root filesystem
ext2fs_check_if_mount: No such file or directory while determining
whether /dev/dasdb1 is mounted.
(null):
The superblock could not be read or does not describe a correct ext2
filesystem. If the device is valid and it really contains an ext2
filesystem (and not swap or ufs or something else), then the superblock
is corrupt, and you might try running e2fsck with an alternate
superblock:
    e2fsck -b 8193 <device>
```

Check if the device or partition specified has had a filesystem created on it--have you specified the name of a device instead of a partition, or vice versa?

In this example the ext2 file system was created on /dev/dasdb, not on the partition /dev/dasdb1.

E.19 Error when running dasdfmt

When running `dasdfmt` on a small minidisk, you may get this error:

```
fixpoint divide exception: 0009
CPU:    0
Process dasdfmt (pid: 128, stackpage=020E1000)

User PSW:    0709e000 c00c248e
task: 020e0000 tss: 020e02d8 ksp: 020e1ca0 pt_regs: 020e1f68
User GPRS:
7ffff988 400c248c 00000003 400c4400
7ffffb48 00000000 00400c70 7ffffd68
7ffffc88 7ffffd68 00000000 7ffffa68
c010bb10 80402a80 80402eaa 7ffffa68
User ACRS:
00000000 00000000 00000000 00000000
00000001 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
Kernel BackChain CallChain  BackChain CallChain
          020e1ca0 00392600
Segmentation fault
```

The minimum size for a minidisk that can be successfully formatted by `dasdfmt` is around twenty (20) 3390 cylinders.

E.20 minidisk.sh

Linux for S/390 does not save the minidisk information with the device nodes in `/proc/dasd/devices`. In the bootup messages, this information is given in the following form:

```
[root@linux6 /root]# dmesg | grep ": dasd"
dasda: (nonl)/      : dasda1
dasdb: (nonl)/      : dasdb1
dasdc: (CMS1)/Z-DISK: dasdc1 (CMS)
dasdd: (CMS1)/LXX121: dasdd1 (CMS)
dasde: (CMS1)/LIN191: dasde1 (CMS)
dasdf: (nonl)/      : dasdf1
dasdg: (nonl)/      : dasdg1
dasdh: (MDSK)/      : dasdh1
dasdi: (nonl)/      : dasdi1
```

We wrote a script `minidisk.sh` that reads this information and combines it with the information in `dmesg`. You could, for example, call the script from `/etc/rc.d/rc.local` to save the information in `/var/log/dasd.devices`:

```
minidisk.sh > /var/log/dasd/devices
```

Here is the `minidisk.sh` shell script:

```
#!/bin/bash
#
dmesg|grep ": dasd" >/tmp/minidisk.sh.$$
grep -v "MAJ" /proc/dasd/devices >>/tmp/minidisk.sh.$$

cat /tmp/$0.$$ | sed "s%(% %g" \
    | sed "s%)% %g" \
    | sed "s%/dev/% %g" \
    | sed "s%/ %g" \
    | awk '
BEGIN {i=0 }
###      { print "1=", $1 " 2=", $2 " 3=", $3 " 4=", $4 " $5=", $5 "
$6=", $6; }
$1 ~ /dasd/ { DASD[NR]=substr($1,1,5); TYP[NR]=$2; LAB[NR]=$3;
PART[NR]=$4; }

$1 !~ /dasd/ { CUA[$4]=$1; MAJ[$4]=$2; MIN[$4]=$3; BLK[$4]=$5; }
```

```

        END { for (i=1; i<=NR/2; i++) {
                print "CUA="CUA[DASD[i]] " DASD="DASD[i] " TYP="TYP[i] "
MAJ="MAJ[DASD[i]] " MIN="MIN[DASD[i]] " PART="PART[i] " BLK="BLK[DASD[i]] "
LAB="LAB[i];
        }
    }
}

```

Following is an example of the script's output:

```

[root@linux6 /root]# ./minidisk.sh
CUA=0500 DASD=dasda TYP=nonl MAJ=94 MIN=0 PART=dasda1 BLK=4096 LAB=:
CUA=0600 DASD=dasdb TYP=nonl MAJ=94 MIN=4 PART=dasdb1 BLK=4096 LAB=:
CUA=019F DASD=dasdc TYP=CMS1 MAJ=94 MIN=8 PART=dasdc1 BLK=4096 LAB=Z-DISK:
CUA=019D DASD=dasdd TYP=CMS1 MAJ=94 MIN=12 PART=dasdd1 BLK=4096 LAB=HELP!::
CUA=0191 DASD=dasde TYP=CMS1 MAJ=94 MIN=16 PART=dasde1 BLK=4096 LAB=LIN191:
CUA=0200 DASD=dasdf TYP=nonl MAJ=94 MIN=20 PART=dasdf1 BLK=4096 LAB=:
CUA=0300 DASD=dasdg TYP=nonl MAJ=94 MIN=24 PART=dasdg1 BLK=4096 LAB=:
CUA=0400 DASD=dasdh TYP=MDSK MAJ=94 MIN=28 PART=dasdh1 BLK=4096 LAB=:
CUA=0192 DASD=dasdi TYP=nonl MAJ=94 MIN=32 PART=dasdi1 BLK=4096 LAB=:

```

E.21 The script with the networking questions is gone

The script `netsetup` is run at boot time in order to ask the user the basic questions needed to start the network, and then this script deletes itself.

In this section, for your reference, we provide the entire `netsetup` script:

```

#!/bin/bash

#
# readln reads a line into $ans.
#
function readln () {
    echo -n "$1"
    IFS='@' read ans || exit 1
    ans=`echo $ans | sed -e 's/^ *//'^`
}

#
# yes_no reads either a yes or a no into $ans
#
function yes_no () {
    while ;; do
        readln "$1"
        case "$ans" in

```

```

        [yY] | [yY]es) ans=yes
                break;;
        [nN] | [nN]o) ans=no
                break;;
    esac
done
}

echo
echo "Welcome to Linux for S/390"

confok=0
while [ $confok = 0 ]; do
    yes_no "Is your machine connected to a network (Yes/No) ? "
    if [ "$ans" = "yes" ]; then
        while :; do
            echo "Select the type of your network device"
            echo "1) for osa token ring"
            echo "2) for osa ethernet"
            echo "3) for channel to channel"
            echo "4) for escon channel"
            readln "Enter your choice (1-4): "
            case "$ans" in
                1)
                    ip_dev=tr0
                    echo "Please type in the options for the lcs module, e.g. to
select"
                    echo "relative port 1 on device 0xfd00 you should enter: "
                    echo "noauto=1 devno_portno_pairs=0xfd00,1"
                    readln "lcs parameter: "
                    lcs_parm=$ans
                    break;;
                2)
                    ip_dev=eth0
                    echo "Please type in the options for the lcs module, e.g. to
select"
                    echo "relative port 1 on device 0xfd00 you should enter: "
                    echo "noauto=1 devno_portno_pairs=0xfd00,1"
                    readln "lcs parameter: "
                    lcs_parm=$ans
                    break;;
                3)
                    ip_dev=ctc0
                    break;;
                4)
                    ip_dev=escon0
                    break;;
            esac
        done
    fi
done

```

```

        esac
    done
    readln "Please enter your host name: "
    ip_host=$ans
    readln "Please enter your IP address: "
    ip_addr=$ans
readln "Please enter the net mask: "
    ip_netmask=$ans
    if [ "$ip_dev" = "ctc0" -o "$ip_dev" = "escon0" ]; then
        readln "Please enter the IP address of your peer: "
        ip_peer=$ans
        ip_gateway=$ans
    else
        readln "Please enter the broadcast address: "
        ip_broadcast=$ans
        readln "Please enter the gateway address: "
        ip_gateway=$ans
    fi
    readln "Please enter the net address: "
    ip_network=$ans
    readln "Please enter the IP address of the DNS server: "
    ip_dns=$ans
    readln "Please enter the DNS search domain: "
    ip_search=$ans
    echo
    echo "Configuration will be:"
    if [ "$ip_dev" = "tr0" -o "$ip_dev" = "eth0" ]; then
        echo "LCS parameter      : $lcs_parm"
    fi
echo "Host name          : $ip_host"
    echo "IP address         : $ip_addr"
    echo "Net mask           : $ip_netmask"
    if [ "$ip_dev" = "ctc0" -o "$ip_dev" = "escon0" ]; then
        echo "Peer IP address    : $ip_peer"
    else
        echo "Broadcast address: $ip_broadcast"
        echo "Gateway address   : $ip_gateway"
    fi
    echo "Net address       : $ip_network"
    echo "DNS IP address    : $ip_dns"
    echo "DNS search domain: $ip_search"
    yes_no "Is this correct (Yes/No) ? "
    if [ "$ans" = "yes" ]; then
        cat > /etc/sysconfig/network <<EOF
NETWORKING=yes
FORWARD_IPV4=no
HOSTNAME=$ip_host

```

```

GATEWAYDEV=$ip_dev
GATEWAY=$ip_gateway
EOF
    if [ "$ip_dev" = "ctc0" -o "$ip_dev" = "escon0" ]; then
        cat > /etc/sysconfig/network-scripts/ifcfg-$ip_dev <<EOF
DEVICE=$ip_dev
USERCTL=no
ONBOOT=yes
BOOTPROTO=none
REMIP=$ip_peer
NETWORK=$ip_network
NETMASK=$ip_netmask
IPADDR=$ip_addr
EOF
    else
        cat > /etc/sysconfig/network-scripts/ifcfg-$ip_dev <<EOF
DEVICE=$ip_dev
USERCTL=no
ONBOOT=yes
BOOTPROTO=none
BROADCAST=$ip_broadcast
NETWORK=$ip_network
NETMASK=$ip_netmask
IPADDR=$ip_addr
EOF
        if [ "$lcs_parm" = "" ]; then
            cat >> /etc/conf.modules <<EOF
alias $ip_dev lcs
EOF
        else
            cat >> /etc/conf.modules <<EOF
alias $ip_dev lcs
options lcs $lcs_parm
EOF
        fi
    fi
    cat > /etc/resolv.conf <<EOF
search $ip_search
nameserver $ip_dns
EOF
    hostname $ip_host
    confok=1
fi
else
    confok=1
fi
done

```

```
rm /etc/rc.d/init.d/netsetup
rm /etc/rc.d/rc3.d/S00netsetup
exit
```

E.22 MTU size problems

Depending upon the network driver you are defining (for example LCS, CTC, IUCV), it is important to check with your network administrator when configuring these types of links to ensure the maximum transmission unit (MTU) parameter is set correctly.

Since this parameter specifies the largest packet that can be sent on a given physical medium, problems can occur in accessing network resources if it is set incorrectly.

Appendix F. Special notices

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

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 the International Business Machines Corporation in the United States and/or other countries:

AFP	AIX
APPN	AS/400
AT	C/MVS
CICS	CT
CUA	Current
DB2	DB2 Connect
DB2 Universal Database	DRDA
ECKD	ESCON
FICON	Home Director
Hummingbird	IBM ®
IBM.COM	Language Environment
Linux for S/390®	MQSeries
Multiprise	Netfinity
Network Station	Nways
OpenEdition	OS/2
OS/390	OS/400
Parallel Sysplex	PR/SM
RACF	RAMAC
RETAIN	RMF
RS/6000	S/370
S/390	SP
System/370	System/390
TCS	ThinkPad
VisualAge	VM/ESA
VSE/ESA	VTAM
WebSphere	XT
3090	400
Redbooks	
Redbooks Logo 	

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.

Linux is a registered trademark of Linus Torvalds.

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

Appendix G. Related publications

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

G.1 IBM Redbooks

For information on ordering these publications see “How to get IBM Redbooks” on page 511.

- *Open Source Software on OS/390*, SG24-5944
- *Porting UNIX Applications to OpenEdition for VM/ESA*, SG24-5458
- *TCP/IP Solutions for VM/ESA*, SG24-5459
- *VM/ESA Performance Tools*, GG24-4152
- *S/390 I/O Connectivity Handbook*, SG24-5303
- *IBM 2216 and Network Utility Host Channel Connections*, SG24-5303
- *Accessing OpenEdition MVS from the Internet*, SG24-4721
- *Getting Started with TCP/IP for VSE/ESA 1.4*, SG24-5626

G.2 IBM Redbooks collections

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at <http://www.redbooks.ibm.com/> for information about all the CD-ROMs offered, updates and formats.

CD-ROM Title	Collection Kit Number
System/390 Redbooks Collection	SK2T-2177
Networking and Systems Management Redbooks Collection	SK2T-6022
Transaction Processing and Data Management Redbooks Collection	SK2T-8038
Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
AS/400 Redbooks Collection	SK2T-2849
Netfinity Hardware and Software Redbooks Collection	SK2T-8046
RS/6000 Redbooks Collection (BkMgr)	SK2T-8040
RS/6000 Redbooks Collection (PDF Format)	SK2T-8043
Application Development Redbooks Collection	SK2T-8037
IBM Enterprise Storage and Systems Management Solutions	SK3T-3694

G.3 Other resources

These publications are also relevant as further information sources:

- *DNS and BIND*, by P. Albitz and C. Liu, published by O'Reilly, ISBN 1565925122
- *Computer Networks*, Third edition, by A. Tanenbaum, published by Prentice Hall, ISBN 0133499456
- *The Linux Network*, by F. Butzen and C. Hilton, published by IDG Books Worldwide, ISBN 155828589X
- *Unix Backup & Recovery*, by W. Curtis Preston, published by O'Reilly, ISBN 1565926420
- *Samba: Integrating UNIX and Windows*, by John D. Blair, published by SSC, ISBN 1578310067
- *Using Samba*, by Robert Eckstein, David Collier-Brown and Peter Kelly, published by O'Reilly, ISBN 1565924495
- *S/370 and S/390 Optical Media Attach/2 Technical Reference, SC53-1201*
- *S/370 and S/390 Optical Media Attach/2 Users Guide, SC53-1200*
- *VM/ESA Planning and Administration, SC24-5750*
- *VM/ESA V2R4.0: TCP/IP Function Level 320 Planning and Customization, SC24-5847*
- *VM/ESA V2R4.0 TCP/IP FL320 User's Guide, SC24-5848*
- *VM/ESA XEDIT User's Guide, SC24-5779*
- *VM/ESA XEDIT Command and Macro Reference, SC24-5780*
- *VM/ESA V2R4.0 VMSES/E Introduction and Reference, GC24-5837*
- *VM/ESA V2R4.0 CP Command and Utility Reference, SC24-5773*
- *VM/ESA V2R4.0 Planning and Administration, SC24-5750*
- *VM/ESA V2R4.0 Performance, SC24-5782*
- *VM/RSCS V3R2.0 Planning and Installation, SH24-5219*
- *OS/390 V2R6.0 NFS Customization and Operation, SC26-7253*
- *OS/390 TCP/IP OpenEdition User's Guide, GC31-8305*
- *TCP/IP for VSE/ESA User's Guide, SC33-6601*
- *Intel Architecture Software Developer's Manual Volume 3*, Intel order number 243192

- *ESA/390 Principles of Operation, SA22-7021*

G.4 Referenced Web sites

These Web sites are also relevant as further information sources:

- <http://metalab.unc.edu/mdw/HOWTO/DNS-HOWTO-4.html>
- <http://www.s390.ibm.com/linux/installfest/l390fin2.pdf>
- <http://www.s390.ibm.com/linux/installfest/l390fpr3.pdf>
- <http://linux.s390.org/download/rpm2html/>
- <http://www.openssh.com/history.html>
- <http://www.nic.com/~dave/SecurityAdminGuide/SecurityAdminGuide.html>
- <http://www.securityportal.com/lasg/>
- <http://www.kernel.org/pub/linux/kernel/v2.2/>
- <http://www.kernel.org/pub/linux/kernel/people/alan/>
- <http://www.linux.s390.org>
- <http://www.ibm.com/s390/corner>
- <http://www.ietf.cnri.reston.va.us>
- <http://www.gnu.org/philosophy/free-sw.html>
- <http://www.linuxbase.org>
- <http://linux390.marist.edu>
- <http://linux.s390.org>
- <http://www.suse.com/PressReleases/ibmsuse.html>
- http://www.turbolinux.com/news/pr/ibm_s390.html
- <http://www.debian.org/News/weekly/2000/15>
- <http://www.linuxfromscratch.org>
- <http://www.opensource.org>
- <http://www.linux.org>
- <http://www.apache.org>
- <http://www.apache-ssl.org>
- <http://www.kernel.org/pub/linux/kernel/v2.2>
- <http://www.kernel.org/pub/linux/kernel/people/alan>
- <http://www.solucorp.qc.ca/linuxconf>

- <http://web.mit.edu/tytso/www/linux/ext2intro.htm>
- <http://www.backupcentral.com>
- <http://www.freshmeat.net>
- <http://oss.software.ibm.com>
- <http://oss.software.ibm.com/developerworks/opensource/jfs/>
- <http://lclint.cs.virginia.edu>
- <http://lclint.cs.virginia.edu/index.html>
- <http://www.lightlink.com/hessling/THE/index.html>
- <http://metalab.unc.edu/mdw/HOWTO/CVS-RCS-HOWTO.html>
- <http://metalab.unc.edu/mdw/HOWTO/CVS-RCS-HOWTO-1.html>
- <http://www.linuxdoc.org/HOWTO/Vim-HOWTO.html>
- <http://www.linuxstart.com/applications/texteditorsreaders.html>
- <http://www.cs.cornell.edu/Info/People/raman/emacspeak/emacspeak.html>
- <http://www.gnu.org/software/ddd>
- <http://www.kdevelop.org>
- <http://sal.kachinatech.com>
- <http://www.dcs.port.ac.uk/~lesterc/humour/G-newpls.html>
- <http://www.lancs.ac.uk/people/cpaap/pfe>
- <http://www.first.gmd.de/cogent/catalog/kits.html>
- <http://www.perl.com>
- <http://www.proftpd.net/>
- <http://www.proftpd.net/download.html>
- <http://linux.s390.org/download/ftp/RPMS/s390>
- <http://linux.s390.org/pub/ThinkBlue/RPMS/s390/>
- http://hamster.wibble.org/proftpd/proftpd_userguide.htm
- <http://www.linux.s390.org/download>
- <http://www.samba.org>
- <http://rootshell.com>
- <http://www.insecure.org>
- <http://www.cert.org>
- <http://lsap.org>

- <http://www.suse.de/de/support/security>
- <http://www.securityfocus.com>
- <http://www.rfc-editor.org>
- <http://www.snipix.freemove.co.uk/hercinst.htm>
- <http://www.elink.ibm.ibm.com/pbl/pbl>
- <http://www.ibm.com/privacy/yourprivacy>
- <http://www.gnu.org>
- <http://www.towergroup.com>
- <http://www.opensource.org/halloween/halloween1.html>
- <http://www.embedded-linux.org>
- http://www.ibm.com/software/webservers/appserv/download_linux.html
- <http://www.ibm.com/vadd>
- <http://www.ibm.com/s390/appsource/>
- <http://www.ibm.com/spc>
- <http://www.lotus.com/home.nfs/welcome/dominolinux>
- <http://www.ibm.com/linux>
- <http://www.ibm.com/developerWorks>
- <http://www.linuxdoc.org>
- <http://www.adelaide.net.au/~rustcorp/ipfwchains/ipfwchains.html>
- <http://www.chiark.greenend.org.uk/~sgtatham/putty/>
- <ftp://ftp.kernel.org>
- ftp://ftp.cert.org/pub/tech_tips/anonymous_ftp_config
- ftp://ftp.cert.org/pub/tech_tips/anonymous_ftp_abuses
- <ftp://ftp.suse.com/pub/suse/s390/SuSEbookS390.pdf>

How to get IBM Redbooks

This section explains how both customers and IBM employees can find out about IBM Redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** <http://www.redbooks.ibm.com/>

Search for, view, download, or order hardcopy/CD-ROM Redbooks from the Redbooks Web site. Also read redpieces and 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.

- **E-mail Orders**

Send orders by e-mail including information from the IBM Redbooks fax order form to:

	e-mail address
In United States	usib6fpl@ibmmail.com
Outside North America	Contact information is in the "How to Order" section at this site: http://www.elink.ibm.com/pbl/pbl

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the "How to Order" section at this site: http://www.elink.ibm.com/pbl/pbl

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the "How to Order" section at this site: http://www.elink.ibm.com/pbl/pbl

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the Redbooks Web site.

IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and Redbooks by accessing the IBM Intranet Web site at <http://w3.itso.ibm.com/> and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access MyNews at <http://w3.ibm.com/> for redbook, residency, and workshop announcements.

Index

Symbols

/etc/fstab 75
/etc/group 200
/etc/passwd 198
/etc/sysconfig/network 76

Numerics

2216
description 262
features 263
protocols 263
LCS (LAN Channel Station) 263
LSA (Link Services Architecture) 263
MPC (Multi-Path Channel) 263

A

access 278
control 439
Activation profile 59, 82
alias 445
Apache 405
CGI 414
customization 410
installation 406
logging 414
operation 412
security 417
server configuration settings 411
SSI 414
SSL 417
startup 413
stopping 413
tar file 406
virtual hosting 412
Apache Web server 405
Application Framework 14
Application Service Providers (ASPs) 7
authoritative server 352

B

backup 211
commands 213
cpio 219
strategies 211

tar 222
using dump 214
Berkeley Internet Name Domain 353
BIND (Berkeley Internet Name Domain) 351
bind8 353
components 354
bison 332
boot device, setup 239
BSD 11
builtin 445
bzip2 317

C

caching-only 350
name server 352, 354
change mode
command 74
Channel-to-channel (CTC)
Description 264
device 276
chmod 202
chmod swap
directory 74
client 277
CMS
FORMAT 88, 102
RESERVE 88
commands
bzip2 317
cat 165, 178
cd 162
chmod 120, 191
chown 202
compress 318
cp 76
cpio 219
crontab 207
dasdfmt 70, 115, 180, 186
dd 74, 177, 190
dump 214, 217
e2fsck 188
export 136
find 200, 221, 224, 444
free 450
ftp 116
fuser 451
gcc 319

- grep 211
- groupadd 200
- groupdel 201
- groupmod 201
- gzip 317
- ifconfig 276
- info 442
- insmod 60, 141, 181
- kill 205
- ln 121, 167
- locate 444
- lpr 433
- make 244, 320, 323
- make clean 246
- make dep 248
- make image 248
- make menuconfig 247
- man 202, 435, 441
- mkdir 71, 115, 240
- mke2fs 71, 115, 194, 240
- mknod 68, 183
- mkswap 74, 120, 189
- modprobe 181
- mount 72, 115, 162, 187, 194
- mv 120
- passwd 198
- pinfo 442
- ping 63
- procmeter 450
- ps 448
- pstree 448
- pwck 200
- pwd 445
- restore 216, 218
- route 276
- rpm 227, 231
- sed 323
- shutdown 80, 175
- silo 195, 240, 250
- swapoff 189
- swapon 74, 120, 189
- tar 222, 234, 318
- touch 221
- uncompress 318
- unzip 319
- useradd 198
- userdel 199
- usermod 199
- Yacc 332

- YAST 142, 208
- YAST2 209
- zip 319
- Common Gateway Interface (CGI) 414
- compression 280
- configuration files 355
- Configure and compile the kernel 246
- connection oriented 268
- Coordinated Universal Time (UTC) 120
- CP directory entry 90
- cpio 219
- Creating 3490 tape 53
- Creating a swap file
 - chmod swap directory 74
 - mkswap command 74
 - swapon command 74
- Creating an IPLable tape 48
- Creating IPLable 3490 tape using VM 55
- Creating mount points
 - mkdir command 71
- CTC 289
 - 3088 264
 - ESCON 264
 - FICON 264
- CTC connection 93
- Customization 410
- Customizing the root file system
 - /etc/fstab 75

D

- daemons 272
- datagrams 268
- DAU 445
- DB2 309
- dd command 74
- ddd 328
- debugger 327
- define virtual devices 92
- df 451
- DHCP 269
- disk usage 451
- DISPLAY 439
- distributions
 - Debian 41
 - Marist 41
 - SuSE 41
 - Turbo-Linux 41
- DNS 270, 349

- database 351
 - server 352
- du 451
- Dynamic IP 269

E

- e-business 6
- echo command 77
- editors 321
- emacs 322
- encrypted connections 437
- export 368, 378
 - DISPLAY 278

F

- File Transfer Protocol 274
- Firewall 421
 - customization 424
 - installation 422
 - operation 429
- flex 331
- Format DASD as swap space
 - mkswap command 74
- Format dasd as swap space
 - swapon command 74
- forwarding 354
- free 450
- FTP 309, 335
 - access control 336
 - anonymous 335
 - binaries to host system 52
 - daemon 338
 - security 336
 - tools 337
- ftpaccess 339
- ftpgroups 341
- ftphosts 341
- ftpshut 343
- ftpusers 341
- function 445
- fuser 451

G

- g++ 319
- gateways 275
- gcc 319
- gdb 327

- generic listener 272
- GID 198
- GNU Public License 12
- gprof 330
- groupadd 200
- groupdel 201
- groupmod 201
- grpck 202
- gzip 317

H

- help 442
- hints file 355
- host.conf 359
- hostile 441
- HOSTNAME 271
 - command 271
- hosts 269
- howtos 443

I

- IBM Public License 13
- IBM WebSphere Application Server (WAS) 20
- ifcfg 75
- ifconfig 275
- in.ftpd 273
- incremental backup 224
- inetd 271, 272
 - inetd.conf 272, 273
- info command 442
- init process 171
- inittab 172
- interactive mode 360
- Internet Control Message Protocol 268
- Internet Protocol 350
- Internet Service Providers (ISPs) 7
- InterNET services Daemon 272
- IPL
 - from DASD 80
 - messages 59
 - record 195
- IUCV 291
 - connection 122
 - definition permanent 296
- IUCV connections 93

J

joe 322
jove 322

K

kernel
 patching 243
 source URL 225
 update 226
kernel initialization 168

L

LAN Channel Station (LCS) 288
LCS driver 288
lease 269
lex 331
lilo 195
Load 58
Load from CDROM 139
localhost 360
logfiles 435, 447
login
 remote 437
loopback devices 275
Low level format
 dasdfmt command 70
LPAR 285
ltrace 329

M

major number 177
man 441
Marist distribution 41
memory
 usage 450
messages 435, 447
minor number 177
mkdir command 71
mke2fs command 71
mkswap command 74
mount
 command 72, 194
 NFS 369
MTU 294
MTU size 114
Multiprise 3000 139
mvlogout 378

MVS system 352
mvslogin 378

N

Name server 358
Name servers 350
named 355
named.conf 355
Native S/390 Installation 45
 Activation profile 82
 Creating & activating swap space 73
 Format DASD as swap space 74
 Recommendations 74
 Creating 3490 tape 53
 Creating a file system
 mke2fs command 71
 Creating a new kernel 77
 Creating a swap file
 chmod swap directory 74
 dd command 74
 mkswap command 74
 swapon command 74
 Creating an IPLable tape 48
 Parameter line file 49
 Creating IPLable 3490 tape using VM 55
 Creating mount points
 mkdir command 71
 Customizing the root file system 75
 /etc/fstab 75
 Network configuration 75
DASD 68
 limitations 68
 major number/minor number 66
echo command 77
Format DASD as swap space
 mkswap command 74
Format dasd as swap space
 swapon command 74
FTP binaries to host system
 OS/2 53
 Win95/NT 52
Hardware management console (HMC)
 Activation profile 59
 IPL 57
 issue commands 62
 Load 58
 Operating messages 59
 Reset clear 57

- Hardware preparation 46
- IPL messages
 - /var/log/dmesg 65
 - IPL messages 65
- Low level format of DASD 66
- Mknod 68
- Network configuration 60
- Parameter line file
 - DASD statement 49
 - DASD statement implications 50
 - lpldelay statement 50
 - Root statement 49
- Parmline dataset attributes 55
- Uncompress the file system
 - tar command 72
- Upload and customize file system 71
- Verifying the IPL 64
- Writing IPL info to DASD
 - Create parameter file 78
 - noinitrd 78
 - silo command 78
 - Write IPL record 78
- Writing parmline to 3490 tape 56
- ndc
 - restart 360
 - start 359
 - stop 359
- netsetup 287
- netstat 275
- network 275
 - clients 275
- Network configuration
 - /etc/conf.modules 76
 - /etc/resolv.conf 76
 - /etc/sysconfig/network 76
 - network-scripts
 - ifcfg 75
- network definitions 103
- Network Layer 267
- network script 272
- network scripts
 - ifcfg 75
- networking 269
- networking services 270
- NFS 367
 - mount 369
- nfsd 367
- noinitrd 78
- non-authoritative 361
- NS (Name Server) 357
- nslookup 360
- nsswitch.conf 359

O

- obscurity 435
- open source 2, 11
- Open Systems Adapter 2 (OSA-2)
 - Asynchronous Transfer Mode (ATM) 259
 - Ethernet 259
 - Fiber Distributed Data Interface (FDDI) 259
 - Token Ring 259
- openssh 279
- Operating messages 59
- OS/309
 - inetd 306
- OS/390 300
 - connectivity 300
 - daemons 302
 - FTP 309
 - NFS 309
 - REXEC 311
 - RSH 311
 - TCP/IP 300
 - Telnet 309
- OSA 286
- OSA/SF
 - graphical user interface (GUI) support 261
 - windows support 261
- OSA-2 286
- OSA-2 ATM
 - feature 260
 - multimode fiber type 261
 - single mode fiber type 261
- OSA-2 Ethernet/Token Ring (ENTR)
 - feature 260
- OSA-2 Fast Ethernet (FENET)
 - auto negotiation 261
 - duplex mode 261
 - feature 261
 - LAN speed 261
- OSA-2 FDDI
 - dual ring attachment 260
 - feature 260
 - optical bypass switch 260
- OSA-2 modes
 - ATM IP Forwarding 262
 - HPDT ATM Native Mode - APPN and TCP/IP

- 262
- SNA Mode (non-shared port) 261
- SNA Mode (shared port) 261
- TCP/IP and SNA Mixed Mode (shared port) 261
- TCP/IP and SNA Mixed Mode for OSA-2 ATM LAN Emulation (ATM LE) 261
- TCP/IP Passthru Mode (non-shared port) 261
- TCP/IP Passthru Mode (shared port) 261

P

- packages 353
- Parallel Enterprise Server 139
- Parameter file 481
 - 3215 485, 486
 - CTC/Escon 484
 - Ctc/Escon
 - Syntax 484
 - DASD 481, 482
 - Mdisk 482, 483
 - Xpram 483, 484
- Parameter line file 49
- Parmline dataset attributes 55
- passwd 203
- patches
 - URL 225
- Perl 320
- pico 322
- ping 274
- portmap 367
- ports 272
- primary name server 350, 352
- Printing 431
 - using OS/390 resources 433
 - using VM resources 432
- proc filesystem 447
- profiling 330
- ProFTPD 346
- PROP 126
- protocols 267, 270
- ps 448
- pseudo-terminal 273
- PTR (Pointer) 357

R

- RACF
 - UNIXPRIV 373
- Raw command
 - dd command 74
 - rc.sysinit 272
 - remote login 437
 - Reset clear 57
 - resolv.conf 358
 - resolvers 350
 - Resource Records 357
 - restore 216
 - RFC
 - Request For Comments 443
 - root 197, 351
 - root.hints 356
 - routing 275, 349
 - RPM 357
 - database 227
 - filenames 226
 - for S/390, URL 225
 - install 230
 - query options 228
 - update 230
 - RS.INTERNIC.NET 357
 - run level 165

S

- S/390 rpm database 353
- SAF 378
- SAFEXP 378
- Samba 389
 - customization 393
 - Installation 389
 - startup 392
 - SWAT 393
 - tools 399
- San Francisco Framework 20
- scp 437
- secondary name server 350, 352
- Secure SHell 279
- security 435
 - Internet sites 439
- sed 323
- server 277
- services 273
- services, unnecessary 435
- SET CLOCK (SCK) 120
- shadow password 438
- shell 203
 - builtin 445
 - function 445

- keyword 445
- showattr 378
- shutdown 175
- Shutting down Linux 80
- silo 195
 - command 78
- Simple Object Access Protocol (SOAP) 9
- Skills 45
- SNA 261
- SOA (Start of Authority) 357
- software dependencies
 - RPM dependencies 229
- software update 226
- ssh 279
- SSL 417
- Start of Authority (SOA) 350
- Static IP 269
- strace 329
- super-server 272
- superuser 197
- SuSE 133
 - bootup the system 163
 - FTP 137
 - loading the DASD driver 141
 - network connection 136
 - network parameter 140
 - NFS 137
- SuSE distribution 41
- swapon command 74
- SWAT 393
- sysinit 174
- syslogd 274
- SYSTEM CONFIG 120

T

- tar 222, 318
 - command 72
 - installation with tar 233
- TCP 261
- TCP/IP 267
 - protocols 270
- tcpd 273, 337, 438
- telnet 272, 273, 277
- Telnetd 273
- texinfo 442
- TFTP 335
- Time of Day (TOD) 120
- tools 274

- top 448
- tr0 75
- Transmission Control Program (TCP)
 - wrapper 438
- Transmission Control Protocol (TCP) 268
- Transport Layer 267
- Turbo-Linux distribution 41
- type of command 445

U

- UID 197
- Uncompress the file system
 - tar command 72
- UNIXPRIV 373
- upgrading 253
- uptime 450
- usage
 - disk 451
 - memory 450
- User Datagram Protocol 268
- useradd 198, 199
- Utility Service Provision 7

V

- Verifying the IPL 64
- vi 321
- virtual console 174
- VM dedicated DASD 88
- VM minidisk driver 88
- VM/ESA 85, 353
 - Booting from tape 98
 - Booting from the reader 98
 - COUPLE 103
 - create a swap file 119
 - CTC device pair 114
 - FTP 313
 - install root file system 115
 - Installing Linux for S/390 87
 - Linux network definitions 103
 - NFS 313
 - REXEC 313
 - RSH 313
 - Telnet 312
- VM/ESA overview 469
- VSE/ESA 353
 - FTP 315
 - NFS 314

W

w (command) 449
warnings 435, 447
wget 357
who 449
Why Linux 17
Write IPL record
 silo command 78
Writing parmline to 3490 tape 56
wwwrun 197

X

X11
 access control 439
xhost 439
xload 450
xosview 450
xpram driver 250
X-Windows 277, 280

Y

yacc 332
YaST 142

Z

zip 319
zone 350
zone file 357

IBM Redbooks review

Your feedback is valued by the Redbook authors. In particular we are interested in situations where a Redbook "made the difference" in a task or problem you encountered. Using one of the following methods, **please review the Redbook, addressing value, subject matter, structure, depth and quality as appropriate.**

- Use the online **Contact us** review redbook form found at ibm.com/redbooks
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Document Number	SG24-4987-00
Redbook Title	Linux for S/390
Review	
What other subjects would you like to see IBM Redbooks address?	
Please rate your overall satisfaction:	<input type="radio"/> Very Good <input type="radio"/> Good <input type="radio"/> Average <input type="radio"/> Poor
Please identify yourself as belonging to one of the following groups:	<input type="radio"/> Customer <input type="radio"/> Business Partner <input type="radio"/> Solution Developer <input type="radio"/> IBM, Lotus or Tivoli Employee <input type="radio"/> None of the above
Your email address: The data you provide here may be used to provide you with information from IBM or our business partners about our products, services or activities.	<input type="checkbox"/> Please do not use the information collected here for future marketing or promotional contacts or other communications beyond the scope of this transaction.
Questions about IBM's privacy policy?	The following link explains how we protect your personal information. ibm.com/privacy/yourprivacy/

IBM



Redbooks

Linux for S/390

(1.0" spine)
0.875" <-> 1.5"
460 <-> 788 pages



Redbooks

Linux for S/390

How can Linux exploit the strengths of S/390?

What different ways can Linux be installed on S/390?

Which Linux applications can run on S/390?

The strengths of S/390 are well known: rock-solid reliability, the ability to run multiple diverse workloads, and highly scalable technology make S/390 an ideal choice for hosting key e-business applications. Now Linux has joined the S/390 family of operating systems, bringing a wealth of open source applications, middleware, and trained developers to help you respond to your business challenges quicker than ever before.

This IBM Redbook will help you install Linux for S/390 in different environments, and documents basic system administration tasks that can help you manage your Linux for S/390 system. It also provides an introduction to a wide range of services, such as Samba, NFS, and Apache. You will learn what each service is, what it is capable of, and how to install it.

The book covers Linux for S/390 Marist distribution (2.2.15).

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

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

For more information:
ibm.com/redbooks

SG24-4987-00

ISBN 0738419141