Redbooks
ibm.com/redbooks

The Virtualization Cookbook for IBM z Systems Volume 3: SUSE Linux Enterprise Server 12

Lydia Parziale

Berthold Gunreben

Filipe Miranda

Paul W Novak

Ken Werner

z Systems

IBM

Redbooks

International Technical Support Organization

The Virtualization Cookbook for IBM z Systems Volume 3: SUSE Linux Enterprise Server 12

October 2015

SG24-8890-00

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

First Edition (October 2015)

This edition applies to Version 6, Release 3 of IBM z/VM, and to SUSE Linux Enterprise Server 12.

Contents

Notices

Trademarks

Preface

This IBM® Redbooks® publication is Volume 3 of a series of three books called *The Virtualization Cookbook for IBM z Systems*. The other two volumes are called:

- *The Virtualization Cookbook for IBM z Systems Volume 1: IBM z/VM 6.3*, SG24-8147
- *The Virtualization Cookbook for IBM z Systems Volume 2: Red Hat Enterprise Linux 7.1 Servers*, SG24-8303

It is suggested that you start with Volume 1 of this series, because IBM z/VM® is the base "layer" when installing Linux on IBM z Systems™. Volume 1 starts with an introduction, describes planning, and then describes z/VM installation into a two-node, single system image (SSI) cluster, configuration, hardening, automation, and servicing. It adopts a cookbook format that provides a concise, repeatable set of procedures for installing and configuring z/VM using the SSI clustering feature.

Volumes 2 and 3 describe how to *roll your own* Linux virtual servers on z Systems hardware under z/VM. The cookbook format continues with installing and customizing Linux.

Volume 3 focuses on SUSE Linux Enterprise Server 12. It consists of the following main chapters:

- Chapter 1, "Installing SUSE Linux Enterprise Server 12 on LNXADMIN" on page 3 describes how to install and configure SUSE Linux Enterprise Server 12 onto the *Linux administration system*, which does the cloning and other tasks.
- Chapter 2, "Automating SUSE Linux Enterprise Server 12 installation using the AutoYaST tool" on page 23 explains how to use AutoYaST2, which enables you to automatically install Linux using a configuration file.
- Chapter 3, "Creating appliances with KIWI" on page 37 explains how to create and use appliances and bootable images from configuration files.
- Chapter 4, "Servicing SUSE Linux Enterprise Server 12" on page 53 provides information about common tasks and tools available to service SUSE Linux Enterprise Server.

Description of volumes in this series

This book series consists of the following volumes:

- *The Virtualization Cookbook for IBM z Systems Volume 1: IBM z/VM 6.3*, SG24-8147 introduces the entire concept of Linux on the mainframe, introduces the system, and describes the z/VM platform, planning, installation, and configuration into a two-member SSI with z/VM 6.3.
- *The Virtualization Cookbook for IBM z Systems Volume 2: Red Hat Enterprise Linux 7.1 Servers*, SG24-8303 describes installing and customizing Red Hat Enterprise Linux.
- *The Virtualization Cookbook for IBM z Systems Volume 3: SUSE Linux Enterprise Server 12*, SG24-8890 describes installing and customizing SUSE Linux Enterprise Server 12.

Each volume has the following parts:

- Part 2, "Other topics" on page 57 includes chapters on the following subjects:
  - Live Guest Relocation (LGR) between SSI members
  - Configuring the Systems Management application programming interface (SMAPI)
  - Enabling IBM RACF® as the external security manager (ESM)
  - Monitoring z/VM and Linux
  - The Linux `systemd` suite of system management daemons, and libraries
  - Miscellaneous "recipes"
- Appendix A, "References, cheat sheets, and blank worksheets" on page 141 includes references, cheat sheets, and blank worksheets.
- Appendix B, "Additional material" on page 151.

Conventions

This section describes the conventions used in this book.

Font conventions used in this book

`Monospace and bold`	Commands entered by the user on the command line
`monospace`	Linux file, directories, and commands
`MONOSPACE CAPITALS`	z/VM files, virtual machine and minidisk names, and commands
`Monospace italics`	Values used to test this book, such as TCP/IP addresses. This font convention is used to signify that you should replace the *`example value`* with the correct value for your system or enterprise.

Command conventions used in this book

- IBM z/VM commands are prefixed with three Equal signs and a greater than symbol (**===>**)
- IBM z/VM XEDIT subcommands are prefixed with the same, but four Equal signs (**====>**)
- Linux commands running as root are prefixed with the Number sign (**#**)
- Linux commands running as non-root are usually prefixed with the Dollar sign (**$**)

Operating system releases used

The following releases of operating systems were used in the writing of this book:

z/VM 6.3:	General availability (GA) code, July 2013
SUSE Linux Enterprise Server 12:	GA code, 2015

Authors

This book was produced by a team of specialists from around the world working at the IBM International Technical Support Organization (ITSO), Poughkeepsie Center.

Lydia Parziale is a Project Leader for the ITSO team in Poughkeepsie, New York, with domestic and international experience in technology management, including software development, project leadership, and strategic planning. Her areas of expertise include Linux on z Systems and database management technologies. Lydia is a Certified IT Specialist with an MBA in Technology Management, and has been employed by IBM for over 25 years in various technology areas.

Berthold Gunreben is a Build Service Engineer at SUSE in Germany. He has 14 years of professional experience in Linux, and is responsible for the administration of the mainframe system at SUSE. In addition to his expertise with Linux on z Systems, he is also a Mainframe System Specialist certified by the European Mainframe Academy: http://www.mainframe-academy.de. His areas of expertise include high availability (HA) on Linux, Realtime Linux, automatic deployments, storage administration on the IBM DS8000®, virtualization systems with Xen, KVM, and z/VM, and documentation. Berthold has written extensively in many of the SUSE manuals.

Filipe Miranda is the Global Lead for Red Hat Enterprise Linux Servers for IBM z Systems and IBM Power Systems™ for Red Hat Inc.® His key responsibility is to help shape the overall Linux on z Systems and Power Systems strategy for Red Hat. With more than 13 years of experience in Linux and open source technologies, he joined Red Hat Inc. 9 years ago, and is based in Southern California, US. He holds two degrees, one in Data Communication from University of California Los Angeles, and another in Computer Science from the University Paulista in Sao Paulo, Brazil.

Paul W Novak is a member of the IBM Washington Systems Center z/VM and Linux zGrowth (formerly ATS) team in Endicott, New York. Paul came from IBM Service Delivery, where he served as the IBM Global Account webmaster on a team responsible for the implementation of IBM WebSphere® and IBM Collaboration Solutions (ICS) on Linux under z/VM in the world's largest production IBM web middleware environment. He is a Senior Certified IT Specialist with a BSBA in Management Information Systems, and has held positions in field services, user support, software development, enterprise hosting, and enterprise architecture. Paul is a fourth-generation IBMer with more than 20 years of Linux and open source technology experience.

Ken Werner is a Linux specialist working for the Systems unit in IBM Research & Development, Germany. He has seven years of professional experience with Linux on different platforms, and has contributed to various open-source projects, such as the GNU Debugger, libunwind, and OpenEmbedded. Currently, Ken works with the Linux on z Systems development team on continuous integration, providing the latest code to developers, testers, and performance evaluators.

Special thanks

Thanks to the following people for their contributions:

IBM ITSO Center Poughkeepsie
Dave Bennin, Rich Conway, Robert Haimowitz

IBM Endicott

Bruce Hayden
Marci Beach
Timothy Greer
Emily Hugenbruch
Brian Hugenbruch
Alan Altmark
Brian Wade
Susan Timashenka
Bill Bitner

IBM Böblingen

Steffen Maier
Pradeep Parameshwaran
Hendrik Brückner
Dominik Klein
Elisabeth Puritscher
Volker Sameske
Ekaterina Teplova

IBM Gaithersburg

Fred Bader

Red Hat Inc.

Chris Mackowski
Jan Stodola
Dan Horak

SUSE

Mike Friesenegger

Thanks to Michael MacIsaac for the original inception of this cookbook, and for his efforts in continually moving the cookbook forward over the years.

Thanks to many others in IBM Endicott and Poughkeepsie, and to the many who answered questions on the `Linux-390` and `IBMVM` list servers.

Thanks to the authors of the previous editions of this book:

- Authors of the previous IBM Redbooks edition, *The Virtualization Cookbook for IBM z Systems Volume 1: IBM z/VM 6.3*, SG24-8147, last updated 22 February 2011: Lydia Parziale, Marian Gasparovic, Berthold Gunreben, Michael MacIsaac, Filipe Miranda, and Daniel Ruutz
- Authors of the previous IBM Redbooks edition, *z/VM and Linux on IBM System z: The Virtualization Cookbook for SLES 11 SP1*, SG24-7931, last updated 22 February 2011: Marian Gasparovic and Michael MacIsaac
- Authors of the previous IBM Redbooks edition, *z/VM and Linux on IBM System z: The Virtualization Cookbook for Red Hat Enterprise Linux 6.0*, SG24-7932, last updated 18 February 2011: Brad Hinson and Michael MacIsaac

Now you can become a published author, too

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time. Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run two - six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Learn more about the residency program, browse the residency index, and apply online:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us.

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

- Use the online **Contact us** review Redbooks form:

  ibm.com/redbooks

- Send your comments in an email:

  redbooks@us.ibm.com

- Mail your comments:

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

Stay connected to IBM Redbooks

- Find us on Facebook:

  www.facebook.com/IBMRedbooks

- Follow us on Twitter:

  http://twitter.com/ibmredbooks

- Look for us on LinkedIn:

  www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  ibm.com/redbooks/redbooks.nsf/subscribe

- Stay current on recent Redbooks publications with RSS Feeds:

  ibm.com/redbooks/rss.html

Summary of changes

This section describes the technical changes made in this edition of the book, and in previous editions. This edition might also include minor corrections and editorial changes that are not identified.

Summary of changes for *The Virtualization Cookbook for IBM z Systems-Volume 3: SUSE Linux Enterprise Server 12*, SG24-8890 as created or updated on May 6, 2016.

Summary of changes in this book

The following changes were made to this book from the prior publication:

- The z/VM-related chapters have been updated and moved to *The Virtualization Cookbook for IBM z Systems Volume 1: IBM z/VM 6.3*, SG24-8147.
- The Red Hat Enterprise Linux Servers-related chapters have been updated and moved to *The Virtualization Cookbook for IBM z Systems Volume 2: Red Hat Enterprise Linux 7.1 Servers*, SG24-8303.

The following topics are included for the first time:

- Systemd in Linux.
- This book has been updated to SUSE Linux Enterprise Server 12.
- *SUSE Manager* is available on IBM z Systems, but not a topic of this manual.

The following changes have been made to the setup of IBM z/VM 6.3:

- The setup of IBM z/VM covers usage of the z/VM Single System Image Feature (VMSSI) feature with a directory management product.
- The chapters to clone the SUSE Linux Enterprise Server 12 operating system has been simplified. The reader is encouraged to look into new technologies such as IBM Wave for z/VM or KIWI for automated installation and imaging.

Part 1

SUSE Linux Enterprise Server 12

This part of the book focuses on the installation and configuration of SUSE Linux Enterprise Server 12. Additionally, it explores an automated approach to SUSE Linux Enterprise Server 12 installation, and describes creating appliances using KIWI. It also describes common tools available to apply services to SUSE Linux Enterprise Server 12.

This part includes the following chapters:

1 Installing SUSE Linux Enterprise Server 12 on LNXADMIN

"*The only thing that interferes with my learning is my education.*"

— Albert Einstein

The LNXADMIN IBM z/VM guest is used throughout this Part of the book to coincide with *The Virtualization Cookbook for IBM z Systems Volume 1: IBM z/VM 6.3*, SG24-8147.

The main purpose of the LNXADMIN z/VM guest is to provide a File Transfer Protocol (FTP) installation source for your upcoming SUSE Linux Enterprise Server 12 systems. To install and configure SUSE Linux Enterprise Server 12 onto the `IDENTITY LNXADMIN`, perform the following steps:

1. 1.1, "Preparing the SUSE Linux Enterprise Server bootstrap files" on page 4.
2. 1.2, "Installing SUSE Linux Enterprise Server 12 onto the Linux administration system" on page 6.
3. 1.3, "Configuring the SUSE Linux Enterprise Server 12 administration system" on page 14.

In this book, SUSE Linux Enterprise Server 12 is installed onto `LNXADMIN` on the second single system image (SSI) member, `ITSOZVM2`, as described in *The Virtualization Cookbook for IBM z Systems Volume 1: IBM z/VM 6.3*, SG24-8147.

1.1 Preparing the SUSE Linux Enterprise Server bootstrap files

To perform an initial program load (IPL) on a SUSE Linux Enterprise Server 12 installation system, four bootstrap files need to be copied to your z/VM shared file system (SFS) file pool:

- A Linux kernel
- A parameter file
- A Linux initial random access memory disk (RAMdisk)
- A Restructured Extended Executor executable (REXX EXEC)

The REXX EXEC "punches" the other three files to the reader and starts the reader after.

To prepare the SUSE Linux Enterprise Server 12 bootstrap files, perform the following steps:

1. Log on to your z/VM SSI node as `LNXADMIN`. The `PROFILE EXEC A` file is run, and makes the Transmission Control Protocol/Internet Protocol (TCP/IP) tools, such as NETSTAT and FTP, available. A virtual network interface controller (NIC) is created at virtual addresses `600 - 602`, and two virtual disks for swap spaces are created at virtual addresses `300` and `301`, as shown in Example 1-1.

   Example 1-1 Creating NIC and virtual disks

```
LOGON LNXADMIN
00: HCPLNM107E MAINT 0592 not linked; not in CP directory
00: z/VM Version 6 Release 3.0, Service Level 1501 (64-bit),
00: built on IBM Virtualization Technology
00: There is no logmsg data
00: FILES:   NO RDR,   NO PRT,   NO PUN
00: LOGON AT 14:14:11 EDT TUESDAY 04/14/15
00: Command complete
00: NIC 0600 is created; devices 0600-0602 defined
00: NIC 0600 is connected to VSWITCH SYSTEM VSW1
z/VM V6.3.0    2015-04-09 09:04

DMSVML2060I TCPMAINT 592 linked as 0120 file mode Z
DIAG swap disk defined at virtual address 300 (64988 4K pages of swap space)
DIAG swap disk defined at virtual address 301 (129980 4K pages of swap space)
```

2. Answer no (**n**) to the question asking you to IPL Linux from `100`:

```
Do you want to IPL Linux from minidisk 100? y/n
n
```

3. Copy the bootstrap files using FTP onto your z/VM shared file system (SFS) File Pool, as shown in Example 1-2.

   Example 1-2 Copying the bootstrap files

```
==> ftp 9.60.87.87
VM TCP/IP FTP Level 630
Connecting to 9.60.87.87, port 21
220 (vsFTPd 3.0.2)
USER (identify yourself to the host):
lydiap
>>>USER lydiap
331 Please specify the password.
Password:
```

```
>>>PASS ********
230 Login successful.
Command:

cd linux/sle12d1/boot/s390x
ascii
get sles12.exec sles12.exec (repl
get parmfile sles12.parmfile (repl
locsite fix 80
binary
get Linux sles12.Linux (repl
get initrd sles12.initrd (repl
quit
```

4. Use the **FILELIST** command to verify that the kernel and RAMdisk are copied in fixed-80 byte record format. You should see the following files, as shown in Example 1-3.

Example 1-3 Files copied in fixed-80 bytes

```
==> filelist sles12 * a
LNXADMIN FILELIST A0  V 169  Trunc=169 Size=4 Line=1 Col=1 Alt=0
Directory = LNX:LNXADMIN.
Cmd   Filename Filetype Fm Format Lrecl    Records     Blocks   Date     Time
      SLES12   PARMFILE A1 V         18          2          1  4/14/15 11:59:24
      SLES12   EXEC     A1 V         80         11          1  4/14/15 11:59:20
      SLES12   INITRD   A1 F         80     338769       6617  4/14/15 11:24:19
      SLES12   Linux    A1 F         80     134631       2352  4/14/15 11:24:13
```

5. Quit by pressing **F3**.
6. Edit the **SLES12 EXEC** to change the file mode of kernel, parmfile, and initrd from A to asterisk (*), as shown in Example 1-4.

Example 1-4 Changing the file mode

```
==> xedit SLES12 EXEC
===== * * * Top of File * * *
      !...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
===== /* REXX LOAD EXEC FOR SUSE Linux S/390 VM GUESTS       */
===== /* LOADS SUSE Linux S/390 FILES INTO READER            */
===== SAY ''
===== SAY 'LOADING SLES12 FILES INTO READER...'
===== 'CP CLOSE RDR'
===== 'PURGE RDR ALL'
===== 'SPOOL PUNCH * RDR'
===== 'PUNCH SLES12 Linux    * (NOH'
===== 'PUNCH SLES12 PARMFILE * (NOH'
===== 'PUNCH SLES12 INITRD   * (NOH'
===== 'IPL 00C'
===== * * * End of File * * *
```

7. Quit by typing **FILE** on the input line.
8. Verify that the **SLES12 PARMFILE** has the correct information:

```
==> type sles12 parmfile
TERM=dumb manual=1
```

You are now ready to start the installation.

1.2 Installing SUSE Linux Enterprise Server 12 onto the Linux administration system

In this section, you install Linux onto the `LNXADMIN` z/VM SSI node. If you have not logged off you should still be logged in as `LNXADMIN` on your virtual machine:

1. Define the memory size (storage) to 1 gigabyte (GB) with the **`DEFINE STORAGE`** command:

   ```
   ==> define storage 1g
   00: STORAGE = 1G
   00: Storage cleared - system reset.
   ```

2. Perform an IPL on Conversational Monitor System (CMS) again:

   ```
   ==> ipl cms
   z/VM V6.3.0    2015-04-09 09:04

   DMSVML2060I TCPMAINT 592 linked as 0120 file mode Z
   DIAG swap disk defined at virtual address 300 (64988 4K pages of swap space)
   DIAG swap disk defined at virtual address 301 (129980 4K pages of swap space)
   ```

3. Answer no (**n**) to the question asking you to IPL Linux from `100`:

   ```
   Do you want to IPL Linux from minidisk 100? y/n
   n
   ```

4. Verify the increased memory size with the **`QUERY VIRTUAL STORAGE`** command:

   ```
   ==> q v storage
   00: STORAGE = 1G
   ```

5. Verify that the disks you are going to install SUSE Linux Enterprise Server 12 on are available on the z/VM guest, as shown in Example 1-5.

   Example 1-5 Verifying that disks are available on z/VM guest

   ```
   ==> q v dasd
   00: DASD 0100 3390 VV1569 R/W      10016 CYL ON DASD  1569 SUBCHANNEL = 0000
   00: DASD 0120 3390 VV1560 R/O        140 CYL ON DASD  1560 SUBCHANNEL = 000D
   00: DASD 0190 3390 VV1560 R/O        214 CYL ON DASD  1560 SUBCHANNEL = 0006
   00: DASD 019D 3390 VV1560 R/O        292 CYL ON DASD  1560 SUBCHANNEL = 0007
   00: DASD 019E 3390 VV1560 R/O        500 CYL ON DASD  1560 SUBCHANNEL = 0008
   00: DASD 0200 3390 VV156B R/W      10016 CYL ON DASD  156B SUBCHANNEL = 0001
   00: DASD 0300 9336 (VDSK) R/W     524288 BLK ON DASD  VDSK SUBCHANNEL = 000E
   Ready; T=0.01/0.01 14:28:29
   ```

6. Run the **`SLES12 EXEC`** to purge the reader, punch the bootstrap files, and IPL from the reader. You should see the Linux RAMdisk getting loaded into memory. Look for the contents of the parameter file that you created, as shown in Example 1-6.

   Example 1-6 Running the EXEC

   ```
   ==> sles12
   sles12

   LOADING SLES12 FILES INTO READER...
   00:      NO FILES PURGED
   00: RDR FILE 0013 SENT FROM LNXADMIN PUN WAS 0013 RECS 135K CPY  001 A NOHOLD
   NO
   KEEP
   ```

```
00: RDR FILE 0014 SENT FROM LNXADMIN PUN WAS 0014 RECS 0002 CPY  001 A NOHOLD
NO
KEEP
00: RDR FILE 0015 SENT FROM LNXADMIN PUN WAS 0015 RECS 339K CPY  001 A NOHOLD
NO
KEEP
00:      NO FILES CHANGED
Initializing cgroup subsys cpuset
Initializing cgroup subsys cpu
Initializing cgroup subsys cpuacct
Linux version 3.12.28-4-default (geeko@buildhost) (gcc version 4.8.3 20140627
[g
cc-4_8-branch revision 212064] (SUSE Linux) ) #1 SMP Thu Sep 25 17:02:34 UTC
201
4 (9879bd4)
setup.1a06a7: Linux is running as a z/VM guest operating system in 64-bit mode
Zone ranges:
  DMA      [mem 0x00000000-0x7fffffff]
  Normal   empty
Movable zone start for each node
Early memory node ranges
  node   0: [mem 0x00000000-0x3fffffff]
PERCPU: Embedded 12 pages/cpu <A7>000000000367d000 s18432 r8192 d22528 u49152
Built 1 zonelists in Zone order, mobility grouping on.  Total pages: 258560
Kernel command line: TERM=dumb manual=1
...
```

Note: In the following description, the Enter key must often be pressed twice. The reason for this is that, when operating a 3215 terminal (like we do here), pressing Enter without any changed content switches the terminal into central processor (CP) mode. Any subsequent command would go directly to the Hypervisor.

Because we only want to submit an empty line, another pressing of Enter switches back to the previous operating system (linuxrc in this case) and sends the empty line. The change of the mode is also reflected at the lower right of the terminal by the statements `VM READ` and `RUNNING`.

7. At the Main Menu, enter `1` to select `Start Installation`:

```
Main Menu

0) <-- Back <--
1) Start Installation
2) Settings
3) Expert
4) Exit or Reboot

> 1
```

8. Enter `1` to select `Installation`:

```
Start Installation

0) <-- Back <--
1) Installation
2) Upgrade
3) Rescue System
4) Boot Installed System
5) Network Setup

> 1
```

9. Enter `2` to select `Network` as the source medium:

```
Choose the source medium.

0) <-- Back <--
1) DVD / CD-ROM
2) Network
3) Hard Disk

> 2
```

10. Enter `1` to select `FTP` as network protocol:

```
Choose the network protocol.

0) <-- Back <--
1) FTP
2) HTTP
3) HTTPS
4) NFS
5) SMB / CIFS (Windows Share)
6) TFTP

> 1
```

11. Enter `1` to select `IBM OSA Express Network card (0.0.0600)` as the network device:

```
Detecting and loading network drivers
We are running on z/VM
Loading the IUCV network driver
netiucv.8db02b: driver initialized

Choose the network device.

0) <-- Back <--
1) IBM OSA Express Network card (0.0.0600)
2) IBM OSA Express Network card (0.0.0601)
3) IBM OSA Express Network card (0.0.0602)
4) IBM IUCV

> 1
```

12. Enter `0` for the relative port number. Other port numbers than 0 are only needed when using a dedicated Open Systems Adapter (OSA) card. For VSWITCH-attached network interfaces, this is always 0:

```
Enter the relative port number. (Enter '+++' to abort).
> 0
```

13. You should be prompted for the read, write, and data channels. Press **Enter** twice to accept the defaults:

```
Device address for read channel. (Enter '+++' to abort).
[0.0.0600]> Enter Enter

Device address for write channel. (Enter '+++' to abort).
[0.0.0601]> Enter Enter

Device address for data channel. (Enter '+++' to abort).
[0.0.0602]> Enter Enter
```

14. When prompted for `Portname to use` press **Enter** twice:

```
Portname to use. (Enter '+++' to abort).
> Enter Enter
```

15. Enter `1` to select `Enable OSI Layer 2 support`:

```
Enable OSI Layer 2 support?

0) <-- Back <--
1) Yes
2) No
> 1
```

16. When prompted for `MAC address` press **Enter** twice:

```
MAC address. (Enter '+++' to abort).
> Enter Enter
```

17. Enter `2` to select `No` to configure your IP settings manually:

```
(Layer2)
qeth.933eb7: register layer 2 discipline
(Port 0)
qdio: 0.0.0602 OSA on SC b using AI:1 QEBSM:0 PRI:1 TDD:1 SIGA:RW A
qeth.cc0c57: 0.0.0600: MAC address 02:00:0b:00:00:05 successfully registered on
device eth0
qeth.736dae: 0.0.0600: Device is a Virtual NIC QDIO card (level: V631)
with link type Virt.NIC QDIO (portname: )

Automatic configuration via DHCP?

0) <-- Back <--
1) Yes
2) No
> 2
```

18. Enter your IP address followed by the network prefix:

```
Enter your IP address with network prefix.

You can enter more than one, separated by space, if necessary.
Leave empty for autoconfig.

Examples: 192.168.5.77/24 2001:db8:75:fff::3/64. (Enter '+++' to abort).
> 9.12.7.97/20
```

19. Enter the IP address of your gateway:

```
Enter your gateway IP address.

You can enter more than one, separated by space, if necessary.
Leave empty if you don't need one.

9.12.4.1
```

20. Enter the IP address of your Domain Name System (DNS) server:

```
Enter your name server IP address.

You can enter more than one, separated by space, if necessary.
Leave empty if you don't need one.

Examples: 192.168.5.77 2001:db8:75:fff::3. (Enter '+++' to abort).
> 9.12.6.6
```

21. Enter your search domain (we use `itso.ibm.com`):

```
Enter your search domains, separated by a space. (Enter '+++' to abort).
> itso.ibm.com
```

22. Enter the IP address or name of your FTP server that contains the contents of the SUSE Linux Enterprise Server 12 DVD images:

```
Enter the name of the FTP server. (Enter '+++' to abort).
> 9.12.5.251
```

23. Enter the path to the first SUSE Linux Enterprise Server 12 DVD image on the FTP server:

```
Enter the directory on the server. (Enter '+++' to abort).
> /ftp/linux/sles/12
```

24. Enter `1` to provide the credentials for the FTP server:

```
Do you need a username and password to access the FTP server?

0) <-- Back <--
1) Yes
2) No
> 1
```

25. Enter the FTP user ID:

```
Enter the user name with which to access the FTP server. (Enter '+++' to abort).

> ftpuser
```

26. Enter the password for the FTP user ID:

```
Enter the password for the FTP server. (Enter '+++' to abort).
> lnx4vm
```

27. Enter `2` to omit the Hypertext Transfer Protocol (HTTP) proxy server settings:

```
Use a HTTP proxy?

0) <-- Back <--
1) Yes
2) No
> 2
```

28. You should see messages showing the installation system being loaded. For example:

```
...
Loading Installation System (1/5) -              0%       1%       2%       3%
   4%       5%       6%       7%       8%       9%      10%      11%      12%      13%
  14%      15%      16%      17%      18%      19%      20%      21%      22%      23%
...
```

If you do not see these, verify that all is well with the FTP server:

a. Try accessing the files on the FTP server from a different system.
b. Be sure that a firewall is not running.
c. Verify the IP settings, the path to the files on the FTP server, and so on.

29.Select 2 to set Virtual Network Computing (VNC) as the display type:

```
0) <-- Back <--
1) X11
2) VNC
3) SSH
4) ASCII Console

> 2
```

30.Specify the VNC password:

```
Enter your VNC password. (Enter '+++' to abort).
> 12345678
```

31.A VNC server process will be started. You should see the following messages:

```
...
starting VNC server...
A log file will be written to: /var/log/YaST2/vncserver.log ...

***
***           You can connect to <host>, display :1 now with vncviewer
***           Or use a Java capable browser on http://<host>:5801/
***

(When YaST2 is finished, close your VNC viewer and return to this window.)

Active interfaces:

eth0      Link encap:Ethernet  HWaddr 02:00:0B:00:00:0B
          inet addr:9.12.7.97  Bcast:9.12.15.255  Mask:255.255.240.0
--
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0


*** Starting YaST2 ***
```

The preparations to run the actual installer are now complete. From this point on, most of the installation is similar to the installation on any other architecture.

To access the installer, continue with the next procedure:

1. Start a VNC viewer session to the installation process. In this example, a TigerVNC client connects to **9.12.7.97:1** as shown on the left side of Figure 1-1. Enter the password specified in the parameter file (**12345678** in this example) as shown on the right.

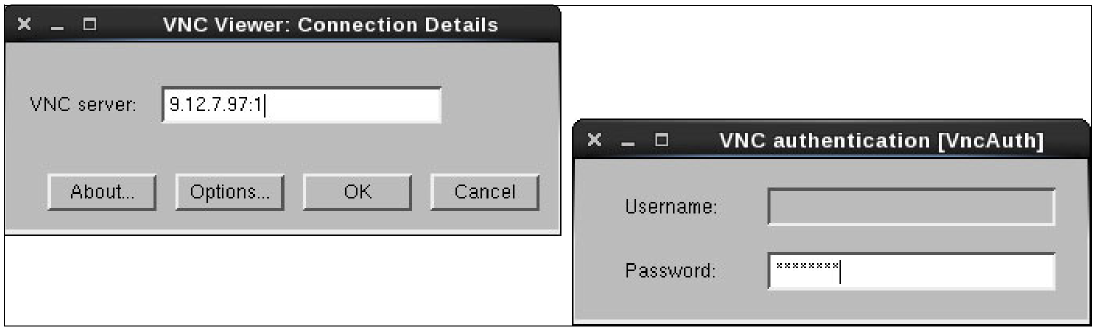
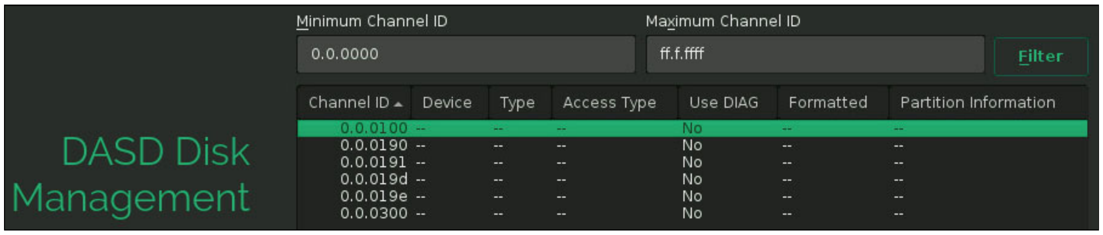


Figure 1-1 Using the VNC viewer

2. You should see the Yet another Setup Tool (YaST) graphical user interface (GUI) to select your Language and Keyboard. After reading the License Agreement, click the **I Agree to the License Terms** check box, then click **Next**.
3. At the Disk Activation window, click **Configure DASD Disks**.
4. At the DASD Disk Management window, perform the following steps:
   a. Select the disk with the channel ID **100**, as shown in Figure 1-2.
   b. Click **Activate** on the Perform Action pop-up menu. If the DASD has not been formatted previously, you are asked if you want to format them now. Format the DASD now. This takes a few minutes.

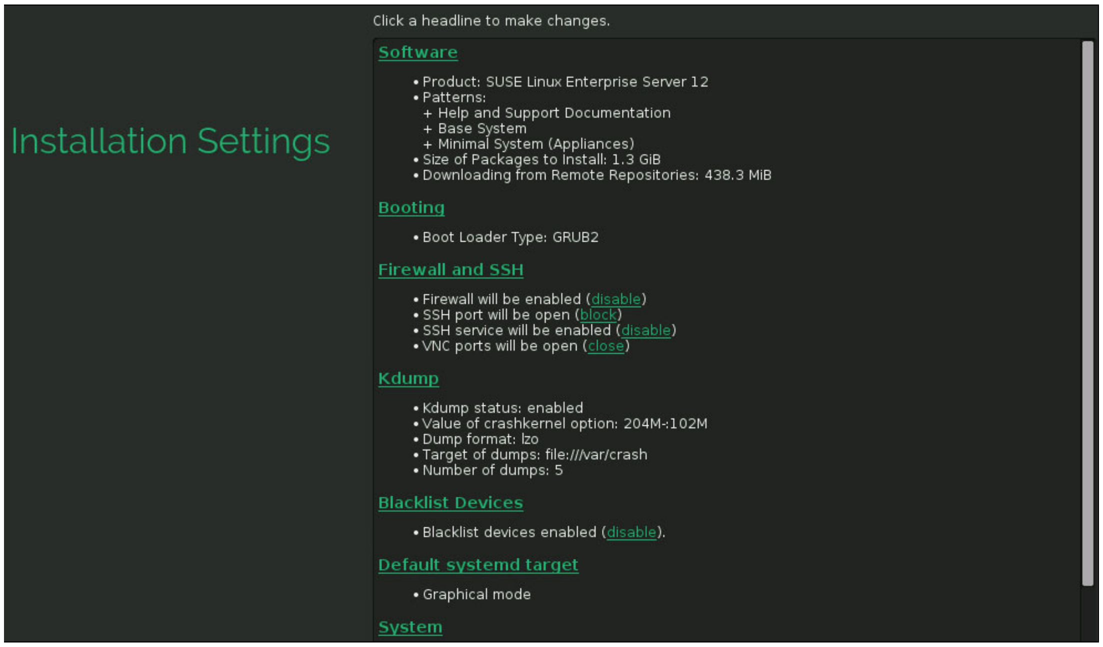

DASD Disk Management

Minimum Channel ID: 0.0.0000
Maximum Channel ID: ff.f.ffff

Channel ID	Device	Type	Access Type	Use DIAG	Formatted	Partition Information
0.0.0100	--	--	--	No	--	--
0.0.0190	--	--	--	No	--	--
0.0.0191	--	--	--	No	--	--
0.0.019d	--	--	--	No	--	--
0.0.019e	--	--	--	No	--	--
0.0.0300	--	--	--	No	--	--

Figure 1-2 Activating DASD on the Linux administration system

   c. In the DASD Disk Management window, click **Next**.
   d. In the Disk Activation window, click **Next**.
5. Register SUSE Linux Enterprise Server 12 using the Registration window. Otherwise, you won’t receive any updates.
6. You are asked to install the latest available online updates using the SUSE Linux Enterprise Server 12 update repositories, which is advised. Select **Yes**, then **Next**.
7. In the Extension and Module Selection window you can include more product extensions if wanted, and then click **Next**.
8. In the Suggested Partitioning window, you can view the partitioning defaults. Then select **Next**.
9. The Clock and Time Zone window enables you to specify the region and time zone.
10. The Create New User window enables you to create another user, but you don’t have to do so. The next window enables you to specify the root password.
11. In the Installation Settings window, select **Software** and clear all except the following items:
   – Help and Support Documentation
   – Base System
   – Minimal System (Appliances)
12. Also in the Installation Settings window, at the Firewall and SSH section, open the Secure Shell (SSH) port, as shown in Figure 1-3 on page 13.


Installation Settings

Click a headline to make changes.

Software

- Product: SUSE Linux Enterprise Server 12
- Patterns:
  - + Help and Support Documentation
  - + Base System
  - + Minimal System (Appliances)
- Size of Packages to Install: 1.3 GiB
- Downloading from Remote Repositories: 438.3 MiB

Booting

- Boot Loader Type: GRUB2

Firewall and SSH

- Firewall will be enabled (disable)
- SSH port will be open (block)
- SSH service will be enabled (disable)
- VNC ports will be open (close)

Kdump

- Kdump status: enabled
- Value of crashkernel option: 204M-:102M
- Dump format: lzo
- Target of dumps: file:///var/crash
- Number of dumps: 5

Blacklist Devices

- Blacklist devices enabled (disable).

Default systemd target

- Graphical mode

System

Figure 1-3 Installation Settings for SUSE Linux Enterprise Server 12 on LNXADMIN

13. Also in the Installation Settings window, set the Blacklist Devices to `disabled`.
14. The last step in the Installation Settings window is to click **Install**.
15. In the Confirm Installation window, click **Install**. This begins the process of laying down RPM Package Managers (RPMs) onto disk. Copying the RPMs should take about 5 - 10 minutes. When the RPMs are copied, your VNC session ends and the system restarts.
16. Go back to your 3270 session. You see messages indicating the Linux image is being restarted. You might need to clear the panel several times. At the end of the restart, the login prompt is displayed, as shown in Example 1-7.

Example 1-7 Login prompt

```
...
00: zIPL v1.24.1-38.17 interactive boot menu
00:
00:  0. default (grub2)
00:
00:  1. grub2
00:  2. skip-grub2
00:
00: Note: VM users please use '#cp vi vmsg <input> <kernel-parameters>'
00:
00: Please choose (default will boot in 16 seconds):
...
Welcome to SUSE Linux Enterprise Server 12  (s390x) - Kernel 3.12.39-47-default
(ttyS0).

linux-4zuk login:
```

17. When you see the login prompt, **`DISCONNECT`** using the **`#CP`** prefix:

```
==> #cp disconnect
```

You have now installed the Linux administration system. You should be able to access the new system using SSH.

1.3 Configuring the SUSE Linux Enterprise Server 12 administration system

Now that your Linux administration system is installed, it needs to be configured. The following steps are involved:

1. "Enabling swap on the virtual disks" on page 14
2. "Setting up the data DASD disk" on page 15
3. "Copying the SUSE Linux Enterprise Server 12 Installation tree to LNXADMIN" on page 17
4. "Configuring the FTP server" on page 17
5. "Reconfigure the Software Repositories" on page 19
6. "Configuring graphical YaST" on page 20
7. "Applying service if necessary: Online update" on page 21
8. "Access z/VM CMS disks from Linux" on page 133
9. "Rebooting the system" on page 22
10. "Verify that the system comes up correctly" on page 22

1.3.1 Enabling swap on the virtual disks

Every time the LNXADMIN runs the common `PROFILE EXEC`, two virtual disks at virtual addresses `300` and `301` are being created. On the Linux guest, these two in-memory disks can be used as swap devices. This section explains how to do this.

Note: You might wonder why they were not enabled during installation using the DASD Disk Management window. The reason is that `dracut` within SUSE Linux Enterprise Server 12 would refer to these swap disks by their Universally Unique Identifier (UUIDs), because this is the default.

Because the UUIDs of the virtual disks change every time they are created with the `PROFILE EXEC` command, the Linux guest would be unable to access the swap disks. This section shows how to change the default behavior of dracut to access the swap disks by their paths rather than their UUIDs.

To enable swap on the virtual disks perform the following steps:

1. Change the persistent policy:

   ```
   # echo 'persistent_policy=by-path' >> /etc/dracut.conf
   ```

2. Update the boot loader:

   ```
   # grub2-install
   ```

   This updates the initramfs image of the first stage boot loader (`/boot/zipl/initrd`).

3. Update the initramfs image of the running kernel (`/boot/initrd`):

   ```
   # dracut -f
   ```

4. Activate the direct access storage devices (DASDs) using either the YaST DASD module or the `dasd_configure` command-line utility. This example shows the usage of the latter:

   ```
   # dasd_configure 0.0.0300 1
   # dasd_configure 0.0.0301 1
   ```

5. Check which DASDs are online using the `lsdasd` command-line utility:

```
# lsdasd
Bus-ID     Status      Name      Device  Type  BlkSz  Size      Blocks
==============================================================================
0.0.0100   active      dasda     94:0    ECKD  4096   7042MB    1802880
0.0.0300   active      dasdb     94:4    FBA   512    256MB     524288
0.0.0301   active      dasdc     94:8    FBA   512    512MB     1048576
```

6. Activate the swap partition on that disk using the `swapon` command-line utility:

```
# swapon -p 5 /dev/disk/by-path/ccw-0.0.0300-part1
# swapon -p 4 /dev/disk/by-path/ccw-0.0.0301-part1
```

7. Check the activated swap devices:

```
# swapon --show
NAME        TYPE        SIZE USED PRIO
/dev/dasdc2 partition   992M   0B   -1
/dev/dasda1 partition 253.9M   0B    5
/dev/dasdb1 partition 507.8M   0B    4
```

8. Add the swap disks to the `/etc/fstab`:

```
# echo '/dev/disk/by-path/ccw-0.0.0300-part1 swap swap pri=5 0 0' >> /etc/fstab
# echo '/dev/disk/by-path/ccw-0.0.0301-part1 swap swap pri=4 0 0' >> /etc/fstab
```

1.3.2 Setting up the data DASD disk

Because we want to use the LNXADMIN Linux guest to function as an FTP server that is used to install other Linux guests, we set up a dedicated disk to put the data on. A Logical Volume Manager (LVM) logical volume is used to enable you to grow the space using additional disks in the future. To set up the data disk perform the following steps:

1. Activate the DASD:

```
# dasd_configure 0.0.0200 1
Device 0.0.0200 is unformatted
```

2. Format the DASD:

```
# dasdfmt -d cdl -b 4096 -yp /dev/disk/by-path/ccw-0.0.020
cyl   10016 of   10016 |##################################################|100%

Finished formatting the device.
Rereading the partition table... ok
```

3. Create a partition that covers the entire disk:

```
# fdasd -a /dev/disk/by-path/ccw-0.0.0200
reading volume label ..: VOL1
reading vtoc ..........: ok

auto-creating one partition for the whole disk...
writing volume label...
writing VTOC...
rereading partition table...
```

4. Verify that the partition has been created:

```
# lsblk /dev/disk/by-path/ccw-0.0.0200-part1
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
dasdd1  94:13   0  6.9G  0 part
```

5. Create a volume group on the newly created partition that implicitly also creates a physical volume:

```
# vgcreate -M2 -s 4096k vg_data /dev/disk/by-path/ccw-0.0.0200-part1
```

6. Verify that the volume group has been created:

```
# vgs vg_data
  VG      #PV #LV #SN Attr   VSize VFree
  vg_data   1   1   0 wz--n- 6.88g    0
```

7. Create the logical volume `lv_data` that uses all of the physical extents on that disk:

```
# lvcreate  -l 100%FREE --name lv_data vg_data
```

8. Verfify that the `lv_data` logical volume has been created in the `vg_data` volume group:

```
# lvs vg_data
  LV      VG      Attr       LSize Pool Origin Data%  Move Log Cpy%Sync Convert
  lv_data vg_data -wi-a---- 6.88g
```

9. Create a file system on the logical volume:

```
# mkfs.xfs -m crc=1 /dev/vg_data/lv_data
meta-data=/dev/vg_data/lv_data isize=512    agcount=4, agsize=450304 blks
         =                       sectsz=4096  attr=2, projid32bit=1
         =                       crc=1        finobt=0
data     =                       bsize=4096   blocks=1801216, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0 ftype=1
log      =internal log           bsize=4096   blocks=2560, version=2
         =                       sectsz=4096  sunit=1 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
```

10. Move the original contents of `/srv`:

```
# mkdir /srv_orig
# mv /srv/* /srv_orig/
```

11. Add the logical volume to the `/etc/fstab` to mount the volume when the guest boots:

```
# echo '/dev/vg_data/lv_data /srv xfs defaults 1 2' >> /etc/fstab
```

12. Mount the newly created volume group:

```
# mount --target /srv
```

13. Move the original contents of `/srv` back to the newly created data space:

```
# mv /srv_orig/* /srv/
# rmdir /srv_orig
```

1.3.3 Copying the SUSE Linux Enterprise Server 12 Installation tree to LNXADMIN

In this section, you copy the SUSE Linux Enterprise Server 12 installation files to the Linux administration system.

To do so, perform the following steps:

1. Start an SSH session to `LNXADMIN` **on member 2**. That is where the large logical volume is mounted on `/srv`.
2. Verify that there is enough disk space left:

   ```
   # df -h /srv
   Filesystem                   Size  Used Avail Use% Mounted on
   /dev/mapper/vg_data-lv_data  6.9G   33M  6.9G   1% /srv
   ```

   In this example there is 6.9 GB of disk space free.
3. Create a directory that is later used as a mount point for the SUSE Linux Enterprise Server 12 ISO image:

   ```
   # mkdir -p /srv/ftp/SLE-12-Server-DVD-s390x-GM-DVD1
   ```
4. Download the `SLE-12-Server-DVD-s390x-GM-DVD1.iso`:

   ```
   # curl -o /srv/ftp/SLE-12-Server-DVD-s390x-GM-DVD1.iso \
   ftp://lydiap:new2day@9.60.87.87//home/lydiap/linux/SLE-12-Server-DVD-s390x-GM-D
   VD1.iso
   ```

   The size of the ISO is about 2.8G. The download can take some time, depending on your network speed.
5. Check the size of the `/srv/` file system again:

   ```
   # df -h /srv
   Filesystem                   Size  Used Avail Use% Mounted on
   /dev/mapper/vg_data-lv_data  6.9G  2.8G  4.2G  41% /srvv
   ```
6. Add an entry to the `/etc/fstab` to automatically loop mount the ISO image at boot time:

   ```
   # echo '/srv/ftp/SLE-12-Server-DVD-s390x-GM-DVD1.iso\
   /srv/ftp/SLE-12-Server-DVD-s390x-GM-DVD1 iso9660 loop,ro 0 0' >> /etc/fstab
   ```
7. Loop mount the ISO image:

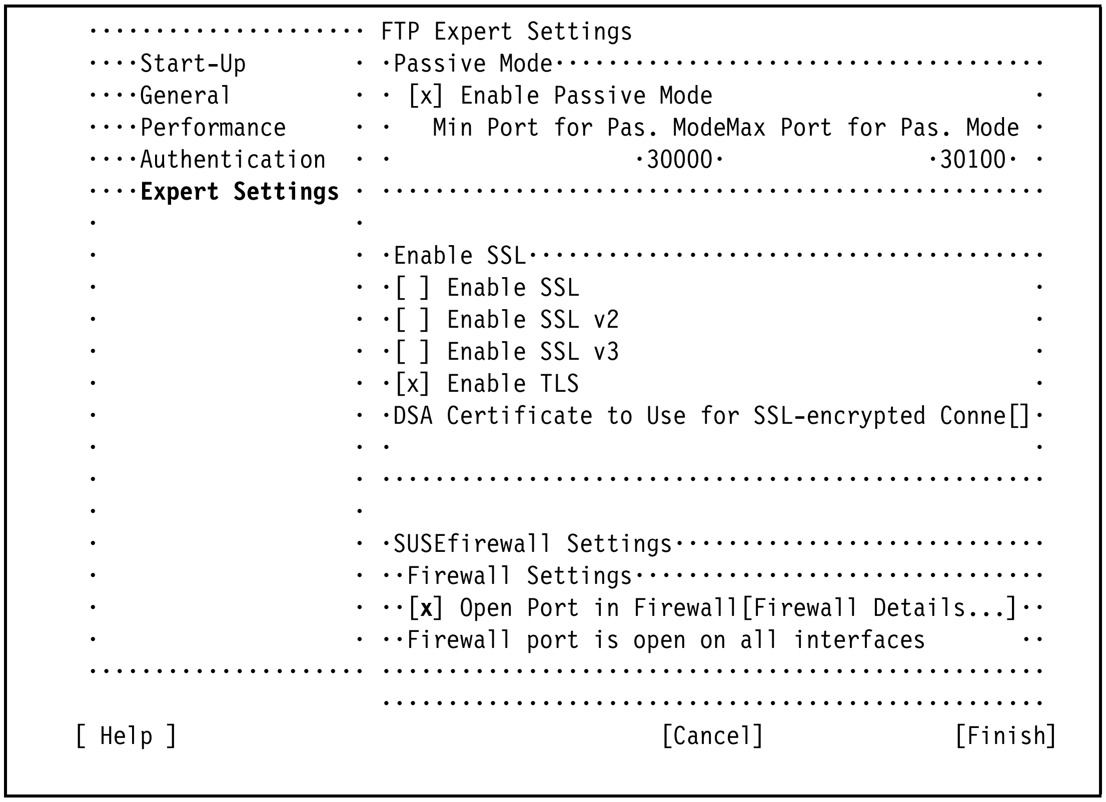
   ```
   # mount /srv/ftp/SLE-12-Server-DVD-s390x-GM-DVD1
   ```

1.3.4 Configuring the FTP server

Set up the Linux administration server as an FTP server by performing the following steps:

1. Start the **`yast`** command.
2. On the left side, use the arrow keys to move down to **Network Services**.
3. On the right side, move down to **FTP Server** and press **Enter**.

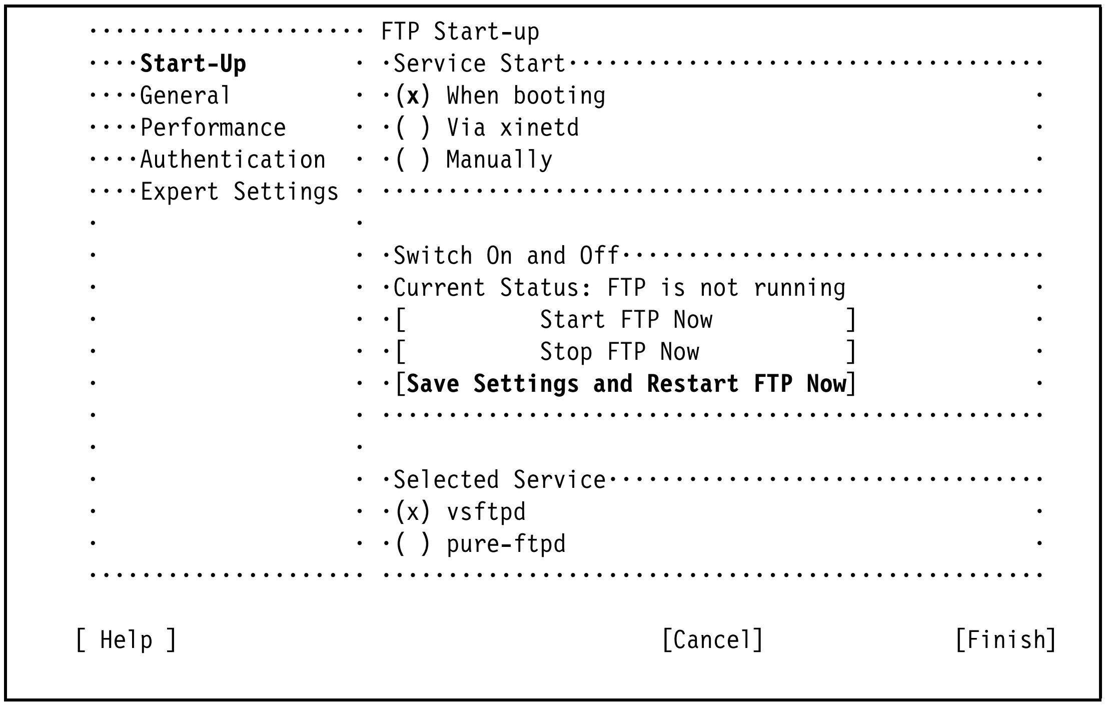
4. In the Expert Settings section, select the **Open Port in Firewall** check box (Figure 1-4).



```
····················· FTP Expert Settings
····Start-Up        · ·Passive Mode····································
····General         · · [x] Enable Passive Mode                          ·
····Performance     · ·   Min Port for Pas. ModeMax Port for Pas. Mode ·
····Authentication  · ·               ·30000·             ·30100· ·
····Expert Settings · ··················································
·                   ·
·                   · ·Enable SSL······································
·                   · ·[ ] Enable SSL                                   ·
·                   · ·[ ] Enable SSL v2                                ·
·                   · ·[ ] Enable SSL v3                                ·
·                   · ·[x] Enable TLS                                   ·
·                   · ·DSA Certificate to Use for SSL-encrypted Conne[]·
·                   · ·                                                 ·
·                   · ··················································
·                   ·
·                   · ·SUSEfirewall Settings···························
·                   · ··Firewall Settings··································
·                   · ··[x] Open Port in Firewall[Firewall Details...]··
·                   · ··Firewall port is open on all interfaces       ··
····················· ··················································
                      ··················································
[ Help ]                                      [Cancel]          [Finish]
```

Figure 1-4 Expert Settings

5. Go to the Start-Up section and select **When booting**, then select **Save Settings and Restart FTP Now** (Figure 1-5).



```
····················· FTP Start-up
····Start-Up        · ·Service Start··································
····General         · ·(x) When booting                              ·
····Performance     · ·( ) Via xinetd                                ·
····Authentication  · ·( ) Manually                                  ·
····Expert Settings · ·················································
·                   ·
·                   · ·Switch On and Off·······························
·                   · ·Current Status: FTP is not running            ·
·                   · ·[          Start FTP Now         ]            ·
·                   · ·[          Stop FTP Now          ]            ·
·                   · ·[Save Settings and Restart FTP Now]           ·
·                   · ·················································
·                   ·
·                   · ·Selected Service································
·                   · ·(x) vsftpd                                    ·
·                   · ·( ) pure-ftpd                                 ·
····················· ·················································

 [ Help ]                                   [Cancel]           [Finish]
```

Figure 1-5 Start-Up options

6. Select **Finish** to exit ftp-server YaST module. This brings you back to the YaST Control Center. Select **Quit** to exit YaST.
7. Create a directory with the proper permissions that will be used for storing AutoYast Extensible Markup Language (XML) files:

```
# install -d -m 755 -o ftp -g ftp /srv/ftp/autoyast
# ls -ld /srv/ftp/autoyast
drwxr-xr-x 2 ftp ftp 5 Apr 18 11:35 /srv/ftp/autoyast
```

The `vsftpd` FTP server should now be configured and running on LNXADMIN.

1.3.5 Reconfigure the Software Repositories

When SUSE Linux Enterprise Server 12 is installed, the location of the *installation source* is remembered. In this case, it is the external FTP Server. Now that the DVD ISO image has been copied to the Linux administration system, you can reset the installation source location to point to itself. The are three options to configure the software repository for the Linux administration system:

- Use the FTP server that just was enabled.
- Use the local path to the loop-mounted ISO image.
- Use the ISO directly.

The following steps describe to use the ISO as a software repository. To accomplish this task, perform the following steps:

1. Start an SSH session as root on the Linux administration system on member 1.
2. Start the **`yast`** command.

3. Accept the default of **Software** in the left column and, using the arrow keys, select **Software Repositories** in the right column and press **Enter**.
4. In the Configured Software Repositories panel, locate the entry for SLES12-12-0 that uses the external FTP server. **Delete** this entry.
5. Accept the default of **Yes** when asked to confirm.
6. Use the Tab key to move to **Add** at the bottom and press **Enter**.
7. Move the cursor down to **Local ISO Image** and press the Spacebar to select. Use the Tab key to move to **Next** at the bottom and press **Enter**.
8. On the Local ISO Image panel, set the **Path to ISO Image** to **`/srv/ftp/SLE-12-Server-DVD-s390x-GM-DVD1.iso`** and click **Next**. It is not necessary to specify a name for the repository.
9. After reading the License Agreement, select the **I Agree to the License Terms** check box, then click **Next**.
10. Back at the Configured Software Repositories panel, you should see the repository you just added. Click **OK**.
11. Select **Quit** to leave YaST.
12. To check if everything is correct, run the **`zypper refresh`** command on the command line:

```
# zypper ref
Repository 'SLES12-12-0' is up to date.
Repository 'SLES12-Pool' is up to date.
Repository 'SLES12-Updates' is up to date.
All repositories have been refreshed.
```

You now have the Linux administration system independant from the external FTP server that was used to install this system.

1.3.6 Configuring graphical YaST

When installing manually with VNC, the installed system automatically activates VNC using the `xinetd`. It also sets the `graphical.target` as the systemd default target. You can verify this using the following steps:

1. The default target can be checked using the following command:

```
# systemctl get-default
graphical.target
```

2. Make sure the `display-manager.service` and `xinetd.service` are enabled and running:

```
# systemctl is-enabled display-manager xinetd.service
enabled
enabled
# systemctl is-active display-manager.service xinetd.service
active
active
```

For more information about the usage of the **`systemctl`** command, see Chapter 8, “Working with systemd” on page 101.

3. To use the graphical YaST, use the following command to install the required packages and their dependencies:

```
# zypper install yast2-control-center libyui-qt
```

4. You should now be able to use the VNC client to connect to the IP address of the Linux administration system with a **:1** appended. The graphical YaST can be started from the SUSE menu at **System** → **YaST**. A sample session is shown in Figure 1-6.

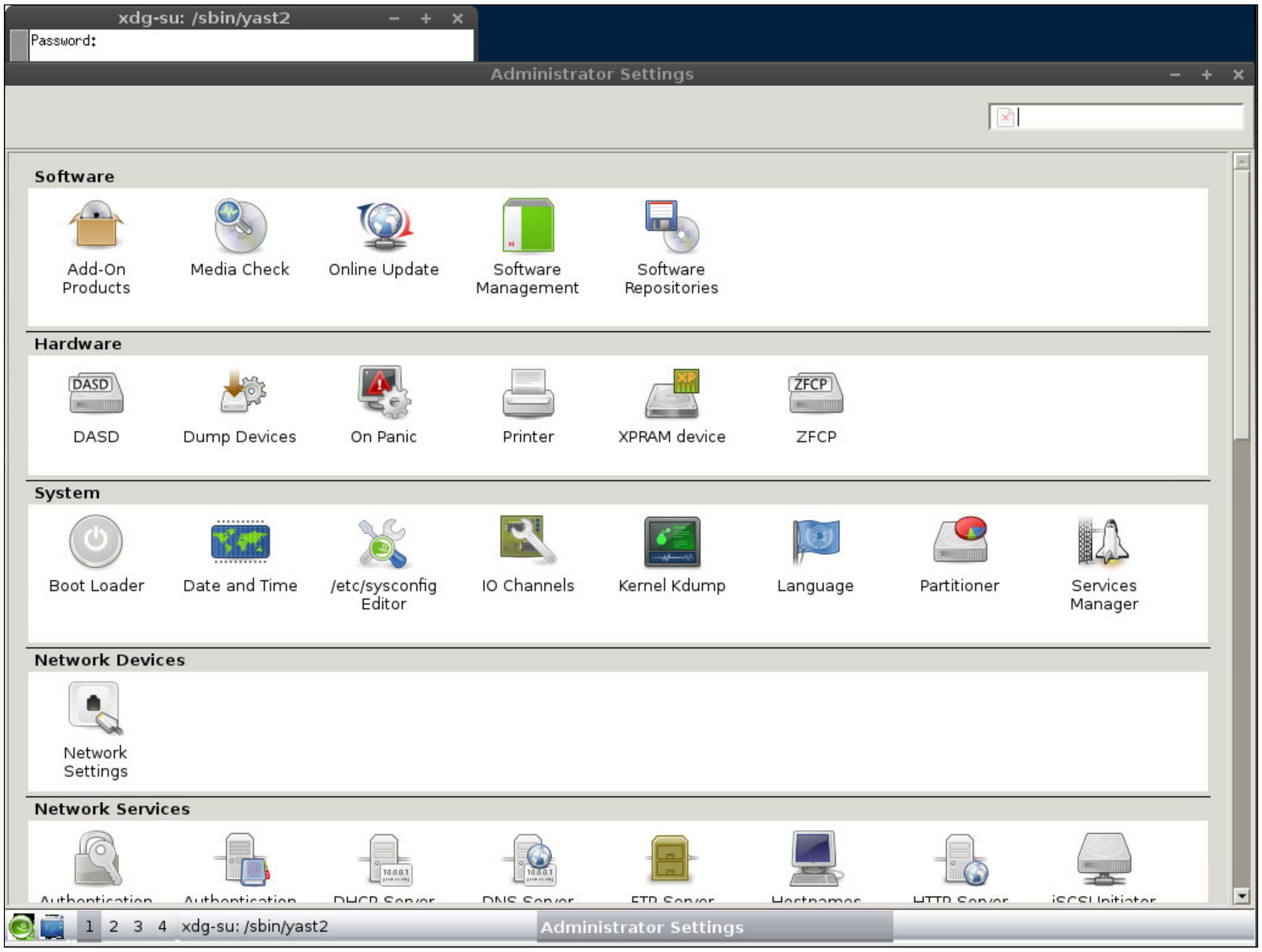


Figure 1-6 Graphical YaST

1.3.7 Applying service if necessary: Online update

You might want to apply service using YaST Online Update. After installing with online repositories activated, you already have all of the latest package versions installed. However, after some time the system will have new patches available.

If the Linux guest has access to the Internet, or an internal online update source, you could either start **YaST** → **Software** → **Online update** or use the following **zypper** command, as shown in Example 1-8.

Example 1-8 Using the zypper command

```
# zypper patch
Loading repository data...
Reading installed packages...
Resolving package dependencies...

The following NEW package is going to be installed:
  kernel-default-3.12.39-47.1
```

```
The following 4 NEW patches are going to be installed:
  SUSE-SLE-SERVER-12-2015-130 SUSE-SLE-SERVER-12-2015-152
  SUSE-SLE-SERVER-12-2015-21 SUSE-SLE-SERVER-12-2015-48

1 new package to install.
Overall download size: 15.1 MiB. Already cached: 0 B  After the operation,
additional 65.0 MiB will be used.
Continue? [y/n/? shows all options] (y): y
```

1.3.8 Rebooting the system

You should now reboot the system to test the changes:

```
# reboot
Connection to 9.12.7.97 closed by remote host.
Connection to 9.12.7.97 closed.
```

Your system should be back in a few minutes. You are now done customizing the Linux administration system Linux image.

1.3.9 Verify that the system comes up correctly

To verify if the system comes up correctly, complete the following steps:

1. Start an SSH session as `root` to the Linux administration system.
2. Confirm that all of the swap spaces are operational:

   ```
   # swapon --show
   NAME        TYPE        SIZE USED PRIO
   /dev/dasda2 partition   992M   0B   -1
   /dev/dasdd1 partition 253.9M   0B    5
   /dev/dasdb1 partition 507.8M   0B    4
   ```

3. Confirm that the FTP service is running:

   ```
   # systemctl status vsftpd
   vsftpd.service - Vsftpd ftp daemon
      Loaded: loaded (/usr/lib/systemd/system/vsftpd.service; enabled)
      Active: active (running) since Fri 2015-04-17 16:51:18 EDT; 3min 26s ago
    Main PID: 1383 (vsftpd)
      CGroup: /system.slice/vsftpd.service
              ··1383 /usr/sbin/vsftpd /etc/vsftpd.conf

   Apr 17 16:51:18 linux-wmye systemd[1]: Starting Vsftpd ftp daemon...
   Apr 17 16:51:18 linux-wmye systemd[1]: Started Vsftpd ftp daemon.
   ```

You have installed and configured a SUSE Linux Enterprise Server 12 Linux system onto the Linux administration system. The next step is to install other Linux guests that are using this server as their installation source.

2

Automating SUSE Linux Enterprise Server 12 installation using the AutoYaST tool

This chapter describes how to install additional SUSE Linux Enterprise Server 12 Linux guests using an automated approach. Only basic instructions are handled here. You can find more information in the official SUSE Documentation, found on the following website:

`https://www.suse.com/documentation/sles-12`

This chapter provides information about the following topics:

2.1 Getting started with AutoYaST

AutoYaST enables you to install SUSE Linux systems without any user interaction. The installation is controlled by information stored in an AutoYaST Extensible Markup Language (XML) file. With AutoYaST, several installation decisions can be done before the actual installation is started using the XML configuration file. The YaST installer still checks the hardware and makes decisions accordingly if some values are not set in the configuration file.

The environment described in this book provides the SUSE Linux Enterprise Server 12 installation tree on an File Transfer Protocol (FTP) server that runs on the Linux administration system. This system is also used to the AutoYaST XML files. This enables you to perform automated SUSE Linux Enterprise Server 12 Linux guest installations over the network.

Using AutoYaST requires the following steps:

1. Create an AutoYaST XML file and upload it to the FTP server on LNXADMIN.
2. Write a parameter file (parmfile) that accesses the AutoYaST2 XML file.
3. Run the automated installation.

2.2 Creating an AutoYaST XML file

There are multiple ways to create an AutoYaST XML file:

- Manually, from scratch
- Using the `/root/autoinst.xml` of an existing SUSE Linux Enterprise Server 12 installation as a starting point
- Using the *Autoinstallation* YaST module

This section describes creating an AutoYaST XML file using the Autoinstallation YaST module because it is both flexible and easy.

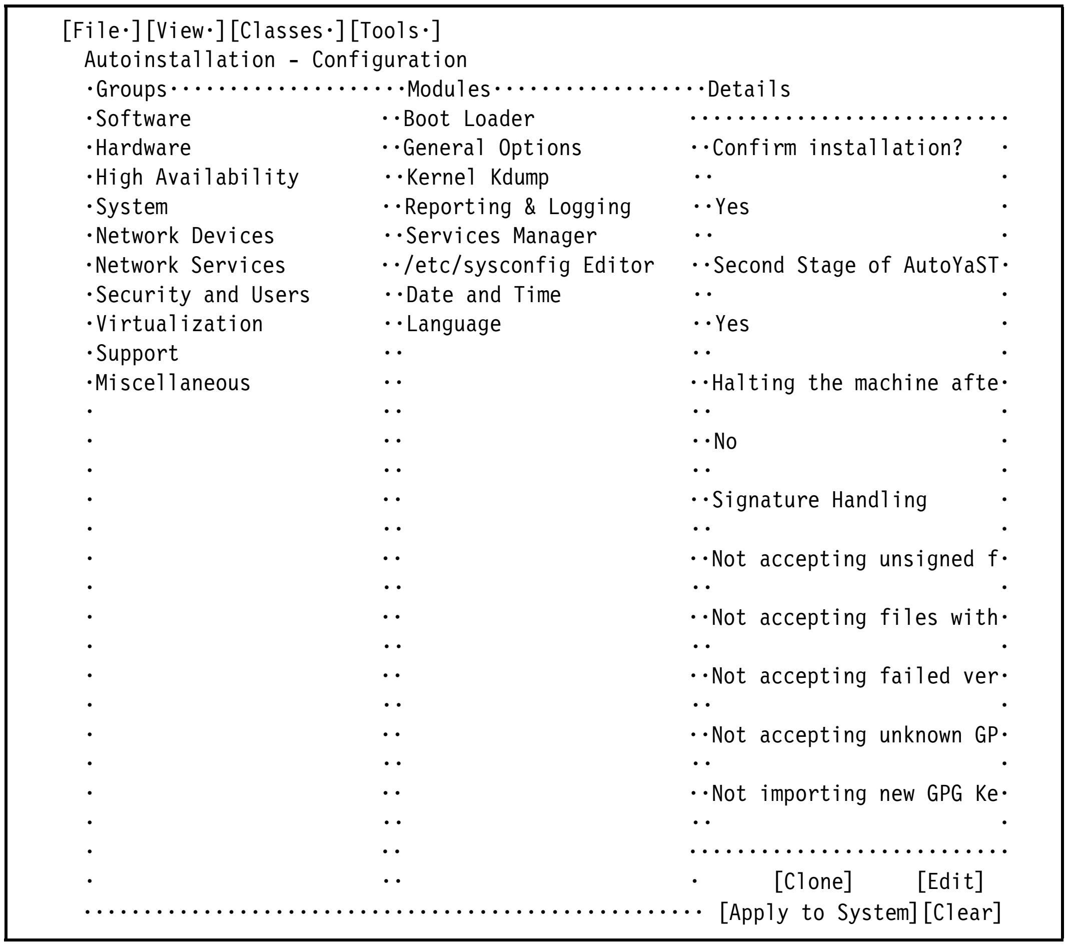
Information: All valid elements of an AutoYaST XML file are defined in a schema. More information about the schema can be found in the `/usr/share/doc/packages/yast2-schema/yast2-schema` file provided by the `yast2-schema` package.

The preferences for all of the settings are taken from the currently running system. To create an AutoYaST XML file using the Autoinstallation YaST module, perform the following steps:

1. Invoke the `yast` command:

   ```
   # yast
   ```

2. On the left side of the main panel, select **Miscellaneous**, and on the right side select **Autoinstallation** and press **Enter** (Figure 2-1).



```
[File·][View·][Classes·][Tools·]
  Autoinstallation - Configuration
  ·Groups···················Modules·················Details
  ·Software               ··Boot Loader           ·························
  ·Hardware               ··General Options       ··Confirm installation?   ·
  ·High Availability      ··Kernel Kdump          ··                        ·
  ·System                 ··Reporting & Logging   ··Yes                     ·
  ·Network Devices        ··Services Manager      ··                        ·
  ·Network Services       ··/etc/sysconfig Editor ··Second Stage of AutoYaST·
  ·Security and Users     ··Date and Time         ··                        ·
  ·Virtualization         ··Language              ··Yes                     ·
  ·Support                ··                      ··                        ·
  ·Miscellaneous          ··                      ··Halting the machine afte·
  ·                       ··                      ··                        ·
  ·                       ··                      ··No                      ·
  ·                       ··                      ··                        ·
  ·                       ··                      ··Signature Handling      ·
  ·                       ··                      ··                        ·
  ·                       ··                      ··Not accepting unsigned f·
  ·                       ··                      ··                        ·
  ·                       ··                      ··Not accepting files with·
  ·                       ··                      ··                        ·
  ·                       ··                      ··Not accepting failed ver·
  ·                       ··                      ··                        ·
  ·                       ··                      ··Not accepting unknown GP·
  ·                       ··                      ··                        ·
  ·                       ··                      ··Not importing new GPG Ke·
  ·                       ··                      ··                        ·
  ·                       ··                      ·························
  ·                       ··                      ·     [Clone]     [Edit]
  ····················································· [Apply to System][Clear]
```

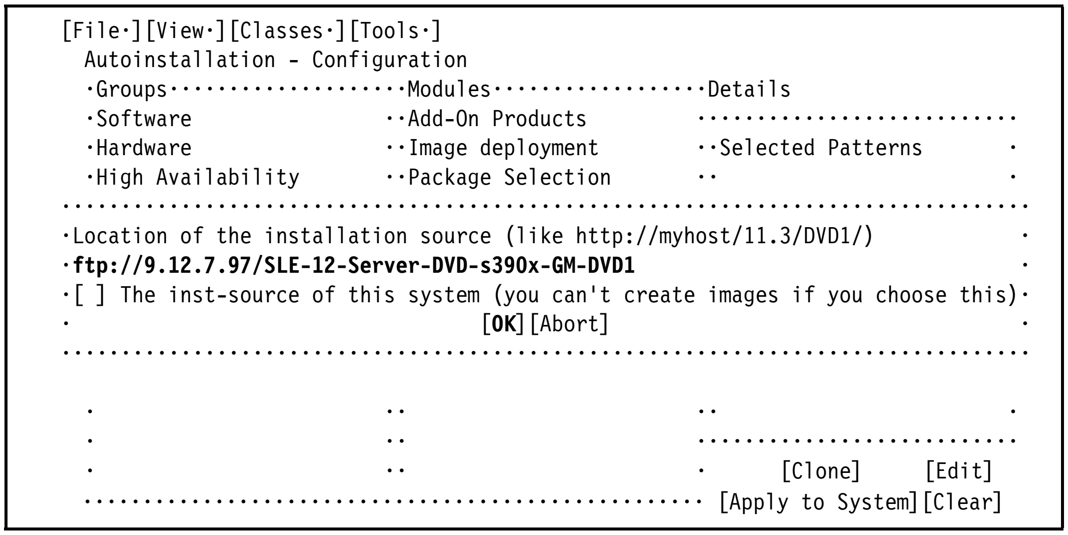
Figure 2-1 Configuring autoinstallation

Configuring the software packages

To begin the configuration for the software packages that AutoYaST will install, perform the following steps:

1. On the left side, select **Software** and press **Enter**.
2. On the right side, with the arrow keys select **Package Selection**.
3. With the tab key, move to **Edit** and select it.

4. When asked for the `Location of the installations source`, clear the check box and enter the Uniform Resource Locator (URL) to the SUSE Linux Enterprise Server 12 installation tree provided by the Linux administration system (Figure 2-2).



```
[File·][View·][Classes·][Tools·]
  Autoinstallation - Configuration
  ·Groups···················Modules·················Details
  ·Software               ··Add-On Products        ··························
  ·Hardware               ··Image deployment       ··Selected Patterns       ·
  ·High Availability      ··Package Selection      ··                        ·
················································································
·Location of the installation source (like http://myhost/11.3/DVD1/)          ·
·ftp://9.12.7.97/SLE-12-Server-DVD-s390x-GM-DVD1                               ·
·[ ] The inst-source of this system (you can't create images if you choose this)·
·                                  [OK][Abort]                                 ·
················································································

  .                        ..                       ..                       ·
  .                        ..                       ··························
  .                        ..                       ·      [Clone]     [Edit]
  ·················································· [Apply to System][Clear]
```

Figure 2-2 Entering the installation tree URL

5. Next, click **OK**.
6. Move the cursor to **Filter**, use the arrow keys to move to **Patterns**, and press **Enter**.
7. On the left side, move the cursor to **Base System** and press the Spacebar to select it. You should see that Minimal System is automatically selected.
8. With the Tab key, move the cursor and select **Accept**.
9. You should see a new panel of more Automatic Changes. Select **OK**.

Note: When you save the the AutoYaST XML file, the Software section will look like this:

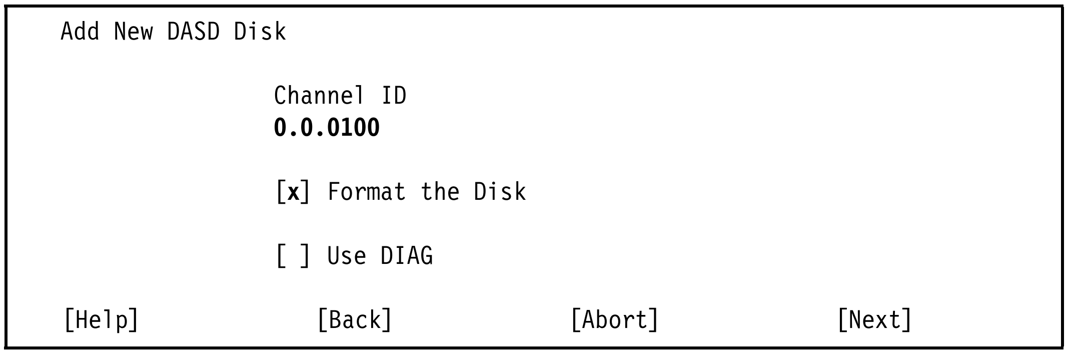
```
<software>
  <instsource>ftp://9.12.7.97/SLE-12-Server-DVD-s390x-GM-DVD1</instsource>
  <packages config:type="list">
    <package>autoyast2-installation</package>
  </packages>
  <patterns config:type="list">
    <pattern>Minimal</pattern>
    <pattern>base</pattern>
  </patterns>
</software>
```

Configuring the DASD

To specify the configuration for the direct access storage device (DASD) to be used by AutoYaST for the automated installation, perform the following steps:

1. On the left side, elect **Hardware** and press **Enter**.
2. On the right side, move to the **DASD** and select **Edit** to launch the Yast Autoinstallation DASD module.
3. You are now in the DASD Disk Management panel. Click **Add** to add information about the DASD that will be used for the installation.

4. In the Add New DASD Disk panel, add the **Channel ID**, which in our example is `0.0.0100`. Also select the **Format the Disk** check box (Figure 2-3).



```
  Add New DASD Disk

                  Channel ID
                  0.0.0100

                  [x] Format the Disk

                  [ ] Use DIAG

  [Help]              [Back]              [Abort]              [Next]
```

Figure 2-3 Adding new DASD disk

5. Then click **Next**, which brings you back to the the DASD Disk Management panel where you see your newly added DASD.
6. Select **Next** to exit the Yast Autoinstallation DASD module.

Note: When you save the the AutoYaST XML file, the DASD section will look like this:

```
<dasd>
  <devices config:type="list">
    <listentry>
      <channel>0.0.0100</channel>
      <diag config:type="boolean">false</diag>
      <format config:type="boolean">true</format>
    </listentry>
  </devices>
  <format_unformatted config:type="boolean">false</format_unformatted>
</dasd>
```

Configuring zFCP Devices

To specify the configuration for zFCP devices to be used by AutoYaST for the automated installation, complete the following steps:

1. Log on as the USER that should be installed with zFCP.
2. Detect the attached zFCP devices with the following command, as shown in Example 2-1.

Example 2-1 The query virtual fcp command

```
==> query virtual fcp
00: FCP  FC00 ON FCP   B804 CHPID 70 SUBCHANNEL = 0002
00:      FC00 DEVTYPE FCP         VIRTUAL CHPID FF FCP REAL CHPID 70
00:      FC00 QDIO-ELIGIBLE       QIOASSIST-ELIGIBLE
00:      FC00 DATA ROUTER ELIGIBLE
00:      WWPN C05076DD90000490
00: FCP  FD00 ON FCP   B904 CHPID 71 SUBCHANNEL = 0003
00:      FD00 DEVTYPE FCP         VIRTUAL CHPID 71 FCP REAL CHPID 71
00:      FD00 QDIO-ELIGIBLE       QIOASSIST-ELIGIBLE
00:      FD00 DATA ROUTER ELIGIBLE
00:      WWPN C05076DD90000AF0
```

3. Run the **SCSIDISC** utility on devices fc00 and fd00, as shown in Example 2-2.

Example 2-2 The SCSIDISC command

```
==> SCSIDISC
Please choose a number corresponding to an FCP device, 'ALL' to select all FCP
devices or 'QUIT'
     0) 0000FC00     1) 0000FD00
Select all fcp devices:
==> ALL
For virtual FCP device 0000FC00
Please choose a number corresponding to a WWPN, 'ALL' to select all WWPNs or
'QUIT'
         0) C05076DD90000490                   1) 500507630500C74C
```

Depending on the number of Ports on the Storage device, there can be multiple worldwide port names (WWPNs). The WWPN `C05076DD90000490` is just the N_Port ID Virtualization (NPIV) adapter we use in this machine, the WWPN `500507630500C74C` is the number from the storage device.

4. Enter `1` to get the logical unit number (LUN) information from the WWPN of the storage device:

```
==> 1
For virtual FCP device 0000FC00 and WWPN 500507630500C74C
Please choose a number corresponding to a LUN, 'ALL' to select all LUNs
or'QUIT'
         0) 4010401A00000000
```

5. Depending on the number of devices you have, repeat the previous steps to find all of the available storage area network (SAN) information. Note that within Linux, the letters used in the hexadecimal number are commonly lowercase.

   Edit the AutoYaST XML file, and add the following information, as shown in Example 2-3.

Example 2-3 Adding the <zfcp> element to the AutoYaST XML file

```
# vi linux4.xml
  <zfcp>
    <devices config:type="list">
      <listentry>
        <controller_id>0.0.fc00</controller_id>
        <wwpn>0x500507630500c74c</wwpn>
        <fcp_lun>0x4010401a00000000</fcp_lun>
      </listentry>
    </devices>
  </zfcp>
```

Important: During the writing of this book, there was an issue with installing to zFCP from AutoYaST. You might have to use a *driver update disk* from the SUSE Customer Center to get around these issues.

Setting the password for root

To set the root password for automatically generated systems, perform the following steps:

1. On the left side of the Autoinstallation-Configuration panel, select **Security and Users** and press **Enter**.
2. On the right side, select **User and Group Management** and press **Enter**.

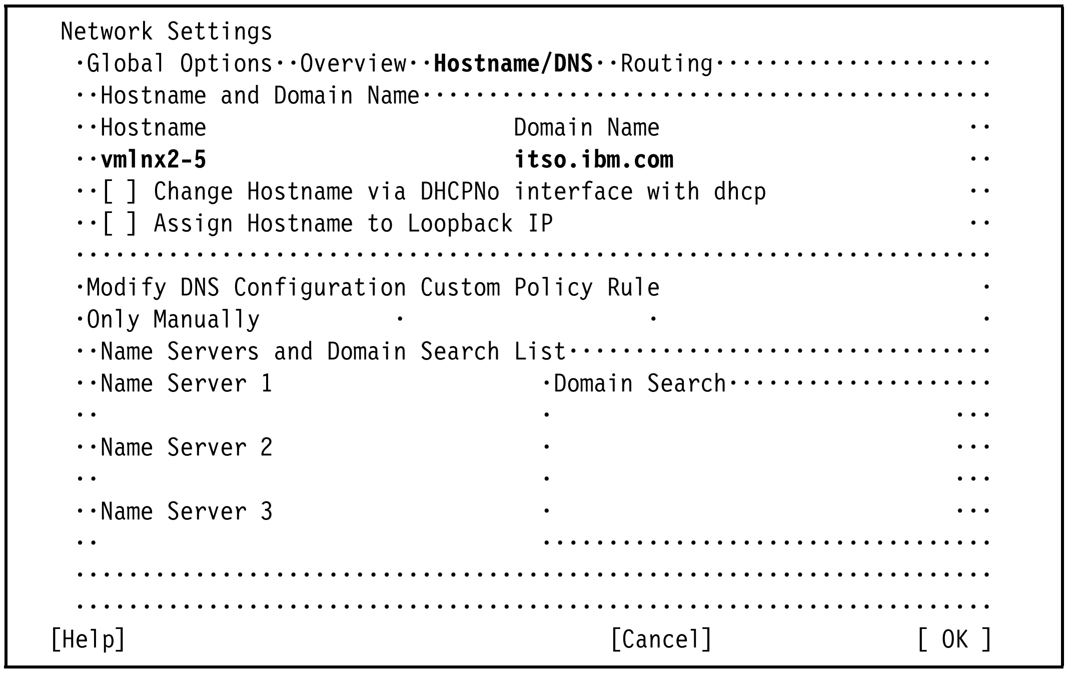
3. Move the cursor with the tab key and select **Edit**.
4. Scroll down with the arrow keys to the **root** user and press **Enter**.
5. In the User Data-Details-Password Settings panel, change the password in both password fields and use the Tab key to select **OK**.
6. Back in the User and Group Administration panel select **OK**.

You should now have the root password set for automatically generated systems.

Configuring the host and domain names

To configure the host and domain, perform the following steps:

1. On the left side of the Autoinstallation-Configuration panel, select **Network Devices** and press **Enter**. On the right side, **Network Settings** should be highlighted.
2. Move the cursor to **Edit** and press Enter.
3. In the Network Settings panel, move the cursor to the top line and use the arrow keys to move to the **Hostname/DNS** tab.
4. Set the **Hostname** and **Domain Name** fields to the correct values for your environment. In this example, it is `vmlnx2-5` and `itso.ibm.com` (Figure 2-4).



```
 Network Settings
  ·Global Options··Overview··Hostname/DNS··Routing····················
  ··Hostname and Domain Name··········································
  ··Hostname                    Domain Name                         ··
  ··vmlnx2-5                    itso.ibm.com                        ··
  ··[ ] Change Hostname via DHCPNo interface with dhcp              ··
  ··[ ] Assign Hostname to Loopback IP                              ··
  ····································································
  ·Modify DNS Configuration Custom Policy Rule                       ·
  ·Only Manually         ·                ·                          ·
  ··Name Servers and Domain Search List·······························
  ··Name Server 1                 ·Domain Search······················
  ··                              ·                                ···
  ··Name Server 2                 ·                                ···
  ··                              ·                                ···
  ··Name Server 3                 ·                                ···
  ··                              ····································
  ····································································
  ····································································
[Help]                                       [Cancel]            [ OK ]
```

Figure 2-4 Setting the Hostname and Domain Name

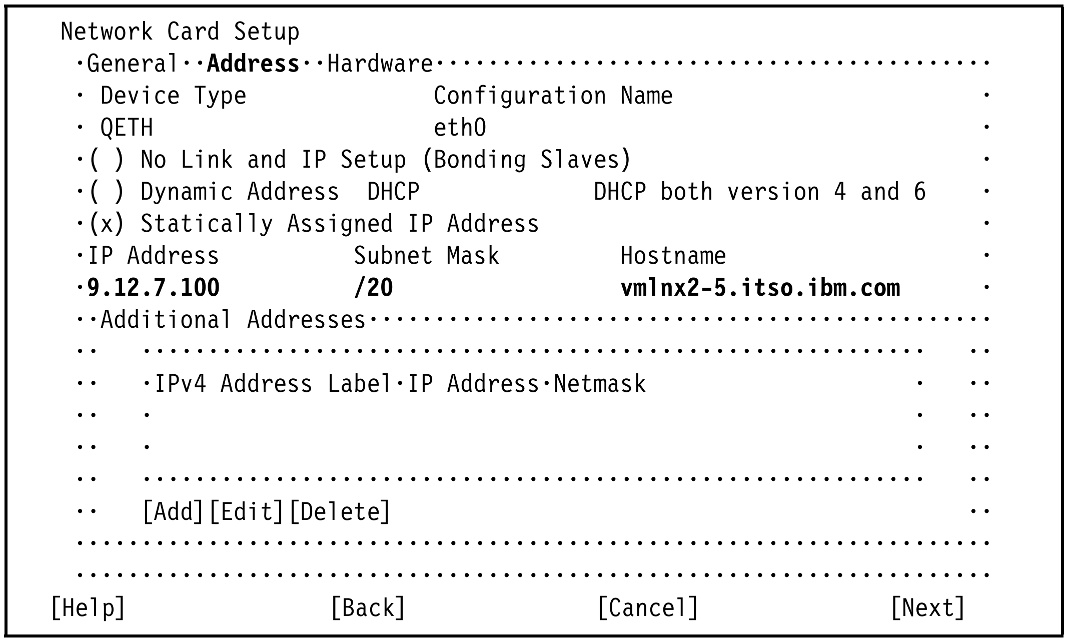
The global host name should now be set correctly.

Configuring the network interface

To configure the network interface for automatically generated systems, perform the following steps:

1. On the left side of the Autoinstallation-Configuration panel, select **Network Devices** and press **Enter**.
2. On the right side, select Network Settings should be selected. Move the cursor to **Clone** and press **Enter**. Details should be shown in the right-most panel.

3. Move the cursor and select **Edit**.
4. The Network Settings panel should appear. Move the cursor to the main window and use the arrow keys to select the **eth0** network card (`eth0`). In this example, it is `0.0.0600`.
5. Move to **Edit** and press **Enter**.
6. Change the **IP Address**, **Subnet Mask**, and **Hostname** to the values that match your environment. In this example, the IP address is `9.12.7.100`, the Subnet Mask is `255.255.240.0`, which translates to a `/20` subnet mask, and the host name is `vmlnx2-5.itso.ibm.com` (Figure 2-5).



```
   Network Card Setup
    ·General··Address··Hardware·······································
    · Device Type             Configuration Name                      ·
    · QETH                    eth0                                    ·
    ·( ) No Link and IP Setup (Bonding Slaves)                        ·
    ·( ) Dynamic Address  DHCP            DHCP both version 4 and 6   ·
    ·(x) Statically Assigned IP Address                               ·
    ·IP Address          Subnet Mask        Hostname                  ·
    ·9.12.7.100          /20                vmlnx2-5.itso.ibm.com     ·
    ··Additional Addresses············································
    ··   ···················································    ··
    ··   ·IPv4 Address Label·IP Address·Netmask                ·    ··
    ··   ·                                                     ·    ··
    ··   ·                                                     ·    ··
    ··   ···················································    ··
    ··   [Add][Edit][Delete]                                        ··
    ··································································
    ··································································
  [Help]               [Back]              [Cancel]              [Next]
```

Figure 2-5 Setting network card values

7. Move the cursor and select **Next**. You should see the new IP address in the Network Settings panel.
8. Move the cursor and select **OK**. This will return you to the main Autoinstallation-Configuration panel.

AutoYaST is now configured to define specific network values.

Omitting the confirmation panel of the installation

By default, the installation will still ask for confirmation before anything is written to disk. Because this involves manual interaction, we configure the system to omit this step.

To turn off the installation confirmation panel, perform the following steps:

1. On the left side of the Autoinstallation-Configuration panel, select **System** and press **Enter**.
2. On the right side, **General Options** should be selected. Move the cursor and select **Edit**.
3. On the Other Options panel, use the Spacebar to clear the **Confirm installation** check box, and select **Next**.
4. On the Ask Options panel, press **F10** to finish.

With this setting, confirmation before writing to disk will no longer be required.

Saving the configuration to disk

Now that the settings are complete, you can save them to an AutoYaST XML configuration file using the following steps:

1. On the Autoinstallation-Configuration panel, the **File** menu should be highlighted. Press **Enter**.
2. Use the arrow keys to move down and select **Save As**.
3. Change the directory to `/srv/ftp/autoyast/`.
4. Move the cursor to the **File name** field and set it to an appropriate name. In this example, `linux3.xml` is used.
5. Move the cursor and select **OK**.
6. The creation of the XML file might take a few seconds. After it has finished, you should see a message similar to “`File /srv/ftp/autoyast/linux3.xml was saved successfully.`” Select **OK**.
7. The **File** menu should be highlighted. Press **Enter** and use the arrow keys to select **Exit**.
8. At the main YaST2 Control Center panel, select **Quit** to exit YaST.

The XML file should now be saved in the `/srv/ftp/autoyast/` directory. You can review the newly created XML file using the following command, as shown in Example 2-4.

Example 2-4 Reviewing the linux3.xml file

```
   # vi /srv/ftp/autoyast/linux3.xml
   <?xml version="1.0"?>
   <!DOCTYPE profile>
   <profile xmlns="http://www.suse.com/1.0/yast2ns"
   xmlns:config="http://www.suse.com/1.0/configns">
     <dasd>
       <devices config:type="list">
         <listentry>
           <channel>0.0.0100</channel>
...
```

Creating a custom partition layout

This step is optional. If the partition layout is not specified explicitly, AutoYaST will use its default scheme. A custom partition layout provides a flexible way to influence the partitioning of the installer. Example 2-5 shows a custom partition layout on a DASD disk at `0.0.0100`.

Example 2-5 Custom partition layout

```
<partitioning config:type="list">
    <drive>1
      <device>/dev/disk/by-path/ccw-0.0.0100</device>
      <disklabel>dasd</disklabel>
      <enable_snapshots config:type="boolean">false</enable_snapshots>
      <initialize config:type="boolean">true</initialize>2
      <use>all</use>3
      <partitions config:type="list">4
        <partition>5
          <filesystem config:type="symbol">ext2</filesystem>
          <format config:type="boolean">true</format>
          <mount>/boot/zipl</mount>
          <mountby config:type="symbol">path</mountby>
          <size>200M</size>
        </partition>
        <partition>6
          <filesystem config:type="symbol">swap</filesystem>
          <format config:type="boolean">true</format>
          <mount>swap</mount>
          <mountby config:type="symbol">path</mountby>
          <size>256M</size>
        </partition>
        <partition>7
          <filesystem config:type="symbol">btrfs</filesystem>
          <format config:type="boolean">true</format>
          <mount>/</mount>
          <mountby config:type="symbol">path</mountby>
          <size>max</size>
          <subvolumes config:type="list">
            <listentry>boot/grub2/s390x-emu</listentry>
            <listentry>home</listentry>
            <listentry>opt</listentry>
            <listentry>srv</listentry>
            <listentry>tmp</listentry>
            <listentry>usr/local</listentry>
            <listentry>var/crash</listentry>
            <listentry>var/lib/mailman</listentry>
            <listentry>var/lib/named</listentry>
            <listentry>var/lib/pgsql</listentry>
            <listentry>var/log</listentry>
            <listentry>var/opt</listentry>
            <listentry>var/spool</listentry>
            <listentry>var/tmp</listentry>
          </subvolumes>
        </partition>
      </partitions>
    </drive>
  </partitioning>
```

Notes:

The following numbers correspond to the marked lines in Example 2-5 on page 31:

1 All of the disks that will be partitioned are handled within `<drive>...</drive>` elements. In this example it is just one disk. Make sure that the disk has been specified in the `<dasd>` or `<zfcp>` section.

2 If true, **`initialize`** wipes out all of the partitions on that disk.

3 The whole disk is going to be used.

4 The `partitions` element is a container for several partitions. In the example described here, three partitions are set up.

5 An `Ext2` partition with a size of 200 megabytes (MB) will be created and mounted at `/boot/zipl`. This partition will have the kernel and initramfs for the Kernel that provides the Grub2 shell.

6 A Swap partition with a size of 265 MB.

7 Btrfs partition with several subvolumes will be created on the remaining disk space. This partition will be used as the root file systems for SUSE Linux Enterprise Server 12.

More information about the AutoYaST partition configuration options can be found on the following website:

http://www.suse.com/documentation/sles-12/singlehtml/book_autoyast/book_autoyast.html#CreateProfile.Partitioning

Add a post install script

AutoYaST enables you to run custom scripts in different stages of the installation. The snippet in Example 2-6 shows how to add a post-installation shell script to the AutoYaST XML.

Example 2-6 Adding a script

```
<scripts>
    <post-scripts config:type="list">
      <script>
        <interpreter>shell</interpreter>
        <source>
<![CDATA[
# change the dracut persistent policy to access disks by path
echo 'persistent_policy=by-path' >> /etc/dracut.conf
# update the initramfs image of the running kernel
/usr/bin/dracut -qf
# updates the initramfs image of the first stage boot loader (/boot/zipl/initrd)
/usr/sbin/grub2-install

# Enable the VDISKs for swap
/sbin/cio_ignore -r 0.0.0300,0.0.0301
/sbin/cio_ignore -r 0.0.0301
/sbin/dasd_configure 0.0.0300 1
/sbin/dasd_configure 0.0.0301 1
/sbin/swapon -p 5 /dev/disk/by-path/ccw-0.0.0300-part1
/sbin/swapon -p 4 /dev/disk/by-path/ccw-0.0.0301-part1
echo '/dev/disk/by-path/ccw-0.0.0300-part1 swap swap pri=5 0 0' >> /etc/fstab
echo '/dev/disk/by-path/ccw-0.0.0301-part1 swap swap pri=4 0 0' >> /etc/fstab

# Detach CMS DASD disks for z/VM SSI LGR after boot up
# Note: Additionaly the commands are piped into bash to detach the disks when
AutoYaST runs post-scripts
tee /etc/rc.d/boot.local <<\EOF | /bin/bash
# Detach CMS DASD disks for z/VM SSI LGR
for d in 0190 019D 019E 0592; do
  /sbin/vmcp q v $d &> /dev/null && echo -n "z/VM disk " && /sbin/vmcp detach $d
done
true
EOF

# Link CMS DASD disks again on shutdown/reboot
cat <<\EOF >> /etc/init.d/halt.local
# Link CMS DASD disks again
for d in 0190 019D 019E; do
  /sbin/vmcp link MAINT $d $d &> /dev/null && echo "z/VM disk $d linked"
done
/sbin/vmcp link TCPMAINT 0592 0592 && echo "TCPMAINT 0592 linked"
true
EOF
# Enable persistent logging for the systemd journal
mkdir -p /var/log/journal
]]>
        </source>
      </script>
    </post-scripts>
  </scripts>
```

On the installed Linux guest, AutoYaST captures the commands plus their output and writes a log of the post install script to: /var/adm/autoinstall/logs/post-scripts.log.

More information about custom user scripts can be found on the following web page:

http://www.suse.com/documentation/sles-12/book_autoyast/data/createprofile_scripts.html

As a reference, the configuration used in this section is supplied with the files associated with this book. These should be available on the following website:

ibm.com/vm/pubs/redbooks/SG248147

2.3 Run the automated installation

To start the installation, a new parameter file has to be created that includes information about the DASD to be used, the network setup, the location of the installation source, and the location of the AutoYast XML. To do so, perform the following steps:

1. Log on as `LINUX3`, when asked to IPL from minidisk 100 choose not to (**n**).
2. Copy the existing parameter file, of the SFS File Pool from `LINUXADMIN` (file mode d) to the SFS File Pool of `LINUX3` (file mode a). In this example, it is `LINUX5`:

   ```
   ==> copyfile sles12 parmfile d = = a
   Ready; T=0.01/0.01 14:52:31
   ```

3. Edit the copied parmfile and change manual=1 to **manual=0**. Then add the information information about the DASD to be used, the network setup, the location of the installation source, and the location of the AutoYast XML, as shown in Example 2-7.

   Example 2-7 Adding information to the parmfile

   ```
   ==> xedit sles12 parmfile
   00000 * * * Top of File * * *
   00001 TERM=dumb manual=0 DASD=0100
   00002 ReadChannel=0.0.0600 WriteChannel=0.0.0601 DataChannel=0.0.0602
   00003 InstNetDev=osa Layer2=1 OSAInterface=qeth OSAMedium=eth PortName= PortNo=
   00004 Hostname=vmlnx2-5.itso.ibm.com HostIP=9.12.7.100 Gateway=9.12.4.1
   00005 Netmask=255.255.240.0 Nameserver=9.12.6.6 Domain=itso.ibm.com
   00006 Install=ftp://9.12.7.97/SLE-12-Server-DVD-s390x-GM-DVD1
   00007 Autoyast=ftp://9.12.7.97/autoyast/linux3.xml
   00008 UseVNC=1 VNCPassword=12345678 UseSSH=1 SSHPassword=12345678
   00009 linuxrclog=/dev/console
   00011 * * * End of File * * *
   ```

   Remember: A parmfile has a limit of 10 lines with a maximum of 79 characters per line.

4. Save the file.
5. Start the installation with the install **EXEC**. The autoyast line will be read, and the xml file will be used for an automated installation:

   ```
   ==> sles12
   ```

   Now the entire installation process should be entirely automated. The installation takes some time to complete.
6. When the installation process is complete, you should be able to log in using SSH to your newly created virtual server.

3

Creating appliances with KIWI

KIWI provides a mechanism to create images of SUSE Linux Enterprise Servers. These can be installation DVD images, live images that can run without deployment to disk, preloaded operating systems, or readily configured appliances.

KIWI procedures are close to cloning an existing system, where AutoYaST is an automated installation. However, unlike cloning, KIWI uses configuration files and repositories, and the actual imaging process does not need any further manual interaction. One important difference from cloning is that the images created have never been booted during the setup process. So KIWI uses the advantages of both cloning and automated building of a system.

Images that have never been booted, and performing the final configuration during the first boot, are also called *preload images* or *firstboot images*. The significant advantage of this is that all unique strings and files inside an operating system are generated when a service is started up the first time.

This is different from cloning, where all the unique strings and files must be removed, which can be a problem if you do not know all of the possible locations. On a preload image, these unique strings and files have never been generated, and therefore it is not necessary to know about the locations.

Compared to automated installations, starting a preconfigured image is much faster and also uses much less hardware resources. Preparing such an image is easy for small images with only a little functionality. Preparing an image with complex functionality can become a time-consuming process. As a rule, it is worthwhile to create an image when one of the following statements is true:

- The same image is used multiple times.
- The image should be always available with the latest updates.
- It must be possible to reproduce the image from a set of rules.
- The operational procedures require a process similar to cloning.

This section provides information about the following topics:

- 3.1, “Set up KIWI on LNXADMIN” on page 39
- 3.2, “Options to adapt the KIWI image” on page 45
- 3.3, “Create and clone the image” on page 50
- 3.4, “More information” on page 51

A good candidate for an image is a small version of the operating system that is used as a minimal configuration for all servers. This is also known as *Just enough Operating System* (JeOS). The following requirements apply for this system:

- It uses only standard packages.
- The software update and installation stack must be fully functional (zypper).
- It is directly bootable.

There is a development front end to KIWI available, which is called *SUSE Studio*. However, this is *not* available at no cost, but is a priced feature. In addition to providing a nice front end, SUSE Studio also adds several development features, such as checking for differences between versions of images, or inheritance of other images. For x86, this front end is publicly available on the Internet on condition that only open source packages are used:

https://susestudio.com

For mainframe environments, SUSE Studio is a priced feature that must be deployed in a local environment. This chapter goes forward only with the methods and features available from the software development kit (SDK) of SLE12.

Note: When running in an environment where many rebuilds of packages are expected, and some automated deployment process must be implemented for those packages, it could be worthwhile to take a closer look at the *Open Build Service* (OBS).

The OBS provides a sophisticated infrastructure to create packages for different architectures and different distributions. It is available as open source software, and used by SUSE to run all needed build jobs. Some of its main features are:

- Version control of all software packages
- Support for multiple architectures
- Support for different package formats, such as RPM Package Manager (RPM), in addtion to image builds
- Automated resolving of package dependencies
- Automatic rebuild of all depending packages if binary changes occur
- Publishing to repositories that might be registered by zypper
- Code review before check in to a repository
- And many more

All of the features can publicly be used for the openSUSE project by using the **`osc`** command-line tool. A tutorial for this is available on the following site:

http://en.opensuse.org/openSUSE:Build_Service_Tutorial

The central documentation page with references about how to deploy the OBS locally is available on the following website:

http://en.opensuse.org/Portal:Build_Service

3.1 Set up KIWI on LNXADMIN

KIWI needs some preparation steps before it can be used. For example, extra disk space is needed to store images.

3.1.1 Add disks for KIWI

Two extra minidisks are added to `LNXADMIN` on the second single system image (SSI) member for storing images created by KIWI. The root file system is kept in `/var/tmp/kiwi/`, as shown in Table 3-1, adhering to the Linux File Hierarchy Standard (FHS).

Table 3-1 Disks used for KIWI

Disk	Cylinders	Mount point	Usage
201	3338	NONE	Imaging disk
150	Emulated direct access storage device (EDEV)	/var/tmp/kiwi/	Preparation of root file system

The first steps are to add two entries to the appropriate `SUBCONFIG` section of the `IDENTITY LNXADMIN`:

1. Add the minidisk. In this example, the minidisk has the label `VV1222`:

   ```
   ==> dirm foruser LNXADM-2 AMDISK 201 3390 AUTOV 3338 VV1222 MR
   ```

   The full procedure is described in *The Virtualization Cookbook for IBM z Systems: Volume 1 IBM z/VM 6.3*, SG24-8147, “Configure z/VM to use Minindisks”.

2. Also dedicate the emulated DASD (EDEV) to the lnxadm-2 subconfig:

   ```
   ==> dirmaint foruser LINUX2 dedicate 150 3005
   ```

Set up the 201 minidisk

The `201` minidisk does not need a file system. However, it must be preformatted with **`dasdfmt`**. Complete the following steps to preformat the 201 minidisk:

1. Start an Secure Shell (SSH) session as root to the Linux running on `LNXADMIN`.
2. Run the **`dasd_configure`** command to make the disk available across reboots:

   ```
   # dasd_configure 0.0.0201 1 0
   Device 0.0.0201 is unformatted
   ```

3. Check the resulting device letter with the **`lsdasd`** command:

   ```
   # lsdasd
   Bus-ID     Status      Name      Device  Type  BlkSz  Size      Blocks
   ==============================================================================
   ....
   0.0.0201   n/f         dasde     94:20   ECKD
   ```

   In this example, the device `/dev/dasde` is the newly created device, which now must be formatted.

4. Use the **dasdfmt** command to format the disk. In this example, it is /dev/dasde:

```
# dasdfmt -b 4096 -f /dev/dasde
Drive Geometry: 3338 Cylinders * 15 Heads =  50070 Tracks

I am going to format the device /dev/dasde in the following way:
   Device number of device : 0x201
   Labelling device        : yes
   Disk label              : VOL1
   Disk identifier         : 0X0201
   Extent start (trk no)   : 0
   Extent end (trk no)     : 50069
   Compatible Disk Layout  : yes
   Blocksize               : 4096

--->> ATTENTION! <<---
All data of that device will be lost.
Type "yes" to continue, no will leave the disk untouched: yes
```

   This command will take some time.

5. Use the **fdasd** command to create a single partition on this disk:

```
# fdasd -a /dev/dasde
reading volume label ..: VOL1
reading vtoc ..........: ok

auto-creating one partition for the whole disk...
writing volume label...
writing VTOC...
rereading partition table...
```

The new partition, /dev/dasde1, is ready to use as an imaging disk.

Set up the 150 disk

One of the nice features of EDEV is that the sizes of logical unit numbers (LUNs) can be increased from the storage system. Therefore, using a volume manager is not necessary. Complete the following steps to create and mount 150 EDEV over /var/tmp/kiwi/:

1. Run the **dasd_configure** command to make the disk available across reboots:

```
# dasd_configure 0.0.0150 1 0
```

2. Check the resulting device letter with the **lsdasd** command:

```
# lsdasd
Bus-ID     Status      Name      Device  Type  BlkSz  Size      Blocks
==============================================================================
....
0.0.0150   active      dasdf     94:16   FBA   512    10240MB   20971520
```

3. For EDEV devices, **dasdfmt** does not work.
4. Use the **fdisk** or **parted** command to create a single partition on this disk:

```
# fdisk /dev/dasdf
```

5. Create an XFS file system with the **mkfs** command:

```
# mkfs -t xfs /dev/disk/by-path/ccw-0.0.0150-part1
meta-data=/dev/dasdf1            isize=512    agcount=4, agsize=655104 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=0        finobt=0
data     =                       bsize=4096   blocks=2620416, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0 ftype=0
log      =internal log           bsize=4096   blocks=2560, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
```

6. Make a backup copy of the working /etc/fstab file:

```
# cd /etc
# cp fstab fstab.works
```

7. Edit the fstab file and add a line so that the new logical volume is mounted over the /var/tmp/kiwi/ directory:

```
# vi fstab
/dev/disk/by-path/ccw-0.0.0300-part1 swap                 swap       pri=2  0 0
/dev/disk/by-path/ccw-0.0.0301-part1 swap                 swap       pri=1  0 0
/dev/disk/by-path/ccw-0.0.0100-part1 /                   ext3   acl,user_xattr
1 1
/dev/lnxadmin-vg/srv /srv                xfs        defaults              1 2
/dev/disk/by-path/ccw-0.0.0150-part1 /var/tmp/kiwi        xfs      defaults 0 0
...
```

8. Make the /var/tmp/kiwi/ directory for the mount point:

```
# mkdir /var/tmp/kiwi
```

9. Use the **mount -a** command to mount all file systems. This has the side effect of testing the syntax in the /etc/fstab file:

```
# mount -a
```

10. Verify that the new logical volume is mounted:

```
# mount | grep kiwi
/dev/dasdf1 on /var/tmp/kiwi type xfs (rw)
```

A new volume has been created by using the 150 EDEV and will be mounted over /var/tmp/kiwi/ at boot time.

3.1.2 Install the required packages

Several packages are needed to run KIWI on LNXADMIN. Some of these are only available from the software development kit (SDK). Complete the following steps to install the required packages:

1. Start an SSH session to LNXADMIN as root.
2. Ensure that you have the SDK available with the following **zypper** command:

```
# zypper lr -d
```

   If you do not have the SDK available, add it with **yast** → **Software** → **Software Repositories**.

3. Install the RPMs `kiwi` and `kiwi-desc-oemboot`. Several dependencies are automatically resolved, which leads to installation of several more packages:

```
# zypper in kiwi kiwi-desc-oemboot
Loading repository data...
Reading installed packages...
Resolving package dependencies...

The following NEW packages are going to be installed:
....
Continue? [y/n/?] (y): y
```

4. Answer with **y**. The packages are now added to the system.

KIWI and all related packages should now be installed.

3.1.3 Create a basic KIWI configuration

When KIWI runs, it expects all of the configurations below a directory that is given by the command line. Inside this directory, there is the `config.xml` file and a file named `.checksum.md5`. In addition to this, a root file system with overlay files can be added to this directory. Complete the following steps to create a basic KIWI configuration:

1. Start an SSH session to `LNXADMIN` as `root`.
2. Create the `/root/kiwi/template/root/` directory with the following **`mkdir`** command:

```
# mkdir -p /root/kiwi/template
Create a basic KIWI configuration:
# kiwi --clone suse-SLE12-JeOS -d /root/kiwi/template
Apr-24 14:11:55 <1> : Cloning image suse-SLE12-JeOS -> /root/kiwi/template...
done
Apr-24 14:11:55 <1> : KIWI exited successfully
```

3. Change the working directory to the kiwi directory and list the template:

```
# cd /root/kiwi/
# ls -l template
total 12
-rw-r--r-- 1 root root 1205 Aug 29  2014 README
-rwxr-xr-x 1 root root 1435 Mar  4  2014 config.sh
-rw-r--r-- 1 root root 1869 Apr 24 14:17 config.xml
drwxr-xr-x 1 root root    6 Apr 20 13:44 root
```

4. Look at the resulting KIWI configuration file (Example 3-1), and edit it according to your needs:

Example 3-1 KIWI configuration file

```
# cd /root/kiwi/template
# vi config.xml
<?xml version="1.0" encoding="utf-8"?>1

<image schemaversion="6.1" name="LimeJeOS-SLE12">2
    <description type="system">3
        <author>Marcus Schaefer</author>
        <contact>cthiel@novell.com</contact>
        <specification>SUSE Linux Enterprise 12 JeOS</specification>
    </description>
    <preferences>4
```

```
        <version>1.1.0</version>
        <packagemanager>zypper</packagemanager>
        <bootsplash-theme>SLES</bootsplash-theme>
        <bootloader-theme>SLES</bootloader-theme>
        <rpm-excludedocs>true</rpm-excludedocs>
        <locale>en_US</locale>
        <keytable>us.map.gz</keytable>
        <timezone>Europe/Berlin</timezone>
        <hwclock>utc</hwclock>
        <type image="vmx" file system="ext3" boot="vmxboot/suse-SLES12"
bootloader="zipl" primary="true"/>
        <type image="oem" filesystem="ext3" boot="oemboot/suse-SLES12"
bootloader="zipl">
            <systemdisk/>
            <oemconfig>
                <oem-swap>true</oem-swap>
                <oem-swapsize>512</oem-swapsize>
            </oemconfig>
        </type>
        <type image="pxe" file system="ext3" boot="netboot/suse-SLES12"
bootloader="zipl"/>
    </preferences>
    <users group="root">5
        <user password="$1$wYJUgpM5$RXMMeASDc035eX.NbYWFl0" home="/root"
name="root"/>
    </users>
    <repository type="yast2">6
        <source path="file:///srv/ftp/SLE-12-Server-DVD-s390x-GM-DVD1.iso"/>
    </repository>
    <packages type="image">7
        <namedCollection name="base"/>
        <product name="SLES"/>
        <package name="cmsfs"/>
        <package name="kernel-default"/>
        <package name="iputils"/>
        <package name="vim"/>
        <package name="s390-tools"/>
    </packages>
    <packages type="bootstrap">
        <package name="file system"/>
        <package name="glibc-locale"/>
        <package name="cracklib-dict-full"/>
    </packages>
</image>
```

Notes:

The following numbers correspond to the marked lines in Example 3-1 on page 42:

1 This file is defined as an Extensible Markup Language (XML) file, following XML version 1.0. The allowed character set is UTF-8.

2 All KIWI image configurations have `image` as root element. For those that are familiar with XML schemas, the definition for this schema can be found in the `/usr/share/kiwi/module` directory.

3 Here, add yourself to the description. The specification will likely be changed for each new appliance that is created.

4 In the **preferences** element, several basic decisions are made. Especially the `<type>` element has a major effect on the resulting image. For mainframe, the most useful type is `oem`. With this type, a preload image is created, which can easily be deployed to other disks by simply dumping it on a device.

5 The default password printed here resolves to *Linux*.

6 To make KIWI work correctly, *change the path to an installation server available to you*. Here, it is also possible to add several more repositories. If the build systems have been set up correctly, all dependencies are resolved automatically.

The following list includes common repository sources:

- ftp: Install from a File Transfer Protocol (FTP) source.
- file: Access a locally available ISO image.
- dir: Access a locally available directory tree.

7 The packages that are installed can be complete patterns, or just packages. Possible dependencies are automatically resolved. Therefore, the number of packages that is actually installed might be higher than expected.

5. After creating the configuration file, check that the XML is well-formatted:

```
# xmlwf config.xml
```

6. Do a first KIWI imaging run. This time, KIWI uses a loop device to create the image, which results in a image with 512-Byte blocks. Because DASD has 4 kilobyte (KB) block sizes, this does not boot on DASD:

```
# kiwi --build template/ -d /var/tmp/kiwi/template --type oem
```

Note: If it happens that KIWI complains about an existing volume group, you can remove this with the following commands:

```
# lvchange -a n kiwiVG
# vgremove -f kiwiVG
```

7. The build takes some time. After some time, it will come back with the following message:

```
Apr-24 14:40:29 <1> : Installing zipl on device:
/var/tmp/kiwi/template/oem/LimeJeOS-SLE12.s390x-1.1.0.raw done
Apr-24 14:40:29 <1> : Saving disk label in MBR: 0x8003c2e5... done
Apr-24 14:40:29 <1> : Find build results at: /var/tmp/kiwi/template done
Apr-24 14:40:29 <1> : KIWI exited successfully
Apr-24 14:40:29 <1> : Complete log file at: /var/tmp/kiwi/template/build/image-root.log
```

8. To build the same image as an image for DASD, the real DASD prepared at virtual address `0.0.0201` is needed for the image creation process. In our case, the device was found at `/dev/dasde`. Be careful to give the correct disk, because the image creation process overwrites any data on the target device:

```
# kiwi --build template/ -d /var/tmp/kiwi/template --type oem --targetDevice /dev/dasde
```

The resulting raw image can then be deployed to other DASD.

3.2 Options to adapt the KIWI image

The starting point for configuring most servers is a minimal image, which has just the needed functionality to connect to it and to install software. SUSE also calls such an operating system *JeOS*. This is what has been created in the last section.

Deployment of such an image as standard procedure provides a means to have a consistent setup for all minimal systems. It is also possible to have some standard login mechanism or security measure that is preinstalled for every machine that way.

With the JeOS as base, it is possible to add funtionality as wanted.

3.2.1 Add more software packages

The selection of packages can be a time-consuming task. The exact selection of packages also might vary between different deployments. The image definition that is found below `/root/kiwi/template` only uses a minimal set of packages. Just like with YaST or zypper, there are several ways to add more software.

Adding software patterns

The SUSE Linux Enterprise Server 12 operating system has several software patterns prepared that can be used to simplify the software installation for a specific purpose. To list all of the patterns installed to the local machine, run the command shown in Example 3-2.

Example 3-2 Listing all installed patterns

```
# zypper search -t pattern
Loading repository data...
Reading installed packages...

S | Name              | Summary                            | Type
--+-------------------+------------------------------------+--------
  | 32bit             | 32-Bit Runtime Environment         | pattern
  | Basis-Devel       | C/C++ Compiler and Tools           | pattern
i | Minimal           | Minimal System (Appliances)        | pattern
  | WBEM              | Web-Based Enterprise Management    | pattern
  | apparmor          | AppArmor                           | pattern
i | base              | Base System                        | pattern
  | dhcp_dns_server   | DHCP and DNS Server                | pattern
  | directory_server  | Directory Server (LDAP)            | pattern
  | documentation     | Help and Support Documentation     | pattern
  | file_server       | File Server                        | pattern
  | fips              | FIPS 140-2 specific packages       | pattern
  | gateway_server    | Internet Gateway                   | pattern
  | gnome-basic       | GNOME Desktop Environment          | pattern
  | hwcrypto          | System z HW crypto support         | pattern
  | kvm_server        | KVM Host Server                    | pattern
  | kvm_tools         | KVM Virtualization Host and tools  | pattern
  | lamp_server       | Web and LAMP Server                | pattern
  | mail_server       | Mail and News Server               | pattern
  | oracle_server     | Oracle Server Base                 | pattern
  | printing          | Printing                           | pattern
  | x11               | X Window System                    | pattern
```

To add a specific pattern like `documentation`, add an element called **`namedCollection`** to `template/config.xml`:

1. Change to the directory `/root/kiwi`:
   ```
   # cd /root/kiwi
   ```
2. Edit the file `config.xml`:
   ```
   # vi template/config.xml
   ```
3. Search for the packages element:
   ```
   /packages
   ```
4. Add the documentation collection to the content of the first package element, as shown in the following code:
   ```
       <packages type="image">

           <namedCollection name="base"/>
           <namedCollection name="documentation"/>
           <product name="SLES"/>
           <package name="cmsfs"/>
           <package name="cmsfs"/>
           <package name="kernel-default"/>
           <package name="iputils"/>
           <package name="vim"/>
           <package name="s390-tools"/>
       </packages>
   ```
5. Rebuild the image:
   ```
   # kiwi --build template/ -d /var/tmp/kiwi/template --targetDevice /dev/dasdf --type oem
   ```

The new image also includes all documentation packages that are included in this pattern.

3.2.2 Add software repositories

Software is often spread over several different repositories. Even after just registering SUSE Linux Enterprise Server 12 with the SUSE Customer Center, several extra repositories are available. Also, if some local *Open Build Service* instance is used, the resulting repositories must be added to the kiwi configuration, before any packages can be used:

1. Edit `/root/kiwi/template/kiwi.xml`:
   ```
   # vi /root/kiwi/template/kiwi.xml
   ```
2. Search for the element repository:
   ```
   /repository
   ```
3. Add a new repository. To see the repositories of the current server, use the command:
   ```
   # zypper lr -d
   ```
4. After adding the update repository from the local LNXADMIN to the **`XML`** configuration, the result would look like this:
   ```
   <repository type="yast2">
       <source path="file:///srv/ftp/SLE-12-Server-DVD-s390x-GM-DVD1.iso"/>
   </repository>
   <repository type="rpm-md">
       <source path="dir:///srv/ftp/SLE-12-Server-Update"/>
   </repository>
   ```
5. From here on, the repository is available just as in an installation. You can add packages from there, as described in 3.2.1, “Add more software packages” on page 45. After enabling the update repository, all images will be built with all updates readily installed.

3.2.3 Update configurations and services in the target system

The default configuration not only provides a `config.xml` file, but also a file called `config.sh`. Several functions are prepared in there that can be called. Two config files are sourced during the execution of `config.sh`:

- `.kconfig`
- `.profile`

If you already followed the instructions of this chapter, these files are found at `/var/tmp/kiwi/template/build/image-root`. The `.kconfig` file provides several functions that can be called during the image build. The `.profile` file contains data about the current build.

To active the special little endian (LSB) **`boot.clone`** init script, add this service to `config.sh` at the `Activate services` section:

```
# vi /root/kiwi/template/config.sh
/Activate services
#======================================
# Activate services
#--------------------------------------
suseActivateDefaultServices
suseRemoveService gpm
suseRemoveService nfs
suseRemoveService boot.cpi
suseInsertService clone
suseInsertService sshd
```

3.2.4 Reduce the initial zipl boot timeout

The default timeout for zipl during the first boot of the image is set to 200 seconds. This provides the user enough time to send any parameter to zipl for the kernel command line. However, for many environments, this is not needed.

To reduce the timeout, proceed as follows:

1. Edit the XML configuration of the virtual appliance:

   ```
   # vi /root/kiwi/template/config.xml
   ```

2. Search for the image type oem:

   ```
   /oem
   ```

3. Add the attribute `boottimeout="5"` to the **`type`** element:

   ```
   <type image="oem" filesystem="ext3" boot="oemboot/suse-SLES12"
   bootloader="zipl" boottimeout="5">
   ```

4. After this, the initial boot timeout will be 5 seconds rather than 200.

3.2.5 Update the System during first boot

One of the changes in this book was to obsolete LNXMAINT and the shared 191 disk. Instead, all Linux guests now have a directory on the shared file pool. The file pool though cannot be accessed from within Linux, and therefore there is no way to use **`cmsfs`** to gather system data from CMS. It would of course be possible to reenable it just for the KIWI imaging purpose, but this would have been a little overkill from a management perspective.

To overcome this, a new setup to distribute the configuration data has been done. Now, the configuration data is sent to the reader of the target system by LNXADMIN. Within the target system, a systemd unit file runs a special script found at the files attached to this book, which in turn updates the system during first bootup, as shown in Figure 3-1.

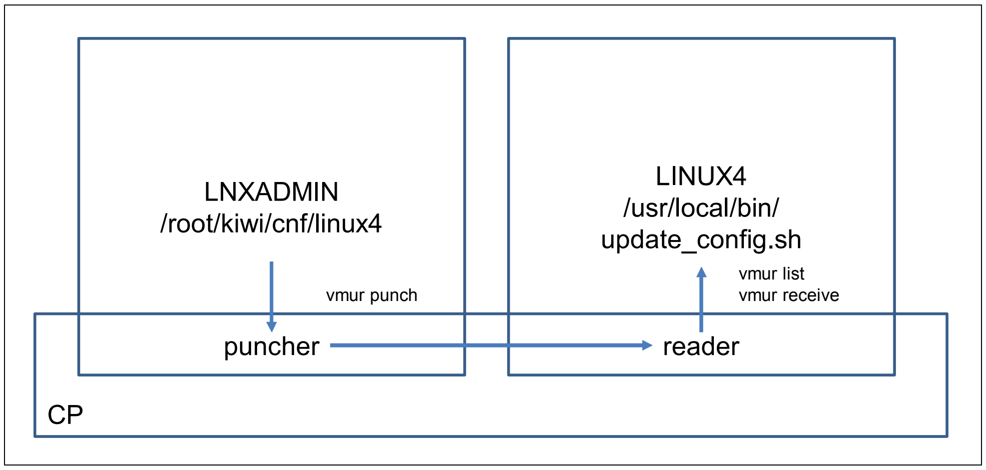


Figure 3-1 KIWI firstboot configuration of target systems

The final configuration of a clone system is done with the script **`update_config.sh`**:

1. Copy this file from the files attached to this book to `/root/kiwi/template/root/usr/local/bin/`:

```
# mkdir -p /root/kiwi/template/root/usr/local/bin
# cp update_config.sh /root/kiwi/template/root/usr/local/bin
# chmod 755 template/root/usr/local/bin/update_config.sh
```

2. To enable the configuration updater, add the following systemd unit file to the overlay file system:

```
# mkdir -p /root/kiwi/template/root/etc/systemd/system
# vi /root/kiwi/template/root/etc/systemd/system/clone.service
[Unit]
Description=System Update Service for cloning systems
After=local-fs.target
Before=wickedd.service wickedd-nanny.service wickedd-dhcp4.service
wickedd-dhcp6.service

[Service]
Type=oneshot
ExecStart=/usr/local/bin/update_config.sh

[Install]
WantedBy=network.target
```

   The script **`update_config.sh`** only starts modification to the system if the z/VM user `LNXADMIN` sends a file with name `CLONE SYSCTL` to the reader of the current guest. If that file exists on the reader, it is received and run. That way, it is even possible to update the systemd configuration during the first bootup of a cloned system.

3. To prepare the KIWI image with these changes, run the following command:

```
# kiwi --build template/ -d /var/tmp/kiwi/myimage --targetDevice /dev/dasdf --type oem
```

3.2.6 Replace or add configuration files

To update files within the virtual KIWI appliance, you can add the needed files to the overlay file system found in `/root/kiwi/template/root`. The method described in this book also enables you to run a special script during bootup of the appliance. This script must be sent by the z/VM user `LNXADMIN` to the respective guest. During the next boot, it will be run early in the boot process, and can update the system configuration.

The script **`kiwi_clone.sh`** provided with the download material for this manual has been updated to create a network setup script from a configuration file. The configuration file looks similar to what used to be a parm file, but has no limitations in terms of line numbers. The script **`clone_kiwi.sh`** takes the information from there, and creates a shell script that is sent to the reader of the cloned guest. The guest itself has a service file and an additional script that reads the configuration script from the reader and updates the guest during first bootup.

To add a configuration file that updates the network configuration of a generic virtual KIWI appliance to the network configuration of **LINUX4**, proceed as follows:

1. Log on to `LNXADMIN` as root.
2. If not already done, change to the KIWI configurations subdirectory:
   ```
   # mkdir -p /root/kiwi/cnf
   ```
3. Edit the configuration file:
   ```
   # vi /root/kiwi/cnf/LINUX4
   Hostname="LINUX4"
   HostIP="9.12.7.101"
   Netmask="255.255.240.0"
   Gateway="9.12.7.97"
   ReadChannel=0.0.0600
   WriteChannel=0.0.0601
   DataChannel=0.0.0602
   Layer2=1
   ```
4. The configuration is then sent to the cloned guest during the run of **`kiwi_clone.sh`**:
   ```
   # cd /root/kiwi
   # bin/kiwi_clone.sh -v file \
   /var/tmp/kiwi/template/LimeJeOS-SLE12.s390x-1.1.0.raw \
   to linux4 at 156d -d cnf
   ```
5. The actual cloning process takes some time. After this process has finished, the guest is started with **`xautolog`**.
6. To watch the actual bootup of the guest, log on to 3270 as maint and run the command:
   ```
   ==> set secuser LINUX4 MAINT
   ```
7. After the system finished bootup, you can ssh to the newly cloned system.

3.2.7 Manually provide an update script

To manually provide a configuration script, the script must be prepared before the cloning script is started. Any script is possible, that can be started with a command like this:

```
# /bin/bash <script>
```

However, because this is transferred to the reader as a text file, it must not exceed 80 characters a line. If more lines are needed, you must create a special script with self-extracting content.

To prepare the script for the cloned KIWI guest, punch it to the virtual reader of the guest:

```
# vmur punch -t -u LINUX4 -N CLONE.SYSCTL
```

The `clone.service` on LINUX4 then starts the script `/usr/local/bin/update_config.sh`. This script will in turn search on the reader for a file that meets the following requirements:

- The file must have been punched by `LNXADMIN`.
- The file name must be `CLONE`.
- The file mode must be `SYSCTL`.

The file will be then copied to `/tmp`, and called as argument to `/bin/bash`.

After the file has been punched to the read, start the script **kiwi_clone** without the **-d** parameter:

```
# bin/kiwi_clone.sh -v file \
/var/tmp/kiwi/template/LimeJeOS-SLE12.s390x-1.1.0.raw to linux4 at 156d
```

In that case, **kiwi_clone.sh** does not punch a configuration and your script is found on the reader.

3.2.8 Add more software not available in SUSE Linux Enterprise Server 12

KIWI provides the possibility to provide overlay files. Any file, that is below the overlay directory tree, will be copied to the system after all RPMs have been unpacked. When adding third party software, always follow the rules that come with that software. If general guidelines are followed, the software is installed to a directory below `/opt/`*<vendor>*`/`*<product>*. If the actual installation is difficult, it is possible to copy the directory tree from that directory to the overlay directory tree.

Overlay files are also used to add the clone service to the built virtual appliance as described in 3.2.3, “Update configurations and services in the target system” on page 47.

If a script must be run during the first bootup to initialize the third-party product, add this to the script **CLONE.SYSCTL** as described in 3.2.6, “Replace or add configuration files” on page 49.

3.3 Create and clone the image

To add a more complex configuration, clone the template configuration to a new directory. To create a new appliance from the existing template, run the following command:

```
# cd /root/kiwi
# kiwi --clone template -d SLES12-appliance
```

This creates a new file system tree from the existing template directory. After doing all needed modifications for that appliance, the commands to rebuild the image is as follows:

```
# kiwi --build SLES12-appliance -d /var/tmp/kiwi/SLES12-appliance --targetDevice \
/dev/dasdf --type oem
```

This reuses the same real DASD for building that was already used during the template configuration. Note, that you can clone directly from that DASD if wanted. If the source and target DASD are of the same size, and IBM FlashCopy® is available, this dramatically increases deployment speed.

The deployment is done similar to the one during the template preparation:

1. Look up the file name of the actual configuration file in the KIWI log file.
2. Set up the network information for the clone target in the `/root/kiwi/cnf` directory.
3. Run **`kiwi_clone.sh`** to deploy the KIWI appliance:

   ```
   # bin/kiwi_clone.sh -v file <image_file> to <guest_name> at <disk_of_guest> -d cnf
   ```

After doing this, the image is automatically started with **`XAUTOLOG`**. If you followed the recommendations of this book, you just have to wait until the system is reachable by network.

3.4 More information

KIWI is a complex imaging system. There is much documentation available to explain this topic.

The following sources provide good entry points:

- The openSUSE website contains helpful information:

  https://en.opensuse.org/Main_Page

- A cookbook is available at the following site:

  http://doc.opensuse.org/projects/kiwi/doc

- To get into the software, see the following website:

  https://github.com/openSUSE/kiwi

- SUSE Studio provides a development environment for images. It also provides the generated kiwi configuration files for download:

  http://susestudio.com

At the time of writing, all the KIWI images had a hardcoded bootloader timeout of 200 seconds. The actual **`boottimeout`** parameter is put into place after the first boot has happened. It is likely that in the meantime, there is an update for KIWI available that removes that limitation.

4

Servicing SUSE Linux Enterprise Server 12

This chapter provides information about common tasks tools available to service SUSE Linux Enterprise Server. The actual setup and configuration of the tools is described in detail in the respective product documentation, and therefore not repeated here:

- 4.1, “System patching philosophies” on page 54
- 4.2, “The Subscription Management Tool” on page 55
- 4.3, “SUSE Manager on z Systems” on page 55
- 4.4, “Managing software and repositories with zypper” on page 56
- 4.5, “The btrfs and grub2 features on SUSE Linux Enterprise Server 12” on page 56

4.1 System patching philosophies

When the first enterprise distribution for mainframe from SUSE appeared in 2000, a new concept was needed to provide updates and patches in a defined and predictable way. The following set of rules was agreed upon (and later adopted by many other distributions):

- Code errors should be fixed in the same package version as was distributed in general availability (GA) level.
- Updates are only done when really needed. The product management must decide if an update is really needed.
- The programming interfaces must stay stable within major releases.
- The source code for all architectures must be consistent. This also means that an RPM Package Manager (RPM) that fails quality assurance (QA) on the mainframe architecture also blocks the same patch for x86.

A process was set up around this, including decision management of product management, the actual code fixing, quality assurance, creating documentation for every patch, and finally making defined release dates possible for fixes. The basic structure of this update process has proven solid and useful over the time and therefore is still in place.

The Linux patching process on SUSE Linux Enterprise Server is different from the one on other operating systems. It does not require the system to keep a golden image, or a preproduction version.

When doing an imaging process, the use of golden images is also deprecated. Instead use the imaging software KIWI to create virtual appliances that in turn can be deployed the same way as golden images can be. Updating such an appliance is just a matter of adding the update repository to the image definition and rebuilding the appliance.

All system files of the distribution are kept in an RPM database, which can correlate between the actual file and a package. These packages are signed with a distribution key, and therefore can be checked by the administrator for validity. That way, RPM also can detect if a system file changed.

A patch for SUSE Linux Enterprise Server commonly contains one or more packages, but also might contain only a script or an image file. All patches are signed and can be checked for validity. The packages that are kept within a patch relate to one or more fixes, which are explained in the patch documentation.

If some unexpected issue shows up with the installer of the operating system, or if for example a new driver is necessary for some specific hardware, it is possible to update the installer by supplying a driver update disk (DUD). With SUSE Linux Enterprise Server 12, such a disk is likely to be needed when doing autoyast installations. In the parmfile, add the parameter **`dud=<url>`** to specify where to find that disk.

4.2 The Subscription Management Tool

To simplify the update process and to reduce bandwidth requirements, SUSE provides a tool at no initial charge called *Subscription Management Tool*. There is currently no specific version for SUSE Linux Enterprise Server 12 available, however, the version available with SUSE Linux Enterprise Server 11 also supports SUSE Linux Enterprise Server 12 update management.

To use this tool, you need a SUSE Linux Enterprise 11 SP3 along with a Maintenance Update to support SUSE Linux Enterprise Server 12.

Most frequently asked questions are covered on the following website:

`https://www.suse.com/solutions/tools/faq.html`

The documentation to this tool is online available on the following website:

`https://www.suse.com/documentation/smt11/book_yep/data/book_yep.html`

4.3 SUSE Manager on z Systems

For businesses that run a huge number of different systems, having a management utility in place is a must. SUSE Manager is based on the open source project Spacewalk and provides many options to manage a multitude of Linux operation systems.

On z Systems, SUSE Manager is available as an appliance that can directly be deployed to emulated DASD (EDEV). Therefore, this can even be installed without performing a single manual installation of Linux on the mainframe.

SUSE Manager not only provides a modern user interface, but also a sophisticated application programming interface (API) that enables other software products to use the functionality provided with SUSE Manager.

For more information about SUSE Manager, visit the following websites:

`https://www.suse.com/products/suse-manager`
`https://www.suse.com/products/suse-manager/features/suse-manager-2-1.html`
`https://www.suse.com/documentation/suse_manager`

4.4 Managing software and repositories with zypper

In Linux there is a distinction between the package manager and a repository manager. The package manager does have any information about what packages would be available to install. It just knows everything about the packages that are already installed, and it can operate on existing packages.

The repository management in SUSE Linux Enterprise is handled by *zypper*. This tool is able to handle several repositories, including the respective debug, update, and source repositories. On a server system, there can easily be more than 10 different repositories that all must be resolved regarding updates and dependencies. Especially with SUSE Linux Enterprise 12 and the newly available extensions, the number of repositories grows substantially.

One of the nice features is to automate the update of all pending patches except those that require interaction. To accomplish this, use the following command in a regular job:

```
# zypper up -t patch --skip-interactive
```

When doing this in an automated way, some kind of monitoring should check if the system is up-to-date. Especially kernel updates still require a reboot, and are not installed automatically with the preceding command.

Find more information about **zypper** at man 8 zypper or the following websites:

https://en.opensuse.org/SDB:Zyper_usage
https://en.opensuse.org/openSUSE:Zypper_features

4.5 The btrfs and grub2 features on SUSE Linux Enterprise Server 12

Some new features on V have more implications on the administration than seen at first. With *btrfs* and *grub2*, there are two new features available that can make a difference with regards to servicing SUSE Linux Enterprise Server.

If the installer finds a disk that is large enough during the installation of the operating system, it enables the snapshot feature of btrfs for system updates. Going back to the system before the update is just a reboot into the system before the update.

Such a reboot would be difficult, if you must add an extra menu entry for **zipl** for each of the snapshots done by snapper. Therefore, SUSE Linux Enterprise Server 12 invented **grub2-emu** to be able to boot into snapshots without having to rewrite the bootloader.

The command **grub2-emu** can even be used to boot into a snapshot from the currently running system. To prevent errors, this requires an additional parameter **--kexec**.

For more information about btrfs and grub2 see the official documentation:

https://www.suse.com/documentation/sles12

Part 2

Other topics

This part of the book includes the following chapters:

5

Working with disks

"*Learn from yesterday, live for today, hope for tomorrow. The important thing is not to stop questioning.*"

— Albert Einstein

This chapter concentrates on the tasks to be performed on Linux, for the z/VM perspective, see *The Virtualization Cookbook for IBM z Systems Volume 1: IBM z/VM 6.3*, SG24-8147 in the section entitled "Considerations for disk storage types". It has the following sections related to working with disks, both IBM extended count key data (IBM ECKD™) direct access storage device (DASD), and Fibre Channel Protocol or Small Computer System Interface (FCP/SCSI) tasks that you might want to perform:

- 5.1, "Add disk space to virtual machines" on page 60
- 5.2, "Add a logical volume" on page 64
- 5.3, "Extend an existing logical volume" on page 68
- 5.4, "Moving a physical volume" on page 71

5.1 Add disk space to virtual machines

This section describes how to add more disk space to a Linux virtual machine. This disk space could come from different types of disks. The respective types are described in *The Virtualization Cookbook for IBM z Systems Volume 1: IBM z/VM 6.3*, SG24-8147, in the section entitled, “Considerations for disk storage types”.

Important: If you add minidisks or to the user directory for a certain virtual machine, it is possible to attach them to a running Linux system without “bouncing” it.

For example, if you added a minidisk at virtual address 104, you can use the following commands to link to the disk and then enable it:

```
# vmcp link '* 104 104 mr'
# chccwdev -e 104
```

5.1.1 Make new minidisks or CKD DASD available in SUSE Enterprise Linux 12

After getting new minidisks or count key data (CKD) DASD, for example at the addresses 0.0.0102, 0.0.0103, and 0.0.0104, make the new disks available by performing the following steps:

1. Make the disks visible with the **cio_ignore** command:

```
# cio_ignore -r 102
# cio_ignore -r 103
# cio_ignore -r 104
```

2. Use the **dasd_configure** command to enable minidisks 102, 103, and 104:

```
# dasd_configure 0.0.0102 1
Configuring device 0.0.0102
Setting device online
# dasd_configure 0.0.0103 1
Configuring device 0.0.0103
Setting device online
# dasd_configure 0.0.0104 1
Configuring device 0.0.0104
Setting device online
```

3. View the available disks again with the **lsdasd** command:

```
# lsdasd
Bus-ID     Status      Name      Device  Type  BlkSz  Size      Blocks
==============================================================================
0.0.0301   active      dasda     94:0    FBA   512    512MB     1048576
0.0.0300   active      dasdb     94:4    FBA   512    256MB     524288
0.0.0100   active      dasdc     94:8    ECKD  4096   3521MB    901440
0.0.0101   active      dasdd     94:12   ECKD  4096   3521MB    901440
0.0.0102   n/f         dasde     94:16   ECKD
0.0.0103   n/f         dasdf     94:20   ECKD
0.0.0104   n/f         dasdg     94:24   ECKD
```

4. Format the disks in parallel with the **dasdfmt** command using a **for** loop and putting them in the background:

```
# for i in e f g
> do
>   dasdfmt -b 4096 -y -f /dev/dasd$i &
> done
[1] 1923
[2] 1924
[3] 1925
Finished formatting the device.
Rereading the partition table... ok
Finished formatting the device.
Rereading the partition table... ok
Finished formatting the device.
Rereading the partition table... ok

[1]   Done                   dasdfmt -b 4096 -y -f /dev/dasd$i
[2]-  Done                   dasdfmt -b 4096 -y -f /dev/dasd$i
[3]+  Done                   dasdfmt -b 4096 -y -f /dev/dasd$i
```

5. Create one partition from each of the disks using a bash **for** loop and the **fdasd -a** command:

```
# for i in e f g
> do
>   fdasd -a /dev/dasd$i
> done
reading volume label ..: VOL1
reading vtoc ..........: ok
auto-creating one partition for the whole disk...
...
```

The three new minidisks should now be low-level-formatted, partitioned, and configured to be active at boot time.

If you are creating a new logical volume, see 5.2.1, “Create a logical volume and file system” on page 64. If you are extending an existing logical volume, skip ahead to 5.3, “Extend an existing logical volume” on page 68.

5.1.2 Making new EDEV available in SUSE Enterprise Linux 12

After getting new emulated DASD (EDEV), for example at address `0.0.0150`, dedicated to your system, the process to integrate them into the system is similar to what is done with CKD DASD. The main difference is that other tools are used to format and partition these disks:

1. Make the disk visible with the command **cio_ignore**:

```
# cio_ignore -r 150
```

2. Use the **dasd_configure** command to enable the DASD 150:

```
# dasd_configure 0.0.0150 1
Configuring device 0.0.0150
Setting device online
```

3. View the available disks again with the **lsdasd** command:

```
# lsdasd
Bus-ID     Status      Name      Device  Type  BlkSz  Size      Blocks
==============================================================================
0.0.0301   active      dasda     94:0    FBA   512    512MB     1048576
0.0.0300   active      dasdb     94:4    FBA   512    256MB     524288
0.0.0100   active      dasdc     94:8    ECKD  4096   3521MB    901440
0.0.0101   active      dasdd     94:12   ECKD  4096   3521MB    901440
0.0.0150   active      dasde     94:16   FBA   512    10240MB   20971520
```

4. Create a partition on the disk using the **parted** command:

```
# parted -s /dev/dasde mklabel msdos mkpart primary 0% 100%
```

The new DASD should now be partitioned, and configured to be active at boot time.

If you are creating a new logical volume, see 5.2.1, “Create a logical volume and file system” on page 64. If you are extending an existing logical volume, skip ahead to 5.3, “Extend an existing logical volume” on page 68.

5.1.3 Make new zFCP LUN available in Linux

To use FCP in a single system image (SSI) environment, it is necessary to understand that within Linux there are also more adapters to be handled than visible in just one SSI node. SUSE Linux Enterprise Server 12 changed the behavior of FCP to automatic logical unit number (LUN) detection. This means, it is sufficient to configure the host adapters and only use the multipathed device for disk configurations.

This section is assuming that no previous zFCP was available. The needed setup for IBM z/VM is described in detail in *The Virtualization Cookbook for IBM z Systems Volume 1: IBM z/VM 6.3,* SG24-8147 in the section entitled, “Direct attached Fibre Channel”. The planning according to this manual creates two FCP adapters at the addresses `0.0.fc00` and `0.0.fd00`:

1. Start a Secure Shell (SSH) session to the target system.
2. Verify that there are two devices available by using the **CP QUERY FCP** command:

```
#  vmcp q v fcp
FCP  FC00 ON FCP   B801 CHPID 70 SUBCHANNEL = 0001
     FC00                        TOKEN      = 00000007F62EA280
     FC00 DEVTYPE FCP         VIRTUAL CHPID FF FCP REAL CHPID 70
     FC00 QDIO ACTIVE         QIOASSIST ACTIVE        QEBSM
     FC00
     FC00 INP + 01 IOCNT = 00001346  ADP = 128 PROG = 000 UNAVAIL = 000
     FC00          BYTES = 0000000000000000
     FC00 OUT + 01 IOCNT = 00001464  ADP = 000 PROG = 128 UNAVAIL = 000
     FC00          BYTES = 00000000005711FE
     FC00 DATA ROUTER ACTIVE
     WWPN C05076DD90000404
FCP  FD00 ON FCP   B901 CHPID 71 SUBCHANNEL = 0002
     FD00                        TOKEN      = 00000007F62EA380
     FD00 DEVTYPE FCP         VIRTUAL CHPID 71 FCP REAL CHPID 71
     FD00 QDIO ACTIVE         QIOASSIST ACTIVE        QEBSM
     FD00
     FD00 INP + 01 IOCNT = 00001338  ADP = 128 PROG = 000 UNAVAIL = 000
     FD00          BYTES = 0000000000000000
     FD00 OUT + 01 IOCNT = 00001428  ADP = 000 PROG = 128 UNAVAIL = 000
     FD00          BYTES = 000000000052EF86
     FD00 DATA ROUTER ACTIVE
     WWPN C05076DD90000A64
```

3. Make the disk visible with the command **cio_ignore**:

```
# cio_ignore -r fc00
# cio_ignore -r fd00
```

4. Enable the FCP adapters **zfcp_host_configure** command:

```
# zfcp_host_configure 0.0.fc00 1
# zfcp_host_configure 0.0.fd00 1
```

5. Verify that the auto LUN scan feature detected all the paths to the LUNs:

```
# lsluns
Scanning for LUNs on adapter 0.0.fc00
        at port 0x500507630500c74c:
                0x4010401700000000
        at port 0x50050763050bc74c:
                0x4010401700000000
Scanning for LUNs on adapter 0.0.fd00
        at port 0x500507630510c74c:
                0x4010401700000000
        at port 0x50050763051bc74c:
                0x4010401700000000
```

6. Set up a multipath configuration, if not already configured:
   a. Make sure that the `multipath-tools` RPM Package Manager(RPM), formerly known as Red Hat Package Manager, is installed with the following **zypper** command:

   ```
   # zypper in multipath-tools
   ```

   b. Run the multipath daemon:

   ```
   # systemctl enable multipathd
   # systemctl start multipathd
   ```

   c. Create a partition on the disk using the **parted** command:

   ```
   # parted -s /dev/mapper/mpatha mklabel msdos mkpart primary 0% 100%
   ```

   d. Use YaST to set up the partitioning for the multipath device. In this case, the FCP disk becomes the LUN in an LVM group for the `/srv/` directory:
      i. Run **yast** → **System** → **Partitioner**.
      ii. Click **Yes** if you are asked if you really want to use this tool.
      iii. Select **System View** → **Hard Disks** and press the Plus sign (+).
      There is a new device available that represents the multipathed FCP disks.
      iv. Add a partition that covers the full disk. Use **Raw Volume** and **Do not format partition** and **Do not mount partition**.
      v. Select **System View** → **Volume Management**.
      vi. Click **Add** → **Volume Group**.
      Use `vg_srv` as Volume Group Name.
      vii. Add the device to the volume group.
      viii.Click **Finish**.
      ix. Select **System View** → **Volume Management**.
      x. Click **Add** → **Logical Volume**.
      xi. Set the name of the Logical Volume to `srv` and then click **Next**.
      xii. Use **Maximum Size** and click **Next**.
      xiii.Select Format partition and use file system XFS.
      xiv.Select **Mount partition** and set the **Mount Point** to `/srv`.
      xv. Click **Finish** and **Next**.
      xvi.Click **Finish** and leave YaST with **Quit**.

e. Check if all paths are online:

```
# multipath -ll
36005076305ffc74c0000000000001119 dm-4 IBM,2107900
size=10G features='1 queue_if_no_path' hwhandler='0' wp=rw
`-+- policy='service-time 0' prio=1 status=active
  |- 0:0:0:1075396625 sda 8:0  active ready running
  |- 0:0:1:1075396625 sdb 8:16 active ready running
  |- 1:0:0:1075396625 sdc 8:32 active ready running
  `- 1:0:1:1075396625 sdd 8:48 active ready running
```

7. To only activate a newly-created LUN to an existing volume group, it is sufficient to run the command:

```
# rescan_scsi_bus -a
```

5.2 Add a logical volume

There are times when you require more disk space than a single direct access storage device (DASD) volume provides. For example, if you want to have a shared `/home/` directory you want it to be of sufficient size for many users to write data to. When this is the case, you can use the Logical Volume Manager (LVM) to combine multiple DASD volumes into one logical volume. This example does not create a large logical volume, but it shows all the steps necessary to do so.

The following sections describe a logical volume with more DASD on a Linux guest. Use the following overall steps to add a logical volume:

1. “Create a logical volume and file system”
2. “Update the file system table” on page 67

5.2.1 Create a logical volume and file system

The following overall steps are involved in creating a logical volume:

1. Create physical volumes from the two partitions.
2. Create a single volume group.
3. Create a single logical volume.
4. Make a file system from the logical volume.

Figure 5-1 on page 65 shows a block diagram of the logical volume manager.

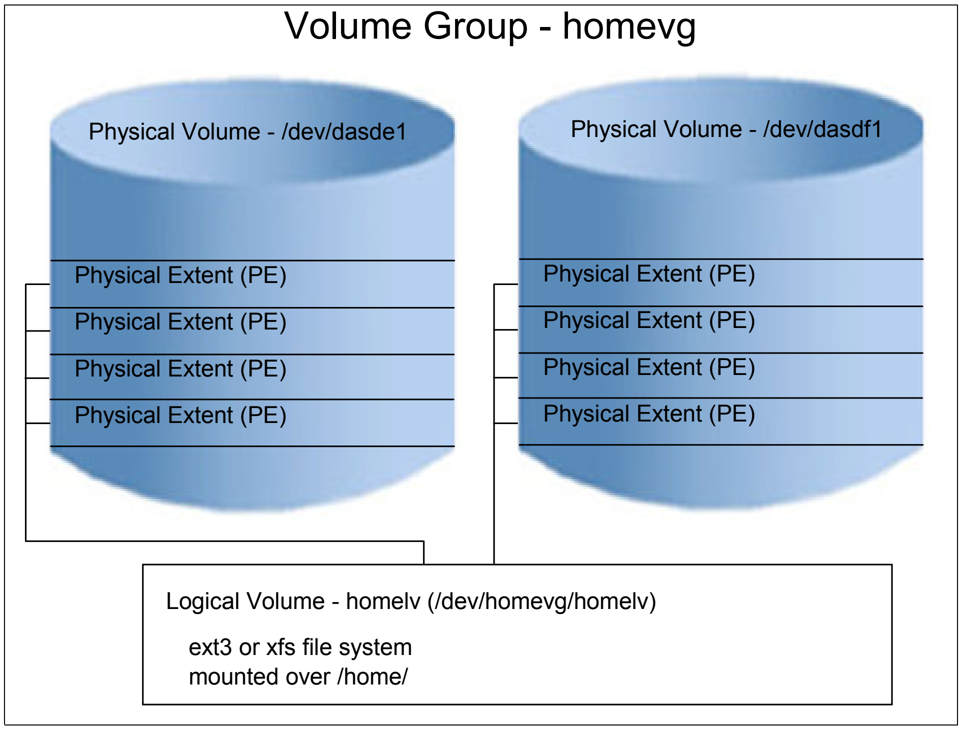


Figure 5-1 LVM block diagram

Create physical volumes from two minidisks

To create physical volumes from new minidisks at virtual device addresses 102 and 103, complete the following steps:

1. Check the devices on your system with the **lsdasd** command.
2. The **pvcreate** command initializes partitions for use by LVM. Initialize the two new DASD partitions:

```
# pvcreate /dev/dasde1 /dev/dasdf1
  Physical volume "/dev/dasde1" successfully created
  Physical volume "/dev/dasdf1" successfully created
```

3. Verify that the physical volumes were created using the **pvdisplay** command:

```
# pvdisplay /dev/dasde1 /dev/dasdf1
  "/dev/dasde1" is a new physical volume of "3.44 GiB"
  --- NEW Physical volume ---
  PV Name               /dev/dasde1
  VG Name
  PV Size               3.44 GiB
  Allocatable           NO
  PE Size               0
  Total PE              0
  Free PE               0
  Allocated PE          0
  PV UUID               sOugfl-hlV3-fYnf-1adW-4mOI-4HTJ-HdAOTU
```

```
    "/dev/dasdf1" is a new physical volume of "3.44 GiB"
    --- NEW Physical volume ---
    PV Name               /dev/dasdf1
    VG Name
    PV Size               3.44 GiB
    Allocatable           NO
    PE Size               0
    Total PE              0
    Free PE               0
    Allocated PE          0
    PV UUID               vO2PJY-gy4x-M9Hj-kt51-TO4J-B4n5-Ntvkje
```

Create a single volume group

The **vgcreate** command is used to create a volume group named `homevg` from the two partitions. Use the **vgdisplay homevg** command to verify that the volume group was created:

```
# vgcreate homevg /dev/dasde1 /dev/dasdf1
Volume group "homevg" successfully created
# vgdisplay homevg
  --- Volume group ---
  VG Name               homevg
  System ID
  Format                lvm2
  Metadata Areas        2
  Metadata Sequence No  1
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                0
  Open LV               0
  Max PV                0
  Cur PV                2
  Act PV                2
  VG Size               6.88 GiB
  PE Size               4.00 MiB
  Total PE              1760
  Alloc PE / Size       0 / 0
  Free  PE / Size       1760 / 6.88 GiB
  VG UUID               acSF65-56Ie-kVoY-Af6I-Hma4-VVuN-ggJEs5
```

In this example, there are 1760 free physical extents.

Create a single logical volume

In this section, you create a single logical volume using the **lvcreate** command:

1. The **lvcreate** command is used to create a logical volume. The **-i** (a lowercase I) flag specifies the number of stripes, in this case two, because there are two volumes in the volume group. The **-l** (a lowercase L) flag specifies the number of logical extents, 1760 in this example. The **-n homelv** specifies the name of the new logical volume. The last argument **homevg** specifies the name of the volume group from which the logical volume will be created:

   ```
   # lvcreate -i 2 -l 1760 -n homelv homevg
     LUsing default stripesize 64.00 KiB
     Logical volume "homelv" created
   ```

2. Use the **lvdisplay** command to verify. The parameter is the full path of the logical volume, not just the logical volume name:

```
# lvdisplay /dev/homevg/homelv
  --- Logical volume ---
  LV Path                /dev/homevg/homelv
  LV Name                homelv
  VG Name                homevg
  LV UUID                qNcyDp-Eeqs-gfBl-XU5Z-Jt3K-QfvV-pf3Kos
  LV Write Access        read/write
  LV Creation host, time virtcook3.itso.ibm.com, 2013-06-17 15:32:39 -0400
  LV Status              available
  # open                 0
  LV Size                6.88 GiB
  Current LE             1760
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     512
  Block device           253:4
```

Make a file system from the logical volume

Create a file system from the new logical volume. When using SUSE Linux Enterprise Server 12, XFS is the suggested file system for data. Use the following command to make the file system:

```
# mkfs.xfs /dev/homevg/homelv
...
```

The file system created from the logical volume is now ready to be mounted.

5.2.2 Update the file system table

You could at this point mount the file system manually. However, if you add the mount to the file system table file, `/etc/fstab`, you can effectively test the change by using the **mount** command with only one argument. Perform the following steps:

1. Make a backup copy of the file then add the following line to it:

```
# cd /etc
# cp fstab fstab.works
```

2. Add one line to the `fstab` file:

```
# vi fstab
/dev/homevg/homelv      /home                  xfs     defaults        0 0
...
```

3. Before mounting over `/home/`, you might want to check that it is empty. If a non-root user exists and a new file system is mounted over it, the contents of the directory are *hidden*. In this example, there is no data in the file system:

```
# ls -a /home
.  ..
```

4. Mount the `/home/` file system with one argument. By using just one argument, you are testing the change to `/etc/fstab`. Use the **`df -h`** command to verify that it is mounted:

```
# mount /home
# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/dasdc1          1008M  184M  774M  20% /
tmpfs                 246M     0  246M   0% /dev/shm
/dev/mapper/system_vg-opt_lv
                      504M   17M  462M   4% /opt
/dev/mapper/system_vg-tmp_lv
                      504M   17M  462M   4% /tmp
/dev/mapper/system_vg-usr_lv
                      2.0G  1.3G  617M  68% /usr
/dev/mapper/system_vg-var_lv
                      504M   92M  388M  20% /var
/dev/mapper/homevg-homelv
                      6.8G  144M  6.3G   3% /home
```

5. Test a reboot to verify that the new logical volume is successfully mounted over `/home/`:

```
# reboot
Broadcast message from root@virtcook3.itso.ibm.com
        (/dev/pts/0) at 15:51 ...

The system is going down for reboot NOW!
```

When the system comes back, you should see the new logical volume mounted over `/home/`.

5.3 Extend an existing logical volume

This section describes the process of adding a minidisk to an existing LVM. This is useful when your logical volume has run out of space. In this example, the `/var/` file system is filling up on `LINUX3`:

```
# df -h /var/
Filesystem            Size  Used Avail Use% Mounted on
/dev/mapper/system_vg-var_lv
                      504M  392M   88M  82% /var
```

A 3390-9 was added as minidisk `106` in section 5.1, “Add disk space to virtual machines” on page 60.

Attention: It is possible to attach minidisks to a running Linux system without rebooting it. For example, if you added a minidisk at virtual address `106`, from a root SSH session, use the **`vmcp link * 106 106 mr`** command to link to the minidisk, then **`chccwdev -e 106`** to enable it.

To extend the logical volume using this disk, perform the following steps:

1. Use the **vgdisplay** command to see the free space in the volume group system_vg:

   ```
   # vgdisplay system_vg
     --- Volume group ---
     VG Name               system_vg
     System ID
     Format                lvm2
     Metadata Areas        2
     Metadata Sequence No  6
     VG Access             read/write
     VG Status             resizable
     MAX LV                0
     Cur LV                5
     Open LV               4
     Max PV                0
     Cur PV                2
     Act PV                2
     VG Size               5.88 GiB
     PE Size               4.00 MiB
     Total PE              1504
     Alloc PE / Size       1504 / 5.88 GiB
     Free  PE / Size       0 / 0
     VG UUID               4i89gF-b0xm-dkHo-blWP-3Kca-0xCI-V6TAXk
   ```

   This shows that there are no free extents in the volume group.

2. Use the **lsdasd** command to show the enabled disks:

   ```
   # lsdasd
   Bus-ID     Status      Name      Device  Type  BlkSz  Size      Blocks
   ==============================================================================
   0.0.0100   active      dasda     94:0    ECKD  4096   3521MB    901440
   0.0.0301   active      dasdb     94:4    FBA   512    512MB     1048576
   0.0.0300   active      dasdc     94:8    FBA   512    256MB     524288
   0.0.0101   active      dasdd     94:12   ECKD  4096   3521MB    901440
   0.0.0102   active      dasde     94:16   ECKD  4096   3521MB    901440
   0.0.0103   active      dasdf     94:20   ECKD  4096   3521MB    901440
   0.0.0104   active      dasdg     94:24   ECKD  4096   7042MB    1802880
   ```

   This shows that minidisk 104 is at /dev/dasdg.

3. Make minidisk 104 a physical volume with the **pvcreate** command:

   ```
   # pvcreate /dev/dasdg1
     Physical volume "/dev/dasdg1" successfully created
   ```

4. Use the **vgextend** command to add the minidisk to the volume group:

   ```
   # vgextend system_vg /dev/dasdg1
     Volume group "system_vg" successfully extended
   ```

5. Use the **vgdisplay** command again to show the free extents in the volume group:

```
# vgdisplay system_vg
  --- Volume group ---
  VG Name               system_vg
  System ID
  Format                lvm2
  Metadata Areas        3
  Metadata Sequence No  7
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                5
  Open LV               4
  Max PV                0
  Cur PV                3
  Act PV                3
  VG Size               12.75 GiB
  PE Size               4.00 MiB
  Total PE              3264
  Alloc PE / Size       1504 / 5.88 GiB
  Free  PE / Size       1760 / 6.88 GiB
  VG UUID               4i89gF-bOxm-dkHo-blWP-3Kca-OxCI-V6TAXk
```

This shows that there are now 1760 free extents in the volume group.

6. Use the **mount** command to determine the name of the logical volume mounted over `/var/`:

```
# mount | grep "\/var "
/dev/mapper/system_vg-var_lv on /var type ext4 (rw)
```

In this example, it is `/dev/mapper/system_vg-var_lv/`.

7. Use the **lvextend** command to extend the volume group with all of the new extents:

```
# lvextend -l +1760 /dev/mapper/system_vg-var_lv
  Extending logical volume var_lv to 7.38 GiB
  Logical volume var_lv successfully resized
```

8. Use the **resize2fs** command to increase the size of the EXT4 file system while it is still mounted:

```
# resize2fs /dev/mapper/system_vg-var_lv
resize2fs 1.41.12 (17-May-2010)
File system at /dev/mapper/system_vg-var_lv is mounted on /var; online resizing
required
old desc_blocks = 1, new_desc_blocks = 1
Performing an online resize of /dev/mapper/system_vg-var_lv to 1933312 (4k)
blocks.
The file system on /dev/mapper/system_vg-var_lv is now 1933312 blocks long.
```

9. Use the **xfs_growfs** command to increase the size of the XFS while it is still mounted:

```
# xfs_growfs /dev/mapper/system_vb-var_lv
```

10. Use the **df** command to show the file system size before and after you extend it, as the following example shows:

```
# df -h /var
File system                    Size  Used Avail Use% Mounted on
/dev/mapper/system_vg-var_lv    7.3G  393M  6.6G   6% /var
```

This shows that the `/var/` file system now has 6.6 GB of free space.

5.4 Moving a physical volume

This section has shows how to move data from one physical volume to another without taking the file system offline.

In addition to file systems getting larger, you might need to move data off one or more volumes onto another or target set of volumes. If your data is in LVM, the **pvmove** and **vgreduce** commands were designed for this, and can be used with the file system online.

In this example, two physical volumes exist, `/dev/dasde1` and `/dev/dasdf1`. Data is populated on the first volume, and later moved to the second. This is done while the file system is online.

To complete this test, perform the following steps:

1. Create a volume group from the first logical volume. In this example, it is named `homelv`:

   ```
   # vgcreate homevg /dev/dasde1
     Volume group "homevg" successfully created
   ```

2. Observe the number of physical extents:

   ```
   # vgdisplay homevg | grep "Total PE"
     Total PE              1760
   ```

3. Create a logical volume from the volume group. In this example, it is named `homelv` and all physical extents are used:

   ```
   # lvcreate -l 1760 -n homelv homevg
     Logical volume "homelv" created
   ```

4. Create a file system from the logical volume. In this example, it is of type ext4:

   ```
   # mkfs.ext4 /dev/homevg/homelv
   ```

5. Add the new file system to the file system table and mount it:

   ```
   # vi /etc/fstab
   ...
   # grep home /etc/fstab
   /dev/homevg/homelv      /home                  ext4    defaults        0 0
   # mount /home
   ```

6. Create a sizable file on it with the **dd** command and show file system usage:

   ```
   # dd if=/dev/zero of=/home/bigfile bs=1M count=500
   500+0 records in
   500+0 records out
   524288000 bytes (524 MB) copied, 3.0718 s, 171 MB/s
   # df -h | grep home
   /dev/mapper/homevg-homelv  6.8G  644M  5.8G  10% /home
   ```

7. Show the volume group usage with the **vgdisplay** command:

   ```
   # vgdisplay homevg
     --- Volume group ---
     VG Name               homevg
     VG Size               6.88 GiB
     PE Size               4.00 MiB
     Total PE              1760
     Alloc PE / Size       1760 / 6.88 GiB
     Free  PE / Size       0 / 0
     VG UUID               YIQgoN-865f-3Vbf-tjH1-eXhO-Aa6W-PcxHri
   ```

   This shows that all physical extents in the volume group are used.

8. Add a second physical volume, which will be the target of the data move, to the volume group:

```
# vgextend homevg /dev/dasdf1
  Volume group "homevg" successfully extended
```

9. Show the volume group usage again:

```
# vgdisplay homevg
  --- Volume group ---
  VG Name               homevg
  ...
  VG Size               13.75 GiB
  PE Size               4.00 MiB
  Total PE              3520
  Alloc PE / Size       1760 / 6.88 GiB
  Free  PE / Size       1760 / 6.88 GiB
  VG UUID               YIQgoN-865f-3Vbf-tjH1-eXhO-Aa6W-PcxHri
```

This shows that the volume group doubled in size, and there are now an equal number of free extents.

10. Move the data off the source physical volume with the **pvmove** command. The target does not need to be specified:

```
# pvmove /dev/dasde1
  /dev/dasde1: Moved: 0.0%
  /dev/dasde1: Moved: 8.0%
  /dev/dasde1: Moved: 18.9%
  /dev/dasde1: Moved: 34.2%
  /dev/dasde1: Moved: 49.1%
  /dev/dasde1: Moved: 63.2%
  /dev/dasde1: Moved: 77.6%
  /dev/dasde1: Moved: 92.7%
  /dev/dasde1: Moved: 100.0%
```

11. Show the volume group usage again:

```
# vgdisplay homevg
  --- Volume group ---
  VG Name               homevg
  ...
  VG Size               13.75 GiB
  PE Size               4.00 MiB
  Total PE              3520
  Alloc PE / Size       1760 / 6.88 GiB
  Free  PE / Size       1760 / 6.88 GiB
  VG UUID               YIQgoN-865f-3Vbf-tjH1-eXhO-Aa6W-PcxHri
```

These free and used extents are the same, however, the data has been moved.

12. Show the free and used extents on the source and target physical volumes with the **pvdisplay** command:

```
# pvdisplay /dev/dasde1 /dev/dasdf1
  --- Physical volume ---
  PV Name               /dev/dasde1
  VG Name               homevg
  PV Size               6.88 GiB / not usable 2.41 MiB
  Allocatable           yes
  PE Size               4.00 MiB
  Total PE              1760
```

```
Free PE               1760
Allocated PE          0
PV UUID               Jo2fa3-5cc0-y2Xs-e0DQ-wQXc-i3er-MPcckW

--- Physical volume ---
PV Name               /dev/dasdf1
VG Name               homevg
PV Size               6.88 GiB / not usable 2.41 MiB
Allocatable           yes (but full)
PE Size               4.00 MiB
Total PE              1760
Free PE               0
Allocated PE          1760
PV UUID               hme2qP-6ytn-Drg8-Wba4-rTU1-q1sV-pVZO3g
```

13. Remove the source physical volume:

```
# vgreduce homevg /dev/dasde1
  Removed "/dev/dasde1" from volume group "homevg"
```

The source volume should now be ready for reassignment, or retirement.

6

Monitoring IBM z/VM and Linux

“*Not everything that can be counted counts, and not everything that counts can be counted.*”

— Albert Einstein

This chapter briefly describes how to monitor IBM z/VM and Linux. For another source on z/VM performance and monitoring, see the chapter about monitoring performance and capacity in the manual *Getting Started With Linux on System z*, SC24-6096, which is available on the web at the following site:

`http://publib.boulder.ibm.com/cgi-bin/bookmgr/download/HCSXOC20.pdf?DT=20130528134905&XKS=hcsh2ac2`

There are several z/VM monitoring tools such as the CA VM:Monitor, IBM z/VM Performance Toolkit, IBM Tivoli® OMEGAMON® XE for z/VM and Linux, and products from IBM Velocity Software. The IBM z/VM Performance Toolkit is briefly described in this chapter.

This chapter contains the following sections:

- 6.1, “Using basic z/VM commands” on page 76
- 6.2, “The z/VM Performance Toolkit” on page 79
- 6.3, “Collect and use raw CP monitor data” on page 87
- 6.4, “Monitoring Linux performance for troubleshooting” on page 91

6.1 Using basic z/VM commands

IBM z/VM has many commands to monitor the state of the system. **CP INDICATE** is the most commonly used, and there are other commands that are addressed. For more information, see the *z/VM Performance Resources* web page at the following site:

http://www.vm.ibm.com/perf

6.1.1 Using the INDICATE command

z/VM has some basic commands, such as **INDICATE**. There are many **INDICATE** parameters that can be included as command-line options. Use the **HELP INDICATE** command for a basic understanding, and then press **F11** for help on each parameter.

INIDICATE LOAD

If no parameter is specified, **INDICATE LOAD** is the default option. There are two versions of this, depending on whether the issuing virtual machine has privilege class G or class E. Class G users can use **INDICATE** to display recent contention for system resources, display environment characteristics, and measurements of resources used by their virtual machine.

The output from virtual machines with class E privilege (for example, MAINT, OPERATOR) is shown here. The lines are numbered for clarity with a description that follows:

```
==> ind load
1  AVGPROC-000% 04
2  MDC READS-000068/SEC WRITES-000001/SEC HIT RATIO-099%
3  PAGING-0/SE
4  Q0-00001(00000)                           DORMANT-00012
5  Q1-00000(00000)           E1-00000(00000)
6  Q2-00001(00000) EXPAN-001 E2-00000(00000)
7  Q3-00001(00000) EXPAN-001 E3-00000(00000)
8
9  PROC 0000-000% CP   VM     PROC 0001-000% CP   VL
10 PROC 0002-000% IFL  VM     PROC 0003-000% IFL  VL
11
12 LIMITED-00000
```

The **INDICATE LOAD** command gives a snapshot of current system performance. Except for the counts of virtual machines in various queues and the limited list, the values you see here are a smoothed average over the past 4 minutes. The following areas are where z/VM performance analysts tend to focus:

- AVGPROC on line 1 gives the overall processor utilization, 38% in this example. The number following it is the number of online processors, 3 in this example. The individual processor utilization is shown on lines 9 and 10. Take a glance at these to see if they are somewhat balanced. There are cases where an imbalance is okay.

  This would include very low usage scenarios or cases where there are not enough users ready to run virtual processors to keep the physical processors busy. One of the processors is a Master, all of the others Alternative, and some imbalance can result from performing these functions. Another imbalance can also come from vertical CPU management.

- Multi-dimensional clustering (MDC) statistics are given on the second line. The effectiveness of MDC can be judged by the combination of the `READS` rate and the `HIT RATIO`. If both are high, many physical input/output operations (I/Os) are avoided due to the MDC feature.

  For a system that has an appreciably high I/O rate, which is composed of reads plus writes, and a high proportion of reads, and a good hit ratio for those reads (tending to 90% or greater), the real, physical I/O avoidance can be very high. The authors have seen the avoidance as high as 50% in some cases. Conversely, however, a high `HIT RATIO` with a low value for the `READS` rate should not be taken as good (a 100% hit ratio, when doing only one I/O per second, is effectively meaningless).
- Line **3** describes more storage (memory) management. The `PAGING` rate is important. Higher values often affect performance. This can be at least partially offset by increasing the number of page volumes, but a more thorough examination of this problem is advisable whenever it arises.
- On lines **4** - **7**, you also see a series of counters that represent the users in various queues. The z/VM scheduler classifies work into three different classes (1 - 3) and a special additional class labeled *zero*. So the column of Q_x and E_x values represents the virtual machines in the dispatch list and the eligible list.

  The most important value to validate here is that there are no virtual machines in the Eligible list: E1, E2, E3. This implies z/VM has stopped dispatching some virtual machines to avoid over-committing resources. Such a system would require further investigation, possibly leading to some tuning work, or even hardware addition in extreme cases. Do not worry about the values in parenthesis.

INDICATE QUEUES EXP

Another useful command to help you understand the state of the system is **INDICATE QUEUES EXP**, as shown in the following example:

```
==> ind q exp
MAINT          Q1 R00   00001623/00001552   .I..  .0004
TCPIP          Q0 PS    00003496/00003178   .I..  99999
```

This is another class E command, and displays the virtual processors associated with a given virtual machine (that can have multiple virtual processors), what queue (dispatch list, eligible list, or limit list) they are in, and what state they are in. This is a snapshot in time. Again, you want to check this output to ensure that there are no virtual machines in the eligible list. Normal virtual processors in the dispatch list are Q_x (x=1, 2, or 3). The eligible list would be marked as E_x. The third column in the example also gives the state of the virtual processor.

This can be helpful to get an idea of how the virtual processors might be constrained. Virtual processors that are actually running in the snapshot period are marked with an `R`*`NN`* where *`NN`* is the processor number they are on. An `R` without a number means that the virtual processor is ready to run, but there is not an available processor.

Important: The virtual machine that issues the **INDICATE** command is always one of the running machines.

Other states are documented in the help for **IND Q EXP**. One does not have to be concerned about the other columns unless detailed analysis is required, or if IBM support requests it. Also, always remember that this is just a snapshot in time, so repeating this command often over time can give a more accurate picture of your z/VM system. A single snapshot cannot be regarded as indicative.

6.1.2 Use other basic commands

Some other useful basic commands are briefly mentioned. All examples are shown from the MAINT virtual machine. The results are different for users with fewer privileges.

Getting help

To get help on the system, use the **HELP** command. Sometimes it is hard to find help for exactly the command that you are looking for. Some useful help commands are included in the following list:

```
==> help            // for basic help
==> help menus      // for menu of all z/VM help menus
==> help cp menu    // for a menu of all CP commands
==> help cpquery    // for a menu of all CP QUERY command
==> help cpset      // for a menu of all CP SET commands
```

Determining who is logged on

To see who is logged on to the system, use the **QUERY NAMES** command. For example:

```
==> q n
DIRMSAT2 - SSI
ZMAPVM62 - DSC , LINUX153 - DSC , LNXADMIN - DSC , LINUX157 - DSC
VSMEVSRV - DSC , VSMPROXY - DSC , VSMREQIU - DSC , VSMREQI6 - DSC
VSMREQIN - DSC , DTCSMAPI - DSC , PERSMAPI - DSC , VSMWORK3 - DSC
VSMWORK2 - DSC , VSMWORK1 - DSC , FTPSERVE - DSC , VSMGUARD - DSC
TCPIP    - DSC , DIRMAINT - DSC , DTCVSW2  - DSC , DTCVSW1  - DSC
VMSERVP  - DSC , VMSERVR  - DSC , VMSERVU  - DSC , VMSERVS  - DSC
OPERSYMP - DSC , DISKACNT - DSC , EREP     - DSC , OPERATOR - DSC
MAINT    -L0004
VSM     - TCPIP
```

Determining storage or memory

To see how much main storage (memory) is installed and allocated to a system, use the **QUERY STORAGE** command. For example:

```
==> q stor
STORAGE = 16G CONFIGURED = 16G INC = 256M STANDBY = 0  RESERVED = 0
```

This shows that there is 16 gigabytes (GB) of central memory (storage).

Determining processors or CPUs

To see how many processors (CPs, Integrated Facilities for Linux (IFLs), or CPUs) you have allocated at system level, use the **QUERY PROCESSORS** command:

```
==> q proc
PROCESSOR 00 MASTER CP
PROCESSOR 01 ALTERNATE CP
PROCESSOR 02 ALTERNATE CP
PROCESSOR 03 ALTERNATE CP
PROCESSOR 04 ALTERNATE CP
PROCESSOR 05 ALTERNATE CP
PROCESSOR 06 ALTERNATE CP
PROCESSOR 07 ALTERNATE CP
PROCESSOR 08 ALTERNATE CP
PROCESSOR 09 ALTERNATE CP
```

Determining software level

To determine what level of CP your system is at, use the **QUERY CPLEVEL** command:

```
==> q cplevel
z/VM Version 6 Release 3.0, service level 1301 (64-bit)
Generated at 06/28/13 14:58:28 EDT
IPL at 09/04/13 10:48:34 EDT
```

Determining system cylinder allocation

The **QUERY ALLOC MAP** command shows you the system allocation of spool, paging, and directory space:

```
==> q alloc map
                EXTENT     EXTENT                         % ALLOCATION
VOLID  RDEV      START        END  TOTAL IN USE   HIGH USED TYPE
------ ---- ---------- ---------- ------ ------ ------ ---- -------------
JV1030 1030          1         20     20      1      1   5% DRCT ACTIVE
JV1031 1031          1       3338 600840  87022  91029  14% SPOOL
JV1131 1131          -          -      0      0      0   0  SHARED
JP1260 1260          0      10016  1761K     27     56   1% PAGE
JP1261 1261          0      10016  1761K     75     75   1% PAGE
JV1032 1032          1       3338 600840     52     63   1% PAGE
```

Determining DASD, OSA, and virtual resources

The **QUERY DASD** and **QUERY DASD FREE** commands show you what direct access storage device (DASD) is assigned to the system, and what DASD is free to be assigned. Similarly, the **QUERY OSA** and **QUERY OSA FREE** commands will report on the Open Systems Adapter (OSA) resources. Finally, the **QUERY VIRTUAL ALL** command can be useful. The following list gives the short form of these commands without any of the associated outputs shown:

```
==> q da
==> q da free
==> q osa
==> q osa free
==> q v all
```

6.2 The z/VM Performance Toolkit

To use the z/VM Performance Toolkit, the product must be ordered. You should configure the product only if you ordered it. z/VM Performance Toolkit is part of the z/VM base installation and it is installed as disabled. It is a priced feature of z/VM. Much more detail can be found in the following books:

- *z/VM Performance Toolkit Guide,* SC24-6156*, z/VM Performance Toolkit Reference*, SC24-6157, on the web starting at the z/VM v6.3 bookshelf:

  http://publib.boulder.ibm.com/cgi-bin/bookmgr/XKS/hcsh2ac2

  Search for `Toolkit` on that page.

- *The Program Directory for Performance Toolkit for VM*, GI10-0785:

  http://www.vm.ibm.com/progdir/6vmptk30.pdf

- The *Linux on IBM eServer zSeries and S/390: Performance Toolkit for VM*, SG24-6059, on the web at the following site:

  http://www.redbooks.ibm.com/abstracts/sg246059.html

The section that follows *briefly* describes how to set up and use the IBM Performance Toolkit:

- "Configure the IBM Performance Toolkit for VM" on page 80
- "Use the IBM Performance Toolkit for VM" on page 85

6.2.1 Configure the IBM Performance Toolkit for VM

The Performance Toolkit is installed with z/VM. Configuration is described in the Program Directory for Performance Toolkit for VM, which can be found at the following website:

`http://www.ibm.com/eserver/zseries/zvm/library`

The following section provides a summary of how to turn it on. Again, you should configure the product only if you ordered it:

1. Query which priced products are enabled with the **`QUERY PRODUCT`** command:

```
==> q product
Product  State    Description
IBMVMSSI Enabled  IBM z/VM Single System Image Feature
6VMDIR30 Disabled 00/00/00.00:00:00.$BASEDDR DIRECTORY MAINTENANCE FACILITY
(DirMaint)
6VMPTK30 Disabled 00/00/00.00:00:00.$BASEDDR PERFORMANCE TOOLKIT FOR VM
6VMRAC30 Disabled 00/00/00.00:00:00.$BASEDDR RACF Security Server
6VMRSC30 Disabled 00/00/00.00:00:00.$BASEDDR RSCS Networking
```

2. To enable IBM Performance Toolkit for VM, log on to **`MAINT630`** and enter the following command:

```
==> service perftk enable
VMFSRV2760I SERVICE processing started
...
VMFSUT2760I VMFSUFTB processing started
VMFSUT2760I VMFSUFTB processing completed successfully
VMFSRV2760I SERVICE processing completed successfully
```

   You should see a few screens of messages scroll by and finally the success messages shown previously. This enables Performance Toolkit for the current z/VM session.

3. The `SYSTEM CONFIG` file is modified by having a line appended to the end. Verify that this has been added with the following commands:

```
==> vmlink pmaint cf0
DMSVML2060I PMAINT CF0 linked as 0120 file mode Z
==> type system config z
...    // many screens cleared
PRODUCT PRODID 6VMPTK30 STATE ENABLED DESCRIPTION '06/05/13.15:22:55.MAINT630
PE
RFKIT Minidisk Install and Service'
```

4. The **`QUERY PRODUCT`** command shows the change:

```
==> q product
Product  State    Description
IBMVMSSI Enabled  IBM z/VM Single System Image Feature
6VMDIR30 Disabled 00/00/00.00:00:00.$BASEDDR DIRECTORY MAINTENANCE FACILITY
(DirMaint)
6VMPTK30 Enabled  06/05/13.15:22:55.MAINT630 PERFKIT Minidisk Install and
Service
6VMRAC30 Disabled 00/00/00.00:00:00.$BASEDDR RACF Security Server
6VMRSC30 Disabled 00/00/00.00:00:00.$BASEDDR RSCS Networking
```

The Performance Toolkit is now enabled. You can also verify this by running the **QUERY PRODUCT** command again.

6.2.2 Configuring web browser support

After the product is enabled, the Transmission Control Protocol/Internet Protocol (TCP/IP) profile must be modified to enable web access to the Performance Toolkit. The following example sets the port to 80, which is the default for a web browser:

1. Log on to TCPMAINT. Edit the TCP/IP configuration file (the default name is PROFILE TCPIP) and search for the string reserve ports. This is where z/VM TCP/IP ports are reserved:

   ```
   ==> x profile tcpip d
   ====> /port
   ```

2. Add the following line under the PORT entries:

   ```
   ...
   PORT
     20   TCP FTPSERVE  NOAUTOLOG ; FTP Server
     21   TCP FTPSERVE            ; FTP Server
     23   TCP INTCLIEN            ; TELNET Server
   ; 25   TCP SMTP                ; SMTP Server
     80   TCP PERFSVM             ; Performance Toolkit
   ; 111  TCP PORTMAP             ; Portmap Server
   ; 111  UDP PORTMAP             ; Portmap Server
   ; 143  TCP IMAP                ; IMAP Server
   ...
   ```

3. Save your changes.
4. To change TCP/IP dynamically, use the **OBEYFILE** command:

   ```
   ==> netstat obey port 80 tcp perfsvm
   VM TCP/IP Netstat Level 630       TCP/IP Server Name: TCPIP

   OBEY command response is: OK
   OBEY return code = 0
   ```

5. Issue the **NETSTAT CLIENTS** command to verify your configuration. You want to see that the service named PERFSVM is a client. This should be shown after a few screens of output:

   ```
   ==> netstat clients
   ...
   Client: PERFSVM                 Authorization: {none}
   Notes Handled: none
   Last Touched:   0:03:23
   Vmcf error count: 0
   ```

If you are configuring central monitoring in a single system image (SSI) cluster, it is enough to configure the web server on only one of the members. Central monitoring enables one member to monitor the other members of the SSI cluster.

6.2.3 Configure PERFSVM

The `PERFSVM` virtual machine is the Performance Toolkit service machine. Follow these steps to configure it:

1. Log on to `PERFSVM`. If you successfully enabled the product, you enter a Performance Toolkit session and see the following text at the top of the screen:

   ```
   FCX001                Performance Toolkit for VM               Autoscroll 12
    FCXBAS500I Performance Toolkit for VM FL630
    16:14:15 Monitor event started -- recording is activated
    16:14:15 Monitor sample started -- recording is activated
   ```

2. Press **F12** twice to get to a CMS prompt.
3. Copy the default configuration files, which are on `PERFSVM`'s D disk, to your A disk:

   ```
   ==> copy * * d = = a
   ```

4. The main configuration file is `FCONX $PROFILE`. Edit that file and search for the string `VMCF`:

   ```
   ==> x fconx $profile
   ====> /vmcf
   ```

   This should take you to line 190 where the next eight lines are comments starting with an asterisk (`*`). Perform the following changes:

   a. Uncomment the 2nd, 4th, 6th, and 8th lines by changing `*C` to **`FC`**
   – Change port `81` to **`80`** on the fourth line. This enables you to use a browser interface without having to specify port 81 on the URL (with a `:81` suffix).

   The modified lines should be as follows. Save your changes with the **`FILE`** subcommand:

   ```
   *    Following command activates VMCF data retrieval interface
   FC MONCOLL VMCF ON
   *    Define the maximum allowed number of Internet connections
   FC MONCOLL WEBSERV MAXCONN 100
   *    Define the timeout of inactive Internet connections in minutes
   FC MONCOLL WEBSERV TIMEOUT 30
   *    Following command activates Internet interface
   FC MONCOLL WEBSERV ON TCPIP TCPIP 80
   *    Following command activates Internet interface with SSL
   ...
   ====> file
   ```

   If you are configuring central monitoring in an SSI cluster, enable the four FC commands only on one member, which will serve as a web server. On the other members, allow only the first FC statement (**`FC MONCOLL VMCF ON`**).

5. Create a remote data retrieval authorization file with your z/VM system identifier (replace ***`ZVM63A`*** with your system identifier):

   ```
   ==> x fconrmt authoriz
   ====> a 2
   ZVM63A PERFSVM  S&FSERV DATA
   ```

   If you are configuring central monitoring in an SSI cluster, allow the member that serves as the web server to access the other members. The authorization file on a second member would look like that shown in the following example:

   ```
   ZVM63A PERFSVM  DATA
   ZVM63B PERFSVM  S&FSERV DATA
   ```

6. Create a system identification file that links your z/VM systems and `PERFSVM` to a special resource name (replace ***ZVM63A*** with your system identifier):

   ```
   ==> x fconrmt systems
   ====> a
   ZVM63A PERFSVM  z/VM6.3 N FCXC1R01
   ```

   If you are configuring central monitoring in an SSI cluster, specify all other members as well. Make sure that each uses a unique resource name. For example, the first member might be **FCXC1R01**, the second member **FCXC1R02**, and so on.

   ```
   ZVM63A PERFSVM  z/VM6.3 N FCXC1R01
   ZVM63B PERFSVM  z/VM6.3 N FCXC1R02
   ZVM63C PERFSVM  z/VM6.3 N FCXC1R03
   ZVM63D PERFSVM  z/VM6.3 N FCXC1R04
   ```

   System identification files on all members must be the same.

7. Set up a resource override for the default resource name (enter the resource name that you used in `FCONRMT AUTHORIZ`):

   ```
   ==> x ucomdir names
   ====> a 6
   :nick.FCXRES00 :luname.*IDENT
               :tpn.FCXC1R01
               :security.SAME
   :nick.FCXSYSTM :luname.*IDENT
               :tpn.FCXC1S01
               :security.SAME
   ```

   If you are configuring central monitoring in an SSI cluster, specify resource override on each member. The second member uses **FCXC1R02** and **FCXC1S02**, the third member uses **FCXC1R03** and **FCXC1S03**, and the fourth member uses **FCXC1R04** and **FCXC1S04**.

8. Make CP start collecting performance data and start Performance Toolkit automatically after initial program load (IPL):

   a. Log on to `AUTOLOG1`.

   b. Before pressing Enter at the `VM READ` prompt, type `acc (noprof` so that the **PROFILE EXEC** is not run:

   ```
   LOGON AUTOLOG1
   z/VM Version 6 Release 3.0, Service Level 0000 (64-bit),
   built on IBM Virtualization Technology
   There is no logmsg data
   FILES:   NO RDR, 0008 PRT,   NO PUN
   LOGON AT 12:13:55 EDT THURSDAY 06/06/13
   z/VM V6.3.0    2013-06-04 12:50
   acc (noprof
   Ready; T=0.01/0.01 12:14:01
   ```

   c. Edit the profile exec in the following way:

   ```
   ==> x profile exec a
   ...
   /*******************************************************************/
   /* Customer processing can be added here                           */
   /*******************************************************************/
   "CP XAUTOLOG TCPIP"             /* Autolog TCPIP                   */
   "CP SET MDC STOR OM 256M"       /* Limit minidisk cache in CSTOR   */
   "CP SET SIGNAL SHUTDOWN 600"    /* Allow guests 10 min to shut down */
   "CP XAUTOLOG LNXADMIN"          /* Start the Linux admin machine   */
   ```

```
"CP MONITOR SAMPLE ENABLE PROCESSOR" /* Setup CP MONITOR parameters  */
"CP MONITOR SAMPLE ENABLE STORAGE"
"CP MONITOR SAMPLE ENABLE USER ALL"
"CP MONITOR SAMPLE ENABLE I/O ALL"
"CP MONITOR SAMPLE ENABLE NETWORK"
"CP MONITOR SAMPLE ENABLE APPLDATA ALL"
"CP MONITOR SAMPLE ENABLE ISFC"
"CP MONITOR SAMPLE ENABLE SSI"

"CP MONITOR EVENT  ENABLE STORAGE"
"CP MONITOR EVENT  ENABLE I/O ALL"
"CP MONITOR EVENT  ENABLE NETWORK"
"CP MONITOR EVENT  ENABLE ISFC"
"CP MONITOR EVENT  ENABLE SSI"

"CP MONITOR SAMPLE INTERVAL 1 MIN"  /* Set sampling interval */

"CP XAUTOLOG PERFSVM"           /* Start Performance Toolkit       */
```

d. Save the file using the following command:

```
====> file
```

Note: If you do not plan to perform an IPL before you try Performance Toolkit, you should run all **`CP MONITOR`** commands you just added to the **`PROFILE EXEC`** file so that CP starts to collect performance data.

e. Log off from `AUTOLOG1`.

6.2.4 Start the IBM Performance Toolkit for VM

To start the Performance Toolkit, perform the following steps:

1. Log on to the `PERFSVM` virtual machine.
2. Press Enter and the performance toolkit should start through the PROFILE EXEC:

```
FFCX001                Performance Toolkit for VM              Autoscroll 12
 FCXBAS500I Performance Toolkit for VM FL630
 12:32:15 FCXAPP530I Connected to *IDENT for resource FCXC1R01
 12:32:15 FCXAPF530I Connected to *IDENT for resource FCXC1S01
 12:32:15 FCXTCP571I Connected to TCP/IP server TCPIP on path 0003
 12:32:15 FCXAPP527I User PERFSVM connected on path 0006
 12:32:15 FCXAPC535I Connected to resource FCXC1R01 on path 0005, for S&F-Coll
 12:32:15 FCXTCP575I WebServer host IP address is 9.12.7.11:00080
 12:32:15 FCXTCP590I WebServer interface activated
 12:32:15 Monitor event started -- recording is activated
 12:32:15 Monitor sample started -- recording is activated
```

3. Disconnect from `PERFSVM` now.

```
Command ===> disc
```

The Performance Toolkit should now be configured and running.

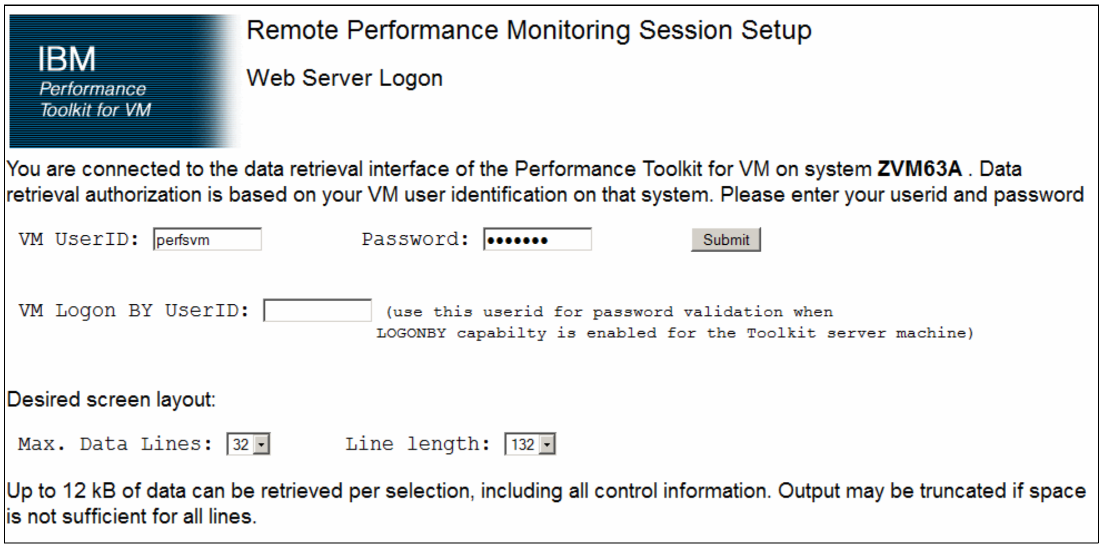
6.2.5 Use the IBM Performance Toolkit for VM

The Performance Toolkit can be used with a web browser or 3270 interface.

Using a web browser interface

To use the web-enabled Performance Toolkit, perform the following steps:

1. Point a browser to your z/VM system (for example, `http://`***`9.12.7.11`***).
2. You should see a splash screen, then the Web Server Logon window, as shown in Figure 6-1.



Remote Performance Monitoring Session Setup

Web Server Logon

You are connected to the data retrieval interface of the Performance Toolkit for VM on system **ZVM63A** . Data retrieval authorization is based on your VM user identification on that system. Please enter your userid and password

VM UserID: perfsvm Password: •••••••

VM Logon BY UserID: (use this userid for password validation when LOGONBY capabilty is enabled for the Toolkit server machine)

Desired screen layout:

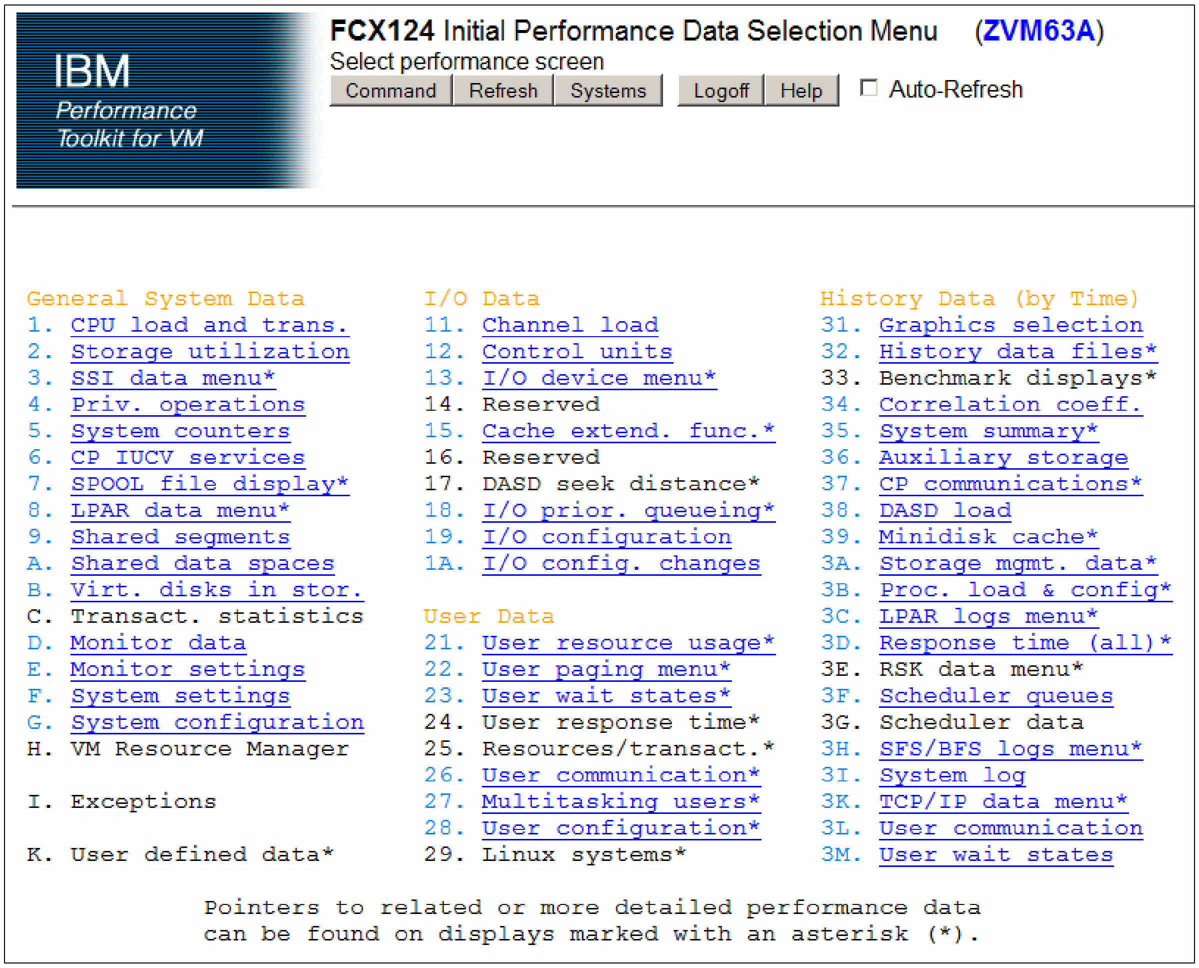
Max. Data Lines: 32 Line length: 132

Up to 12 kB of data can be retrieved per selection, including all control information. Output may be truncated if space is not sufficient for all lines.

Figure 6-1 Performance Toolkit logon window

3. Enter any valid user ID and password and click **Submit**. In this example, `PERFSVM` is used.
4. The Central Monitoring System Load Overview appears with your system identifiers (Node-ID) on the left side.

5. Click your system identifier and the Initial Performance Data Selection Menu window displays, as shown in Figure 6-2. From this window, you can drill down into many different types of reports.



FCX124 Initial Performance Data Selection Menu (**ZVM63A**)

Select performance screen

```
General System Data          I/O Data                     History Data (by Time)
1. CPU load and trans.       11. Channel load             31. Graphics selection
2. Storage utilization       12. Control units            32. History data files*
3. SSI data menu*            13. I/O device menu*         33. Benchmark displays*
4. Priv. operations          14. Reserved                 34. Correlation coeff.
5. System counters           15. Cache extend. func.*     35. System summary*
6. CP IUCV services          16. Reserved                 36. Auxiliary storage
7. SPOOL file display*       17. DASD seek distance*      37. CP communications*
8. LPAR data menu*           18. I/O prior. queueing*     38. DASD load
9. Shared segments           19. I/O configuration        39. Minidisk cache*
A. Shared data spaces        1A. I/O config. changes      3A. Storage mgmt. data*
B. Virt. disks in stor.                                   3B. Proc. load & config*
C. Transact. statistics      User Data                    3C. LPAR logs menu*
D. Monitor data              21. User resource usage*     3D. Response time (all)*
E. Monitor settings          22. User paging menu*        3E. RSK data menu*
F. System settings           23. User wait states*        3F. Scheduler queues
G. System configuration      24. User response time*      3G. Scheduler data
H. VM Resource Manager       25. Resources/transact.*     3H. SFS/BFS logs menu*
                             26. User communication*      3I. System log
I. Exceptions                27. Multitasking users*      3K. TCP/IP data menu*
                             28. User configuration*      3L. User communication
K. User defined data*        29. Linux systems*           3M. User wait states

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

Figure 6-2 Browser interface to the Performance Toolkit

Use a 3270 interface

You can also use a 3270 interface. To do so, perform the following steps:

1. Log on to PERFSVM.
2. If you had disconnected, pressing Enter should get you back to the Performance Toolkit command line. If the virtual machine was logged off, the **PROFILE EXEC** should run and get you to the command line. Enter the **MONITOR** command:

```
Command ==> monitor
```

The Performance Screen Selection panel then appears, as shown in Figure 6-3.

```
FCX124         Performance Screen Selection  (FL630        )    Perf. Monitor

General System Data       I/O Data                  History Data (by Time)
1. CPU load and trans.    11. Channel load          31. Graphics selection
2. Storage utilization    12. Control units         32. History data files*
3. SSI data menu*         13. I/O device menu*      33. Benchmark displays*
4. Priv. operations       14. Reserved              34. Correlation coeff.
5. System counters        15. Cache extend. func.*  35. System summary*
6. CP IUCV services       16. Reserved              36. Auxiliary storage
7. SPOOL file display*    17. DASD seek distance*   37. CP communications*
8. LPAR data menu*        18. I/O prior. queueing*  38. DASD load
9. Shared segments        19. I/O configuration     39. Minidisk cache*
A. Shared data spaces     1A. I/O config. changes   3A. Storage mgmt. data*
B. Virt. disks in stor.                             3B. Proc. load & config*
C. Transact. statistics   User Data                 3C. LPAR logs menu*
D. Monitor data           21. User resource usage*  3D. Response time (all)*
E. Monitor settings       22. User paging menu*     3E. RSK data menu*
F. System settings        23. User wait states*     3F. Scheduler queues
G. System configuration   24. User response time*   3G. Scheduler data
H. VM Resource Manager    25. Resources/transact.*  3H. SFS/BFS logs menu*
                          26. User communication*   3I. System log
I. Exceptions             27. Multitasking users*   3K. TCP/IP data menu*
                          28. User configuration*   3L. User communication
K. User defined data*     29. Linux systems*        3M. User wait states

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

Figure 6-3 Performance Screen Selection panel

Drilling down into report screens

You should now be able to use the active report screens. To drill down into these screens, move the cursor to any of the titles that are active (active titles display the number or letter in white, inactive titles are in green). The following reports are some of the more useful report screens to drill down into:

```
21. User resource usage
22. User paging load
23. User wait states
28. User configuration
29. Linux systems
33. Benchmark displays
```

6.3 Collect and use raw CP monitor data

Although the Performance Toolkit formats and displays current performance data, it is often needed to look at older data as well. Typical use would be to compare the current system performance to the past performance to have data available for troubleshooting, or to generate reports.

6.3.1 Collect CP monitor data

CP monitor records are collected by the **MONWRITE** utility and written to a disk or tape. The resulting file contains all of the original unprocessed data. This data can be used later to generate reports or the Performance Toolkit can use it in Monitor Data Scan Mode to look at historical data as if it were current:

1. Log on to the **MONWRITE** virtual machine.
2. Edit the **PROFILE EXEC**:

```
LOGON MONWRITE
z/VM Version 6 Release 3.0, Service Level 0000 (64-bit),
built on IBM Virtualization Technology
There is no logmsg data
FILES:   NO RDR,   NO PRT,   NO PUN
LOGON AT 10:40:31 EDT FRIDAY 06/07/13
z/VM V6.3.0    2013-06-04 12:50

Ready; T=0.01/0.01 10:40:34
==> x profile exec a
input
/* ALL MONITOR COMMANDS ARE LOCATED IN AUTOLOG1'S PROFILE EXEC */
'MONWRITE MONDCSS *MONITOR DISK CLOSE 480'
==> file
```

3. Execute the REXX exec named **profile**

```
==> profile
HCPMOW6272I Now recording in file D060713 T110146 A1
HCPMOW6265A MONITOR WRITER CONNECTED TO *MONITOR
```

4. Disconnect from MONWRITE

```
==> #cp disc
```

The **CLOSE 480** statement tells **MONWRITE** to close the output file every 8 hours (480 minutes), starting from midnight. It means, regardless of when it starts recording, it will close the file at 08:00, 16:00, and 24:00. The file name will clearly show the date and time when recording started.

To collect **MONWRITE** data automatically, start the **MONWRITE** virtual machine when z/VM is IPLed. To do so, add a line to the **PROFILE EXEC** of the `AUTOLOG1 191` disk (or `AUTOLOG2 191` if an external security manager, such as Resource Access Control Facility (RACF), is running).

```
==> x profile exec
...
"CP XAUTOLOG MONWRITE"             /* Start the MONWRITE VM       */
...
```

MONWRITE's A-disk is delivered as 300 cylinders, which is quite small. Depending on the monitor interval activity of the system and the number of samples and events, it can fill up quickly. When the disk is full, **MONWRITE** cannot write anymore. It is important to monitor the space on **MONWRITE**'s A-disk. Another possibility is to use some utility that would archive old files and clean up the space automatically. An example of such a utility would be **MONCLEAN**. You can download it from the following site:

`http://www.vm.ibm.com/download/packages/descript.cgi?MONCLEAN`

Follow these steps for the **MONCLEAN** installation:

1. Use File Transfer Protocol (FTP) binary to transfer `MONCLEAN VMARC` to `MONWRITE`'s 191 disk.
2. Run **MONWRITE VMARC** using the **pipe** command:

```
==> pipe < monclean vmarc a | fblock 80 00 | > monclean vmarc A F 80
```

3. Unpack the `MONCLEAN VMARC` file with the **VMARC** command:

```
==> vmarc unpk monclean vmarc a
MONCLEAN EXEC     A1. Bytes in=      4080, bytes out=      7678 (   188%).
MONCLEAN README   A1. Bytes in=      1040, bytes out=      2240 (   215%).
```

4. Check documentation in `MONCLEAN README`.
5. Modify **PROFILE EXEC**:

```
==> x profile exec
/* ALL MONITOR COMMANDS ARE LOCATED IN AUTOLOG1'S PROFILE EXEC */
'MONWRITE MONDCSS *MONITOR DISK CLOSE 60 EXEC MONCLEAN'
```

6. Start recording:

```
==> profile
HCPMOW6272I Now recording in file D061213 T131724 A1
HCPMOW6265A MONITOR WRITER CONNECTED TO *MONITOR
```

7. `MONWRITE` closes the output file every hour and runs `MONCLEAN EXEC`. If the `MONCLEAN EXEC` was not modified, it removes the oldest file when the disk reaches 80% full.
8. Example 6-1 shows **MONWRITE**'s 191 disk when **MONCLEAN** is running.

Example 6-1 MONWRITE 191 disk

```
MAINT    FILELIST A0  V 169  Trunc=169 Size=19 Line=1 Col=1 Alt=0
Cmd   Filename Filetype Fm Format Lrecl    Records     Blocks   Date     Time
      D061313  T100016  Z1 F       4096      49275      49275  6/13/13 10:29:16
      D061313  T090016  Z1 F       4096      99407      99407  6/13/13 10:00:15
      D061313  T080015  Z1 F       4096      99392      99392  6/13/13  9:00:15
      D061313  T070015  Z1 F       4096      99348      99348  6/13/13  8:00:15
      D061313  T060015  Z1 F       4096      99348      99348  6/13/13  7:00:15
      D061313  T050016  Z1 F       4096      99348      99348  6/13/13  6:00:15
      D061313  T040016  Z1 F       4096      99348      99348  6/13/13  5:00:15
      D061313  T030015  Z1 F       4096      99348      99348  6/13/13  4:00:15
      D061313  T020016  Z1 F       4096      99348      99348  6/13/13  3:00:15
      D061313  T010015  Z1 F       4096      99348      99348  6/13/13  2:00:15
      D061313  T000015  Z1 F       4096      99348      99348  6/13/13  1:00:15
      D061213  T230015  Z1 F       4096      99348      99348  6/13/13  0:00:15
      D061213  T220015  Z1 F       4096      99356      99356  6/12/13 23:00:15
      D061213  T210015  Z1 F       4096      99357      99357  6/12/13 22:00:15
      D061213  T200015  Z1 F       4096      99348      99348  6/12/13 21:00:15
      PROFILE  EXEC     Z1 V         65          2          1  6/12/13 11:35:49
      MONCLEAN EXEC     Z1 V         75        194          2  6/12/13 11:32:13
      MONCLEAN README   Z1 F         80         28          1  6/12/13 11:32:13
      MONCLEAN VMARC    Z1 F         80         64          2  6/12/13 11:32:13
```

6.3.2 Use CP monitor data

The Performance Toolkit subcommand **MONSCAN** enables you to select a CP monitor file on disk or tape (created by the standard **MONWRITE** utility) as input for performance data analysis. When the specified file is found, a performance data scan mode is entered which looks almost identical to the normal real-time monitoring mode, and which enables you to browse through the accumulated monitor data.

As the **PERFSVM** virtual machine is used to show the current performance data, it is better to use a different virtual machine to perform **MONSCAN**. The following example uses the MAINT user ID:

1. Link and access PERFSVM's 201 minidisk:

```
==> vmlink perfsvm 201
DMSVML2060I PERFSVM 201 linked as 0120 file mode Z
```

2. Link and access MONWRITE's 191 minidisk:

```
==> vmlink monwrite 191
DMSVML2060I MONWRITE 191 linked as 0121 file mode X
```

3. Check files available from MONWRITE:

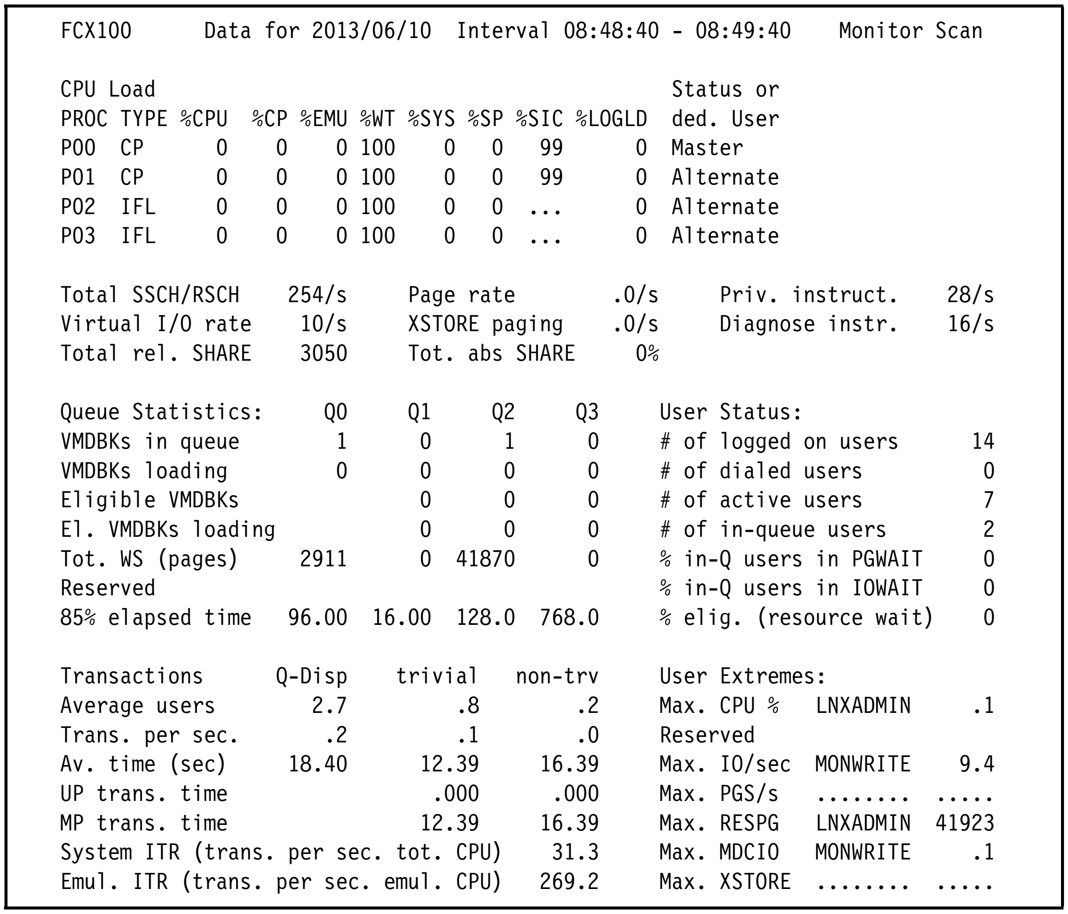
```
==> filel * * x
MAINT    FILELIST A0  V 169  Trunc=169 Size=4 Line=1 Col=1 Alt=0
Cmd   Filename Filetype Fm Format Lrecl    Records     Blocks   Date     Time
      D061013  T084824  X1 F       4096      53930      53930  6/10/13  9:20:43
      PROFILE  EXEC     X1 V         65          3          1  6/10/13  8:48:21
```

4. Run the **MONSCAN** subcommand:

```
==> perfkit monscan D061013 T084824 X
```

The Regular Performance Screen Selection appears, as shown previously in Figure 6-3 on page 87.

5. Make a selection, for example `1 - CPU Load`. The first panel does not contain any data. Enter the **`nexts`** (next sample) command and a panel with real numbers displays, as shown in Figure 6-4. You can see the interval on the top of the panel.



```
FCX100      Data for 2013/06/10  Interval 08:48:40 - 08:49:40    Monitor Scan

CPU Load                                          Status or
PROC TYPE %CPU  %CP %EMU %WT %SYS %SP %SIC %LOGLD  ded. User
P00  CP      0    0    0 100    0   0   99      0  Master
P01  CP      0    0    0 100    0   0   99      0  Alternate
P02  IFL     0    0    0 100    0   0  ...      0  Alternate
P03  IFL     0    0    0 100    0   0  ...      0  Alternate

Total SSCH/RSCH    254/s     Page rate        .0/s     Priv. instruct.    28/s
Virtual I/O rate    10/s     XSTORE paging    .0/s     Diagnose instr.    16/s
Total rel. SHARE    3050     Tot. abs SHARE     0%

Queue Statistics:     Q0     Q1     Q2     Q3     User Status:
VMDBKs in queue        1      0      1      0     # of logged on users      14
VMDBKs loading         0      0      0      0     # of dialed users          0
Eligible VMDBKs               0      0      0     # of active users          7
El. VMDBKs loading            0      0      0     # of in-queue users        2
Tot. WS (pages)     2911      0  41870      0     % in-Q users in PGWAIT     0
Reserved                                          % in-Q users in IOWAIT     0
85% elapsed time   96.00  16.00  128.0  768.0     % elig. (resource wait)    0

Transactions      Q-Disp    trivial   non-trv     User Extremes:
Average users        2.7         .8        .2     Max. CPU %   LNXADMIN     .1
Trans. per sec.       .2         .1        .0     Reserved
Av. time (sec)     18.40      12.39     16.39     Max. IO/sec  MONWRITE    9.4
UP trans. time                 .000      .000     Max. PGS/s   ........  .....
MP trans. time                12.39     16.39     Max. RESPG   LNXADMIN  41923
System ITR (trans. per sec. tot. CPU)    31.3     Max. MDCIO   MONWRITE     .1
Emul. ITR (trans. per sec. emul. CPU)   269.2     Max. XSTORE  ........  .....
```

Figure 6-4 Monitor Scan panel

6.4 Monitoring Linux performance for troubleshooting

Previous sections described how the Performance Toolkit can show resource consumption of the Linux guest, as measured and dispatched by the z/VM hypervisor. z/VM is not aware of the nature of the guest, and it cannot understand what is happening inside the guest. For that reason, it is important to be able to measure performance data from within the Linux guest itself. To monitor Linux performance data at this level, a data gatherer process must be running within each Linux guest that you want to monitor.

There are different ways of gathering this data. There are many commercial and non-commercial solutions for long-term monitoring as well. This book cannot cover all of the requirements for long-term monitoring (low CPU consumption, data storage, and similar). This chapter shows how to monitor Linux performance in short periods, especially when troubleshooting performance problems.

6.4.1 Monitoring Linux performance from z/VM

This section describes how to gather Linux performance data in Linux and provide this data to z/VM for a consolidated overview.

To monitor Linux performance data directly from the kernel, the following statements must be true:

- The `APPLMON` option must be set in the user directory.
- Applmon data monitoring must be built into the kernel.

The first requirement should be true, because the `OPTION APPLMON` was set for the Linux virtual machines. For the second requirement, from SUSE Linux Enterprise Server 11 SP3 onwards has had this feature built in.

The following steps provide a quick description of how to use this built-in monitoring function:

1. Start an Secure Shell (SSH) session to a Linux system. In this example, `LINUX3` is used.
2. There are three modules that are built into the kernel but are not loaded by default. They are named `appldata_mem`, `appldata_os`, and `appldata_net_sum`. You can verify that they are not loaded with the **`lsmod`** and **`grep`** commands:

   ```
   # lsmod | grep appldata
   ```

3. There is no output, so no modules with the string `appldata` are loaded. Load those modules with the **`modprobe`** command and verify that they have been loaded:

   ```
   # modprobe appldata_mem
   # modprobe appldata_os
   # modprobe appldata_net_sum
   ```

4. Now, if you repeat the **`lsmod`** command, you should see the following result:

   ```
   # lsmod | grep appldata
   appldata_net_sum        1966  0
   appldata_os             2989  0
   appldata_mem            2008  0
   ```

5. The directory in the virtual `/proc/` file system where the monitoring variables exist is `/proc/sys/appldata/`. In this directory, there are five files:

`timer`	Controls whether any data gathering is in effect.
`interval`	Sets the interval, in milliseconds, that samples are taken.
`mem`	Controls the memory data gathering module.
`os`	Controls the CPU data gathering module.
`net_sum`	Controls the net data gathering module.

6. To turn on the built-in kernel monitoring, use the **`echo`** command to send a nonzero value into four of the five monitoring variables in the `/proc/` virtual file system:

   ```
   # echo 1 > /proc/sys/appldata/timer
   # echo 1 > /proc/sys/appldata/mem
   # echo 1 > /proc/sys/appldata/os
   # echo 1 > /proc/sys/appldata/net_sum
   ```

Built-in kernel monitoring should now be turned on. You might want to leave the monitoring on for only specific periods of time. As Linux monitoring data is captured, the Performance Toolkit's minidisk space can fill up relatively quickly.

View performance data from the Linux kernel in the Performance Toolkit

After the system has had some time to collect data, you should be able to use the Performance Toolkit to view Linux performance data. To view that data, complete the following steps:

1. Drill down into menu 29, `Linux systems`. This can be done either from the browser interface or the 3270 interface, as shown in Example 6-2.

Example 6-2 Linux Displays

```
FCX242      CPU 2817  SER 23BD5          Linux Displays         Perf. Monitor

   Linux screens selection
 S Display     Description
 . LINUX       RMF PM system selection menu
 . LXCPU       Summary CPU activity display
 S LXMEM       Summary memory util. & activity display
 . LXNETWRK    Summary network activity display
```

2. Then, type `S` over the period on the left side of the submenu panel in the row corresponding to the report that you want to see. You should see a new report panel with the Linux guest systems memory overview, as shown in Example 6-3.

Example 6-3 Linux guests memory overview

```
FCX244      CPU 2817  SER 23BD5       Initial  14:22:57         Perf. Monitor
______            .      .       .      .      .       .      .       .      .
          <------------ Memory Allocation (MB) -------------> <------- Swapping
Linux     <--- Main ---> <--- High --->        Buffers  Cache <-Space (MB)-> <-
Userid    M_Total %MUsed H_Total %HUsed Shared /CaFree   Used S_Total %SUsed
>System<    491.6   25.8      .0     .0     .0     8.6   46.3   761.6     .0
LINUX3      491.6   25.8      .0     .0     .0     8.6   46.3   761.6     .0
```

3. You can also use a web interface to view the same data.

6.4.2 Monitoring Linux performance from inside Linux

There are many tools for Linux performance monitoring. This section describes just some of the most commonly used. They are all platform independent, and they work on Linux in general.

The top command

When running the **top** command without any parameters, it shows a system overview and running tasks similar to those shown in Example 6-4. Output is refreshed every 3 seconds automatically. To leave **top**, press "q".

Example 6-4 The top command

```
top - 17:26:52 up  7:30,  1 user,  load average: 0.06, 0.08, 0.05
Tasks:  99 total,   1 running,  98 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni, 99.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  1015640 total,   666560 free,   104228 used,   244852 buff/cache
KiB Swap:  1828452 total,  1828452 free,        0 used.   863024 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
    1 root      20   0    9564   6800   4016 S   0.0  0.7   0:00.81 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.00 kthreadd
    3 root      20   0       0      0      0 S   0.0  0.0   0:00.01 ksoftirqd/0
    5 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:0H
    7 root      rt   0       0      0      0 S   0.0  0.0   0:00.00 migration/0
...
```

The vmstat command

Another useful command is **vmstat**, which reports information about processes, memory, paging, block I/O, traps, and CPU activity. When running **vmstat** without any parameters, it shows just one line summarizing averages since the last IPL, which is not very useful. Example 6-5 shows **vmstat 5** output. This shows the first line with averages since the last IPL, and then writes a new line every 5 seconds with the current data.

Example 6-5 The vmstat command

```
# vmstat 5
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 0  0      0 666832   4596 239924    0    0     4     1   79  126  0  0 100  0  0
 0  0      0 666768   4596 239956    0    0     0     0  149  248  0  0 100  0  0
 0  0      0 666752   4596 239956    0    0     0     0  149  246  0  0 100  0  0
 0  0      0 666720   4596 239956    0    0     0     0  171  266  0  0 100  0  0
 0  0      0 666752   4596 239956    0    0     0     0  169  258  0  0 100  0  0
 0  0      0 666752   4596 239956    0    0     0     0  166  254  0  0 100  0  0
 0  0      0 666752   4596 239956    0    0     0     0  160  247  0  0 100  0  0
 0  0      0 666720   4596 239956    0    0     0     0  152  250  0  0 100  0  0
```

The `wa` column shows a wait time, and represents a percentage of time while the system waited for I/O. The higher the percentage, the more time tasks waste nonproductively.

The `st` column shows what is known as *stolen time*. It represents the time that CPU was stolen from a guest by the hypervisor. This can mean several different things: CPU contention at the z/VM level, heavy z/VM paging, heavy virtual switch usage, and so on. The higher the number, the more time a guest can spend nonproductively.

The sysstat package

The following tools are part of the `sysstat` package, which might not be installed automatically. To install the `sysstat` package, perform the following command from your Linux guest, as shown in Example 6-6.

Example 6-6 Installing the sysstat package

```
# zypper install sysstat
Loading repository data...
Reading installed packages...
Resolving package dependencies...

The following NEW package is going to be installed:
  sysstat

1 new package to install.
Overall download size: 174.0 KiB. After the operation, additional 662.0 KiB will
be used.
Continue? [y/n/?] (y): y
Retrieving package sysstat-8.1.5-7.45.24.s390x (1/1), 174.0 KiB (662.0 KiB
unpacked)
Installing: sysstat-8.1.5-7.45.24 [done]
# rpm -qa|grep sysstat
sysstat-8.1.5-7.45.24
```

The iostat command

Besides reporting overall system performance, **iostat** provides detailed I/O statistics for devices. Example 6-7 shows an output of the **iostat -x 5 2** command. It displays two samples with a 5-second interval.

Example 6-7 The iostat command

```
Linux 3.10.0-229.el7.s390x (linux1.itso.ibm.com)       04/29/2015      _s390x_ (2 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.01    0.00    0.04    0.01    0.00   99.93

Device:         rrqm/s   wrqm/s     r/s     w/s    rkB/s    wkB/s avgrq-sz avgqu-sz   await r_await w_await  svctm  %util
dasda             0.07     0.02    0.41    0.15     9.76     1.51    40.02     0.00    5.99    4.27   10.65   0.31   0.02
dm-0              0.00     0.00    0.02    0.00     0.06     0.00     8.00     0.00    1.10    1.10    0.00   0.50   0.00
dm-1              0.00     0.00    0.42    0.17     9.56     1.41    37.13     0.00    6.07    4.18   10.92   0.28   0.02
dasdb             0.04     0.00    0.02    0.00     0.08     0.00     9.25     0.00    0.06    0.06    0.00   0.06   0.00
dasdc             0.04     0.00    0.02    0.00     0.08     0.00     9.29     0.00    0.06    0.06    0.00   0.06   0.00
sdb               0.00     0.00    0.02    0.00     0.07     0.00     8.02     0.00    0.23    0.24    0.00   0.23   0.00
sda               0.00     0.00    0.02    0.00     0.08     0.00     8.26     0.00    0.32    0.30    2.50   0.29   0.00
sdc               0.00     0.00    0.02    0.00     0.07     0.00     8.18     0.00    0.20    0.20    0.00   0.20   0.00
sdd               0.00     0.00    0.02    0.00     0.07     0.00     7.98     0.00    0.23    0.23    0.00   0.20   0.00
dm-2              0.00     0.00    0.03    0.00     0.14     0.00     8.30     0.00    0.23    0.23    0.00   0.23   0.00
dm-3              0.00     0.00    0.00    0.00     0.02     0.00     8.00     0.00    0.49    0.49    0.00   0.49   0.00

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.00    0.00    0.00    0.00    0.00  100.00

Device:         rrqm/s   wrqm/s     r/s     w/s    rkB/s    wkB/s avgrq-sz avgqu-sz   await r_await w_await  svctm  %util
dasda             0.00     0.00    0.00    0.20     0.00     0.30     3.00     0.00    0.00    0.00    0.00   0.00   0.00
dm-0              0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00    0.00    0.00   0.00   0.00
dm-1              0.00     0.00    0.00    0.20     0.00     0.30     3.00     0.00    0.00    0.00    0.00   0.00   0.00
dasdb             0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00    0.00    0.00   0.00   0.00
dasdc             0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00    0.00    0.00   0.00   0.00
sdb               0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00    0.00    0.00   0.00   0.00
sda               0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00    0.00    0.00   0.00   0.00
sdc               0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00    0.00    0.00   0.00   0.00
sdd               0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00    0.00    0.00   0.00   0.00
dm-2              0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00    0.00    0.00   0.00   0.00
dm-3              0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00    0.00    0.00   0.00   0.00
```

The avgqu-sz column shows the average queue size for a given device. The larger the number, the more contention exists for a device.

The await column displays average wait time for a device. It includes the time that requests spent in queue, and the time spent servicing requests in a storage device. The higher the number, the more time is wasted by a program waiting for I/O.

The sar and sadc commands

Ad hoc data can be gathered by calling the data collector with the **sadc** command:

```
# /usr/lib64/sa/sadc -S ALL -F 5 60 /tmp/sadc.out
```

Data is collected for 5 minutes in 5-second intervals (5 x 60 = 300, s = 5 minutes).

Output that is produced by **sadc** is a binary file. To process it and generate text output, the **sar** command is used:

```
# sar -A -f /tmp/sadc.output > outfile.txt
```

The **sar** command generates much detailed performance information.

If **sar** is configured as a service and it gathers data automatically, its data is stored in the /var/log/sa directory. Data files are of the form sa<*dd*>, where <*dd*> is the day of the month. Text files are of the form sar<*dd*>, as shown in the following example:

```
# cd /var/log/sa
# file *
sa09:  data
sa10:  data
sa11:  data
sa12:  data
sa13:  data
sar09: ASCII text
sar10: ASCII text
sar11: ASCII text
sar12: ASCII text
```

7

Configuring Linux for cloning

Cloning is a process, where new systems are created from an existing master system, also called a *golden image*. This chapter has greatly simplified the process of cloning when compared to the previous release of this book. The cloning procedure benefits from several improvements to the z/VM environment therefore making the clone process simple enough to fit into just a handful of steps.

Linux operating systems over time tend to have more and more unique identifiers, such as, with the introduction of *systemd*, a new machine ID has been added. All of these identifiers must be re-created on the cloned system. However, the process to know all these identifiers and to re-create them requires in-depth knowledge of the *golden image*. Failure to update all of these identifiers could cause unforeseen trouble later, including the possibilities of data corruption or security issues.

If you are unsure of all of the unique identifiers for your *golden image*, and you prefer not to follow the cloning process, refer to the automated installation procedures for KIWI imaging instead. Find information about these in the following chapters:

- Chapter 2, “Automating SUSE Linux Enterprise Server 12 installation using the AutoYaST tool” on page 23
- Chapter 3, “Creating appliances with KIWI” on page 37

This chapter describes an example of the cloning process. A basic cleanup procedure has been supplied on a best-effort basis. The cleanup script should be reviewed by the user, and it requires updates whenever a more specialized golden image is prepared.

After the cleanup, the configuration update of the target system has been prepared as an example in this chapter. Depending on the services used, this also needs updates from the administrator, and should be reviewed in the first place.

This chapter provides information about the following topics:

- 7.1, “Creating golden image for cloning” on page 98
- 7.2, “Cloning the golden image using dirmaint” on page 98
- 7.3, “Sending configuration update to the clone system” on page 99
- 7.4, “Starting the clone system” on page 100

7.1 Creating golden image for cloning

A golden image is a special type of Linux installation that is used as a baseline to generate new Linux systems. Special care must be taken to prepare the golden image. All unique identifiers must be updated for the resulting new Linux system, known as the *cloned system*.

Before starting, have the files associated with this book available. You can find these files in the Appendixes, specifically in "Important IBM z/VM files" on page 142 and "Locating the web material" on page 151.

The user name of the z/VM guest in this book is assumed to be `GOLD`. Be sure to make all needed changes that you want to have for your template system before proceeding.

To prepare the Linux system as the golden image, follow these steps:

1. Copy the **`prepare_system.sh`** script to the system. It removes files specific to the one installation:

   ```
   # scp prepare_system.sh golden.itso.ibm.com:/usr/local/bin
   ```

2. Log on to the golden image, and make sure that the scripts below `/usr/local/bin` are executable:

   ```
   # ssh golden.itso.ibm.com
   # chmod 755 /usr/local/bin/*
   ```

3. Shut down the image by calling the script **`prepare_system.sh`**:

   ```
   # /usr/local/bin/prepare_system.sh

   Warning: This utility will prepare the root disk for cloning by removing some
            configuration files. After it has finished it will halt this systemd.

   press Ctrl+C within the next 15 seconds to abort

   ln -s '/etc/systemd/system/clone.service'
   '/etc/systemd/system/network.target.wants/clone.service'
   Removing SSH keys
   Removing the network configuration
   Removing this script (/usr/local/bin/prepare_system.sh)
   halt the system
   ```

This is the time to do a backup of the image.

7.2 Cloning the golden image using dirmaint

To perform the copy of the golden image to a new user, dirmaint is used. To clone the golden image, to a user with user name `LINUX6`, use the following procedure.

Make sure that the needed resources, such as disks, are already prepared. The needed tasks are described in *The Virtualization Cookbook for IBM z Systems Volume 1: IBM z/VM 6.3*, SG24-8147 in Chapter 6, "Planning and preparation for Linux workloads":

1. Log off the source system (golden).
2. Log on as `MAINT`.

3. If the z/VM guest hasn't been created yet, use **dirmaint** to do so:

   ```
   ==> dirmaint add LINUX6 like LNXPROTO pw <NewPassword>
   ```

4. Ensure that the source and target systems are logged off:

   ```
   ==> query linux5 #query linux6
   query linux5
   query linux6
   HCPCQU045E LINUX5 not logged on
   Ready(00045); T=0.01/0.01 10:11:14
   HCPCQU045E LINUX6 not logged on
   Ready(00045); T=0.01/0.01 10:11:14
   ```

5. Copy the disk 0100 from the source to the target system:

   ```
   ===> dirmaint for linux6 clonedisk 0100 linux5 0100
   dirmaint for linux6 clonedisk 0100 linux5 0100
   DVHXMT1191I Your CLONEDISK request has been sent for processing to
   DVHXMT1191I DIRMAINT at ITSOZVM1 via DIRMSAT2.
   Ready; T=0.01/0.01 08:41:24
    DVHREQ2288I Your CLONEDISK request for LINUX6 at
    DVHREQ2288I * has been accepted.
   ...
    DVHBIU3428I Changes made to directory entry
    DVHBIU3428I DATAMOV2 have been placed online.
    DVHSHN3430I CLONEDISK operation for LINUX6
    DVHSHN3430I address 0100 has finished (WUCF
    DVHSHN3430I 01085901).
   ```

6. Log off MAINT.

7.3 Sending configuration update to the clone system

Before being operational, the system needs some information about its configuration. This is achieved by sending a configuration script to the READER of the target system, LINUX6.

For SUSE Linux Enterprise Server, it is called **send_config_sles.sh**. To simplify the configuration, it is sufficient to create a small configuration file. This looks similar to Example 7-1.

Example 7-1 Configuration file to update network configuration of LINUX6

```
GENERATE_MACHINEID="1"
Hostname="LINUX6"
HostIP="9.12.7.103/20"
Gateway="9.12.4.1"
ReadChannel=0.0.0600
WriteChannel=0.0.0601
DataChannel=0.0.0602
Domain=itso.ibm.com
Nameserver=9.12.6.6
Layer2=1
```

1. Log in as root on LNXADMIN
2. Copy the **send_config_sles.sh** shell script to /usr/local/bin:

   ```
   # cp send_config_sles.sh /usr/local/bin
   ```

3. Make sure that the script is executable:

```
# chmod +x /usr/local/bin/send_config*
```

4. Create a configuration subdirectory to `/root`:

```
# cd /root
# mkdir /root/cnf
```

5. Create and adapt the additional configuration script as shown in Example 7-1 on page 99. The file name must match the `USERID` of the cloned system.

```
# vi /root/cnf/LINUX6
```

6. Run the **send_config** script with the target `USERID` as argument:

```
SLES: # send_config_sles.sh LINUX6
```

This results in a configuration update script being sent to the reader of `LINUX6`, but with the following limitations:

- The file name sent to the reader is `CLONE`.
- The file mode sent to the reader is `SYSCTL`.
- The file must be sent by `LNXADMIN`.
- The lines of the script must not exceed 80 characters.

The script sent to `LINUX6` is then run during the first bootup.

7.4 Starting the clone system

To activate the clone system, just start it. This can be done logged on or, if the system has been set up with a common **PROFILE EXEC** that starts the system disk, as a remote user with `XAUTOLOG`:

```
==> ipl 100
```

During the IPL, the systemd **clone.service** reads and runs the **clone.sysclt** from the reader.

The **clone.sysctl** script contains the information needed to finalize the target system.

Finally, restart the new system to make sure that the new `machine-id` is consistently used with the cloned system.

8

Working with systemd

"You can be discouraged by failure, or you can learn from it. So go ahead and make mistakes, make all you can. Because, remember that's where you'll find success - on the far side of failure."

— Thomas J. Watson Sr.

On modern Linux installations systemd plays a major role. Understanding the concepts and knowing the utilities provided by systemd is key to any Linux administrator. This chapter describes how to work with a SUSE Linux Enterprise Server 12 system that uses **`systemd`**. It includes the following topics:

8.1 Getting started with systemd

As a system and services manager, systemd replaced the SysV-style init system in SUSE Linux Enterprise Server. It is the first user space process that the kernel starts when booting. This process is responsible for starting all of the services and their dependencies that allow the system to act as a server. systemd uses units that can be dependent on other units. There are different unit types:

Service units	Used to start services.
Socket units	Allow socket-based activations.
Device units	Trigger reactions for devices as they appear or disappear.
Mount point units	Control mount points.
Target units	Allow grouping of units to act as synchronization points.

For example, before the **`local-fs.target`** is reached, the local mount units need to be activated. These local units are dependent on the device units. By default, the system boots into the `default.target` that pulls in all of the dependencies necessary to start the system. More details about the boot process can be found in 8.4, "The system boot process" on page 109. The units are described in text in the unit files at `/etc/systemd/system` and `/usr/lib/systemd/system` where systemd searches and loads them.

Some unit files are generated dynamically by systemd unit generators. For example, the `systemd-fstab-generator` parses the `/etc/fstab` and creates mount units for the entries if necessary. RPM Package Manager (RPM) packages that provide services usually install their service unit files at `/usr/lib/systemd/system`. An example of the `/usr/lib/systemd/system/sshd.service` unit file is shown in Example 8-1.

Example 8-1 The sshd.service unit file

```
[Unit]
Description=OpenSSH server daemon
After=network.target sshd-keygen.service
Wants=sshd-keygen.service

[Service]
EnvironmentFile=/etc/sysconfig/sshd
ExecStart=/usr/sbin/sshd -D $OPTIONS
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure
RestartSec=42s

[Install]
WantedBy=multi-user.target
```

8.2 Using systemd units

This section describes how to manage services and isolate systemd targets.

8.2.1 Managing services

Because systemd starts all of the services, one of the first things you want to know is how to get a list of the available services. Use the **systemctl** utility to control services and inspect their state.

Listing active service units (running services)

Use the following command to list active service units:

```
# systemctl -t service
UNIT                    LOAD   ACTIVE SUB     DESCRIPTION
after-local.service     loaded active exited  /etc/init.d/after.local Compatibility
cpi.service             loaded active exited  LSB: Set Control Program
cron.service            loaded active running Command Scheduler
dbus.service            loaded active running D-Bus System Message Bus
...
```

Note: The **systemctl** command attempts to fit the contents of its output into your terminal window. If the output has more lines than your terminal window can display, it pipes the output into `less`, the default pager. You can change that behavior by adding the `--no-pager` option. In case the width of your terminal window is too narrow to display the full contents you can add the `--full` or `-l` option to show the full contents. Of course, the `--no-pager` and the `--full` options can be combined if wanted.

Listing failed service units

Use the following command to list failed service units:

```
# systemctl -t service --state=failed
```

Querying the status of a service unit

Use the following command to query the status of a service unit:

```
# systemctl status sshd.service
sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled)
   Active: active (running) since Thu 2015-04-23 16:40:03 EDT; 24h ago
 Main PID: 1454 (sshd)
   CGroup: /system.slice/sshd.service
           ··1454 /usr/sbin/sshd -D

Apr 23 16:40:03 linux2.itso.ibm.com systemd[1]: Started OpenSSH server daemon.
Apr 23 16:40:03 linux2.itso.ibm.com sshd[1454]: Server listening on 0.0.0.0 ...
Apr 23 16:40:03 linux2.itso.ibm.com sshd[1454]: Server listening on :: port 22.
Apr 23 16:58:13 linux2.itso.ibm.com sshd[2906]: Accepted password for ken f...2
...
```

In addition to the information that you would expect there, the output also shows the last ten log messages from the journal. If you just want to know if a service unit is active or not, you can type the following command:

```
# systemctl is-active sshd.service
active
```

Stopping, starting, and restarting a service unit

Use the following commands to stop, start, and restart a service unit:

```
# systemctl stop vsftpd.service
# systemctl start vsftpd.service
# systemctl restart vsftpd.service
```

You can also omit the suffix and specify multiple units:

```
# systemctl restart vsftpd sshd
```

As a Linux administrator you might change the configuration of a running service and want to have it to reload its configuration without restarting the service. Service unit files can use the ExecReload= option to specify a command line (like sending a signal) that triggers the reload of the configuration. If your service supports it, you can trigger reload using the following command:

```
# systemctl reload sshd
```

Not all services support that command:

```
# systemctl reload vsftpd
Failed to reload vsftpd.service: Job type reload is not applicable for unit
vsftpd.service.
```

To reload the configuration of services that support it and restart services that do not, type:

```
# systemctl reload-or-restart sshd vsftpd
```

Listing installed service units

Use the following command to list installed service units:

```
# systemctl list-unit-files -t service
UNIT FILE                               STATE
after-local.service                     static
atd.service                             disabled
auditd.service                          disabled
autofs.service                          disabled
```

Listing enabled service units

To list the service units that are wanted or required by another unit (such as the `multiuser.target`) type:

```
# systemctl list-unit-files -t service --state=enabled
UNIT FILE                               STATE
btrfsmaintenance-refresh.service        enabled
cio_ignore.service                      enabled
cron.service                            enabled
dm-event.service                        enabled
getty@.service                          enabled
```

Disabling a service

Use the following command to disable a service:

```
# systemctl disable vsftpd
rm '/etc/systemd/system/multi-user.target.wants/vsftpd.service'
```

Enabling a service

Use the following command to enable a service:

```
# systemctl enable vsftpd
ln -s '/usr/lib/systemd/system/vsftpd.service'
'/etc/systemd/system/multi-user.target.wants/vsftpd.service'
```

Note: Issuing the `enable` command for a service unit doesn't trigger the activation of the service unit. It doesn't start the service. It merely creates a dependency on another unit, for example the `multi-user.target` unit. This dependency causes the start of the service whenever the unit the service depends on is activated, for example the `multi-user.target`. Disabling the service unit removes the dependency.

Checking which socket units activate which services

Use the following command to check which socket units activate which services:

```
# systemctl list-sockets
LISTEN                 UNIT                       ACTIVATES
/dev/initctl           systemd-initctl.socket     systemd-initctl.service
/dev/log               systemd-journald.socket    systemd-journald.service
/run/dmeventd-client   dm-event.socket            dm-event.service
...
```

8.2.2 Managing systemd target units

Target units group units and act as synchronization points. The concept of systemd targets is somewhat similar to the concept of runlevels of the SysV-style init system, but there are a couple of differences. Rather than using runlevel numbers the systemd targets have names, because they are units. Also, with systemd there is usually more than one target active, because a target unit might depend on another target unit.

Listing active targets

Use the following command to list active targets:

```
# systemctl --type target
UNIT                LOAD   ACTIVE SUB    DESCRIPTION
basic.target        loaded active active Basic System
cryptsetup.target   loaded active active Encrypted Volumes
getty.target        loaded active active Login Prompts
local-fs-pre.target loaded active active Local File Systems (Pre)
local-fs.target     loaded active active Local File Systems
multi-user.target   loaded active active Multi-User System
...
```

Listing all targets

Use the following command to list all targets:

```
# systemctl --type target --all
UNIT                   LOAD   ACTIVE   SUB    DESCRIPTION
basic.target           loaded active   active Basic System
cryptsetup.target      loaded active   active Encrypted Volumes
emergency.target       loaded inactive dead   Emergency Mode
final.target           loaded inactive dead   Final Step
getty.target           loaded active   active Login Prompts
graphical.target       loaded inactive dead   Graphical Interface
local-fs-pre.target    loaded active   active Local File Systems (Pre)
local-fs.target        loaded active   active Local File Systems
multi-user.target      loaded active   active Multi-User System
...
```

Querying the default target the system boots into

Use the following command to query the default target that the system boots into:

```
# systemctl get-default
graphical.target
```

Setting the default target the system should boot into

Use the following command to set the default target:

```
# systemctl set-default multi-user.target
ln -s '/usr/lib/systemd/system/multi-user.target'
'/etc/systemd/system/default.target'
```

Switching to a target

Use the following command to switch to a target:

```
# systemctl isolate multi-user.target
```

The **isolate** command activates the specified target and all of its dependent units. All other units are stopped. By default, only a few targets are isolatable to prevent unusable system states. Table 8-1 maps important systemd targets to SysV runlevels.

Table 8-1 Mapping systemd targets to SysV runlevels

systemd Target	SysV Runlevel	Notes
poweroff.target	0	Halts the system.
rescue.target	1, s, single	Single-user mode that provides a base system and a rescue shell.
multi-user.target	2, 3, 4	Multi-user, non-graphical but with Network and Services running.
graphical.target	5	Multi-user, Graphical.
reboot.target	6	Reboot the system.
emergency.target	emergency	Emergency shell. This is a special systemd target unit that can be be specified as a kernel command line argument: **systemd.unit=emergency.target**.

8.3 Working with the systemd journal

This section describes how to enable persistent journal data, and how to view the journal.

8.3.1 Getting started with the journal

The journal is part of systemd, and provides a modern logging mechanism. It enables you to capture Kernel log messages, regular syslog messages, the `stdout` and `stderr` written by services, and messages from the early boot stages. In addition to the log message text, the journal stores metadata, such as the product identifier (PID), user identifier (UID), group identifier (GID), executable file, and so on (see man `systemd.journal-fields`). All of this information is indexed, and can be queried by the administrator.

In SUSE Linux Enterprise Server 12, the data of the journal itself is not stored permanently by default. The systemd-journald is configured to store its data onto a small in-memory disk (`/run/log/journal`, which is a tmpfs) and to forward the log messages to syslog, which means that they can be picked up by any traditional log daemon. Both operating systems have the rsyslogd service enabled, which receives the messages from the journald and stores them persistently to `/var/log/messages`.

Therefore, only the latest log data can be queried from the journal directly. To change the systemd-journald to store its data persistently on disk, create the `/var/log/journal` directory. When the journald gets restarted, it detects the existence of this directory and start to store the journal permanent on disk. Then, the traditional log daemon can be turned off.

Enabling persistent journal data at /var/log/journal

Use the following command to enable persistent journal data:

```
# mkdir /var/log/journal
# systemctl restart systemd-journald.service
```

Disabling and stopping the traditional log daemon (optional)

This is an optional step. If wanted you can also use both, the systemd-journald and the traditional log daemon. In case you don’t rely on the traditional log daemon, it can be turned off:

```
# systemctl disable rsyslog.service
rm '/etc/systemd/system/multi-user.target.wants/rsyslog.service'
rm '/etc/systemd/system/syslog.service'
# systemctl stop rsyslog.service
```

8.3.2 Viewing the journal

Use the `journalctl` utility to view and filter the journal. If started without any parameters, it shows the entire journal in a format that is similar to traditional syslog files. The `journalctl` utility pipes the output into `less`, the default pager:

```
# journalctl
-- Logs begin at Wed 2015-04-22 14:59:04 EDT, end at Wed 2015-04-22 16:44:17 EDT.
Apr 22 16:43:51 linux1.itso.ibm.com systemd-journal[64]: Runtime journal is using
Apr 22 16:43:51 linux1.itso.ibm.com systemd-journal[64]: Runtime journal is using
Apr 22 16:43:51 linux1.itso.ibm.com kernel: Initializing cgroup subsys cpuset
...
```

View the journal and advise the `less` pager to jump to the end of the log:

```
# journalctl -e
```

Show the last twenty log events (like **`tail -n20`**):

```
# journalctl -n20
```

A live view of the journal can be obtained using the `--follow` option. It will show the last ten lines like **`tail -f`**, and then display updates as soon as new log messages are being added to the journal:

```
# journalctl -f
```

The live view can be stopped using Ctrl+C. That sends a `SIGINT` signal to the *journalctl* process, which causes it to end.

The default log output is intended to be readable for humans. There are other output formats that are intended to be machine-friendly:

```
# journalctl -o verbose
# journalctl -o json-pretty
```

8.3.3 Filtering the journal

This section describes several actions you can perform to filter the journal:

- Show the log messages of the current boot (filters out the messages from previous boots):

  ```
  # journalctl -b
  ```

- Show today's log messages:

  ```
  # journalctl --since today
  ```

- Show kernel messages of the current boot only:

  ```
  # journalctl -b -k
  ```

- Only show errors:

  ```
  # journalctl -p err
  ```

- Show the log messages of a specific unit (like the sshd.service unit):

  ```
  # journalctl -u sshd.service
  ```

- Show the messages logged by a specific executable file:

  ```
  # journalctl /usr/sbin/sshd
  ```

- Show the messages logged by a specific PID, UID, or GID:

  ```
  # journalctl _PID=0
  # journalctl _UID=0
  # journalctl _GID=0
  ```

  For more fields see `man systemd.journal-fields`.

- Show the space on disk occupied by the journal data:

  ```
  # journalctl --disk-usage
  ```

Journals take up 40.0 megabytes (MB) on disk.

8.4 The system boot process

This section provides an overview of the boot process on Linux instances with systemd.

The boot loader

The zipl boot loader loads the Linux kernel with the initramfs and the boot parameters into memory and starts the kernel. Then the kernel unpacks the initramfs and starts **`/sbin/init`** provided by the initramfs, which is usually a symlink to systemd these days.

Note: On SUSE Linux Enterprise Server 12, the Linux that zipl loads is a grub2 shell that starts the actual SUSE Linux Enterprise Server 12 kernel with the initramfs and the boot parameters using kexec. This is why the default zipl menu has an entry called grub2.

The initramfs command

The initramfs contains the user space that is supposed to bring the disk that contains the root file system online and switch over to it. For example, if your rootfs is on a logical volume it needs to activate the volume group. After the rootfs is mounted to `/sysroot`, it is used as the new root directory. This is done by the `initrd-switch-root.service`. It runs **`systemctl switch-root /sysroot`** that switches the root directory and starts the systemd as PID 1. The job of the initramfs is done.

The systemd command

From now on, systemd is responsible to start your system. Its ultimate goal is to reach the default target. First it needs to load all of the unit files. Some of the unit files are generated dynamically by generators. For example, the systemd-fstab-generator creates mount units for the entries found in the `/etc/fstab`. After systemd has loaded the unit files, it knows the dependency tree and activates all units necessary for the default target.

8.5 Analyzing Linux instances that use systemd

This section shows a few commands to retrieve performance statistics and information about the dependencies between systemd units.

8.5.1 Retrieving performance statistics

The following list provides examples of actions you can take to retrieve performance statistics:

- Show the time spent in the various stages of the boot process, after the boot loader started the kernel:

  ```
  # systemd-analyze time
  Startup finished in 386ms (kernel) + 904ms (initrd) + 3.005s (userspace) =
  4.296s
  ```

- Show the time it took to initialize the service units, sorted by time:

```
# systemd-analyze blame
           537ms kdump.service
           509ms firewalld.service
           448ms lvm2-monitor.service
           401ms network.service
           363ms postfix.service
           345ms tuned.service
           220ms lvm2-pvscan@94:2.service
           190ms auditd.service
           119ms device_cio_free.service
           ...
```

- Display the time-critical chain of a systemd unit:

```
# systemd-analyze critical-chain default.target
The time after the unit is active or started is printed after the "@"
character.
The time the unit takes to start is printed after the "+" character.

multi-user.target @3.001s
··postfix.service @2.464s +363ms
  ··network.target @2.462s
    ··network.service @2.060s +401ms
      ··NetworkManager.service @1.941s +117ms
        ··firewalld.service @1.430s +509ms
          ··basic.target @1.429s
            ...
```

- Generate a Scalable Vector Graphics (SVG) image (as shown in Figure 8-1) that shows the details about the boot process:

  ```
  # systemd-analyze plot > plot.svg
  ```

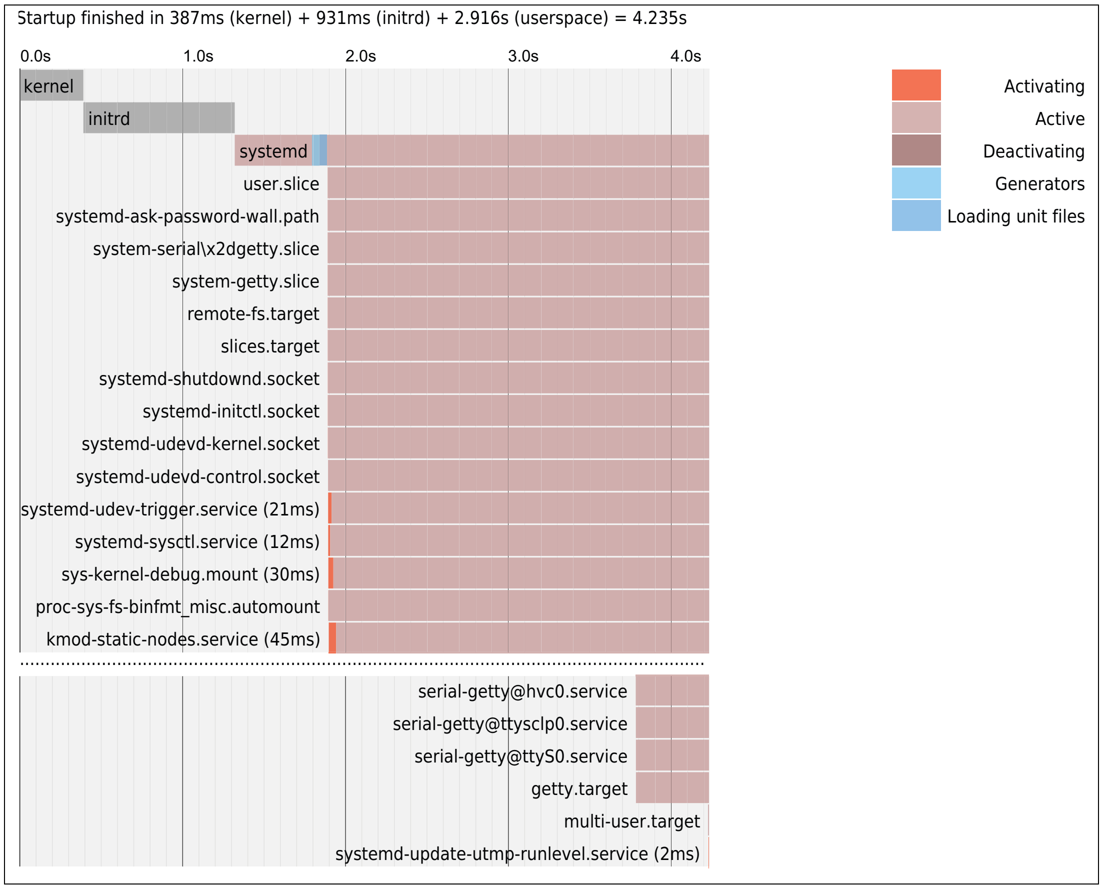


Figure 8-1 A systemd-analyze plot

Note: Scalable Vector Graphics (SVG) files can easily be viewed using your internet browser.

8.5.2 Retrieving information about unit dependencies

With systemd, there are two different types of dependencies between units:

- Dependencies that affect the activation of units (Requires/Wants/Conflicts).
- Dependencies that affect the order of units (After/Before).

The activation perspective (Requires/Wants/Conflicts)

Show units that the specified unit requires or wants:

```
# systemctl list-dependencies sshd.service
sshd.service
··sshd-keygen.service
··system.slice
··basic.target
  ··firewalld.service
  ...
```

Show units that require or want the specified unit:

```
# systemctl list-dependencies --reverse sshd.service
sshd.service
··multi-user.target
  ··graphical.target
```

The order perspective (After/Before)

List the units the specified unit has an “After” dependency on (shows units that need to started up prior the given unit):

```
# systemctl list-dependencies --after sshd.service
sshd.service
··sshd-keygen.service
··system.slice
··systemd-journald.socket
··basic.target
...
```

List the units the specified unit has a “Before” dependency on (shows units that need to delayed until the specified unit has started up):

```
# systemctl list-dependencies --before sshd.service
sshd.service
··multi-user.target
· ··systemd-readahead-done.service
· ··systemd-readahead-done.timer
· ··systemd-update-utmp-runlevel.service
· ··graphical.target

...
```

9

Miscellaneous recipes

“*Try not to become a man of success, but rather try to become a man of value.*”

— Albert Einstein

This chapter contains information that falls under the miscellaneous category. These are topics that help make administration easier, save time, increase functionaility, or add capabilities to your systems.

This chapter provides information about the following topics:

9.1 Rescuing a Linux system

This section describes how to boot your Linux server into different modes for troubleshooting purposes. It covers booting Linux into single user mode, and also entering a rescue environment when you require more advanced troubleshooting.

9.1.1 The initrd shell and systemd targets

When using previous versions, Linux SysV offered special run levels, such as the single user mode or the emergency mode, that could be used for special tasks. The systemd command introduces a new concept, called *targets*. It offers the same functionality, but in a different way. For more information, review Chapter 8, "Working with systemd" on page 101.

9.1.2 Using rescue mode with SUSE Linux Enterprise Server

The procedure to enter rescue mode in SUSE Linux Enterprise Server is similar to a manual installation. To enter rescue mode for `LINUX6` in this example, proceed as follows:

1. Log on to `LNXMAINT`.
2. Copy the `SLES12 EXEC` file to a new file named `RESCUE EXEC`, and copy the parameter file to a new file (`LINUX6 RESCUE` in this example):

```
===> copy sles12 exec d rescue = =
===> copy linux6 parm-s11 d = rescue =
```

3. Edit `RESCUE EXEC` to point to the new `RESCUE` file:

```
===> x rescue exec d
...
/* */
'CL RDR'
'PURGE RDR ALL'
'SPOOL PUNCH * RDR'
'PUNCH KERNEL  IMG * (NOH'
'PUNCH rescue  PRM * (NOH'
'PUNCH INITRD  IMG * (NOH'
'CH RDR ALL KEEP NOHOLD'
'I 00C'
```

4. Edit the `LINUX6 RESCUE` file, deleting the autoyast line and set `Manual=1`:

```
===> x linux6 rescue d
LINUX6   RESCUE   D1  F 80  Trunc=80 Size=10 Line=0 Col=1 Alt=1

ramdisk_size=65536 root=/dev/ram1 ro init=/linuxrc
TERM=dump HostIP=9.12.7.6 Hostname=virtcook6.itso.ibm.com
Gateway=9.12.4.1 Netmask=255.255.240.0 Layer2=1
ReadChannel=0.0.0600 WriteChannel=0.0.0601
DataChannel=0.0.0602 OSAHWAddr=
Nameserver=9.12.6.7 Portname= Portno=0
Install=ftp://9.12.7.8/SLES-12
UseVNC=1 VNCPassword=12345678
UseSSH=1 SSHPassword=12345678
InstNetDev=osa OsaInterface=qdio OsaMedium=eth Manual=1
```

5. Log off from `LNXMAINT`.

6. Log on to `LINUX6`.
7. Answer **no** to the initial program load (IPL) from `100` question.
8. Define storage to 1 gigabyte (GB):

```
===> def stor 1g
00: STORAGE = 1G
00: Storage cleared - system reset.
Enter
```

9. IPL CMS:

```
===> ipl cms
```

10. Run the **`RESCUE EXEC`**:

```
===> rescue
00: 0000001 FILE  PURGED
00: RDR FILE 0002 SENT FROM LINUX6   PUN WAS 0002 RECS 113K CPY  001 A NOHOLD
NO
KEEP
00: RDR FILE 0006 SENT FROM LINUX6   PUN WAS 0006 RECS 0011 CPY  001 A NOHOLD
NO
KEEP
00: RDR FILE 0010 SENT FROM LINUX6   PUN WAS 0010 RECS 204K CPY  001 A NOHOLD
NO
KEEP
00: 0000003 FILES CHANGED
00: 0000003 FILES CHANGED
...
Kernel command line: ramdisk_size=65536 root=/dev/ram1 ro init=/linuxrc
                     TERM=dump HostIP=9.12.7.6 Hostname=virtcook6.itso.ibm.com
                     Gateway=9.12.4.1 Netmask=255.255.240.0 Layer2=1
                     ReadChannel=0.0.0600 WriteChannel=0.0.0601
                     DataChannel=0.0.0602 OSAHWAddr=
                     Nameserver=9.12.6.7 Portname= Portno=0
                    Install=ftp://9.12.7.87/SLE-12-Server-DVD-s390x-GM-DVD1
                     UseVNC=1 VNCPassword=12345678
                     UseSSH=1 SSHPassword=12345678
                     InstNetDev=osa OsaInterface=qdio OsaMedium=eth Manual=1
...
>>> Linuxrc v3.3.91 (Kernel 3.0.76-0.9-default) <<<

Main Menu

1) Start Installation
2) Settings
3) Expert
4) Exit or Reboot
```

11. Select **1** for Start Installation:

```
Start Installation

1) Start Installation or Update
2) Boot Installed System
3) Start Rescue System
```

12. Select **3** for Start Rescue System. The rest of the startup procedure just follows the normal installation process. After bootup, you are presented a login prompt:

```
Master Resource Control: runlevel 3 has been   [80C [10D [1mreached [m

Rescue login:
```

13. Enter root as the login user.
14. You are asked about the terminal type to use. In doubt, use the value `vt100`. Only line mode can be used.
15. If a special key combination like [Ctrl]-[C] is needed in this terminal, write the characters ^C instead.
16. To leave the rescue system, type **exit**.

This section has described how to rescue a damaged Linux system.

9.2 Setting up Memory Hotplugging

Linux *Memory Hotplug* allows the amount of memory in a Linux system to be increased or decreased without a reboot. You must first have standby memory defined to the virtual machine in which Linux is running. You can issue the **`CP DEFINE STORAGE`** command to configure standby memory (storage).

To set up standby storage for Linux memory hotplug, using, for example, `LINUX1` as the virtual machine, perform the following steps:

1. To give the virtual machine an additional 1 gigabyte (GB) of standby memory, this change can be done either in the directory level or per specific virtual machine using CMS command-line mode. The following examples show how to change a specific Linux virtual machine to define a standby memory using the CMS command. The second example shows you how to make the changes in the directory, and then how the Linux guest will load the changes:

   – If you want this to be a temporary change for this bootup of Linux only, or for some other reason decide not to change your virtual machine's directory entry, type the following **`DEFINE`** statement from a 3270 terminal:

```
===> define storage 1GB standby 1GB
00: HCPZPM003E Invalid option - 1GB
Ready(00003); T=0.01/0.01 14:53:04
define storage 1G  standby 1G
00: STORAGE = 1G MAX = 2G INC = 2M STANDBY = 1G RESERVED = 0
00: Storage cleared - system reset.
===> ipl 100
```

   – If you would like to make this a permanent change, you must update the user directory entry for this virtual machine. To do so, reference *The Virtualization Cookbook for IBM z Systems Volume 1: IBM z/VM 6.3*, SG24-8147 in the section entitled "Change the amount of memory a user is assigned". When directory changes have been made, log on to `LINUX1`. You should see the standby memory reported:

```
LOGON LINUX1
00: z/VM Version 6 Release 3.0, Service Level 1501 (64-bit),
00: built on IBM Virtualization Technology
00: There is no logmsg data
00: FILES: 0003 RDR,   NO PRT,   NO PUN
00: LOGON AT 15:27:29 EDT MONDAY 04/27/15
```

```
00: STORAGE = 1G MAX = 2G INC = 2M STANDBY = 1G RESERVED = 0
00: Storage cleared - system reset.
00: Command complete
00: NIC 0600 is created; devices 0600-0602 defined
00: NIC 0600 is connected to VSWITCH SYSTEM VSW1
z/VM V6.3.0    2015-04-09 09:04
LINUX1   AT ITSOZVM1 VIA RSCS     2015-04-27 15:27:30 EDT      MONDAY
DMSACR723I D (LNX:LNXADMIN.) R/O
DIAG swap disk defined at virtual address 0300 (64988 4K pages of swap
space)
DIAG swap disk defined at virtual address 0301 (129980 4K pages of swap
space)
DO YOU WANT TO IPL LINUX FROM MINIDISK 100? Y/N
Y
```

2. Start an SSH session as root and view the memory in the `/sys/` file system. Change directory to `/sys/devices/system/memory/` and list the files:

```
# cd /sys/devices/system/memory
# ls
block_size_bytes  memory1  memory3  memory5  memory7  uevent
memory0           memory2  memory4  memory6  power
```

3. Type the `block_size_bytes` file with the **cat** command:

```
# cat block_size_bytes
10000000
```

This number is the number of bytes in hexadecimal. 10000000 in hex is 256 megabytes (MB) in decimal. So the block size is 256 MB and there are eight blocks: `memory0-memory7`, which are represented as directories. Each of the memory blocks has a state, which is represented as a file.

4. Show the state of each memory block with the following command:

```
# cat memory*/state
online
online
online
online
offline
offline
offline
offline
```

This shows that the first 1 GB is online, and the next four blocks are offline.

5. You can also show information about memory with the **free -m** command:

```
# free -m
            total        used        free      shared  buff/cache   available
Mem:         991          94          766      0          130          858
Swap:       1785           0         1785
```

This shows 766 MB of free memory available (some of the memory is used internally by Linux).

6. You can turn on memory by sending the string **online** to the state file. Turn on an additional 512 MB of memory with the following commands:

```
# echo online > memory4/state
# echo online > memory5/state
```

7. Show that the memory is now online:

```
# cat memory*/state
online
online
online
online
online
online
offline
offline
```

8. Again, confirm with the **free -m** command:

```
# free -m
              total        used        free      shared  buff/cache   available
Mem:           1503          96        1276           0         130        1367
Swap:          1785           0        1785
```

This shows that 1276 MB of free memory is now available.

9. You can also give the memory back by echoing **offline** to the state file:

```
# echo offline  > memory4/state
# echo offline  > memory5/state
```

10. Verify that the memory has to be returned:

```
# cat memory*/state
online
offline
offline
offline
# free -m
              total        used        free      shared  buff/cache   available
Mem:            991          94         768           0         129         859
Swap:          1785           0        1785
```

This section has shown how to configure virtual machines with standby memory, and how to "hot-plug" the memory from Linux. This function can increase your system's performance and availability.

9.3 Using the cpuplugd service

The **cpuplugd** service allows Linux to enable or disable processors (CPUs) and memory, based on a set of rules. It can improve performance by setting the correct number of processors and amount of memory for Linux systems, depending on their current load. It can also prevent the Linux scheduler from queue balancing in partial load situations.

9.3.1 Determining the virtual CPUs being used

To start work with **cpuplugd**, perform the following steps:

1. Start an SSH session to the Linux system to determine how many CPUs Linux has online:

```
# lscpu
Architecture:          s390x
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Big Endian
CPU(s):                2
On-line CPU(s) list:   0,1
Thread(s) per core:    1
Core(s) per socket:    1
Socket(s) per book:    1
Book(s):               2
Vendor ID:             IBM/S390
BogoMIPS:              20325.00
Hypervisor:            z/VM 6.3.0
Hypervisor vendor:     IBM
Virtualization type:   full
Dispatching mode:      horizontal
L1d cache:              128K
L1i cache:               96K
L2d cache:             2048K
L2i cache:             2048K
```

2. Observe the status of the **cpuplugd** service:

```
# systemctl status cpuplugd
cpuplugd.service - LSB: Start the cpu hotplug daemon for Linux on System z
   Loaded: loaded (/etc/rc.d/init.d/cpuplugd)
   Active: inactive (dead)
```

3. Start the cpuplugd daemon:

```
# systemctl start cpuplugd
# systemctl status cpuplugd
cpuplugd.service - LSB: Start the cpu hotplug daemon for Linux on System z
   Loaded: loaded (/etc/rc.d/init.d/cpuplugd)
   Active: active (running) since Mon 2015-04-27 16:07:50 EDT; 1s ago
  Process: 4814 ExecStart=/etc/rc.d/init.d/cpuplugd start (code=exited, 
status=0/SUCCESS)
 Main PID: 4821 (cpuplugd)
   CGroup: /system.slice/cpuplugd.service
           ââ4821 /usr/sbin/cpuplugd -c /etc/sysconfig/cpuplugd
```

4. Wait a few minutes and run the **lscpus** script again:

```
# lscpu
Architecture:          s390x
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Big Endian
CPU(s):                2
On-line CPU(s) list:   0
Off-line CPU(s) list:  1
Thread(s) per core:    1
Core(s) per socket:    1
Socket(s) per book:    1
Book(s):               1
```

```
Vendor ID:              IBM/S390
BogoMIPS:               20325.00
Hypervisor:             z/VM 6.3.0
Hypervisor vendor:      IBM
Virtualization type:    full
Dispatching mode:       horizontal
L1d cache:               128K
L1i cache:                96K
L2d cache:              2048K
L2i cache:              2048K
```

The output shows that now only one of the two virtual CPUs are active. The **cpuplugd** service turned off the other one.

5. The **cpuplugd** configuration file is `/etc/sysconfig/cpuplugd`. Some middleware products recommend a minimum of two virtual processors. If most of your Linux servers will be running a workload that recommends two processors, change the default for `CPU_MIN` to `2`. An exception would be when only a single physical processor is available. View the non-comments and lines that are not blank in the configuration file with the following command:

```
# cd /etc/sysconfig
# egrep -v '^$|^#' cpuplugd
CPU_MIN="1"
CPU_MAX="0"
UPDATE="1"
CMM_MIN="0"
CMM_MAX="131072"        # 512 MB
pgscan_d="vmstat.pgscan_direct_dma[0] + vmstat.pgscan_direct_normal[0] + vmstat.
pgscan_direct_movable[0]"
pgscan_d1="vmstat.pgscan_direct_dma[1] + vmstat.pgscan_direct_normal[1] + vmstat
.pgscan_direct_movable[1]"
pgscanrate="(pgscan_d - pgscan_d1) / (cpustat.total_ticks[0] - cpustat.total_tic
ks[1])"
avail_cache="meminfo.Cached - meminfo.Shmem"
user_0="(cpustat.user[0] - cpustat.user[1])"
nice_0="(cpustat.nice[0] - cpustat.nice[1])"
system_0="(cpustat.system[0] - cpustat.system[1])"
user_2="(cpustat.user[2] - cpustat.user[3])"
nice_2="(cpustat.nice[2] - cpustat.nice[3])"
system_2="(cpustat.system[2] - cpustat.system[3])"
CP_Active0="(user_0 + nice_0 + system_0) / (cpustat.total_ticks[0] - cpustat.tot
al_ticks[1])"
CP_Active2="(user_2 + nice_2 + system_2) / (cpustat.total_ticks[2] - cpustat.tot
al_ticks[3])"
CP_ActiveAVG="(CP_Active0+CP_Active2) / 2"
idle_0="(cpustat.idle[0] - cpustat.idle[1])"
iowait_0="(cpustat.iowait[0] - cpustat.iowait[1])"
idle_2="(cpustat.idle[2] - cpustat.idle[3])"
iowait_2="(cpustat.iowait[2] - cpustat.iowait[3])"
CP_idle0="(idle_0 + iowait_0) / (cpustat.total_ticks[0] - cpustat.total_ticks[1]
)"
CP_idle2="(idle_2 + iowait_2) / (cpustat.total_ticks[2] - cpustat.total_ticks[3]
)"
CP_idleAVG="(CP_idle0 + CP_idle2) / 2"
CMM_INC="meminfo.MemFree / 40"
CMM_DEC="meminfo.MemTotal / 40"
HOTPLUG="((1 - CP_ActiveAVG) * onumcpus) < 0.08"
HOTUNPLUG="(CP_idleAVG * onumcpus) > 1.15"
MEMPLUG="0"
MEMUNPLUG="0"
```

The following default rules are for the plugging and unplugging of CPUs in the configuration file:

```
HOTPLUG="((1 - CP_ActiveAVG) * onumcpus) < 0.08"
HOTUNPLUG="(CP_idleAVG * onumcpus) > 1.15"
```

The variables in the statements have the following meaning:

`loadavg`	The current average CPU load
`onumcpus`	The number of CPUs that are online
`runable_proc`	The current number of processes that can be run
`idle`	The current idle percentage

These CPU hot-plugging and unplugging values will be used in the next section. In the default setup, **cpuplugd** only changes the virtual processor configuration. The auto-adaptive adjustment of the memory using the `cmm` feature (module) is deactivated by default, and also not available when running in a native logical partition (LPAR) environment.

9.3.2 Generating a workload to see cpuplugd work

You can now generate a workload to show how the **cpuplugd** turns on CPUs.

Important: Running the following command generates significant CPU use. Verify that there is not a mission-critical workload running on this IBM z/VM LPAR, because this test might affect it. Also, be sure to stop the processes after seeing **cpuplugd** in action.

Perform the following steps:

1. Put 10 looping jobs in the background with the following **for** loop:

```
# for i in `seq 1 10`
> do
>   bash -c "cat /dev/zero > /dev/null" &
> done
[1] 2441
[2] 2442
[3] 2443
[4] 2444
[5] 2445
[6] 2446
[7] 2447
[8] 2448
[9] 2449
[10] 2453
```

2. See that the jobs are running (you can also use the **top** command):

```
# pstree -G | grep cat
    ··sshd···sshd···bash···50*[bash···cat]
```

3. Run the **lscpu** command. The following example shows that, after a minute or so, **cpuplugd** has started the other spare processor:

```
# lscpu
Architecture:          s390x
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Big Endian
CPU(s):                2
On-line CPU(s) list:   0,1
```

```
Thread(s) per core:    1
Core(s) per socket:    1
Socket(s) per book:    1
Book(s):               2
Vendor ID:             IBM/S390
BogoMIPS:              20325.00
Hypervisor:            z/VM 6.3.0
Hypervisor vendor:     IBM
Virtualization type:   full
Dispatching mode:      horizontal
L1d cache:             128K
L1i cache:             96K
L2d cache:             2048K
L2i cache:             2048K
```

After a few more minutes, all of the CPUs should be activated.

4. Stop the created workload before using the **killall** command:

```
# killall cat
[1]   Exit 143               bash -c "cat /dev/zero > /dev/null"
[2]   Exit 143               bash -c "cat /dev/zero > /dev/null"
[3]   Exit 143               bash -c "cat /dev/zero > /dev/null"
[4]   Exit 143               bash -c "cat /dev/zero > /dev/null"
[5]   Exit 143               bash -c "cat /dev/zero > /dev/null"
...
[48]   Exit 143               bash -c "cat /dev/zero > /dev/null"
[49]-  Exit 143               bash -c "cat /dev/zero > /dev/null"
[50]+  Exit 143               bash -c "cat /dev/zero > /dev/null"
...
```

5. Run the **lscpu** command. The following example shows that, after a minute or so, **cpuplugd** has stopped the other processor:

```
# lscpu
Architecture:          s390x
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Big Endian
CPU(s):                2
On-line CPU(s) list:   0
Off-line CPU(s) list:  1
Thread(s) per core:    1
Core(s) per socket:    1
Socket(s) per book:    1
Book(s):               1
Vendor ID:             IBM/S390
BogoMIPS:              20325.00
Hypervisor:            z/VM 6.3.0
Hypervisor vendor:     IBM
Virtualization type:   full
Dispatching mode:      horizontal
L1d cache:             128K
L1i cache:             96K
L2d cache:             2048K
L2i cache:             2048K
```

9.3.3 Setting memory sizes with cpuplugd

Memory sizes can also be set by the **cpuplugd** service. However, unlike CPUs, there is no good generic default value. The following example is in the *Device Drivers* book:

```
MEMPLUG = "swaprate > freemem+10 & freemem+10 < apcr"
MEMUNPLUG = "swaprate < freemem + 10000"
```

However, this is just a starting point to explain the syntactical structure of a rule. Do not use this configuration in production. You should test any setting that you want to implement against a representative workload that your Linux systems will be running. Details are beyond the scope of this section.

More information about **cpuplugd** can be found in the manual *Linux on z Systems Device Drivers, Features, and Commands* for SUSE Linux Enterprise Server 12, on the web at the following site:

http://ibm.com/developerworks/linux/linux390/documentation_novell_suse.html

9.4 Hardware cryptographic acceleration support for OpenSSH using SUSE Linux Enterprise Server 12

This section shows how to copy a test file with OpenSSH, first without any crypto acceleration. Then, crypto acceleration for OpenSSH is enabled and the same file is copied again. A much higher throughput rate should be observed. The prerequisite for using hardware cryptography is to have a firmware level of LIC 3863 installed on your IBM z Systems central processor complex (CPC).

This section is based on the white paper *First experiences with hardware Cryptographic Support for OpenSSH with Linux for z Systems*, by Manfred Gnirss, Winfried Münch, Klaus Werner, and Arthur Winterling, on the web at the following site:

http://ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101690

This section shows only a single example of crypto-acceleration. For a much more complete and detailed analysis, see the previously mentioned white paper.

To test copying a file with and without cryptographic acceleration, perform the following steps:

1. Start an SSH session as root to any Linux.
2. Create a 1 GB test file for copying in the /tmp/ directory:

   ```
   # cd /tmp
   # dd if=/dev/zero of=testdata.txt bs=1048576 count=1000
   1000+0 records in
   1000+0 records out
   1048576000 bytes (1.0 GB) copied, 13.5595 s, 77.3 MB/s
   ```

3. Copy the file locally with the **scp** command, two times with specific encryption algorithms and one time without, prefixing all with the **time** command:

   ```
   # time scp -c 3des-cbc testdata.txt localhost:/dev/null
   testdata.txt                                      100% 1000MB  27.8MB/s   00:36

   real    0m39.799s
   user    0m32.102s
   sys     0m0.829s
   ```

4. Determine if the necessary cryptographic-related RPMs are installed:

   ```
   # rpm -qa | grep openssl-ibmca
   ```

   No output shows that they are not installed.

You can then proceed with the following steps:

1. Install the RPMs with the **zypper install** command:

   ```
   # zypper in openssl-ibmca libica-2_1_0
   Loading repository data...
   Reading installed packages...
   'libica-2_1_0' not found in package names. Trying capabilities.
   Resolving package dependencies...

   The following 2 NEW packages are going to be installed:
     libica-2_3_0 openssl-ibmca

   2 new packages to install.
   Overall download size: 62.0 KiB. Already cached: 0 B  After the operation,
   additional 185.5 KiB will be used.
   Continue? [y/n/? shows all options] (y):
   ....
   (2/2) Installing: openssl-ibmca-1.2.0-144.2
   ..............................[done]
   ```

2. Show the new RPMs with the following command:

   ```
   # rpm -qa | egrep "libica|ibmca"
   openssl-ibmca-1.2.0-141.13.1
   libica-2_1_0-2.1.0-0.12.28
   libica-2_1_0-2.1.0-0.12.28
   openssl-ibmca-32bit-1.2.0-141.13.1
   ```

3. Verify that CP Assist for Cryptographic Function (CPACF) operations are supported:

```
# icainfo
The following CP Assist for Cryptographic Function (CPACF) operations are
supported by libica on this system:
SHA-1:        yes
SHA-256:      yes
SHA-512:      yes
DES:          yes
TDES-128:     yes
TDES-192:     yes
AES-128:      yes
AES-192:      yes
AES-256:      yes
PRNG:         yes
CCM-AES-128:  yes
CMAC-AES-128: yes
CMAC-AES-192: yes
CMAC-AES-256: yes
```

You should now have the cryptographic packages installed on SUSE Linux Enterprise Server.

To configure cryptography on SUSE Linux Enterprise Server 12, perform the following steps:

1. Make a backup of the SSL configuration file, `/etc/ssl/openssl.cnf`:

   ```
   # cd /etc/ssl
   # cp openssl.cnf openssl.cnf.orig
   ```

2. Append the sample SSL configuration file in the package documentation to the actual SSL configuration file, `/etc/openssl.cnf`:

   ```
   # cat /usr/share/doc/packages/openssl-ibmca/openssl.cnf.sample >> openssl.cnf
   ```

3. Edit the appended file and search for the line with the **openssl_conf** variable. Move that line from the bottom to the top and save the file, as shown in the following example:

   ```
   # vi openssl.cnf
   /openssl_conf
   #
   # OpenSSL example configuration file.
   # This is mostly being used for generation of certificate requests.
   #

   # This definition stops the following lines choking if HOME isn't
   # defined.
   HOME                    = .
   RANDFILE                = $ENV::HOME/.rnd
   openssl_conf = openssl_def
   ...
   ```

4. Rerun the same **scp** commands:

   ```
   # cd /tmp
   # time scp -c 3des-cbc testdata.txt localhost:/dev/null
   testdata.txt                                  100% 1000MB 250.0MB/s   00:04

   real    0m3.987s
   user    0m3.189s
   sys     0m0.646s
   # time scp -c aes128-cbc testdata.txt localhost:/dev/null
   testdata.txt                                  100% 1000MB 333.3MB/s   00:03

   real    0m3.019s
   user    0m2.194s
   sys     0m0.648s
   ```

5. To check the usage of the hardware functions, run the **icastats** command:

   ```
   # icastats
    function | # hardware | # software
   ----------+------------+------------
       SHA-1 |      18434 |          0
     SHA-224 |          0 |          0
     SHA-256 |        180 |          0
     SHA-384 |          0 |          0
     SHA-512 |          0 |          0
      RANDOM |         17 |          0
    MOD EXPO |          0 |          5
     RSA CRT |          0 |          5
     DES ENC |          0 |          0
     DES DEC |          0 |          0
    3DES ENC |     147200 |          0
   ```

```
 3DES DEC |     147200 |          0
  AES ENC |      73641 |          0
  AES DEC |      73641 |          0
 CMAC GEN |          0 |          0
 CMAC VER |          0 |          0
  CCM ENC |          0 |          0
  CCM DEC |          0 |          0
 CCM AUTH |          0 |          0
  GCM ENC |          0 |          0
  GCM DEC |          0 |          0
 GCM AUTH |          0 |          0
```

6. To set a preferred cipher for the connections, edit the `/etc/ssh/sshd_config` file and add the following line at the end of the configuration file:

   ```
   # cd /etc/ssh
   # vi sshd_config
   ...
   Ciphers aes128-cbc
   ```

7. The changed default for the cipher improves the throughput:

   ```
   # time scp testdata.txt localhost:/dev/null
   testdata.txt                                  100% 1000MB 250.0MB/s   00:04

   real    0m3.472s
   user    0m2.655s
   sys     0m0.656s
   ```

   You should see an improved throughput as a result of using the cryptographic hardware.

8. Delete the test file:

   ```
   # rm /tmp/testdata.txt
   ```

9.5 Use Crypto-Express to seed /dev/random

Linux has a difficult time getting enough entropy from low interrupts and keyboard events on mainframe computers just as on newer distributed servers. Largely, this comes from the fact that the upper interrupts are not used to seed the entropy pool, and usually there is no keyboard or mouse attached to the system. With SUSE Linux Enterprise Server, there is an entropy generator installed by default that increases entropy when needed.

To check the availability of entropy on a server, complete the following steps:

1. Log on to the server:

   ```
   # ssh LINUX4
   ```

2. Check the current entropy:

   ```
   # cat /proc/sys/kernel/random/entropy_avail
   3965
   ```

The values of the available entropy are always kept `128 - 4096`. Therefore, `3965` is a good value. However, although **`haveged`** is likely to produce good entropy, some environments might want to go with the hardware random generator from the Crypto-Express card.

To check what happens without an entropy generator, complete the following steps:

1. Start an SSH session as `root` to a Linux system.
2. Disable the **haveged** service:

```
# rchaveged stop
Shutting down haveged daemon                                                  done
```

3. Run a small loop to see how the entropy develops over time:

```
# for i in {1..20}; do cat /proc/sys/kernel/random/entropy_avail ; done
2985
2857
...
681
553
```

   The numbers getting smaller can be explained with the page randomization of the Linux kernel. For each new process, the Linux kernel uses a small amount of entropy, which leads to a decrease of the available entropy. As entropy decreases, programs like **sshd** and web servers with Secure Sockets Layer (SSL) will have issues.

To enable the hardware random number generator from the Crypto-Express card, a dedicated crypto-domain is needed. To create this, perform the following steps:

1. Log on as `MAINT`.
2. Check for an available crypto-domain:

```
===> query crypto ap
AP 00 CEX4C  Domain 06 available    shared                unspecified
```

3. Dedicate a crypto-domain to one Linux guest:

```
===> dirmaint for LINUX4 CRYPTO DOMAIN 6 APDED 0
```

4. Query the crypto-device again:

```
===> query crypto ap
AP 000 CEX5C  Domain 006 available    dedicated to LINUX4   dedication
```

5. Log on to the virtual machine and start Linux.
6. Start an SSH session as `root`.
7. Turn the **haveged** service off:

```
# systemctl disable haveged
# systemctl stop haveged
```

8. Make sure that the package `libica-2_1_0` is installed:

```
# zypper in libica-2_1_0
```

9. Enable the service **z90crypt**:

```
# systemctl enable z90crypt
# systemctl start z90crypt
```

10. Check the availability of entropy with an endless loop:

```
# watch -n 0.1 cat /proc/sys/kernel/random/entropy_avail
```

11. Stop the program by entering Ctrl+C.

This section has shown how to use the cryptographic hardware to increase entropy to generate numerous random bytes.

9.6 The X Window system

For many years, UNIX-like operating systems have been using the X Window system (commonly just "X"). This system was designed to provide a client/server, hardware-independent, and network-enabled graphical environment. Linux systems current use X.Org, which is an open source implementation of X Window system.

The X communication protocol by its nature is not secure at all. For this reason, it is often used together with SSH protocol, which tunnels X11 traffic using encrypted (and therefore secure) communications.

X11 itself provides the ability to display graphics on raster display, nothing more. If the user wants to be able to move, resize, and otherwise manage windows, a *window manager* is needed. There are many window managers available; some are lightweight and some are more robust. So using a window manager is a good idea, because it provides functionality that one expects from a graphical user interface (GUI).

When you have Linux installed on your workstation, a window manager is probably not enough. Here you want a full desktop environment with menus, icons, taskbars, and so on, such as Gnome and KDE. However, installing GNOME or KDE on IBM System z® is discouraged because they are resource-intensive.

9.6.1 Virtual Network Computing Server

As mentioned earlier, the X server is run where the mouse, keyboard, and monitor are located (on the workstation). In a nutshell, Virtual Network Computing (VNC) Server provides a virtual workstation with all of these peripherals (virtual). The VNC server starts an embedded X server. Then, any X-based application can send its output to this X server, regardless of if the applications are local or remote to the X server.

One big advantage of VNC is that it is session-oriented. If communication to the VNC server is lost, a new connection is reestablished to the session as it was. Also, applications in a disconnected VNC session still continue to run.

Setting up a VNC Server

Commonly, it is *not* recommended to run X on a server machine. However, sometimes programs require an X server to run. A VNC server can be set up with YaST:

1. Start the YaST remote module:
   ```
   # yast
   ```
2. Change to **Network Services** → **Remote Administration (VNC)**.

```
.........................................
.........................................
··These packages need to be installed:  ··
··xorg-x11-Xvnc                         ··
··                                      ··
··                                      ··
··                                      ··
··                                      ··
.........................................
·                                        ·
·           [Install] [Cancel]           ·
.........................................
```

3. Click **Allow Remote Administration**.

```
·Remote Administration Settings························
·(x) Allow Remote Administration                       ·
·( ) Do Not Allow Remote Administration                ·
·······················································

·Firewall Settings·····································
·[ ] Open Port in Firewall  [Firewall Details...]      ·
·Firewall is disabled                                  ·
·······················································
```

4. Click **OK**.
5. Change to **System** → **/etc/sysconfig Editor**.
6. Select **Desktop** → **Display manager**.
7. Make sure that `DISPLAYMANAGER_ROOT_LOGIN_REMOTE` is set to `yes`.
8. Confirm by clicking **Finish**.
9. Select **System** → **Services Manager** and enable `xinet.d`:

```
Edit /etc/xinet.d/vnc and remove the line disable = yes for the entry vnc1:
service vnc1
{
        socket_type     = stream
        protocol        = tcp
        wait            = no
        user            = nobody
        server          = /usr/bin/Xvnc
        server_args     = -noreset -inetd -once -query localhost -geometry
1024x768 -securitytypes none
        type            = UNLISTED
        port            = 5901
}
```

9.6.2 X server on a workstation

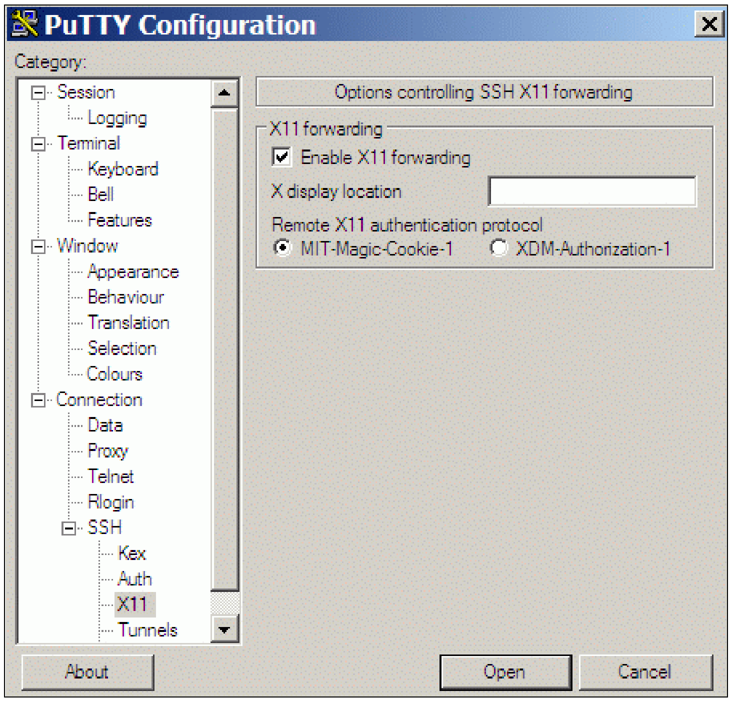
If for some reason VNC is not acceptable, it is possible to use a standard X server on a workstation. Because Linux users usually know the X Window system, an X server running on Windows is described in this section.

Using embedded SSH to forward X with SUSE Enterprise Linux Server

Support for X forwarding with ssh on SUSE Linux Enterprise Server 12 is already included in the installation when installing the patterns `Minimal System (Appliances)` and `Base System`. No further packages are needed to enable this feature.

Using PuTTY

To use PuTTY for X11 forwarding, select X11 forwarding as shown in Figure 9-1.



Options controlling SSH X11 forwarding

X11 forwarding

[✓] Enable X11 forwarding

X display location

Remote X11 authentication protocol

[● MIT-Magic-Cookie-1] XDM-Authorization-1

Figure 9-1 Allow X11 forwarding in PuTTY

When connected to a remote Linux system with X11 forwarding enabled, the `DISPLAY` environment variable contains the special value of **`localhost:10.0`**, which tells PuTTY to forward X11 protocol over SSH to an SSH client address. Remember that for PuTTY to work you require an X Window Server running on your Microsoft Windows workstation.

There are many commercial and free X Window servers available for Microsoft Windows, which provides a free X server that is based on Cygwin.

There are many ways to achieve the same results. It is up to you to choose a solution that best suits your purpose.

9.7 Setting up the IUCV Linux Terminal Server

Implementation of a Linux Terminal Server (LTS) based on z/VM inter-user communication vehicle (IUCV) enables access to the Linux console without a functioning Transmission Control Protocol/Internet Protocol (TCP/IP) stack on Linux.

Many, many IBM customers who run Linux under z/VM consider the implementation of this to be a fundamental requirement for a Linux Virtual Server to be eligible for classification as a production system in their environment. Additionally, it is based on a character mode interface, which enables the use of traditional Linux full-screen tools such as **`vi`**.

The official documentation for setup is on IBM.com at the following website:

http://ibm.com/support/knowledgecenter/linuxonibm/liaaf/lnz_r_zvm_ht.html

Although the documentation has many different options as to how the IUCV LTS can be set up, this section of the cookbook covers implementation using the **`iucvtty`** command.

Implementation of the LTS includes topics that involve changes on both z/VM and Linux. Topics in this section are as follows. Disregard any that do not apply to your Linux distribution:

- "Configuring z/VM for Linux Terminal Server" on page 131
- "Configuring SUSE Linux Enterprise Server 12 for IUCV consoles" on page 131
- "Configuring SUSE Linux Enterprise Server 12 for IUCV LTS" on page 131

9.7.1 Configuring z/VM for Linux Terminal Server

The **IUCV ALLOW** line enables virtual machines to connect to other virtual machines, such as the Linux Terminal Server, by using IUCV. The **IUCV ALLOW** line was included in the **LNXPDFLT PROFILE** entry, covered in *The Virtualization Cookbook for IBM z Systems Volume 1: IBM z/VM 6.3*, SG24-8147, in the section entitled "z/VM User Directory profiles".

9.7.2 Configuring SUSE Linux Enterprise Server 12 for IUCV consoles

A login-protected console IUCV service is already configured by default. No further actions are needed to enable this.

9.7.3 Configuring SUSE Linux Enterprise Server 12 for IUCV LTS

On SUSE Linux Enterprise Server 12, the IUCV hypervisor virtual consoles (HVCs) are configured by default in the system. Therefore, it is only necessary to configure the terminal server, but no changes are required on the systems that need to be connected to.

To configure Linux Terminal Server on SUSE Linux Enterprise Server 12, perform the following steps:

1. Start an SSH session as root to LNXADMIN.
2. Run **yast** → **Network Services** → **IUCV Terminal Server**.
3. Select **z/VM IDs** then press Enter and Tab to switch to the z/VM IDs input field.
4. Add all the machines that LNXADMIN should connect to.
5. Click the **IUCVConn** tab.
6. Select **Enable IUCVConn on Login**.
7. Enter a password for the IUCVCONN user and confirm it.
8. Click **OK** to finish the configuration.
9. Start a new SSH session to lnxadmin. Use one of the machines that were configured and is running Linux at that moment. The password for the connection is just the iucvconn password that was entered before:

```
# ssh linux4@lnxadmin.itso.ibm.com
login as: linux4
Using keyboard-interactive authentication.
Password:
iucvconn_on_login: Connecting to linux4 (terminal ID: lnxhvc0)
```

10. You are now connected to the IUCV terminal hvc0 of the z/VM guest LINUX4 and are prompted for a login. Note that no network connection to that guest is needed at this time.
11. To terminate the connection, a special key sequence is required. The default for this is the key combination control underscore followed by a dot (Ctrl+Shift+_.).

9.8 Issue z/VM CP commands from Linux

The **vmcp** command enables z/VM central processor (CP) commands to be issued from Linux. Here are a few examples:

```
# vmcp query v dasd
DASD 0100 3390 VV1569 R/W      10016 CYL ON DASD  1569 SUBCHANNEL = 0000
DASD 0120 3390 VV1560 R/O        140 CYL ON DASD  1560 SUBCHANNEL = 000D
DASD 0190 3390 VV1560 R/O        214 CYL ON DASD  1560 SUBCHANNEL = 0006
DASD 0191 3390 VV1560 R/O        500 CYL ON DASD  1560 SUBCHANNEL = 0009
DASD 019D 3390 VV1560 R/O        292 CYL ON DASD  1560 SUBCHANNEL = 0007
DASD 019E 3390 VV1560 R/O        500 CYL ON DASD  1560 SUBCHANNEL = 0008
DASD 0200 3390 VV156B R/W      10016 CYL ON DASD  156B SUBCHANNEL = 0001
DASD 0300 9336 (VDSK) R/W     524288 BLK ON DASD  VDSK SUBCHANNEL = 000E
DASD 0301 9336 (VDSK) R/W    1048576 BLK ON DASD  VDSK SUBCHANNEL = 000F

# vmcp query names
LINUX3   - SSI , LINUX4   - SSI , DIRMSAT2 - SSI
DATAMOVE - DSC , MAINT630 -L0006, LNXSERV1 - DSC , DIRMAINT - DSC
FTPSERVE - DSC , SSLSERVE - DSC , TCPIP    - DSC , ZHCP     - DSC
XCAT     - DSC , VSMEVSRV - DSC , VSMREQIU - DSC , VSMREQI6 - DSC
VSMREQIN - DSC , DTCSMAPI - DSC , PERSMAPI - DSC , VSMWORK3 - DSC
VSMWORK2 - DSC , VSMWORK1 - DSC , VSMGUARD - DSC , LNXADMIN -L0007
DTCVSW2  - DSC , DTCVSW1  - DSC , VMSERVP  - DSC , VMSERVR  - DSC
VMSERVU  - DSC , VMSERVS  - DSC , OPERSYMP - DSC , DISKACNT - DSC
EREP     - DSC , OPERATOR - DSC , MAINT    -L0005, LINUX2   -L0009
VSM     - TCPIP

# vmcp query v storage
STORAGE = 1G MAX = 2G INC = 2M STANDBY = 1G RESERVED = 0

# vmcp query vswitch details
VSWITCH SYSTEM VSW1     Type: QDIO    Connected: 3    Maxconn: INFINITE
  PERSISTENT  RESTRICTED    ETHERNET                  Accounting: OFF
  USERBASED
  VLAN Unaware
  MAC address: 02-00-0A-00-00-01    MAC Protection: Unspecified
...
```

More information about CP commands can be found in the z/VM v6.3 CP Commands and Utilities Reference:

http://www.vm.ibm.com/library/zvmpdf.html

9.9 Access z/VM CMS disks from Linux

Data on z/VM CMS disk can be accessed using the CMS file system tools. Alternatively the CMS disk can be mounted using `cmsfs-fuse`.

9.9.1 Use the CMS file system tools

To use the CMS file system tools, the following package needs to be installed:

```
SLES: # zypper install cmsfs
```

Perform the following steps:

1. Set the device with the CMS file system online:

```
# cio_ignore -r 0.0.0120
# chccwdev -e 0.0.0120
Setting device 0.0.0190 online
Done
# lsdasd 0.0.0120
Bus-ID     Status      Name      Device  Type  BlkSz  Size      Blocks
==============================================================================
0.0.0120   active(ro)  dasde     94:16   ECKD  4096   98MB      25200
```

2. Print the table of contents of that CMS disk:

```
# cmsfslst | head
FILENAME FILETYPE FM FORMAT LRECL       RECS     BLOCKS     DATE     TIME
         DIRECTOR P0 F         64       1134          1  4/23/2015 11:11:19
         ALLOCMAP P0 F       4096          3          3  4/23/2015 11:11:19
$RACLIS$ XEDIT    Z2 V        248        585          5 11/04/2005 14:19:43
$RACLIS$ XEDITPWD Z2 V        248        557          5 11/03/2005 10:04:35
$RACPER$ XEDIT    Z2 V        193        685          6  4/25/2013  9:00:51
$RACPER$ XEDITPWD Z2 V        200        629          5 11/03/2005 10:04:35
__CPL    H        Z2 V         71         83          1  9/23/2011 13:44:15
__FTP    H        Z2 V         66         72          1  9/23/2011 13:44:16
__GETIPC H        Z2 V         72        517          7  9/23/2011 13:44:16
```

3. Show the content of a file:

```
# cmsfscat -d /dev/dasdd -a __CPL.H | head
                   ??=ifndef __sys_cpl
                   ??=ifdef __COMPILER_VER__
                     ??=pragma filetag("IBM-1047")
                   ??=endif
                   #define __sys_cpl 1
                   #pragma nomargins nosequence
                   #pragma checkout(suspend)
  /***************************************************************
  * <sys/__cpl.h> header file                                     *
  *                                                               *
```

4. Set the device with the CMS file system offline:

```
# chccwdev -d 0.0.0120
Setting device 0.0.0120 offline
Done
```

9.9.2 Mount a CMS disk using cmsfs-fuse

To mount the CMS file system tools, the following packages need to be installed:

```
SLES: # zypper install fuse s390-tools
```

Complete the following steps:

1. Set the device with the CMS file system online:

```
# cio_ignore -r 0.0.0190
# chccwdev -e 0.0.0190
Setting device 0.0.0190 online
Done
# lsdasd 0.0.0120
Bus-ID     Status      Name      Device  Type  BlkSz  Size      Blocks
==============================================================================
0.0.0190   active      dasdc     94:8    ECKD  4096   150MB     38520
```

2. Mount the CMS disk to /mnt:

```
# cmsfs-fuse -a -o ro /dev/disk/by-path/ccw-0.0.0190 /mnt
```

3. Access data:

```
# ls /mnt/EDIT.EXEC
/mnt/EDIT.EXEC

# tail -n5 /mnt/EDIT.EXEC
&LOOP 3 &K
&IF &&I = LRECL  &&I = WIDTH
&IF &&I = NODISP &&I = NOSCREEN
&I = &I + 1
&GOTO -GO
```

4. Unmount the CMS disk:

```
# fusermount -u /mnt
```

5. Set the device with the CMS file system offline:

```
# chccwdev -d 0.0.0190
Setting device 0.0.0190 offline
Done
```

9.10 NFS mounting the LNXADMIN SFS directory from Linux

Although this would seldom be something that you might typically do, the authors of this book are including it for reference. If you do take advantage of this, it is advised that you unmount as soon as it is no longer required so that you do not affect Shared File System (SFS) performance by having a litany of unnecessary Network File System (NFS) mounts open to the file pool, which could affect performance:

```
itsovm1.itso.ibm.com:lnx:lnxadmin,lines=nl,trans=yes,userid=linux1,password=XXXXXX
XX /mnt/vmlnx/ nfs noauto,rsize=8192,wsize=8192,nosuid,timeo=14,soft,intr 0 0
```

9.11 Printing from z/VM

There might be times where you want to print from z/VM direct to a network-attached printer. The following site provides a wealth of information about how to accomplish this:

http://**ibm.com**/vm/printing

9.12 Install a package from the z/VM Download Library

The VM Download Library is a clearinghouse or repository for tools, documentation, and other nifty gadgets of interest specifically for VMers (served to you by a z/VM web server running on CMS).

The IBM z/VM platform development team in Endicott has set up the library so that both IBMers and non-IBMers can submit content and so that anyone can download content. The library contains a huge number of items, which you are encouraged to explore. IBM provides the Library contents on an "AS IS" basis and might not have reviewed the Library contents, so you must use your discretion to determine whether a package is appropriate to use in your environment.

Should it be that you are interested in downloading a package, this section provides you the information about how to do so from z/VM itself. As you become more experienced with the z/VM platform, you might find that you wind up creating some tools and utilities yourself, which you are encouraged to share.

Note: Before you download anything from the VM Download Library, you must read the license agreement for downloads to ensure that you are able to comply with the terms:

http://**ibm.com**/vm/download/license.html

9.12.1 Use the CMS web browser

To use the CMS web browser, called Charlotte, it must be reblocked, then unpacked. In *The Virtualization Cookbook for IBM z Systems Volume 1: IBM z/VM 6.3*, SG24-8147, in the section entitled "Copy the utilities and REXX EXECs to the cluster-wide SFS pool", the VMARC was placed at `VMPSFS:VMWW2`. Complete the following steps:

1. Log on as either `MAINT` or `MAINT630` on the first node in the cluster.
2. Access `VMPSFS:VMWW2` as W read/write and `VMPSFS:MAINT630.UTILS.VMARC` as V:

```
===> access vmpsfs:vmww2. W (forcerw
===> access vmpsfs:maint630.utils.vmarc V
```

3. Run the VMWW2 VMARC file through this pipeline:

```
===> pipe < vmww2 vmarc W | fblock 80 00 | > vmww2 vmarc W f 80
Ready;
```

4. Unpack the contents:

```
===> VMARC unpack vmww2 W = = W
```

5. You can now use these grant commands to grant access to the browser for all virtual machines:

```
===> grant auth vmpsfs:vmww2. to public ( read newread
===> grant auth * * vmpsfs:vmww2. to public ( read
```

   Alternatively, you can substitute individual users in place of `public`:

```
===> grant auth vmpsfs:vmww2. to FMIRANDA ( read newread
```

   Note: The reason that we did not put this into an SFS directory underneath `MAINT` or `MAINT630` is because *you should not be accessing the web from a master system management virtual machine* such as those. Use your own class G user ID.

6. You can now use Charlotte by issuing this command from any virtual machine you have granted authorization to:

```
===> VMLINK .DIR VMPSFS:VMWW2. < . Z-A * > (INVOKE WW2
```

9.13 Manually formatting DASD for use

From time to time you need to add a new DASD to the system for additional paging, spool, or otherwise. During the initial setup of z/VM, it was mentioned that certain types of DASD require formatting parameters that determine ownership of the volume:

- This example covers the formatting of three additional DASDs:

```
===> query 30D0-30D2
DASD 30D0 VM30D0  , DASD 30D1 VM30D1  , DASD 30D2 VM30D2
===> attach 30D0-30D2 to *
30D0-30D2 ATTACHED TO MAINT630
```

- Allocate `30D0` to `ITSOZVM2` as a SPOOL volume:

```
===> cpfmtxa 30D0
ENTER FORMAT, ALLOCATE, LABEL, OWNER OR QUIT:
format
ENTER THE CYLINDER RANGE TO BE FORMATTED ON DISK 30D0 OR QUIT:
0-END
ENTER THE VOLUME LABEL FOR DISK 30D0:
VP30D0
CPFMTXA:
FORMAT WILL ERASE CYLINDERS 000000000-000003338 ON DISK 30D0
DO YOU WANT TO CONTINUE? (YES | NO)
yes
HCPCCF6209I INVOKING ICKDSF.
ICK030E DEFINE INPUT  DEVICE: FN FT FM, "CONSOLE", OR "READER"
CONSOLE
ICK031E DEFINE OUTPUT DEVICE: FN FT FM, "CONSOLE", OR "PRINTER"
CONSOLE
ICKDSF - CMS/XA/ESA DEVICE SUPPORT FACILITIES 17.0                TIME:
19:37:30        05/21/15     PAGE   1

ENTER INPUT COMMAND:
  CPVOL FMT MODE(ESA) UNIT(30D0) VOLID(VP30D0) NOVFY NFILL -
ENTER INPUT COMMAND:
 RANGE(0,3338)
ICK00700I DEVICE INFORMATION FOR 30D0 IS CURRENTLY AS FOLLOWS:
```

```
          PHYSICAL DEVICE = 3390
          STORAGE CONTROLLER = 3990
          STORAGE CONTROL DESCRIPTOR = E9
          DEVICE DESCRIPTOR = 0A
          ADDITIONAL DEVICE INFORMATION = 4A001F3C
          TRKS/CYL = 15, # PRIMARY CYLS = 3339
ICK04000I DEVICE IS IN SIMPLEX STATE
ICK00091I 30D0 NED=002107.900.IBM.75.0000000AKAZ1
ICK091I   30D0 NED=002107.900.IBM.75.0000000AKAZ1
ICK03020I CPVOL WILL PROCESS 30D0 FOR VM/ESA MODE
ICK03090I VOLUME SERIAL = VS30D0
ICK03022I FORMATTING THE DEVICE WITHOUT FILLER RECORDS
ICK03011I CYLINDER RANGE TO BE FORMATTED IS 0 - 3338
ICK003D REPLY U TO ALTER VOLUME 30D0 CONTENTS, ELSE T
U
ICK03000I CPVOL REPORT FOR 30D0 FOLLOWS:

FORMATTING OF CYLINDER 0 STARTED AT: 19:37:30
         FORMATTING OF CYLINDER 100 ENDED AT: 19:37:30
         FORMATTING OF CYLINDER 200 ENDED AT: 19:37:31
         FORMATTING OF CYLINDER 300 ENDED AT: 19:37:32
         FORMATTING OF CYLINDER 400 ENDED AT: 19:37:32
         FORMATTING OF CYLINDER 500 ENDED AT: 19:37:33
...

          VOLUME SERIAL NUMBER IS NOW = VP30D0

          CYLINDER ALLOCATION CURRENTLY IS AS FOLLOWS:
          TYPE     START      END        TOTAL
          ----     -----      ---        -----
PERM     0          3338       3339

ICK00001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
          19:38:49    05/21/15

ENTER INPUT COMMAND:
 END

ICK00002I ICKDSF PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
ENTER ALLOCATION DATA
TYPE CYLINDERS
.................
SPOL 0 END
END
HCPCCF6209I INVOKING ICKDSF.
ICK030E DEFINE INPUT  DEVICE: FN FT FM, "CONSOLE", OR "READER"
CONSOLE
ICK031E DEFINE OUTPUT DEVICE: FN FT FM, "CONSOLE", OR "PRINTER"
CONSOLE
ICKDSF - CMS/XA/ESA DEVICE SUPPORT FACILITIES 17.0 ...

ENTER INPUT COMMAND:
  CPVOL ALLOC MODE(ESA) UNIT(30D0) VFY(VP30D0) -
ENTER INPUT COMMAND:
        TYPE((SPOL,0,3338))
```

```
ICK00700I DEVICE INFORMATION FOR 30D0 IS CURRENTLY AS FOLLOWS:
          PHYSICAL DEVICE = 3390
          STORAGE CONTROLLER = 3990
          STORAGE CONTROL DESCRIPTOR = E9
          DEVICE DESCRIPTOR = 0A
          ADDITIONAL DEVICE INFORMATION = 4A001F3C
          TRKS/CYL = 15, # PRIMARY CYLS = 3339
ICK04000I DEVICE IS IN SIMPLEX STATE
ICK00091I 30D0 NED=002107.900.IBM.75.0000000AKAZ1
ICK091I   30D0 NED=002107.900.IBM.75.0000000AKAZ1
ICK03020I CPVOL WILL PROCESS 30D0 FOR VM/ESA MODE
ICK03090I VOLUME SERIAL = VP30D0
ICK03024I DEVICE IS CURRENTLY FORMATTED WITHOUT FILLER RECORDS
ICK003D REPLY U TO ALTER VOLUME 30D0 CONTENTS, ELSE T
U
ICK03000I CPVOL REPORT FOR 30D0 FOLLOWS:

          CYLINDER ALLOCATION CURRENTLY IS AS FOLLOWS:
          TYPE     START      END        TOTAL
          ----     -----      ---        -----
          SPOL     0          3338       3339

ICK00001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
          19:39:55    05/21/15

ENTER INPUT COMMAND:
 END

ICK00002I ICKDSF PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0

===> cpfmtxa 30D0
ENTER FORMAT, ALLOCATE, LABEL, OWNER OR QUIT:
owner
```

Part 3

Appendixes

This section consists of the following appendixes:

A

References, cheat sheets, and blank worksheets

This appendix refers to additional materials included for your reference, which can either be printed or downloaded from the Internet as described in the following sections.

Important IBM z/VM files

IBM z/VM differs from Linux in regard to the location and number of configuration files. In Linux, there are many configuration files, and most of them are in or under the `/etc/` directory. On z/VM, there are relatively few configuration files. However, they are on many different minidisks. Table 9-1 provides a summary and the location of important z/VM configuration files.

Table 9-1 Important z/VM configuration files

File	Location	Description
`SYSTEM CONFIG`	`PMAINT CF0`	This is the operating system's main configuration file. It defines the system name, the CP volumes, user volumes, and other settings.
`USER DIRECT`	`MAINT 2CC`	This is the initial z/VM user directory. All virtual machines known to the system are defined here. If a directory maintenance product is in use, this file is no longer authoritative.
`PROFILE TCPIP`	`TCPMAINT 198`	This file defines the resources for the primary z/VM Transmission Control Protocol/Internet Protocol (TCP/IP) stack, including TCP/IP address, Open Systems Adapter (OSA) resources, subnet mask, and gateway. It is initially created by the IPWIZARD tool as `PROFILE TCPIP`.
`SYSTEM DTCPARMS`	`TCPMAINT 198`	This file is created to define the TCP/IP stacks on the system. It is initially created by the IPWIZARD tool.
`TCPIP DATA`	`TCPMAINT 592`	This file defines the Domain Name System (DNS) server, the domain name, and some other settings. It is initially created by the IPWIZARD tool.
`PROFILE EXEC`	`AUTOLOG1 191`	This file is a Restructured Extended Executor (REXX) EXEC that is run when the system starts. It is analogous to the `/etc/inittab` file in Linux.

Cheat sheets

This section contains quick references or “cheat sheets” for the XEDIT and vi editors.

XEDIT cheat sheet

`XEDIT` has line commands that are typed on the command line (====>) and prefix commands, which are typed over the line numbers on the left side of the window.

Line commands

In all of these items, curly brackets { } are used to indicate variables. Do not include the { } in your commands:

```
a                 Add a line
a{n}              Add {n} lines
c/{old}/{new}/{n} {m}
                  Search for string {old} and replace it with {new} for {n} lines
below the current line and {m} times on each line. * can be used for {n} and {m}
/<string>         Search for ‘string’ from the current line
-/<string>        Search backwards for ‘string’
all /<string>/    Show all occurrences of ‘string’ and hide other lines
bottom            Move to the bottom of the file
top               Move to the top of the file
down <n>          Move down ‘n’ lines
up <n>            Move up ‘n’ lines
file              Save the current file and exit XEDIT
ffile             Save the current file and exit but don’t warn of overwrite
save              Save the current file but dont exit
quit              Exit XEDIT if no changes have been made
qquit             Exit XEIDT even if changes have not been saved
left <n>          Shift <n> characters to the left
right <n>         Shift <n> characters to the right
get <file>        Copy file and insert past the current line
input             Enable INPUT mode to insert multiple lines of text beginning at
the current line
:<n>              Move to line ‘n’
?                 Display last command
=                 Execute last command
x <file>          Edit ‘file’ and put it into the XEDIT “ring”
x                 Move to the next file in the ring
```

Prefix commands

The following list describes prefix commands:

```
a                 Add one line
a<n>              Add 'n' lines
c                 Copies one line
cc                Copies a block of lines
d                 Deletes one line
dd                Deletes a block of lines
f                 Line after which a copy (c) or a move (m) is to be inserted
p                 Line before which a copy (c) or a move (m) is to be inserted
i                 Insert a line
i<n>              Insert 'n' lines
m                 Move one line
mm                Move a block of lines
"                 Replicate a line
"<n>              Replicate a line 'n' times
""                Replicate a block of lines
```

A vi cheat sheet

The following section provides only a small subset of `vi` commands, but they are those most commonly used. The vi editor has three modes:

- Input mode. The **Insert** key, **i**, **o** (add a line below), **O** (add a line above), and other commands put you in this mode where you can type text into the file. When you are in this mode, you see the text `--INSERT--` in the last line.
- Command mode. The Esc key gets you out of input mode and into command mode. You can issue the following commands:

`i`	Brings you back to input mode
`dd`	Deletes a line and puts it in the buffer
`<n>dd`	Delete <n> lines
`x`	Delete a character
`dw`	Delete a word
`p`	Add the buffer past the current location
`P`	Add the buffer before the current location
`o`	Add a line and go into insert mode
`/<string>`	Search for ‘string’
`n`	Do the last command again (this can be powerful)
`jkl;`	Cursor movement
`A`	Add text at the end of the line
`<nn>G`	Go to line <nn>
`G`	Go to the last line in the file
`yy`	Yank a line (copy into buffer)
`<n>yy`	Yank <n> lines

- Command-line mode. Pressing the colon (`:`) key brings you to this mode at the bottom of the window. You can issue the following commands:

`:wq`	Save (write & quit)
`:q!`	Quit and discard changes
`:<nn>`	Go to line number <nn>
`:r <file>`	Read <file> into the current file
`:1,$s/old/new/g`	Globally replace <old> with <new>
`:help`	Give help

IBM DirMaint cheat sheet

The following list describes IBM DirMaint™ commands:

`Add`	Add a new user or profile directory entry
`AMDisk`	Adds a new minidisk
`DEDicate`	Add or delete an existing dedicate statements
`DMDisk`	Removes a minidisk
`FILE`	Add or replace a DirMaint control file
`RLDCode`	Reload DirMaint resident operating procedures
`RLDExtn`	Reload DirMaint CONFIG* DATADVH file
`REView`	Review a user or profile directory entry
`MDisk`	Change the access mode and passwords for minidisks
`STorage`	Change logon storage size
`SEND`	Request a copy of a DirMaint control file
`SETOptn`	Add, change, or delete CP options
`CLAss`	Change the CP class for a directory entry
`SPEcial`	Add or delete an existing special statement

DirMaint example commands

The following list provides examples of DirMaint commands:

- Add a new 50 cylinder minidisk 200 to user ID `spiedie`:

  ```
  DIRMAINT FORUSER SPIEDIE AMDISK 200 3390 AUTOG 50 {VOLGROUP}
  ```

- Add a link statement to TCPMAINT's 592 minidisk into the directory entry for user `vmfrau`:

  ```
  DIRMAINT FORUSER VMFRAU LINK TCPMAINT 0592 0592 RR
  ```

Blank planning worksheet

This section contains a blank copy of the planning worksheet used in *The Virtualization Cookbook for IBM z Systems Volume 1: IBM z/VM 6.3*, SG24-8147 in the section entitled "Planning for VMSSI with LGR".

This worksheet is included for your convenience, and is organized to be in the order that you will likely need the data.

It is suggested that you specify all values that will apply, to make your installation process go more smoothly.

IBM Shopz

If you are ordering z/VM using *Shopz*, as described in *The Virtualization Cookbook for IBM z Systems Volume 1: IBM z/VM 6.3*, SG24-8147 in the section entitled "Obtain z/VM through electronic download", use Table 9-2 to record the values you will use.

Table 9-2 Shopz data

Name	Value	Comment
Starting URL	`ibm.com/shopz`	
User ID		Customer number (for IBM employees, it is your intranet user ID and password)
Password		
Order number		

Hardware Management Console

The Virtualization Cookbook for IBM z Systems Volume 1: IBM z/VM 6.3, SG24-8147 section "Start the z/VM installation" describes how to start a z/VM installation from the Hardware Management Console (HMC). Complete Table 9-3 to record the values that you will use.

Table 9-3 HMC values

Name	Value
HMC host name or Uniform Resource Locator (URL)	
HMC user ID	
HMC password	
File Transfer Protocol (FTP) source system (If installing from FTP)	
z/VM installation directory	

z/VM Installation Planning Panels (INSTPLAN)

The Virtualization Cookbook for IBM z Systems Volume 1: IBM z/VM 6.3, SG24-8147 section "Copy a vanilla z/VM system to DASD", describes the **`INSTPLAN`** command run from the Integrated 3270 Console. The following information is necessary.

INSTPLAN panels 1 and 2

Complete Table 9-4 to record the values required in the first two **`INSTPLAN`** panels.

Table 9-4 INSTPLAN values for first two panels

Name	Value	Comment
Language	☐ AMENG ☐ UCENG ☐ KANJI	AMENG (American English), UCENG (uppercase English), or KANJI
DASD model	☐ 3390 Model-3 ☐ 3390 Model-9	3390 Model-3 or Model-9 (Installation to FBA disk is not described in this book)
File pool name		VMPSYS (default) suggested
System type		SSI (Non-SSI is not described in this book)
Non-SSI system name		Used for non-SSI installation only
Number of members		SSI installation only (usually 2 or 4)
SSI cluster name		SSI installation only
Automatic configuration		"No" is strongly advised

INSTPLAN panel 3

Complete Table 9-5 to record the values required in the third INSTPLAN panel. The member names become the z/VM system identifiers, and the logical partition (LPAR) names should be the same names as on the HMC.

Table 9-5 INSTPLAN values for panel 3

Slot	Member name	LPAR name	Comment
1			Member 1 system identifier and LPAR name
2			Member 2 system identifier and LPAR name
3			Member 3 system ID and LPAR name (optional)
4			Member 4 system ID and LPAR name (optional)

INSTPLAN worksheet 3

Complete Table 9-6 to record the volume labels and real device addresses required in the *Installation Volume Definition* `INSTPLAN` panel.

Table 9-6 INSTPLAN values worksheet for volume definition

Type	Default Label	Chosen Label	Address	Comment
COMMON	VMCOM1			Common volume 1
COMMON2	VMCOM2			Common volume 2
RELVOL	630RL1			Release volume 1
RELVOL2	630RL2			Release volume 2
Mem 1 RES	M01R01			Member 1 residence volume
Mem 1 SPOOL	M01S01			Member 1 spool volume
Mem 1 PAGE	M01P01			Member 1 page volume
Mem 1 WORK	M01W01			Member 1 work volume 1
Mem 1 WORK	M01W02			Member 1 work vol 2 (3390-3 only)
Mem 1 WORK	M01W03			Member 1 work vol 3 (3390-3 only)
Mem 2 RES				Member 2 residence volume
Mem 2 SPOOL				Member 2 spool volume
Mem 2 PAGE				Member 2 page volume
Mem 2 WORK				Member 2 work volume 1
Mem 2 WORK				Member 2 work vol 2 (3390-3 only)
Mem 2 WORK				Member 2 work vol 3 (3390-3 only)
Mem 3 RES				Member 3 residence vol (optional)
Mem 3 SPOOL				Member 3 spool volume
Mem 3 PAGE				Member 3 page volume
Mem 3 WORK				Member 3 work volume 1

Type	Default Label	Chosen Label	Address	Comment
Mem 3 WORK				Member 3 work vol 2 (3390-3 only)
Mem 3 WORK				Member 3 work vol 3 (3390-3 only)
Mem 4 RES				Member 4 residence vol (optional)
Mem 4 SPOOL				Member 4 spool volume
Mem 4 PAGE				Member 4 page volume
Mem 4 WORK				Member 4 work volume 1
Mem 4 WORK				Member 4 work vol 2 (3390-3 only)
Mem 4 WORK				Member 4 work vol 3 (3390-3 only)

INSTPLAN worksheet 4

Complete the following table to record the common volume and CTC addresses required in the **INSTPLAN** panel.

Complete the worksheet in Table 9-7 to document **INSTPLAN** values for volume definition. This panel is shown in *The Virtualization Cookbook for IBM z Systems Volume 1: IBM z/VM 6.3*, SG24-8147 section at the end of section “Copy a vanilla z/VM system to DASD”.

If you have only two members in the SSI, you need to specify only two pairs of CTCs (from member 1 to member 2, and vice versa)

Table 9-7 INSTPLAN values worksheet for volume definition

Real addresses for the common volume on each member LPAR:			
Member 1	**Member 2**	**Member 3**	**Member 4**
CTC device addresses:			
From member 1		**From member 2**	
To: member 1	N/A	To: member 1	_______ _______
To: member 2	_______ _______	To: member 2	N/A
To: member 3	_______ _______	To: member 3	_______ _______
To: member 4	_______ _______	To: member 4	_______ _______
From member 3		**From member 4**	
To: member 1	_______ _______	To: member 1	_______ _______
To: member 2	_______ _______	To: member 2	_______ _______
To: member 3	N/A	To: member 3	_______ _______
To: member 4	_______ _______	To: member 4	N/A

z/VM Networking resources

Complete the worksheet in Table 9-8 to list the networking resources that will be needed when invoking the `IPWIZARD`, and when creating a VSWITCH for the Linux virtual machines.

Table 9-8 z/VM and networking resources worksheet

Name	Value	Comment
TCP/IP user ID		TCP/IP is recommended
z/VM host name, member 1		
z/VM host name, member 2		
TCP/IP domain name		System domain name usually set in DNS
TCP/IP gateway		The router to and from the local subnet
DNS server 1		Assigned by the network administrator
DNS server 2/3		Optional
Interface name		
OSA starting device number		Start of OSA *triplet* for z/VM TCP/IP stack
Subnet mask		Assigned by network administrator
OSA device type		
Maximum transmission unit (MTU) size		Check with network administrator
Primary OSA device for virtual switch		Specify the first real device number and the next two device numbers will also be used
Secondary OSA device for virtual switch		Ideally, it should be on a different channel-path identifier (CHPID) or OSA card

z/VM DASD worksheet

Use the worksheet in Table 9-9 to document the z/VM DASD that you will use.

Table 9-9 z/VM DASD blank worksheet

Device number	Label	Type	Notes

Linux resources worksheet

Use the worksheet in Table 9-10 to document the resources associated with the NFS server that will be used to be the installation source of the first Linux on z Systems.

Table 9-10 Linux NFS server resources blank worksheet

Name	Value	Comment
TCP/IP address		
User/password		
NFS-exported installation directory		

Use the worksheet in Table 9-11 to document your Linux on z Systems resources.

Table 9-11 Linux resources blank worksheet

Name	Value	Comment
Linux installation password		
Linux root password		
Linux TCP/IP gateway		
Linux TCP/IP broadcast		
Linux DNS server		
VNC installation password		

Host names and IP addresses worksheet

Use the worksheet in Table 9-12 to document the host names and associated IP addresses and virtual machines that you will use.

Table 9-12 Host names blank worksheet

Host name	IP address	Virtual machine/ LPAR	Notes

B

Additional material

This book refers to additional material that can be downloaded from the Internet.

Locating the web material

The web material associated with this book is available on the Internet. You can find this at the following URL:

http://www.ibm.com/vm/pubs/redbooks/sg248147/files/24814701.tgz

Using the web material

The files associated with this book are in a GNU-compressed tape archive (TAR) file.

The additional web materials that accompany this book are in the following file:

File name	Description
`24814701.tgz`	Code samples in compressed `.tar` format

System requirements for downloading the web material

The web material requires the following system configuration:

Hard disk space	41 kilobytes (KB)
Operating System	Linux

Downloading and extracting the web material

This section lists code associated with this book. The following sections are included:

- “z/VM REXX EXECs and XEDIT macros” on page 152
- “Sample files” on page 164
- “Linux code” on page 164

z/VM REXX EXECs and XEDIT macros

This section lists all of the z/VM code included in the associated TAR file:

- The CPFORMAT EXEC
- The SSICMD EXEC
- PROFILE EXEC for Linux virtual machines
- The SLES12 EXEC
- The SLES12 EXEC

The CPFORMAT EXEC

The following section contains the code for the EXEC that formats multiple DASD using **CPFMTXA**. It is described in in *The Virtualization Cookbook for IBM z Systems Volume 1: IBM z/VM 6.3,* SG24-8147 in the section entitled “Enable basic system automation”:

```
/***********************************************************/
/*                                                         */
/*  This program is provided on an "AS IS" basis, without    */
/*  warranties or conditions of any kind, either express or  */
/*  implied including, without limitation, any warranties    */
/*  or conditions of title, non-infringement,                */
/*  merchantability or fitness for a particular purpose.     */
/*  Neither recipient nor any contributors shall have any    */
/*  liability for any direct, indirect, incidental,          */
/*  special, exemplary, or consequential damages (including  */
/*  without limitation lost profits), however caused and on  */
/*  any theory of liability, whether in contract, strict     */
/*  liability, or tort (including negligence or otherwise)   */
/*  arising in any way out of the use or distribution of     */
/*  the program or the exercise of any rights granted        */
/*  hereunder, even if advised of the possibility of such    */
/*  damages.                                                 */
/*                                                         */
/***********************************************************/
/*                                                         */
/* Purpose:                                                  */
/*   CP format one, a range or multiple ranges of DASD.      */
/*   and label these DASDs.                                  */
/*                                                         */
/* Inputs:                                                   */
/*   dasds - address(es) of the DASD to format.              */
/*   type  - type of formatting to be done: PERM, PAGE, SPOL */
/*           or TEMP.                                        */
/*                                                         */
/* Output:                                                   */
/*   Virtual DASD that is CP formatted and labeled.          */
/*                                                         */
/* Return codes:                                             */
/*   0 - success                                             */
/*   1 - help was asked for or given                         */
/*   2 - user did not respond Y to confirm formatting        */
/*   3 - DASD (minidisk) range is not valid                  */
/*   4 - at least one DASD (minidisk) is reserved to MAINT   */
/*                                                         */
/*                                                           */
/*************************************************************/
  Address COMMAND
  firstchar = 'J'
  Arg dasds 'AS ' type .
  If dasds = '' | dasds = '?' Then Call help
  labelPrefix = firstchar || getLabelPrefix(type)
  numDasd = parseDasd(dasds)
  answer = areYouSure(type)
  If answer = 'Y' Then Do
     /* the user is sure */
     formatted = ''
     retVal = doFormat(labelPrefix numDasd type)
     Call doReport retVal
     End
  Else retVal = 2
  Exit retVal

/*+------------------------------------------------------------------+*/
help:
  Procedure Expose firstchar
/*+------------------------------------------------------------------+*/
  Parse Source . . fn .
  Say
  Say 'Synopsis:'
  Say
  Say '  Format and label DASD as page, perm, spool or temp disk space'
  Say '  The label written to each DASD is' firstchar || '<t><xxxx> where:'
  Say '    <t> is type - P (page), M (perm), S (spool) or T (Temp disk)'
  Say '    <xxxx> is the 4 digit address'
  Say
  Say 'Syntax is:'
  Say "                <---------------<              "
  Say "   >>--CPFORMAT--.-vdev--------.--AS---.-PERM-.---------><"
  Say "                 '-vdev1-vdev2-'       '-PAGE-'"
  Say "                                       '-SPOL-'"
  Say "                                       '-TEMP-'"
  Say
  Exit 1

/*+------------------------------------------------------------------+*/
areYouSure:
  Procedure
/*| Warn the user of possible data loss and ask if it is okay to     |*/
/*|  format the DASD.                                                |*/
/*|  parm 1: format type for the virtual DASD                        |*/
/*|  retVal: first character of response. continue if 'Y'.           |*/
/*+------------------------------------------------------------------+*/
  Arg type
  Say
  Say 'WARNING - this will destroy data!'
  Say 'Are you sure you want to format the DASD as' type 'space (y/n)?'
  Pull answer .
  Return  'LEFT'(answer,1) /* from areYouSure */
```

```
/*+------------------------------------------------------------------+*/
getLabelPrefix:
  Procedure expose firstchar
/*| Return the second character of the virtual DASD label            |*/
/*|  parm 1: format type for the virtual DASD                        |*/
/*+------------------------------------------------------------------+*/
  Arg type .
  firstchar. = 0
  firstchar.PERM = 'M'
  firstchar.PAGE = 'P'
  firstchar.SPOL = 'S'
  firstchar.TEMP = 'T'
  If firstchar.type = 0 Then Do
  /* Incorrect formatting type specified. Provide help and quit. */
     Say 'Error: "AS" must be present, type must be PERM, PAGE, SPOL or TEMP'
     Call help
     End
  Return firstchar.type

/*+------------------------------------------------------------------+*/
parseDASD:
  Procedure Expose dasdList.
/*| parse all dasd into an array verifying all are attached          |*/
/*|  parm 1: dasds - the list of dasd passed in                      |*/
/*|  retVal: number of DASD in dasdList                              |*/
/*+------------------------------------------------------------------+*/
  Arg dasds
  numDasd = 0
  dropheader = ''
  Say
  Say 'Format the following DASD:'
  Do While dasds <> ''
     Parse Upper Var dasds dasd dasds
     dashPos = 'POS'('-',dasd)
     If dashPos = 0 Then Do
     /* There is a singleton DASD specified. */
     /* start and end of range are the same. */
        startrange = dasd
        endrange = dasd
        End
     /* process the range of DASD */
     Else Parse Var dasd startrange  '-' endrange
     Do i = 'X2D'(startrange) To 'X2D'(endrange)
        numDasd = numDasd + 1
        dasdList.numDasd = 'D2X'(i)
        'PIPE CP QUERY MDISK' dasdList.numDasd 'LOCATION',
        dropheader,
        '|CONS'
        If rc <> 0 Then Do
           Say 'Return code from QUERY MDISK =' rc
           /* If RC=40, then HCPxxx40E has been issued and msg below */
           If rc = 40 Then Say 'DASD' dasdList.numDasd 'is not attached.'
           Exit 3
           End
        Call checkReserved(dasdList.numDasd)
        dropheader = '|DROP 1'
        End
     End
  Return numDasd /* from parseDasd */

/*+------------------------------------------------------------------+*/
doFormat:
  Procedure Expose dasdList. formatted
/*| Format all DASD specified using CPFMTXA                          |*/
/*|  parm 1: labelPrefix - the two character label prefix            |*/
/*|  parm 2: numDasd - number of DASD in the array dasdList          |*/
/*|  parm 3: type - the type of DASD format                          |*/
/*|  retVal: 0 = success                                              |*/
/*+------------------------------------------------------------------+*/
  Arg labelPrefix numDasd type
  /* Save the current settings for MORE */
  Parse Value 'DIAG'('08','CP QUERY TERM') With ' MORE' morevalues ','
  'CP TERM MORE 1 1' /* Make MORE brief */

  /* Save system identifier and SSI name */
  'PIPE CP QUERY USERID | SPEC W3 | VAR systemID'
  'PIPE CP QUERY SSI | LOCATE /SSI Name/ | SPEC W3 | VAR SSIname'
  If (SSIname = "SSINAME") Then /* variable not set */
    inSSI = 'no'
  Else
    inSSI = 'yes'

  /* Iterate through all DASD in list */
  Do i = 1 to numDasd
     label = labelPrefix  || 'RIGHT'(dasdList.i,4,'0')
     retVal = formatOne(dasdList.i type label)
     If retVal <> 0 Then Do
        Say 'Error from CPFMTXA on DASD' label 'rc =' retVal
        Leave /* error - abort this format */
        End

     /* add owner info for CP owned devices */
     If (type != 'PERM') Then /* CP owned => owner info is needed */
       If (inSSI = 'yes') Then /* add owner info */
         call addOwnerInfo(dasdList.i label SSIname systemID)
       Else
         call addOwnerInfo(dasdList.i label "NOSSI" systemID)
     formatted = formatted label
     End /* Do i = */
  'CP TERM MORE' morevalues
  Return retVal /* from doFormat */

/*+------------------------------------------------------------------+*/
checkReserved:
  Procedure
/*| Try copying an already formatted DASD Then relabelling it        |*/
/*|  parm 1: dasd - the virtual address of the DASD                  |*/
/*+------------------------------------------------------------------+*/
  Arg dasd
  /* Create a list of reserved virtual DASD addresses. */
  /* Ensure that a system minidisk is not formatted.   */
  resvd = '122 123 124 190 191 193 19D 19E 2CC 401 402 990 CF1 CF3 CFD'
  If 'POS'(resvd,dasd) <> 0 Then Do
     /* MAINT minidisk - ABORT! */
     Say 'Minidisk' dasd 'is a reserved MAINT minidisk'
     Say 'This must be formatted manually using a different vaddr.'
     Exit 4
     End /* If dasd is reserved */
  Return /* from checkReserved */

/*+------------------------------------------------------------------+*/
doReport:
  Procedure Expose dasds formatted
/*| Report on the newly labelled DASD                                 |*/
/*|  parm 1: formatSuccess - 0=all is well, non-0= a format failed   |*/
/*|  retVal: 0 = success                                              |*/
/*+------------------------------------------------------------------+*/
  Arg formatSuccess
  If formatSuccess <> 0 Then
    Say 'Error was encountered! retVal from CPFMTXA =' formatSuccess
  If formatted = '' Then
    Say 'No DASD were successfully formatted'
  Else
    Say 'DASD successfully formatted:' formatted
  'CP DETACH' dasds
  'CP ATTACH' dasds '*'
  Say
  Say 'DASD status after:'
  'CP QUERY MDISK' dasds 'LOCATION'
  Return 0 /* from doReport */

/*+------------------------------------------------------------------+*/
formatOne:
  Procedure
/*| Format a DASD via DDR                                             |*/
/*|  parm 1: disk - the vaddr to be formatted                         |*/
/*|  parm 2: type - PERM, PAGE, SPOL or TEMP                          |*/
/*|  parm 3: label - the six character label                          |*/
/*+------------------------------------------------------------------+*/
  Arg disk type label
  Queue 'FORMAT'
  Queue disk
  Queue '0 END'
  Queue label
  Queue 'YES'
  Queue type '0 END'
  Queue 'END'
  'EXEC CPFMTXA'
  retVal = rc
  Return retVal /* from formatOne */

/*+------------------------------------------------------------------+*/
AddOwnerInfo:
  Procedure
/*| Tag PAGE, SPOL and TDSK volumes with SSI                         |*/
/*|  parm 1: disk - the vaddr to be formatted                          |*/
/*|  parm 2: type - PERM, PAGE, SPOL or TEMP                            |*/
/*|  parm 3: label - the six character label                            |*/
/*+--------------------------------------------------------------------+*/
  Arg disk label SSIname systemID
  Queue 'OWNER'
  Queue disk
  Queue label
  Queue SSIname
  Queue systemID
  'EXEC CPFMTXA'
  retVal = rc
  Return retVal /* from addOwnerInfo */
```

The SSICMD EXEC

The following section contains the code for the EXEC that issues CP commands on all joined members of a subsystem interface (SSI) cluster. It is advised to be on the `MAINT 191` disk:

```
/***********************************************************/
/*                                                         */
/*  This program is provided on an "AS IS" basis, without  */
/*  warranties or conditions of any kind, either express or */
/*  implied including, without limitation, any warranties  */
/*  or conditions of title, non-infringement,              */
/*  merchantability or fitness for a particular purpose.   */
/*  Neither recipient nor any contributors shall have any  */
/*  liability for any direct, indirect, incidental,        */
/*  special, exemplary, or consequential damages (including */
/*  without limitation lost profits), however caused and on */
/*  any theory of liability, whether in contract, strict   */
/*  liability, or tort (including negligence or otherwise)  */
/*  arising in any way out of the use or distribution of   */
/*  the program or the exercise of any rights granted      */
/*  hereunder, even if advised of the possibility of such  */
/*  damages.                                               */
/*                                                         */
/***********************************************************/
/*                                                         */
/* Purpose:                                                */
/*   Issue a command on all members of a cluster using the  */
/*   response from QUERY SSI to find the member names.     */
/*                                                         */
/* Inputs:                                                 */
/*   cmd - the CP command to issue on each member.         */
/*                                                         */
/* Output:                                                 */
/*   The results from issuing the AT command.              */
/*                                                         */
/*                                                         */
/***********************************************************/
  Address COMMAND
/* The command is passed by the caller */
  Arg cmd
  /* Provide help if requested or if no command is specified */
  If cmd = '' | cmd = '?' Then Call Help
  /* Determine the members of the SSI cluster */
  'PIPE CP QUERY SSI',
  '| STEM MSG.',           /* Save the response if error */
  '| XLATE',               /* Make all output upper case */
  '| FRTARGET ALL /SLOT/', /* Just look after 'SLOT' */
  '| LOCATE /JOINED/',     /* JOINED members can do a command */
  '| SPEC W2',             /* Get the member names */
  '| STEM SSI.'            /* Save the member names */
  /* If nonzero return code, show error message and exit */
  If rc <> 0 | ssi.0 = 0 Then Do
     Say 'Error: QUERY SSI return code =' rc
     Say msg.1
     End
  Else Do
  /* Send the command to each member of the SSI cluster */
     Do i = 1 To ssi.0
        Say ssi.i||":"
        'CP AT' ssi.i 'CMD' cmd
        Say
        End
     End
  Exit

help:
  /* Provide syntax information to the user */
  Say 'SSICMD cmd'
  Say
  Say 'cmd is a command to be issued on each of the members'
  Say '  in the SSI cluster using the AT command.'
  Exit
```

PROFILE EXEC for Linux virtual machines

This section lists the code for the `PROFILE EXEC` that is shared among Linux virtual machines from the `LNXMAINT 192` disk:

```
/* PROFILE EXEC for Linux virtual servers */
'CP SET RUN ON'
'CP SET PF11 RETRIEVE FORWARD'
'CP SET PF12 RETRIEVE'
'ACC 592 C'
'SWAPGEN 300 524288' /* create a 256M VDISK disk swap space */
'SWAPGEN 301 1048576' /* create a 512M VDISK disk swap space */
'PIPE CP QUERY' userid() '| var user'
parse value user with id . dsc .
if (dsc = 'DSC') then /* user is disconnected */
  'CP IPL 100'
else  /* user is interactive -> prompt */
 do
  say 'Do you want to IPL Linux from minidisk 100? y/n'
  parse upper pull answer .
  if (answer = 'Y') then 'CP IPL 100'
 end
```

The SLES12 EXEC

This section lists the code for the `sles12 EXEC` that starts a SUSE Linux Enterprise Server 12 installation. It is advised that it be on the `LNXMAINT 192` disk:

```
/************************************************************/
/* REXX LOAD EXEC FOR SUSE LINUX S/390 VM GUESTS       */
/* LOADS SUSE LINUX S/390 FILES INTO READER            */
SAY ''
SAY 'LOADING SLES12 FILES INTO READER...'
'CP CLOSE RDR'
'PURGE RDR ALL'
'SPOOL PUNCH * RDR'
'PUNCH SLES12 LINUX    * (NOH'
'PUNCH SLES12 PARMFILE * (NOH'
'PUNCH SLES12 INITRD   * (NOH'
'IPL 00C'
```

The SWAPGEN EXEC

The following section contains the code for the EXEC that creates Linux swap spaces from z/VM VDISKs:

```
/*******************************************************************
* Program: SWAPGEN EXEC
*
* Original Author: Dave Jones (djones@sinenomine.net)
*
* Description/Purpose:
*
* Generate VDISK swap for Linux on System z guest virtual
* machines
*
* Syntax:
*
* Issue: SWAPGEN ? for syntax etc.
*
* Version History:
...
*/

   address command

   arg vdev blks . '(' options ')'

   debug = 0                                   /* Default to quiet */
   fba = 0                                    /* No FBA option yet */
   reuse = 0                                /* No reuse option yet */
   do while options <> ''                     /* Parse the options */
      parse var options option options            /* Get an option */
      select
         when option = 'DIAG' then fba = 0        /* Use DIAG driver */
         when option = 'FBA' then fba = 1          /* Use FBA driver */
         when option = 'REUSE' then reuse = 1          /* Reuse DASD */
         when option = 'DEBUG' then debug = 1    /* Wants debug chat */
         when option = 'VERSION' then signal Version /* version query*/
         otherwise
         say 'Invalid option "'option'"'            /* Else unknown */
      end
   end

   minblks = 40 - 8 * fba  /* Minimum number of blocks that can work */

   if reuse = 1 then do
      parse value diagrc(8, 'Q V 'vdev) ,     /* Get blocks from ... */
        with rc . 17 msg                     /* ... actual device size */
      if rc <> 0 then signal BadDev
      parse var msg . . . . . newblks .
      if blks = '' then blks = newblks   /* Default to detected size */
      if blks <> newblks then signal WrongBlks    /* Mismatch, error */
   end

   if vdev = '?' then signal Help            /* Wants Help, give it */
   if vdev = '' then signal NoVdev                /* Missing, error */
   if blks = '' then signal NoBlks                /* Missing, error */
   if datatype(blks, 'W') = 0 | blks < minblks then /* Bad/too small */
   signal BadBlks                                       /* So error */
   if datatype(vdev, 'X') = 0 | length(vdev) > 4 then     /* Invalid */
   signal BadVdev                                       /* So error */

   if fba then do    /* If FBA driver, make sure we have the package */
      'NUCEXT RXDASD'                              /* Already got it?? */
      if rc <> 0 then 'NUCXLOAD RXDASD'        /* No, try to load it */
      if debug then say 'SWAPGEN: Loading RXDASD got rc=' rc
      if rc <> 0 then signal NoRXDASD       /* That failed, so error */
   end

   if reuse = 0 then do
      call diag 8, 'DETACH' vdev       /* DETACH any existing device */
      parse value diagrc(8, 'DEFINE VFB-512 AS' vdev 'BLK' blks) ,
        with rc . 17 msg '15'x                    /* Define the V-DISK */
      if debug then say 'SWAPGEN: DEFINE VDEV got rc=' rc
      if rc <> 0 then signal BadDefine      /* That failed, so error */
   end
   call csl 'DMSGETFM rc reacode  fm'        /* find a free filemode */
   if debug then say 'SWAPGEN: Got filemode' fm 'from DMSGETFM'
   if rc <> 0 then signal NoFreeModes /* Weren't any, strange, error */

   if fba then do
      pages = trunc((blks * 512)/4096) - 1              /* FBA case */
      writeit = 'stem swap.'                          /* Pipe stage */
      if debug then say 'SWAPGEN: Computed' pages 'for FBA disk'
   end
   else do                /* Not FBA, we must FORMAT and RESERVE it */
      'MAKEBUF'         /* Guard stack contents if something's there */
      buf = rc    /* Remember buffer number so we drop the right one */
      if debug then say 'SWAPGEN: Acquired buffer' buf ,
        'before non-FBA format.'
      writeit = 'specs number 1 1-* next' ,
        '| mdskupdate LINUX SWAP' fm 'F 512'        /* Pipes stages */
      queue '1'                        /* Yes to the format? question */
```

```
      queue 'LXSWAP'                               /* Disk volume name */
      queue '1'                         /* Yes to the reserve question */
      'PIPE (name SWPFORMAT)' ,         /* FORMAT and RESERVE the disk */
        '| cms FORMAT' vdev fm '(BLKSIZE 512 NOERASE' ,     /* FORMAT */
        '| var rs1' ,                        /* Remember how that went */
        '| hole' ,                           /* And otherwise pitch it */
        '| cms RESERVE LINUX SWAP' fm ,                       /* Do it */
        '| var rs2' ,                        /* Remember how that went */
        '| hole' ,                           /* And otherwise pitch it */
        '| state LINUX SWAP' fm  , /* Look at the reserved swap file */
        '| var reserveok' ,                   /* Keep that information */
        '| specs w6 1'         ,  /* Word 6 is the number of blocks */
        '| specs w1 1' ,                      /* Calculate it modulo 8 */
        'a: word 1 .' ,                        /* Get the token we want */
        '   set #0:=a%8-2' ,        /* Calculate it modulo 8 minus 2 */
        '   print #0 20' ,                                 /* Write it */
        '| specs w2 1' ,      /* Just get the number of usable pages */
        '| var pages'                             /* And remember that */
      if debug then say 'SWAPGEN: Formatted' pages 'pages on disk' ,
        fm 'in PIPE'
      'DROPBUF' buf          /* Not nice to leave trash lying around */
      if debug then say 'SWAPGEN: Dropped buffer' buf
   end

   if debug then say 'SWAPGEN: About to write non-FBA swap signature'

/* Must use separate Pipe to write since mdskupdate commits to 0 */
   'PIPE (name SWPWrite)' ,
     '| var pages' ,                            /* Get number of pages */
     '| specs pad 00 w1 d2c 1.4 right' ,                 /* Format it */
     '| append strliteral x'c2x(copies('00'x, 4086-1033+1) || ,
     '53574150535041434532'x) ,             /* "SWAPSPACE2" in ASCII */
     '| join' ,                        /* Build that into a nice chunk */
     '| preface strliteral x'c2x(copies('00'x, 1027)'01'x) ,   /* 0s */
     '| join' ,                        /* Build that into a nice chunk */
     '| deblock 512',                             /* Break into records */
     '|' writeit   /* And write to disk or variable, per driver type */
   if debug then say 'SWAPGEN: Wrote non-FBA swap signature with rc=' rc
   if rc <> 0 then signal BadWrite

/* If FBA, we have the values, need to use RXDASD to write them */
   if fba then do i = 1 to swap.0     /* If FBA, we didn't write yet */
      if debug then say 'SWAPGEN: About to write FBA signature' i
      rc = DASD('WRITED', vdev, i-1, swap.i)           /* Write one */
      if debug then say 'SWAPGEN: Wrote FBA signature' i 'with rc='rc
      if rc <> 0 then signal BadWrite            /* Failed, so error */
   end

   if fba then type ='FBA'
   else type = 'DIAG'
   say type 'swap disk defined at virtual address' vdev , /* Success! */
     '('pages-1' 4K pages of swap space)'
   call Quit 0
```

```
Quit:
   arg rc
   if rc <> 0 then say 'No Swap disk was created.'
   exit rc

NoVdev:              /* User didn't give us a virtual device address */
   say 'A virtual device address must be specified!'
   signal Help

NoBlks:                    /* User didn't give us several blocks */
   say 'Number of blocks must be specified!'
   signal Help

NoFreeModes:                       /* No free disk modes can be found */
   say 'No free disk modes are available!'
   say 'Please release a minidisk and try again.'
   call Quit 1      /* They invoked it correctly, so don't show help */

BadDev:                              /* REUSE tried to use bad device */
   say 'The device at 'vdev' cannot be used:'
   say msg
   call Quit 24

BadBlks:                /* User gave us an invalid number of blocks */
   say 'Invalid number of blocks "'blks'" specified; must be'
   say 'at least 'minblks' 512-byte blocks.'
   call Quit 24

WrongBlks:              /* Supplied number of blocks does not match */
   say 'REUSE requested with' blks ,
     'and existing disk block count is' newblks'.'
   call Quit 24

BadVdev:          /* User gave us an invalid virtual device address */
   say 'Invalid virtual device address "'vdev'" specified;'
   say 'must be a 1- to 4-digit hexadecimal value.'
   call Quit 24

NoRXDASD:                  /* We don't have the required FBA utility */
   say 'Unable to NUXCLOAD RXDASD MODULE; this is available from:'
   say ' http://www.vm.ibm.com/download/packages'
   call Quit rc

BadDefine:                                /* Error DEFINE-ing the VDISK */
   say 'Error' rc 'from CP DEFINE VFB-512 AS' vdev 'BLK' blks':'
   say msg                                    /* Display error from CP */
   call Quit rc

BadFBA:                            /* Error writing FBA block on disk */
   say 'Error' rc 'from RXDASD'
   call Quit rc

BadWrite:                         /* Error on FORMAT or RESERVE steps */
   select                           /* Figure out where it went wrong */
      when symbol('RESERVEOK') <> 'VAR' then do
```

```
         say 'Error' rc 'from CMS RESERVE LINUX SWAP' fm':'
         say rs2
      end
      when symbol('RS2') <> 'VAR' then do
         say 'Error' rc 'from CMS FORMAT' vaddr fm '(BLKSIZE 512:'
         say rs1
      end
      otherwise
      say 'Error' rc 'calculating swap size, contact support'
   end
   call Quit rc

Help:
   parse source . . fn .

   say 'Syntax is:'
   say ''
   say fn 'vdev #blocks <( <options> <)> >'
   say '   or'
   say fn 'vdev ( REUSE <options> <)>'
   say ''
   say 'where:'
   say ''
   say 'vdev     -- is a virtual device address'
   say '#blocks  -- is a decimal number of 512-byte blocks;'
   say '            minimum 24 (FBA) or 32 (DIAG)'
   say ''
   say 'Options are:'
   say 'DIAG     -- (Default) Use DIAG I/O (requires Linux DIAG driver)'
   say 'FBA      -- use FBA driver instead of DIAG; requires RXDASD'
   say '            package, downloadable from the IBM VM download'
   say '            page at: http://www.vm.ibm.com/download/packages'
   say 'REUSE    -- use existing device at vdev.  WARNING: This will'
   say '            destroy any data on device vdev.  The #blocks'
   say '            parameter may be omitted; the whole device will'
   say '            be used in that case.'
   say 'VERSION  -- display current version number string and date'
   say '            of last module update.'
   say 'DEBUG    -- display progress messages and debugging'
   say '            information about the program logic. '
   say ''
   say fn 'will DETACH any existing virtual device at that address,'
   say 'DEFINE a new VDISK, format it, and write the Linux swap'
   say 'signature on the disk so Linux will recognize it.'
   say ''
   say 'If using FBA mode, SWAPGEN prepares the whole device:'
   say '    /dev/dasdb or /dev/dasd/0151/device'
   say 'so the whole device must then specified in the Linux fstab.'
   say ''
   say 'If using DIAG mode, because the V-DISK is CMS FORMATted,'
   say 'SWAPGEN prepares the partition:'
   say '    /dev/dasdb1 or /dev/dasd/0151/part1'
   say 'so the partition must be specified in the fstab on Linux.'
   call Quit 1
```

```
Version:
   parse source . . fn .

/* These variables should be updated with each release */
   version = 'SNA120601'                /* Release string: SNAyymmvv */
   last_update = '2012-06-20 (yyyy-mm-dd)'       /* Last update date */

   say 'SWAPGEN: Version' version', last updated:' last_update'.'
   call Quit 4
```

Sample files

This section lists sample files described in the book.

The Generic PARM file

This section lists the sample `generic.prm` configuration file:

```
ro ramdisk_size=40000 cio_ignore=all,!condev
ip=9.12.7.96::9.12.4.1:20:vmlnx2-1.itso.ibm.com:enccw0.0.0600:none
rd.znet=qeth,0.0.0600,0.0.0601,0.0.0602,layer2=1
nameserver=9.12.6.7 nameserver=9.60.70.80
inst.repo=ftp://ftproot:myftppassword@9.60.87.87/ftp/linux/rhel71
rd.dasd=0.0.0100 rd.dasd=0.0.0200
vnc vncpassword=12345678
```

Linux code

This section contains listings of the following Linux scripts:

- The SUSE Linux Enterprise Server clone.sh script
- The SUSE Linux Enterprise Server boot.clone script

The SUSE Linux Enterprise Server clone.sh script

This section lists the code for the `/usr/local/sbin/clone.sh` script that clones from a SUSE Linux Enterprise Server 12 golden Linux image to a target virtual machine:

```
#!/bin/sh
#
# clone.sh <LinuxUserID> - clone a Linux server running under z/VM
#
#
# -----------------------------------------------------------------------------
# THE PROGRAM IS PROVIDED ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, EITHER EXPRESS OR IMPLIED INCLUDING, WITHOUT LIMITATION, ANY
# WARRANTIES OR CONDITIONS OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY
# OR FITNESS FOR A PARTICULAR PURPOSE.
# NEITHER RECIPIENT NOR ANY CONTRIBUTORS SHALL HAVE ANY LIABILITY FOR ANY
# DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
# (INCLUDING WITHOUT LIMITATION LOST PROFITS), HOWEVER CAUSED AND ON ANY THEORY
# OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
# NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OR
# DISTRIBUTION OF THE PROGRAM OR THE EXERCISE OF ANY RIGHTS GRANTED
# HEREUNDER, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES
# -----------------------------------------------------------------------------

#+--------------------------------------------------------------------------+
function help()
# give help
#+--------------------------------------------------------------------------+
 {
  echo "Usage: clone [options] from <sourceID> to <targetID>"
  echo ""
  echo "  Clone Linux from sourceID 100 and 101 minidisks to targetID"
  echo "  options:"
  echo "    -v or --verbose: verbose"
  echo ""
  echo "Example: clone.sh from s11gold to linux01"
  exit 1
 }

#+--------------------------------------------------------------------------+
function processArguments()
# Parse command line arguments
# Args: The arguments passed in to the script
#+--------------------------------------------------------------------------+
 {
  verbose="off"
  sourceID="none"
  targetID="none"
  while (( "$#" )); do
    case $1 in
      -v|--verbose)
         verbose="on"
         ;;
      from)
        shift
        sourceID=`echo $1 | tr '[a-z]' '[A-Z]'` # fold source ID to upper case
        ;;
      to)
        shift
        targetID=`echo $1 | tr '[a-z]' '[A-Z]'` # fold target ID to upper case
        ;;
    esac
    shift
  done
  if [ $sourceID = "none" ]; then # source user ID was not passed
    echo "Error: Source Linux user ID not supplied"
    help
  fi
  if [ $targetID = "none" ]; then # target user ID was not passed
    echo "Error: Target Linux user ID not supplied"
    help
  fi
 }
```

```
#+--------------------------------------------------------------------------+
function CPcmd()
# echo a CP command and invoke it via the vmcp module/command
#   Arg1-n: the command to issue
#   Return: the command's return code
#+--------------------------------------------------------------------------+
 {
  echo "Invoking CP command: $@"
# parse output to get return code: awk -F# splits line at '#' with rc at end
  output=`vmcp $@ 2>&1`
  echo "$output"
  retVal=0
  retVal=`echo $output | grep "Error: non-zero CP response" | awk -F# '{print
$2}'`
  return $retVal
 }

#+--------------------------------------------------------------------------+
function checkID()
# Verify user ID exists and is logged off
#    Arg 1: The user ID to check
#+--------------------------------------------------------------------------+
 {
  userID=$1
  echo "Checking that $userID exists and is not logged on ..."
  CPcmd QUERY $userID
  rc=$?
  case $rc in
    0)  # user ID is logged on or disconnected
      echo "$userID user ID must be logged off"
      exit 2
      ;;
    3)  # user ID does not exist
      echo "$userID user ID does not exist"
      exit 3
      ;;
   45) # user ID is logged off - this is correct
      ;;
   *) # unexpected
      echo "Return code of $rc unexpected from QUERY $userID"
      echo "User ID must exist and be logged off"
      exit 4
  esac
 }

#+--------------------------------------------------------------------------+
function prepareIPaddr()
# Set the variable "newIPaddr" by adding a backslash before any "."s
#    Arg 1: The IP address to be modified
#+--------------------------------------------------------------------------+
 {
  newIPaddr=`echo $1 | sed -e 's:\.:\\\.:g'`
 }

#+--------------------------------------------------------------------------+
function prepareVaddr()
# Prepare an address by folding to lower case and prepending leading zeros
# to make it 4 digits
#    Arg 1: The vaddr to be modified
# Return:
#    The new value is written to the global variable newVaddr
#+--------------------------------------------------------------------------+
 {
  newVaddr=`echo $1 | tr '[A-Z]' '[a-z]'`  # fold to lower case
  let leadingZeros=4-${#1}                  # determine number of zeros to add
  let i=0
  while [ $i -lt $leadingZeros ]; do
    newVaddr="0$newVaddr"
    i=$[$i+1]
  done
 }

#+--------------------------------------------------------------------------+
function copyDisk()
# Use FLASHCOPY to copy a disk, if it fails, fall back to dasdfmt then dd
#   Arg 1: Source vaddr
#   Arg 2: Target vaddr
#+--------------------------------------------------------------------------+
 {
  source=$1
  target=$2
  echo ""
  echo "FLASHCOPYing $source to $target ..."
  CPcmd FLASHCOPY $source 0 end to $target 0 end
  if [ $? != 0 ]; then
    echo "FLASHCOPY failed, falling back to dasdfmt and dd ..."
    chccwdev -e $source
    if [ $? != 0 ]; then exit 7; fi
    chccwdev -e $target
    if [ $? != 0 ]; then exit 8; fi
    sleep 1
    srcDev=/dev/$(egrep ^0.0.$source /proc/dasd/devices | awk '{ print $7 }')
    if [ "$?" != 0 ]; then exit 5; fi
    tgtDev=/dev/$(egrep ^0.0.$target /proc/dasd/devices | awk '{ print $7 }')
    if [ "$?" != 0 ]; then exit 6; fi
    echo "dasdfmt-ing $tgtDev ..."
    dasdfmt -y -b 4096 -f $tgtDev
    if [ "$?" != 0 ]; then exit 9; fi
    echo "dd-ing $srcDev to $tgtDev ..."
    dd bs=1M if=$srcDev of=$tgtDev oflag=sync
    if [ "$?" != 0 ]; then exit 10; fi
    sync
    echo "disabling and re-enabling $target ..."
    chccwdev -d $target
    if [ $? != 0 ]; then exit 11; fi
    chccwdev -e $target
    if [ $? != 0 ]; then exit 12; fi
    sync
  fi
 }
```

```
#+--------------------------------------------------------------------------+
function askAreYouSure()
# Ask "Are you sure?" - if not, then exit
#+--------------------------------------------------------------------------+
 {
  echo ""
  echo "WARNING!!: Minidisks 100 and 101 will be copied to $targetID"
  echo "Network data is retrieved from $targetID PARM-S11 on 191 disk"
  echo "during the first boot of $targetID"
  echo -n "Are you sure you want to overwrite these disks (y/n): "
  read ans
  if [ $ans != "y" ]; then
    echo "Aborting clone per user input"
    exit 16
  fi
 }

#+--------------------------------------------------------------------------+
function copySystem()
# For each of two minidisks 100 and 101:
#   -) Link disk
#   -) Enable disk
#   -) Copy disk
#+--------------------------------------------------------------------------+
 {
  echo "Linking source and target 100 disks ..."
  CPcmd detach 1100
  CPcmd link $sourceID 100 1100 rr
  if [ $? != 0 ]; then exit 17; fi
  CPcmd detach 2100
  CPcmd link $targetID 100 2100 mr
  if [ $? != 0 ]; then exit 18; fi
  echo "Copying 100 disks ..."
  copyDisk 1100 2100
  echo "Take 1100 Offline...."
  chccwdev -d 1100
  CPcmd det 1100
  CPcmd det 2100

  echo " "
  echo "--------------------------------------"
  echo "Linking source and target 101 disks ..."
  CPcmd detach 1101
  CPcmd link $sourceID 101 1101 rr
  if [ $? != 0 ]; then exit 19; fi
  CPcmd detach 2101
  CPcmd link $targetID 101 2101 mr
  if [ $? != 0 ]; then exit 20; fi
  echo "Copying 101 disks ..."
  copyDisk 1101 2101
  echo "Taking 1101 Offline..."
  chccwdev -d 1101
  CPcmd det 1101
  echo "Taking 2101 Offline..."
  chccwdev -d 2101
  CPcmd det 2101
 }

# main()
processArguments $@                     # process arguments passed by user
if [ $verbose = "on" ]; then set -vx; fi  # turn on debug
checkID $sourceID                       # user ID must exist and be logged
off
checkID $targetID                       # user ID must exist and be logged
off
# getNetworkInfo                          # get info from parm files
askAreYouSure                           # confirm disks will be overwritten
copySystem                              # copy source disks to target
# modifyClone                             # modify newly copied system
echo "sleeping 10 seconds"
sleep 10
CPcmd XAUTOLOG $targetID                # bring new clone to life
if [ $verbose = "on" ]; then set +vx; fi  # turn off debug
echo "Successfully cloned $sourceID to $targetID"
exit 0
```

The SUSE Linux Enterprise Server boot.clone script

This section lists the code for the /etc/init.d/boot.clone script that runs at “first boot” of a newly cloned SUSE Linux Enterprise Server system:

```
#!/bin/bash
#
# /etc/init.d/boot.clone
#
### BEGIN INIT INFO
# Provides:          boot.clone
# Required-Start:    boot.localfs boot.rootfsck
# Required-Stop:     boot.localfs
# Default-Start:     B
# Default-Stop:
# Short-Description: Change configuration during boot
# Description:       Change the current configuration of the system
#  during first bootup. This script works as follows:
#  1. Run vmcp q userid
#  2. Search for a cms file called userid() PARM-S11
#  3. Get new values for network config from there
#  4. Update the network configuration accordingly
#  This previously used to be the cloning.sh script on linuxadmin.
### END INIT INFO

. /etc/rc.status

rc_reset

#+--------------------------------------------------------------------------+
function CPcmd()
# echo a CP command and invoke it via the vmcp module/command
#   Arg1-n: the command to issue
#   Return: the command's return code
#+--------------------------------------------------------------------------+
 {
# echo "Invoking CP command: $@"
# parse output to get return code: awk -F# splits line at '#' with rc at end
  output=`vmcp $@ 2>&1`
  echo "$output"
  retVal=0
  retVal=`echo $output | grep "Error: non-zero CP response" | awk -F# '{print
$2}'`
  return $retVal
 }

#+--------------------------------------------------------------------------+
function prepareVaddr()
# Prepare an address by folding to lower case and prepending leading zeros
# to make it 4 digits
#    Arg 1: The vaddr to be modified
# Return:
#    The new value is written to the global variable newVaddr
#+--------------------------------------------------------------------------+
 {
  newVaddr=`echo $1 | tr '[A-Z]' '[a-z]'`  # fold to lower case
  let leadingZeros=4-${#1}                  # determine number of zeros to add
  let i=0
  while [ $i -lt $leadingZeros ]; do
    newVaddr="0$newVaddr"
    i=$[$i+1]
  done
 }

#+--------------------------------------------------------------------------+
function getUserid()
# Read current userid with vmcp q userid
#+--------------------------------------------------------------------------+
 {
 modprobe vmcp
 UserID=$(CPcmd q userid | awk '{print $1}')
 echo $UserID
 }

#+--------------------------------------------------------------------------+
function getNetworkInfo()
# Bring 191 minidisk online to check for my parameter files
#+--------------------------------------------------------------------------+
 {
  # recycle 191 to pick up latest changes
  chccwdev -d 191
  chccwdev -e 191
  rc=$?
  if [ $rc != 0 ]; then # unable to enable 191 disk
    echo "unable to enable 191, rc from chccwdev = $rc"
    exit 13
  fi
  udevadm settle
  CMSdisk=`lsdasd | grep 0191 | awk '{ print $3 }'`
  cmsfslst -d /dev/$CMSdisk | grep -i $1 | grep PARM-S11
  rc=$?
  if [ $rc != 0 ]; then
    echo "Error: $1 PARM-S11 not found on 191 minidisk. Exiting"
    exit 14
  fi

# get information about target
  { while read parameter; do
    #echo "parameter: ${parameter%=*}"
    case "${parameter%=*}" in
      Hostname)
        targetHostname=${parameter#*=}
        ;;
      HostIP)
        targetIP=${parameter#*=}
        ;;
      Nameserver)
        targetDNS=${parameter#*=}
        ;;
      Gateway)
        targetGW=${parameter#*=}
        ;;
      Netmask)
        targetMask=${parameter#*=}
        ;;
      Broadcast)
        targetBroadcast=${parameter#*=}
        ;;
      ReadChannel)
        prepareVaddr ${parameter#*=}
        targetReaddev=$newVaddr
        ;;
      WriteChannel)
        prepareVaddr ${parameter#*=}
        targetWritedev=$newVaddr
        ;;
      DataChannel)
        prepareVaddr ${parameter#*=}
        targetDatadev=$newVaddr
        ;;
      *)
        # don't know about any other parameters
        ;;
    esac
  done < <(cmsfscat -a -d /dev/$CMSdisk $1.PARM-S11 | tr '[:space:]' '\n')
  }
 }

#+--------------------------------------------------------------------------+
function createNetworkConfig()
# - remove existing network configuration if it exists
# - create new network configuration from information in CMS parmfile
# - update HOSTNAME, hosts, and resolv.conf
```

```
#+--------------------------------------------------------------------------+
 {
 # delete old configuration
 rm -f /etc/sysconfig/network/ifcfg-eth0
 # setup new configuration
 if [ -n "${targetHostname}" ]; then
   echo "Setting hostname to ${targetHostname}"
   echo ${targetHostname} > /etc/HOSTNAME
 fi
 if [ -n "${targetDNS}" ]; then
   echo "Setting dns resolver to ${targetDNS}"
   sed -i '/nameserver/d' /etc/resolv.conf
   echo "nameserver ${targetDNS}" >> /etc/resolv.conf
 fi
 # echo target stuff
 # will add configuration of different devices when time permits.
 if [ -n "${targetIP}" ]; then
   echo "Setting IP address to ${targetIP}"
   echo "STARTMODE='onboot'" >> /etc/sysconfig/network/ifcfg-eth0
   echo "BOOTPROTO='static'" >> /etc/sysconfig/network/ifcfg-eth0
   echo "IPADDR='${targetIP}'" >> /etc/sysconfig/network/ifcfg-eth0
 fi
 if [ -n "${targetMask}" ]; then
   echo "Setting netmask to ${targetMask}"
   echo "NETMASK='${targetMask}'" >> /etc/sysconfig/network/ifcfg-eth0
 fi
 if [ -n "${targetBroadcast}" ]; then
   echo "Setting broadcast to ${targetBroadcast}"
   echo "BROADCAST='${targetBroadcast}'" >> /etc/sysconfig/network/ifcfg-eth0
 fi
 if [ -n "${targetGW}" ]; then
   echo "Setting default gateway to ${targetGW}"
   sed -i '/default/d' /etc/sysconfig/network/routes
   echo "default ${targetGW} - -" >> /etc/sysconfig/network/routes
 fi
 }
#+--------------------------------------------------------------------------+
function cleanupSSH()
# - remove all existing ssh keys
#+--------------------------------------------------------------------------+
 {
# Delete SSH keys - sshd will recreate them at first boot
  echo "Removing SSH keys"
  rm /etc/ssh/ssh_host*
 }


case "$1" in
    start)
   # update system configuration
        userid=$(getUserid)
        getNetworkInfo $userid
        createNetworkConfig
        cleanupSSH
        chkconfig boot.clone off
   rc_reset
   ;;
    stop|restart)
        # this should never happen
        # nothing to do
   ;;
    status)
   # probably never will be run.
        # nothing to do
   ;;
    *)
   echo "Usage: $0 {start}."
   exit 1
   ;;
esac

rc_exit
```

Related publications

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

IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only:

- *The Virtualization Cookbook for IBM z Systems Volume 1: IBM z/VM 6.3*, SG24-8147
- *The Virtualization Cookbook for IBM z Systems Volume 2: Red Hat Enterprise Linux 7.1 Servers*, SG24-8303
- *Linux on IBM eServer zSeries and S/390: Performance Toolkit for VM*, SG24-6059
- *z/VM and Linux on IBM System z*, SG24-7492
- *Linux on IBM eServer zSeries and S/390: Application Development*, SG24-6807
- *IBM Lotus Domino 6.5 for Linux on zSeries Implementation*, SG24-7021
- *Printing with Linux on zSeries Using CUPS and Samba*, REDP-3864
- *IBM z Systems Connectivity Handbook*, SG24-5444
- *Deploying a Cloud on IBM System z*, REDP-4711
- *Installing Oracle 11gR2 RAC on Linux on System z*, REDP-4788
- *Linux on IBM System z: Performance Measurement and Tuning*, SG24-6926
- *Fibre Channel Protocol for Linux and z/VM on IBM System z*, SG24-7266
- *Security for Linux on System z*, SG24-7728
- *Advanced Networking Concepts Applied Using Linux on IBM System z*, SG24-7995
- *Set up Linux on IBM System z for Production*, SG24-8137
- *Practical Migration from x86 to Linux on IBM System z*, SG24-8217
- *End-to-End High Availability Solution for System z from a Linux Perspective*, SG24-8233
- *Security for Linux on System z: Securing Your Network*, TIPS0981
- *Linux on System z: An Ideal Platform to Migrate Your IT Workload*, TIPS1166
- *Linux on IBM eServer zSeries and S/390: Performance Toolkit for VM*, SG24-6059
- *Printing with Linux on zSeries Using CUPS and Samba*, REDP-3864

You can search for, view, download, or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

ibm.com/redbooks

Other publications

These publications are also relevant as further information sources:

- z/VM online documentation:

  http://www.ibm.com/vm/library

  Under the link labeled **`z/VM V6.3 PDF Files`**, the following books are especially useful:

  – *z/VM CP Messages and Codes*
  – *z/VM TCP/IP Messages and Codes*
  – *z/VM CP Commands and Utilities Reference*
  – *z/VM CP Planning and Administration*
  – *z/VM Getting Started with Linux on System z*
  – *z/VM TCP/IP Planning and Customization*
  – *z/VM Performance Toolkit Guide*, SC24-6156-00
  – *z/VM Performance Toolkit Reference*, SC24-6157-00

  Under the link labeled **`Program Directories`**, the following books are especially useful:

  – *Performance Toolkit for VM*
  – *DirMaint*
  – *RACF Security Server for z/VM*

- *z/VM Performance Toolkit Guide*, SC24-6156-00
- *z/VM Performance Toolkit Reference*, SC24-6157-00

Online resources

These websites are also relevant as further information sources:

- The Linux for zSeries and S/390 portal

  http://linuxvm.org
- The IBMVM list server

  http://listserv.uark.edu/archives/ibmvm.html
- The Linux-390 list server

  http://www.marist.edu/htbin/wlvindex?linux-390
- Documentation for Red Hat distributions

  http://www.ibm.com/developerworks/linux/linux390/documentation_red_hat.html
- Documentation for z Systems Linux Development stream

  http://www.ibm.com/developerworks/linux/linux390/documentation_novell_suse.html
- SUSE Linux Enterprise Server evaluation

  http://www.novell.com/products/linuxenterpriseserver/eval.html
- Red Hat Enterprise Linux Server evaluation download for IBM z Systems at no initial cost

  redhat.com/z
- z/VM publications

  http://www.vm.ibm.com/pubs
- z/VM performance tips

  http://www.vm.ibm.com/perf/tips
- z/VM VDISK for Linux swap performance tips

  ibm.com/vm/perf/tips/lxswpvdk.html
- z/VM TCP/IP planning, customization, and reference

  ibm.com/vm/related/tcpip/tcp-pubs.html
- z/VM TCP/IP cryptographic security

  ibm.com/vm/related/tcpip/vmsslinf.html
- z/VM User's Guides and Command References (XEDIT, CMS, others)

  ibm.com/vm/library/zvmpdf.html
- XEDIT for VM/SystemProduct R3 (Historical reference)

  http://ukcc.uky.edu/ukccinfo/391/xeditref.html

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Redbooks The Virtualization Cookbook for IBM z Systems Volume 3: SUSE Linux Enterprise Server 12

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

SG24-8890-00

ISBN 0738440981

Printed in U.S.A.