Redbooks Paper

Gregory Geiselhart
Roy Costa
Klaus Egeler
Michael MacIsaac

Linux on IBM eserver zSeries and S/390: Managing a Samba Server from z/VM

Preface

This Redpaper describes a z/VM front-end for managing a Linux Samba file server. For shops that have substantial mainframe skills but UNIX skills that are not as strong, managing a Linux Samba server, even with the Samba Web Administration Tool, or SWAT, can be challenging. The z/VM front-end tool described in this paper, EZLNXID, allows you to maintain a Samba Linux system from a z/VM user ID running CMS.

In this document, file sharing from Microsoft Windows desktops is assumed. We use the term “file system” from a Linux perspective, and use the term “share” from a Windows desktop perspective. Users on the same team, or Linux group, will get shares read-write, while other shares may be accessible read-only.

Objectives

The objectives of this paper are to:

- Present a brief overview of Samba; see “Samba overview” on page 2
- Describe planning for and preparing z/VM; see “Managing a Samba server” on page 5
- Discuss preparing z/VM for Linux, Samba and EZLNXID; see “Preparing the z/VM host” on page 7
- Discuss installing and customizing Linux, including setting up Logical Volume Manager (LVM); see “Preparing the Linux guest” on page 10
- Discuss Samba versions; see “Linux distributions and Samba versions” on page 17
- Discuss customization, tuning and usage of Samba; see “Customizing and using Samba” on page 21

- Describe installing and setting up the z/VM front-end tool to Samba; see “Installing and customizing the EZLNXID tool” on page 23
- Describe the z/VM front-end tool to Samba; see “Using EZLNXID” on page 28
- Briefly discuss other topics pertinent to a Samba file serving solution, but outside the scope of this Redpaper; see “Data backup and recovery” on page 37

Samba overview

Samba is a very useful tool. It is commonly used because it is free, it bridges the Windows and Linux worlds—and because the Samba development team excels at staying abreast of new Windows technologies and protocols.

File and print serving

Samba makes a Linux server look like a Windows server. Because of the ubiquity of Windows desktops, you should follow two rules when creating an architecture for a file and print solution:

- **Rule 1**: Windows clients should not have to be modified.
- **Rule 2**: When a change is needed to a Windows client, see Rule 1.

This is a bit of a joke, but is also basically true. Solutions that require Windows desktops to be modified, especially with proprietary software, often never get off the ground or fail for both technical and political reasons.

Samba at a low level

Samba implements a server for the Server Message Block (SMB) protocol. All Windows and OS/2 systems are both SMB clients and servers. Normally Samba provides only the server function on Linux, but there are also some client capabilities (the SMB file system on Linux is rarely used because other file sharing protocols are superior).

SMB was built on top of Network Basic I/O Services (NetBIOS), which is an Application Programming Interface (API), and not a networking protocol. The original PC networking protocol is the NetBIOS Extended User Interface (NetBEUI). It was useful when PCs were breaking into the enterprise and TCP/IP was still only used by the government and academia.

NetBEUI cannot be routed, but on the other hand it is very lightweight, so it is very efficient. NetBIOS over IPX can be done in a Novell NetWare environment; however, NetWare environments are becoming less common.

NetBIOS can be routed over TCP/IP. This protocol was proposed in RFCs 1001 and 1002 in 1987. This marries the large numbers of SMB clients and the global nature of TCP/IP. See Figure 1 on page 3 for a block diagram of these protocols and APIs as they pertain to the ISO Open Systems Interconnect (OSI) model.

NetBIOS provides the following three types of service but only two (session and name) are commonly used.

Session	Sharing disks and printers.
Name	Computer, disk and printer names - sometimes called browse lists and typically accessed via Windows Network Neighborhood.
Datagram	This function is typically not used. It is analogous to UDP/IP.

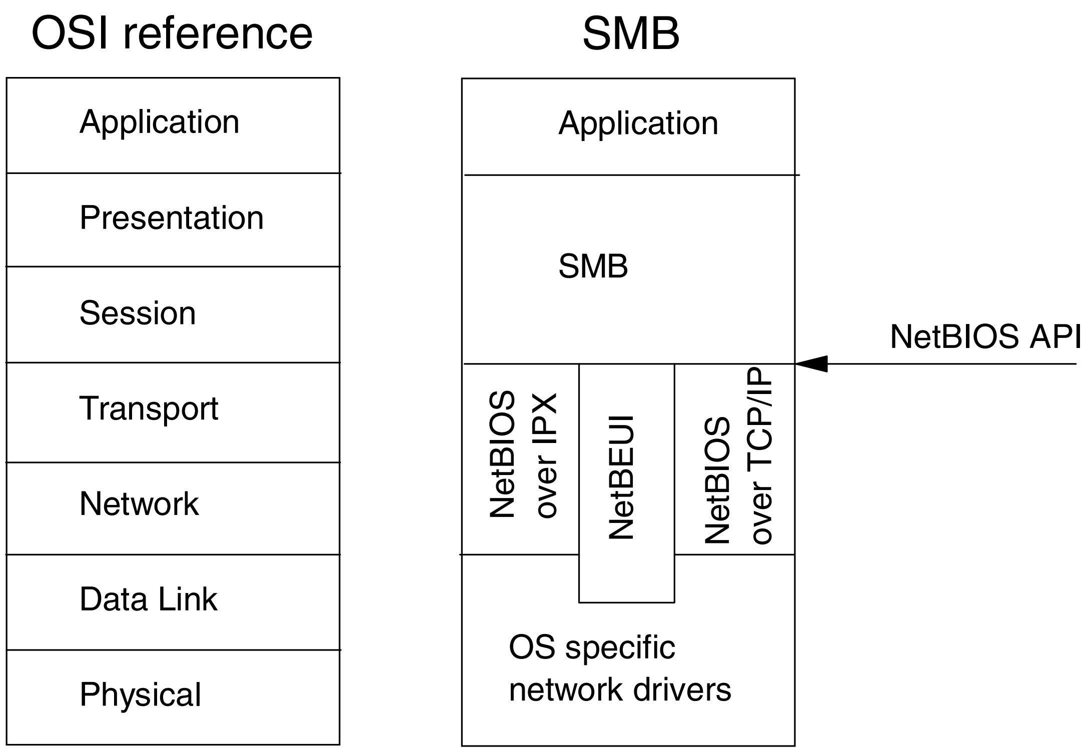


Figure 1 SMB relative to the OSI reference model

Samba services

Samba can provide the following services:

- Supply disk shares
- Supply printer shares, including the automatic download of printer drivers to the client
- Act as a Windows NT Primary Domain Controller (PDC) - v2.2.1 and later
- Be integrated into a Windows Active Directory - v2.2.3 and later

The services Samba provides have been designed to allow Windows clients to easily connect to UNIX and Linux servers.

Samba executables

The Samba package includes many executables, which can be divided into system and user executables.

The *system* executables are:

smbd	Server Message Block Daemon - the main workhorse
nmbd	NetBIOS name daemon - the browse list daemon
swat	The Samba Web administration tool - a mini-Web server that allows a browser interface
winbindd	Resolve user and group information from a Windows NT/2000 server

The more commonly used *user* executables are:

smbclient	Connects to SMB shares using an FTP-like syntax
testparm	Tests the validity of an smb.conf configuration file
testprns	Check printer name for validity with smbd
smbstatus	Report on current Samba connections
smbpasswd	Change a users SMB password
nmblookup	Look up NetBIOS names
wbinfo	Query information from winbind daemon

There is only one Samba configuration file, named smb.conf. Because there is only one configuration file, the parameters for all Samba executables are in one place—however, there are hundreds of parameters that can be set. This file is typically in the /etc/samba/ directory, but is sometimes found in the /etc directory.

Samba documentation

When Samba is installed on your Linux system, the documentation is typically included below the /usr/share/doc directory. On SuSE, the directory is /usr/share/doc/packages/samba. On a Red Hat distribution, the directory is /usr/share/doc/samba-2.2.1a/docs. If you have downloaded the Samba source package, the documentation is in the docs directory.

The book *Using Samba*, by Robert Eckstein, David Collier-Brown and Peter Kelly was donated to the Samba project by O'Reilly Publishers. It is available on the Web at:

http://www.oreilly.com/catalog/samba/chapter/book/index.html

This book is included with all recent Samba packages, and is accessible via SWAT.

Although a great deal of old Samba documentation exists, *The Unofficial Samba HOWTO,* updated in November 2002, is available at:

http://hr.uoregon.edu/davidrl/samba.html

What's new in Samba

Samba 2.2.7a is the latest stable release of Samba at the time of writing. There were a few minor fixes in 2.2.7a. Release 2.2.7 fixed a security hole whereby a user could possibly gain root access (although the Samba team was unable to exploit this possible hole). Both of these versions became available in late 2002.

Samba 2.2.6 became available in October 2002. Several fixes and internal enhancements were made including:

- Fixes for MS-RPC printing issues affecting Windows 2000 clients
- New support for smb.conf generation in SWAT
- Inclusion of several performance enhancements
- Fixes for several file locking bugs and returned status codes

For more detailed information about the new functions in Samba 2.2.6, 2.2.7, and 2.2.7a, see the smb.conf man page and WHATSNEW.txt. For any other questions concerning Samba, refer to the Web site:

http://www.samba.org/

Managing a Samba server

In this section we explain the use of EZLNXID, which is a tool to manage a Samba server running in a z/VM Linux guest.

The EZLNXID administration tool consists of a package of EXECs on z/VM and scripts on Linux. There is a panel-driven front-end program on z/VM that runs on a CMS user ID.

There are two methods for the panel display implementation:

- CMS XEDIT, which is the z/VM editor.
- IOS3270, which is a presentation tool used on some VM systems.

> **Note:** If you do not have IOS3270, it is not supported, but is available on:
>
> `http://www/vm/ibm.com/download/packages/`

Using EZLNXID, Samba administrators can perform the following functions from a CMS session:

- Define new users and sets Linux and Samba passwords
- Define new groups for access to Samba file systems
- Create new file system directories
- Create Samba share definitions for the Linux file system
- Authorize new and existing users to groups
- Change passwords, both for Linux and Samba
- Remove users from groups
- Remove users from Linux system
- List users, groups, Samba shares, and Linux file system usage

EZLNXID will *not* perform the following functions:

- Delete Linux files or directories
- Remove Samba share definitions from the smb.conf file

This Redpaper is based on a SuSE SLES8 Linux distribution running under z/VM 4.3. However, the EZLNXID tool has also been tested using Samba 2.2.0 and with a Red Hat Linux distribution.

Overview of the EZLNXID tool

Figure 2 on page 6 illustrates how the EZLNXID tool operates.

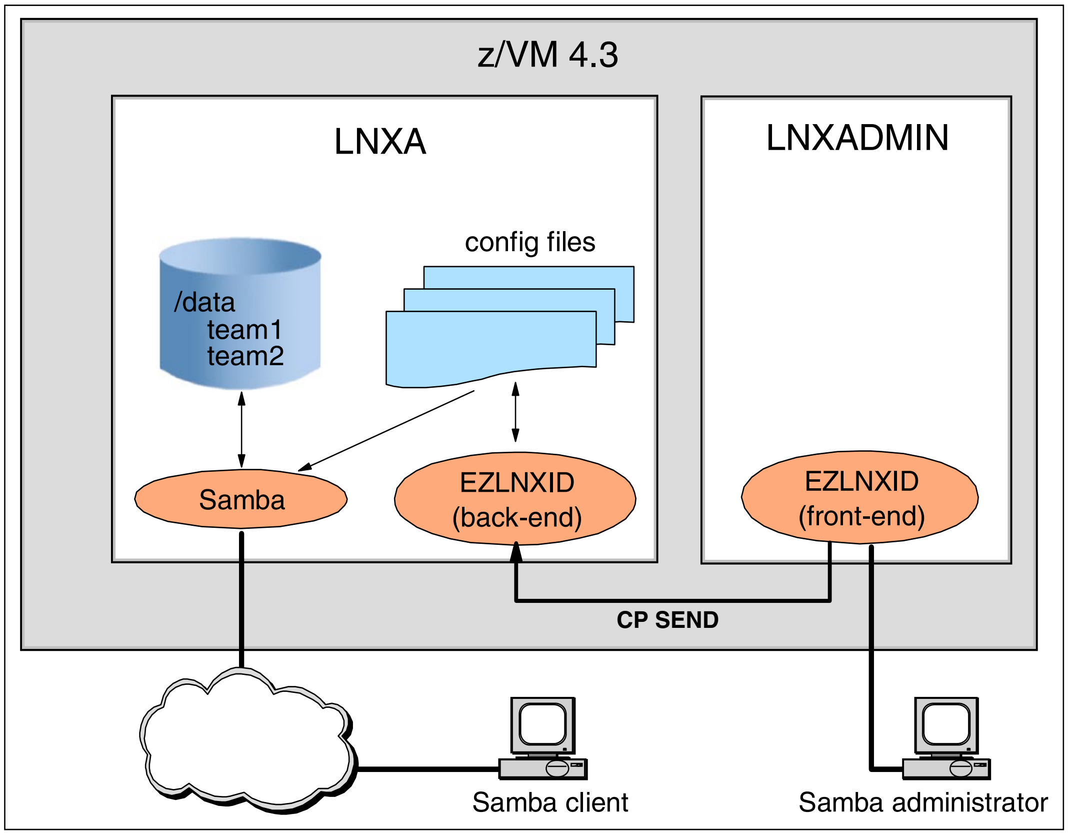


Figure 2 Overview of the EZLNXID tool

In Figure 2, a Samba administrator logs on to the LNXADMIN z/VM user ID and uses the EZLNXID tool to manage a Samba server running in the LNXA Linux guest. Samba clients access the Samba server over a TCP/IP network. Samba shared disks are defined as subdirectories under the /data directory.

Note: To simplify authorization control to Samba shared drives, the EXLNXID tool uses group names when creating shared drives. Figure 2 illustrates two shared Samba drives, team1 and team2. Samba users are granted authorization based on membership in the corresponding group. For instance, Samba users who are members of group team1 will be granted access to the team1 shared drive.

The EZLNXID tool consists of a front-end running on the LNXADMIN virtual machine, and a back-end on the LNXA Linux guest. Commands to administer the Samba server are conveyed from the front-end to the back-end using the CP **SEND** command. Acting on messages received from the front-end, the EZLNXID back-end issues Linux and Samba commands to modify configuration files used by the Samba server.

Note: By default, Samba re-reads its configuration file every 60 seconds. This allows the EZLNXID back-end to change the Samba server configuration (and to have those changes reflected in a running server) *without* having to send a restart signal to the Samba daemon.

Preparing the z/VM host

In this section, we describe the two z/VM virtual machines that we created that will use the EZLNXID tool. The user ID that will run CMS is named LNXADMIN and the user ID that will run Linux is named LNXA.

The LNXADMIN virtual machine definition

The LNXADMIN z/VM user directory entry is shown in Example 1. This virtual machine runs the EZLNXID front-end.

Example 1 The LNXADMIN DIRECT profile

```
USER LNXADMIN LNXADMPW 32M 64M CG                          1
 INCLUDE IBMDFLT
 MACH ESA
 IPL CMS
 MDISK 191 3390 3189 010 430W01  MR RPW WPW MPW            2
```

Following are explanations of the details shown in the example.

1. The user is defined general user class G and privilege class C. This enables the LNXADMIN user to issue the CP **SET SECUSER** command (discussed further in "Using the CP SEND command" on page 8).
2. Defines a small 191 minidisk used for the A-disk CMS guest with read, write, and multiwrite passwords.

The LNXA virtual machine definition

The LNXA z/VM user directory entry is shown in Example 2. This virtual machine runs the EZLNXID back-end and the Samba server in a Linux guest.

Example 2 The LNXA DIRECT profile

```
USER LNXA     LNXAPW 128M 1G   G
 INCLUDE IBMDFLT
 MACHINE ESA
 IPL CMS PARM AUTOCR                                     1
 CONSOLE 009 3215 T LNXADMIN
 DEDICATE 7127 7127                                      2
 DEDICATE 7128 7128
 DEDICATE 7129 7129
 MDISK 191 3390 3199 025 430W01  MR RPW WPW MPW          3
 MDISK 201 3390 0001 200 LX3CA0  MR RPW WPW MPW          4
 MDISK 202 3390 0201 3138 LX3CA0  MR RPW WPW MPW         5
 MDISK 203 3390 0001 3338 LX3FA0  MR RPW WPW MPW         6
 MDISK 204 3390 0001 3338 LX3FC0  MR RPW WPW MPW         6
```

Following are explanations of the details shown in the example.

1. Automatically IPL and run CMS, bypassing the first VM READ.
2. Attach hipersocket network devices to the guest.
3. Define a 191 minidisk for the A-disk CMS guest.
4. Define a 201 minidisk for a swap device on the Linux guest.
5. Define a 202 minidisk for / (root) file system on the Linux guest.

6. Define minidisks at addresses 203 and 204 to be used for a /data file system on the Linux guest. A volume management system will be used to combine these two minidisks.

Note: The minidisk passwords should be unique and protected, since having those passwords would allow others to access these disks. (The passwords listed in the examples of this paper are not the actual passwords of the minidisks for our systems.)

Using the CP SEND command

Using the CP **SEND** command allows the Linux administrator (LNXADMIN) to communicate with the Linux guest (LNXA). In order to use CP **SEND**, the LNXADMN user ID must first issue the z/VM privileged CP **SET SECUSER** command. For this reason, the LNXADMIN user is granted privilege class C.

Note: As an alternative to authorizing LNXADMIN as a class C user, the LNXA user ID could have defined LNXADMIN to be a secondary user. This would allow LNXADMIN to use CP **SEND** as a general class G user.

To define LNXADMIN as a secondary user to LNXA, define the console in the LNXA z/VM user directory entry as:

```
CONSOLE 009 3215 T LNXADMIN
```

When defined as a secondary user, console messages from the LNXA virtual machine are displayed on LNXADMIN console.

Automatically boot Linux on z/VM IPL

To have the LNXA Linux guest boot automatically on z/VM IPL, we first modify the PROFILE EXEC on the LNXA 191 minidisk, adding the lines shown in Example 3 to the end.

Example 3 LNXA profile exec

```
...
if word(diagrc(24,-1),2) = 2 then            /* User is disconnected - rc=2 */
'CP IPL 202 clear'                           /* IPL address of Linux system */
```

If the user is disconnected (as it would be if the user were **AUTOLOG**ed), the Linux system boot disk is IPLed.

Next, we modify the PROFILE EXEC on the AUTOLOG1 191 minidisk by adding the line:

```
CP XAUTOLOG LNXA
```

This causes the LNXA user to be automatically logged on at system IPL.

Note: By default, the AUTOLOG1 service machine is automatically logged on disconnected by the system after IPL of a z/VM system. AUTOLOG1 automatically runs its PROFILE EXEC when logged on. In z/VM, this mechanism is used to start z/VM users (which are analogous to Linux services) on system IPL.

Unlike the CP **AUTOLOG** command, CP **XAUTOLOG** does not require a logon password.

Automatically shut down Linux on z/VM shutdown

We would like to automatically shut down the LNXA Linux guest whenever the z/VM system is shut down. We first create the HALTLNX EXEC script shown in Example 4 on the 191 minidisk of the z/VM user ID that is used to issue the z/VM system SHUTDOWN (usually, MAINT).

Example 4 The HALTLNX EXEC script

```
/* Program to 'halt' linux system prior to system shutdown */

'Pipe cms id() | specs w1 | var user'         /* determine this userid for secuser */
say ''
say 'Going to take down the LINUX systems before z/VM shutdown.'
say ''

/* List your Linux system(s) to be shutdown after variable - lnxsys       */
/* If there is more than one system, list them with a space between each. */
lnxsys = lnxa
 'pipe',
    ' literal 'lnxsys,
    '| split',
    '| stem lnx.'

do i=1 to lnx.0

/* This is where we set up secuser, issue the Linux shutdown, and reset secuser */

   'SET SECUSER 'lnx.i user
   Address Command 'CP SEND 'lnx.i' shutdown -h now'
   'SLEEP 5 SEC'
   'SET SECUSER 'lnx.i' RESET'
end

exit
```

Tip: The highlighted line in Example 4 illustrates how to identify the Linux guest to be shut down. To shut down multiple Linux guests, list the guest users adding a space between each user name; for example:

```
   lnxsys = lnxa lnxb lnxc
```

When shutting down z/VM, the system operator executes the HALTLNX EXEC a few minutes prior to issuing the z/VM system CP **SHUTDOWN** command.

If the operator always uses the same user ID to do the z/VM system shutdown, you can create a SHUTDOWN EXEC on the 191 disk of that user ID. The script would first run the HALTLNX EXEC, wait a few minutes, then issue the CP **SHUTDOWN** command.

Example 5 illustrates an example of a SHUTDOWN EXEC. It first prompts to make sure the correct z/VM system is going to be shut down. (This helps prevent the wrong system from being shut down in a multiple z/VM environment.) It then runs the HALTLNX EXEC, waits two minutes, then shuts down the z/VM system.

Example 5 The SHUTDOWN EXEC script

```
/* Program to take down other resources prior to CP SHUTDOWN */

/* Check to make sure we are going to shutdown the correct system */
'Pipe cms id() | specs w3 | var node'
```

```
say ' '
say ' Are you sure you want to SHUTDOWN' node 'now ? (Y/N)'
say ' '
parse upper pull answer

if substr(answer,1,1) = 'Y' then do
  'EXEC HALTLNX'
  say ''
  say 'Going wait 2 minutes for the LINUX systems to come down.'
  say ''
  'SLEEP 2 MIN'

'CP SHUTDOWN'
end
else do
   say ' '
   say 'SHUTDOWN processing terminated by operator request.'
   say ' '
   exit
end
exit
```

Preparing the Linux guest

The SuSE 2.4.7 SLES7 distribution is used in conjunction with Samba 2.2.7a. Since the environment is dependent on Samba and basic Linux scripts (Regina - REXX), you should be able to run this on any of the major Linux for zSeries distributions. Prototypes of this system have run on a SuSE 7.0 distribution that has 2.2.16 kernel with Samba 2.2.0.

DASD considerations

Factors to consider for DASD are minidisk size, file system type, and volume management.

Minidisk size

3390 DASD is commonly emulated by hardware such as RVA, Shark and Iceberg. Physical 3390 hardware has not been manufactured for many years.

The model number is roughly the number of gigabytes that can be stored; there are 3390-1s (or “mod-1s”), 3300-2s, 3390-3s and 3390-9s; “mod-3s” are by far the most common.

A new, larger model can be emulated, though no physical hardware of this type was ever manufactured; it is referred to as “large disk support” or a “mod-27”.

The models listed in Table 1 on page 10 can be emulated.

Table 1 3390 models

Model	Number of cylinders	Approximate size	Approximate size after Linux dasdfmt
3390-1	1113	924 MB	789 MB
3390-2 (rare)	2226	1.8 GB	1.54 GB
3390-3 (common)	3339	2.7 GB	2.3 GB
3390-9	10017	8.12 GB	7.0 GB

Model	Number of cylinders	Approximate size	Approximate size after Linux dasdfmt
3390-27 (new)	32000	25.9 GB	22.1GB

Minidisks on z/VM cannot span more than one physical DASD volume, and even a new “mod 27” will likely not have enough space for a shared file system. Therefore, several DASD are combined into a file system using a volume management system.

With regard to the number of DASD that can be utilized, there three barriers (the first two can be worked around):

- There are 26 DASD nodes predefined in the /dev directory.

  The Linux for zSeries device file system is newer and avoids this problem because letters are not used in the file name; rather, the device address is used as a directory name. For example, in the device file system, a root partition on minidisk 200 would have the file name /dev/dasd/0200/part1.

  However, most distributions still do not use the device file system and instead use device names /dev/dasda.../dev/dasdz.

  Note: The major number assigned to DASD block devices is 94.

- The device minor number supports up to 64 DASD devices.

  The value of the minor number is stored in one byte and therefore limited to 256. Each DASD uses four minor numbers. Thus, 64 DASD use all available minor numbers.

  To get more than 64 DASD devices, you have to use a different major number. Typically this is done by using the largest “free” major number, 253, and working backwards.

- There are a limited number of available major device numbers.

  Major device numbers are assigned by convention. New devices may added for each major Linux kernel release; consult the Documentation/devices.txt file in the Linux kernel source for the major number assignments.

  Note: As of the 2.4.18 kernel, major numbers 240-253 are available for use locally or for experimentation.

Adding more DASD device nodes

To create more DASD device nodes, you need to know the major and minor node number for each device. The Linux for zSeries DASD driver manages all zSeries disk devices as block devices with a major number of 94. The four minor numbers are assigned to each device in order of definition as shown in Table 2 on page 11.

Table 2 Minor number assignments for DASD devices

Device	Minor numbers assigned
/dev/dasda	0 1 2 3
/dev/dasdb	4 5 6 7
...	...
/dev/dasdaa	104 105 106 107

Example 6 illustrates a simple method to determine the major and minor nodes for DASD devices defined to your system.

Example 6 Determining major and minor DASD assignments

```
# cat /proc/dasd/devices
0201(ECKD) at ( 94:  0) is dasda:active at blocksize: 4096, 36000 blocks, 140 MB
0202(ECKD) at ( 94:  4) is dasdb:active at blocksize: 4096, 564840 blocks, 2206 MB
0227(ECKD) at ( 94:  104) is dasdaa:active at blocksize: 4096, 600840 blocks, 2347 MB
```

Use the **mknod** command to create the additional DASD devices. For example, to create two additional DASD devices (the raw /dev/dasdaa DASD device and the /dev/dasdaa1 partition) to follow /dev/dasdz, use the following:

```
# mknod /dev/dasdaa b 94 104
# mknod /dev/dasdaa1 b 94 105
```

Once created, the new DASD devices can be used like any of the other DASD devices.

File system types

File systems can be created as a *journaled* file system, or as the default ext2 file system (which does not provide a journaling feature). If Linux were to crash, an ext2 file system cannot be mounted until its consistency has been checked via the **e2fsck** command. On a large file system without journaling, this process can be quite lengthy.

Important: A journaled file system offers rapid restart capabilities after a system crash, so it is clearly recommended for file serving.

There are several types of journaled file systems such as ext3, JFS, and ReiserFS and any of them would provide the services needed. ReiserFS is a journaled file system that is already complied with the SuSE distribution; no further installation or configuration is needed to use it. The ext3 journaled file system is built into Red Hat.

Note: Because the example in this paper uses a SuSE Linux distribution, ReiserFS is used.

Volume management types

There are several volume management systems available for Linux today, notably:

- Logical Volume Manager (LVM) - commonly used on SuSE distributions
- RAID tools, sometimes called mdtools - commonly used on Red Hat distributions
- Enterprise Volume Management System (EVMS) - a new system with multiple interfaces and allowing “plug-ins” of other systems

Because the example in this paper uses a SuSE Linux distribution, we’ll cover Logical Volume Manager in detail.

Logical Volume Manager concepts

Logical Volume Manager has its own terminology, which we briefly describe here:

- A DASD volume is called a *physical* volume (PV), because it is the volume where the data is physically stored.

- The PV is divided into several physical *extents* (PE) of the same size. The PEs are like blocks on the PV.
- Several PVs make up a *volume group* (VG), which becomes a pool of PEs available for the *logical* volume (LV).
- The LVs appear as normal devices in /dev/ directory. You can add or delete PVs to/from a VG, and increase/decrease your LVs.

Figure 3 shows a block diagram of Logical Volume Manager.

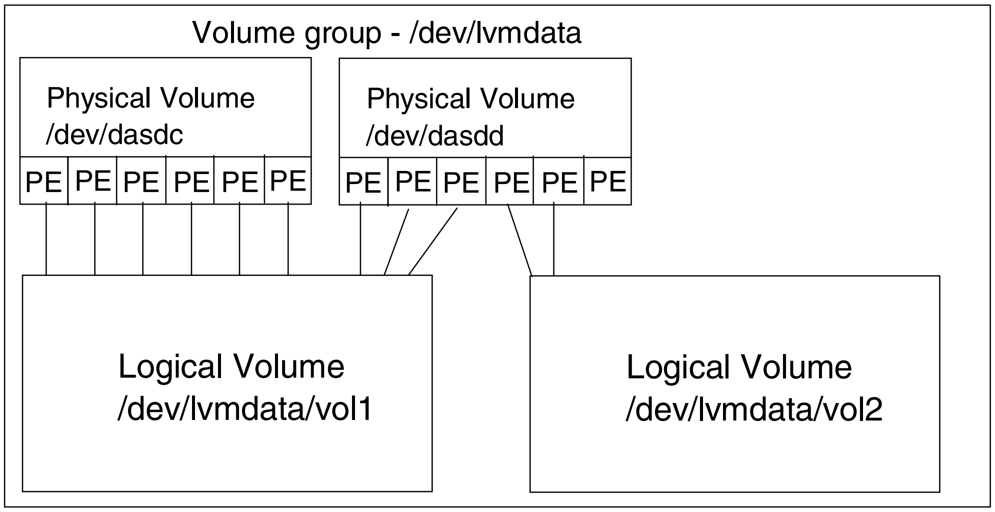


Figure 3 LVM block diagram

Logical Volume Manager can be exercised either from the command line, or by using the setup tool **yast**. If the YaST implementation does not complete successfully, your new logical volume could be left in an inconsistent state that could be difficult to correct. It was also noted that the YaST with SLES-7 would not recognize DASD device nodes beyond /dev/dasdz1. Therefore, we used the Logical Volume Manager line commands.

Example of creating a logical volume

In the sections that follow, we show an example of creating and using a logical volume.

Preparing DASD devices for Logical Volume Manager

Before the physical volumes can be created, they must first be formatted and partitioned for zSeries Linux. This is done using the **dasdfmt** and **fdasd** commands.

1. To see the DASD that are known to Linux:

```
# cat /proc/dasd/devices
0201(ECKD) at ( 94:  0) is dasda : active at blocksize: 4096, 36000 blocks, 140 MB
0202(ECKD) at ( 94:  4) is dasdb : active at blocksize: 4096, 564840 blocks, 2206 MB
0203(ECKD) at ( 94:  8) is dasdc : active at blocksize: 4096, 36000 blocks, 140 MB
0204(ECKD) at ( 94: 12) is dasdd : active at blocksize: 4096, 36000 blocks, 140 MB
```

2. To format a single DASD, a block size of 4 KB is recommended. Use the command:

```
# dasdfmt -b 4096 -f /dev/dasdc
Drive Geometry: 200 Cylinders * 15 Heads =  3000 Tracks

I am going to format the device /dev/dasdc in the following way:
   Device number of device : 0x203
   Labelling device        : yes
   Disk label              : VOL1
   Disk identifier         : 0X0203
   Extent start (trk no)   : 0
   Extent end (trk no)     : 2999
   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
Formatting the device. This may take a while (get yourself a coffee).
Finished formatting the device.
Rereading the partition table... ok
```

Note: Answer `yes` to the query.

3. To create a single partition from a formatted DASD, the **-a** flag to **fdasd** can be used. If the -a flag is omitted, an interactive formatting session similar to the PC **fdisk** command is invoked:

```
# fdasd -a /dev/dasdc
auto-creating one partition for the whole disk...
writing volume label...
writing VTOC...
rereading partition table...
```

Repeat the **dasdfmt** and **fdasd** commands for each DASD device to be used by LVM.

Setting up Logical Volume Manager

Once the DASD devices have been formatted and partitioned, Logical Volume Manager can be installed on the devices. The following commands are used to create a logical volume of /dev/dasdc1 and /dev/dasdd1:

1. Create physical volumes for each DASD that will be included in the logical volume. This can be done for all DASD in one command or one at a time using multiple **pvcreate** commands:

```
# pvcreate /dev/dasdc1 /dev/dasdd1
pvcreate -- physical volume "/dev/dasdc1" successfully created
pvcreate -- physical volume "/dev/dasdd1" successfully create
```

2. To show the currently defined physical volume:

```
# pvscan
pvscan -- reading all physical volumes (this may take a while...)
pvscan -- inactive PV "/dev/dasdc1" is in no VG  [140.53 MB]
pvscan -- inactive PV "/dev/dasdd1" is in no VG  [140.53 MB]
pvscan -- total: 2 [281.06 MB] / in use: 0 [0] / in no VG: 2 [281.06 MB]
```

3. Use the **vgcreate** command to create the volume group. All the physical volumes that are to be in the group must be included in the **vgcreate** command:

```
# vgcreate lvmdata /dev/dasdc1 /dev/dasdd1
vgcreate -- INFO: using default physical extent size 4 MB
vgcreate -- INFO: maximum logical volume size is 255.99 Gigabyte
vgcreate -- doing automatic backup of volume group "lvmdata"
vgcreate -- volume group "lvmdata" successfully created and activated
```

Note: If additional volumes need to be added to the volume group, use the **vgextend** command.

4. Display the volume group you just created:

```
# vgdisplay /dev/lvmdata
--- Volume group ---
VG Name               lvmdata
VG Access             read/write
VG Status             available/resizable
VG #                  0
MAX LV                256
Cur LV                0
Open LV               0
MAX LV Size           255.99 GB
Max PV                256
Cur PV                2
Act PV                2
VG Size               272 MB
PE Size               4 MB
Total PE              68
Alloc PE / Size       0 / 0
Free  PE / Size       68 / 272 MB
VG UUID               fxozbU-5VqJ-v5FG-LfIV-JcyK-1Jbh-KHXd2N
```

Note: The reported VG Size of volume group is used in the next step.

5. Create a logical volume using the volume group VG Size noted above:

```
# lvcreate -L 272M -n vol1 lvmdata
lvcreate -- doing automatic backup of "lvmdata"
lvcreate -- logical volume "/dev/lvmdata/vol1" successfully created
```

6. To display the logical volume you just created:

```
# lvdisplay /dev/lvmdata/vol1
--- Logical volume ---
LV Name                /dev/lvmdata/vol1
VG Name                lvmdata
LV Write Access        read/write
LV Status              available
LV #                   1
# open                 0
LV Size                272 MB
Current LE             68
Allocated LE           68
Allocation             next free
Read ahead sectors     1024
Block device           58:0
```

Creating a file system on the logical volume

After the logical volume is created, a file system is installed on it. For the Reiser file system, the **`mkreiserfs`** command is used:

```
# mkreiserfs /dev/lvmdata/vol1
```

Note: For an ext2 file system, the **`mke2fs`** command is used and for an ext3 file system one would expect a mke3fs command, however, none exists. Rather, the **`mke2fs`** command is again used with the -j flag. For example, to create an ext3 file system, the command would be:

```
# mke2fs -j -b 4096 /dev/lvmdata/vol1
```

The file system can now be mounted using:

```
# mkdir /data
# mount /dev/lvmdata/vol1 /data
```

To check the status of the new file system, use the **df** command as shown in Example 7.

Example 7 Using the df command to display mounted file systems

```
# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/dasdb1           2.1G  1.5G  531M  75% /
/dev/lvmdata/vol1     4.5G  100k  4.4G   1% /data
```

Details on device nodes can be found in *Linux for S/390*, SG24-4987. Further information is available in *LVM - Logical Volume Manager*, found at:

http://www.suse.de/us/whitepapers/lvm/

Automatic logon to Linux as root

The EZLNXID back-end needs to execute commands with root authority (commands include adding Linux new users). Two options exist to enable the back-end to act on CP **SEND** messages from the front-end:

- Append the root password to the CP **SEND** command issued from the front-end.
- Open a root console login on the LNXA Linux guest.

We choose to open a root console login, as this will not expose the root password outside the LNXA virtual machine. To automate root login on startup of the LNXA Linux guest, we create shell script to login as root during Linux initialization. The /root/bin/consoleshell script is shown in Example 8.

Example 8 The /root/bin/consoleshell script

```
#!/bin/sh
exec /bin/login -f root
```

To enable root login at boot time:

1. Create the /root/bin/consolescript file as in Example 8.
2. Set the execute permissions for the file:

   ```
   # chmod 0700 /root/bin/consoleshell
   ```

3. Modify the /etc/inittab file. Change the line:

   ```
   1:1235:respawn:/sbin/mingetty console
   ```

   to:

   ```
   1:1235:respawn:/sbin/agetty -L -n -l /root/bin/consoleshell 9600 console dumb
   ```

   Note: For safety, make a backup copy of the /etc/inittab file before making this change:

   ```
   # cp -p /etc/inittab /etc/inittab.bkup
   ```

The **agetty** terminal executable is specially designed for this type of login.

Patching the kernel

It may be necessary to patch the Linux kernel, because the initial SuSE SLES-7 available on the Internet with Samba 2.2.0 will cause a kernel oops when a Samba share is first written to.

If this occurs, you will either need to build Samba 2.2.6 or later, or you can patch the kernel with patches 3 and 4 (the SLES-7 kernel already contains patches 1 and 2) from the IBM developerWorks site.

Documentation on patching the Linux kernel for z/VM is available from:

- *Linux on IBM @server zSeries and S/390: TCP/IP broadcast on z/VM Guest LAN,* REDP3596, available at:

  http://www.ibm.com/redbooks/abstracts/redp3596.html

- *Patching the Linux/390 kernel*, available at:

  http://linuxvm.org/Info/HOWTOs/krnpatch.html

Linux distributions and Samba versions

There are a number of ways that you can get Samba installed and running on your Linux system. The easiest is to install Samba when you install Linux, or to add it after the Linux is installed similar to how Regina was added. If you have a Linux system and are not sure the level of Samba, use the following command:

```
# rpm -q samba
samba-2.2.5-55
```

This was issued on a SLES-8 system and shows that samba-2.2.5 is installed. Usually only one version of Samba is included with a distribution.

Tip: The `-55` extension on the samba package name is the package release number. The release number indicates the number of times the package has been rebuilt using the same software version.

The Samba versions listed in Table 3 are available for zSeries.

Table 3 Linux distributions and Samba versions

Linux for zSeries distribution	Kernel level	Samba version
SuSE 7.0	2.2.16	2.0.7
SuSE SLES-7	2.4.7	2.2.0a
Red Hat 7.2	2.4.9	2.2.1a
Debian 3.0	2.4.17	2.2.3a
SuSE SLES-8	2.4.19	2.2.5

A Samba version of 2.2.3a or later is recommended because of the many new features that became available at this level (**`winbindd`**, for example). Given this recommendation, only Debian 3.0 and SuSE SLES-8 could use the Samba package that comes with the distribution. However, this is just a recommendation; older versions of Samba will certainly run fine if the additional function is not needed. The latest version of Samba at the time of writing is the 2.2.7a version, which became available in December 2002.

Samba installation via source code

There are many Samba compile-time options to include additional support that is not compiled in by default. To see the complete list, go to the `source` directory and give the command **`./configure --help`**. From that list, these are some of the interesting options:

`--with-acl-support`	Include ACL support (default=no)
`--with-afs`	Include AFS support (default=no)
`--with-winbind`	To also build **`windbindd`** (default: if supported by the OS)
`--with-pam_smbpass`	Include the smbpass PAM module (default=no)
`--with-ssl`	Include SSL support (default=no)
`--with-ldapsam`	Include experimental LDAP SAM support (default=no)
`--enable-cups`	Turn on CUPS support (default=auto)

Get the source code

First you have to download the source code of Samba from:

`http://www.samba.org/`

Find a download *mirror* site and download the latest version. For example, if Samba version 2.2.7a is the latest, download either the samba-2.2.7a.tar.bz2, or samba-2.2.7a.tar.gz file to the /usr/src directory.

Tip: Files with suffixes of .bz2 and .gz use different compression algorithms:

- .bz2 files compress into a smaller space

  Use **`tar -j`** to uncompress this type of file.
- .gz files generally compress and uncompress faster

  Use **`tar -z`** to uncompress this type of file.

Building and installing Samba

Extract the file using the **`tar`** command. By convention this is done in the /usr/src directory and a symbolic link is created using the package name without the version number.

```
# cd /usr/src
# ftp ... <server where Samba is - get samba-2-2-7a.tar.gz in binary>
# tar xzf samba-2.2.7a.tar.gz
# ln -s samba-2.2.7a samba
# cd samba
# ls -F
COPYING   Read-Manifest-Now  docs/      pcp/     testsuite/
Manifest  Roadmap            examples/  source/
README    WHATSNEW.txt       packaging/ swat/
```

The Samba source code is in the subdirectory named source. Because the recipe file to build UNIX software, the makefile, often differs depending on the platform, it is common to create it with a script named configure, as follows:

```
# cd source
# ./configure --with-winbind --enable-cups
...
checking whether to use included popt... no
checking configure summary... yes
updating cache ./config.cache
creating ./config.status
creating include/stamp-h
creating Makefile
creating script/findsmb
creating include/config.h
# make
...
# make install
...
```

The final step will install the files under the directory /usr/local/samba/.

Integrating the new Samba

Because the newly built files are located in a different directory, the startup script /etc/init.d/smb needs to be modified. Now you need to point to the new Samba files. Save the original file:

```
# cd /etc/init.d
# cp smb smb-2.2.5
# vi /etc/init.d/smb
```

Change the following lines:

```
SMB_BIN=/usr/sbin/smbd
NMB_BIN=/usr/sbin/nmbd
SMB_CONF=/etc/smb.conf
SMB_PID=/var/lock/samba/smbd.pid
NMB_PID=/var/lock/samba/nmbd.pid
```

to this:

```
SMB_BIN=/usr/local/samba/bin/smbd
NMB_BIN=/usr/local/samba/bin/nmbd
SMB_CONF=/usr/local/samba/lib/smb.conf
SMB_PID=/usr/local/samba/var/locks/smbd.pid
NMB_PID=/usr/local/samba/var/locks/nmbd.pid
```

Figure 4 on page 20 shows a block diagram of the two versions of Samba.

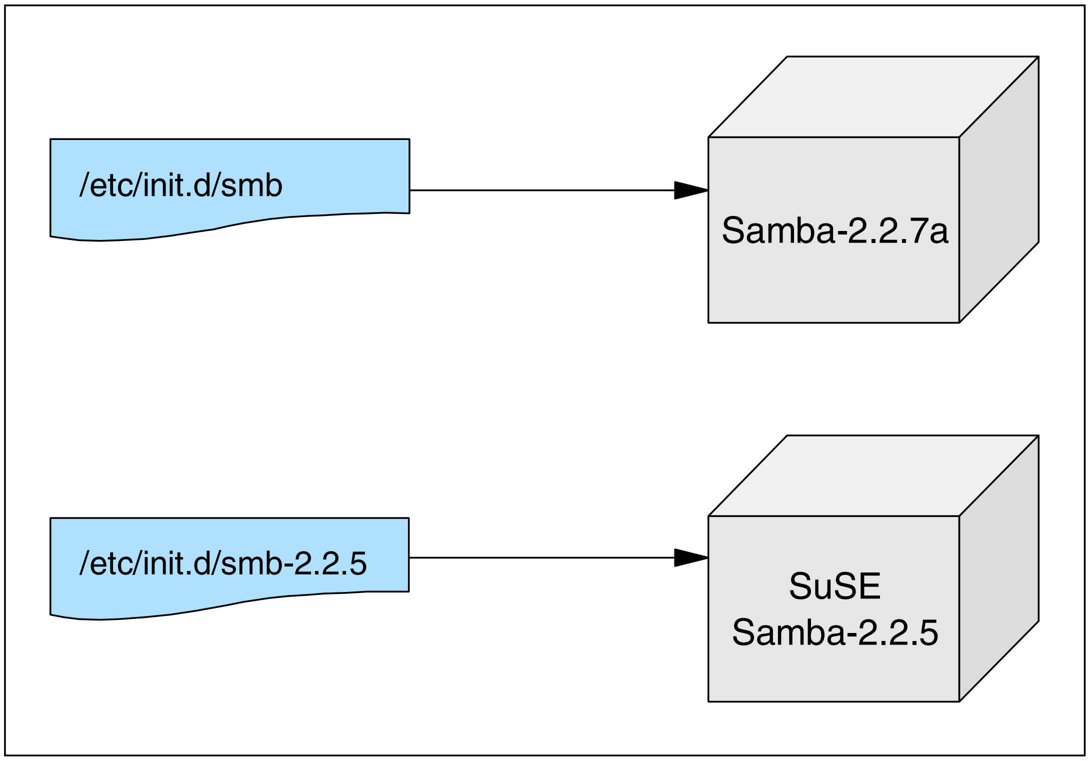


Figure 4   Integrating new Samba via modified smb script

Setting PATH and MANPATH

Now set up your PATH and MANPATH environment variables to pick up the /usr/local/samba/bin directory before the /bin or /sbin directories. This is so you will pick up the newly installed Samba executables and manual pages before any others.

```
# export PATH=/usr/local/samba/bin:$PATH
# export MANPATH=/usr/local/samba/man:$MANPATH
```

To check this:

```
# echo $PATH
/usr/local/samba/bin:/sbin:/usr/sbin:/root/bin:/usr/local/bin:/usr/bin:/usr/X11R6/bin:/b
in:/usr/games/bin:/usr/games:/opt/gnome/bin:/opt/kde2/bin
# echo $MANPATH
/usr/local/samba/man:/usr/share/man:/usr/local/man:/usr/X11R6/man:/usr/man:/usr/openwin/
man
```

You can use the **`which`** and **`man -w`** commands to verify that your changes went into effect:

```
# which smbd
/usr/local/samba/bin/smbd
# man -w smbd
/usr/local/samba/man/man8/smbd.8
```

To maintain these changes between reboots, add the export statements to the file /etc/profile.local:

```
# export PATH=/usr/local/samba/bin:$PATH
# export MANPATH=/usr/local/samba/man:$MANPATH
```

Now that there may be two Samba versions available to the system, it will be easy to forget which configuration files are which, especially for the two important files smb.conf and smbpasswd. To avoid confusion, set up symbolic links from the appropriate files under /usr/local/samba directory to the /etc directory:

```
# ln -s /etc/smb.conf /usr/local/samba/lib/smb.conf
# ln -s /etc/smbpasswd /usr/local/samba/private/smbpasswd
```

Start Samba

Start and stop Samba and check if the modified smb script works correctly.

```
# cd /etc/rc.d
# ./smb stop
Shutting down SAMBA nmbd :                                      done
Shutting down SAMBA smbd :                                      done
# ./smb status
Checking for SAMBA nmbd :                                       unused
Checking for SAMBA smbd :                                       unused
# ./smb start
Starting SAMBA nmbd :                                           done
Starting SAMBA smbd :                                           done
# ./smb status
Checking for SAMBA nmbd :                                       running
Checking for SAMBA smbd :                                       running
```

Important: SuSE SLES-8 has split the smb script into three scripts: smb, nmb, and winbind. This is probably a cleaner architecture in that it allows you to selectively start and stop the three main Samba daemons: smbd, nmbd, and winbindd.

Customizing and using Samba

After the desired version of Samba is set up on your Linux system, some customization may be needed. As previously mentioned, Samba uses a single configuration file, smb.conf, for global and specific setting. The topics that are addressed in this section are:

- Setting up the Samba Web Administration Tool (SWAT)
- Creating an smb.conf file
- Accessing a Samba share from Windows
- Performance considerations and tuning Samba

Setting up SWAT

The Samba Web Administration Tool (SWAT) allows administration of a Samba system via a Web browser. On the server side, it is a mini-Web server that is invoked via **inetd**. Even though the focus of this paper is a z/VM front-end, setting up SWAT is mentioned for completeness.

To use swat, it must be set up to be invoked via **inetd**. It is almost always set to well-known port 901 in the file /etc/services. Usually, it is commented out in the file /etc/inetd.conf. If so, inetd.conf has to be modified to point to the new code location:

```
# cd /etc
# grep swat services inetd.conf
services:swat           901/tcp                 # XXX Samba Web Adminisration Tool
inetd.conf:# swat is the Samba Web Administration Tool
# cp -p inetd.conf inetd.conf.orig
# vi inetd.conf
```

First, find “swat”, then un-comment the line and, if you have rebuilt from source, change the path to point to the new executable, as follows:

```
swat  stream  tcp    nowait.400     root    /usr/local/samba/bin/swat      swat
```

Restart **inetd** with the following command:

```
# rcinetd restart
Shutting down inetd                                                 done
Starting inetd                                                      done
```

Creating an smb.conf file

This smb.conf file can either be modified from an editor such as vi or from SWAT. One advantage of using SWAT is that there is a help interface directly available from the various screens. You should first update the global variables, as shown in Example 9.

Example 9 The smb.conf global variables

```
[global]
          workgroup = FILESERVER
          netbios name = LNXA                                                    1
          interfaces = 9.12.9.14/24                                              2
          bind interfaces only = Yes
          encrypt passwords = Yes
          map to guest = Bad User
          log level = 1
          max log size = 50
          max xmit = 8192
          deadtime = 15                                                          3
          keepalive = 30
          socket options = TCP_NODELAY IPTOS_LOWDELAY SO_SNDBUF=14596 SO_RCVBUF=14596  4
          os level = 2
```

The following are explanations of the details shown in the example.

1. Name of the Linux file server.
2. The IP address of the server, with a mask parameter of 24.
3. The number of minutes after which an inactive connection to Samba will be disconnected.
4. Suggested performance options.

The individual Samba shares will need specific parameters for the proper accesses. The automation tool will create these shares as outlined in Example 10.

Example 10 Samba file system share

```
[ourteam]
          comment = Residency - redp3604
          path = /data/ourteam
          force group = +ourteam        1
          read only = No
          create mask = 0770            2
          directory mask = 0770         2
          force directory mode = 02770  3
```

The following are explanations of the details shown in the example.

1. This sets the group ourteam to be assigned as the default primary group for all users connecting to this service. This will maintain the ourteam group authority.
2. Files and subdirectories will have full permissions for the owner and group, but no access to others. If you want other teams to have read and execute authority, use the value 0775.

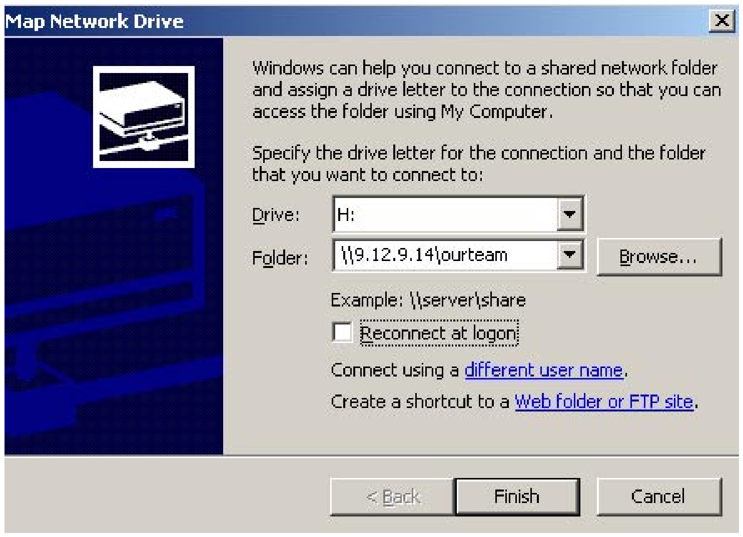
3. Any subdirectory structure created will inherit the main directory permissions. The 2 in the 02770 parameter sets the “set group ID” bit which, when set on a directory, ensures the permissions are inherited to subdirectories created below it.

Accessing Samba shares from Windows

Within a Windows environment, many editing programs and other file manipulation works best when the file system is *network-mapped* as a drive on your Windows client.

Once a Linux user ID and password have been created and the id has been authorized to the group, the Windows client should be able to map to the group’s file system. There are several ways to network-map the file system—for example, by using My Network Places or Network Neighborhood, or from Windows Explorer. In addition, the DOS `net use` command can be used.

For this example, Windows Explorer is used. Choose the **Tools** menu and select **Map Network Drive...**. A dialog similar to that shown in Figure 5 will appear. Specify the Microsoft Universal Naming Convention (UNC) of \\serverName\shareName. Windows NT/2000/XP clients can used the TCP/IP address as the server name; however, Windows 95/98/ME clients must use a NetBIOS name (in which case you might want to use the LMHOSTS file).


Map Network Drive

Windows can help you connect to a shared network folder and assign a drive letter to the connection so that you can access the folder using My Computer.

Specify the drive letter for the connection and the folder that you want to connect to:

Drive: H:

Folder: \\9.12.9.14\ourteam

Example: \\server\share

☐ Reconnect at logon

Connect using a different user name.

Create a shortcut to a Web folder or FTP site.

Figure 5 The Map Network Drive dialog

If the Linux user ID and password is different than the Windows sign-on id and password, you can select **Connect using a different user name** and log on from that screen.

In this example, once connected, the ourteam file system will be connected as your H: drive.

Installing and customizing the EZLNXID tool

To set up EZLNXID, the following high-level steps are done:

- Be sure planning is done - see “Planning” on page 24.
- Obtain the code - see “Obtaining the EZLNXID code” on page 24
- Install the z/VM front-end on CMS - see “Installing EZLNXID on the z/VM host” on page 24.

- Customize the z/VM front-end - see “Customizing the z/VM front-end” on page 25.
- Install the back-end on Linux - see “Installation on the Linux guest” on page 27.
- Customize the Linux back-end - see “Customizing the Linux back-end” on page 27.

Planning

Before installing the EZLNXID tool, initial setup of the z/VM environment and the Linux guest is required. Be sure these steps have been done:

- “Preparing the z/VM host” on page 7 discusses how to configure the z/VM environment for EZLNXID.
- “Preparing the Linux guest” on page 10 discusses how to configure a Linux guest to act as a Samba server. Detailed steps to set up Logical Volume Manager are covered.
- “Linux distributions and Samba versions” on page 17 discusses which Samba versions are provided by several zSeries Linux distributions. A detailed procedure to build Samba from source is provided.
- “Customizing and using Samba” on page 21 discusses how to configure Samba when running on Linux for zSeries.

Obtaining the EZLNXID code

The EZLNXID tool described in this Redpaper is available on the Web at:

```
ftp://www.redbooks.ibm.com/redbooks/REDP3604/
```

The code is packaged in two components; a VMARC file with code that runs on CMS and a tar file with code that runs on Linux:

- ezlnxid.vmarc

  This contains the REXX execs that comprise the EZLNXID front-end in VMARC format. These will be installed on the CMS guest used to manage the Samba server (virtual machine LNXADMIN in Figure 2 on page 6).

- ezlnxid.tar.gz

  This contains the Linux scripts that comprise the EZLNXID back-end in zipped tar format. These will be installed on the Linux guest which runs the Samba server (virtual machine LNXA in Figure 2 on page 6).

  Download both packages to a machine that is preferably running an FTP server.

Installing EZLNXID on the z/VM host

To install the EZLNXID front-end on a z/VM system, you need to obtain and install the VMARC package. If it is not on your system, you can get it from the z/VM Download Library Web site:

```
http://www.vm.ibm.com/download
```

On that page, look for the heading “To download VMARC itself” which points to the module:

```
http://www.vm.ibm.com/download/vmarc.module
```

Once you have downloaded it, upload the file to VM in binary and run the file through this pipeline:

```
PIPE < VMARC MODULE A | deblock cms | > VMARC MODULE A
```

Install the EZLNXID front-end as follows:

1. Using **FTP** on z/VM, copy the ezlnxid.vmarc package in binary with fixed 80 byte records to the 191 minidisk of the LNXADMIN user.

```
# ftp <your freeware server>
ftp> cd <correct directory>
ftp> bin fix 80
ftp> get ezlnxid.vmarc
ftp> quit
```

2. Unpack the file EZLNXID VMARC using the VMARC command. You should see output similar to the following:

```
VMARC UNPACK EZLNXID VMARC A = = A
EZLNXID  EXEC     A2. Bytes in=   12720, bytes out=   31874 (   250%).
EZLNXID  HELPCMS  A2. Bytes in=    1520, bytes out=    1993 (   131%).
EZLNXID  PACKAGE  A2. Bytes in=    1280, bytes out=    2240 (   175%).
EZLNXID  XEDIT    A2. Bytes in=    1600, bytes out=    2216 (   138%).
EZLNXIDC XEDIT    A2. Bytes in=    1920, bytes out=    2975 (   154%).
EZLNXIDS XEDIT    A2. Bytes in=    4880, bytes out=   10515 (   215%).
EZLNXPFX XEDIT    A2. Bytes in=    1360, bytes out=    1786 (   131%).
EZZVM    EXEC     A1. Bytes in=    1200, bytes out=    1868 (   155%).
```

The EZLNXID front-end is now ready for customization.

Customizing the z/VM front-end

On the z/VM side, the automation tool consists of the following files (described in the EZLNXID PACKAGE); the file package list is shown in Example 11.

Example 11 The EZLNXID PACKAGE file

```
**********************************************************************
* :nick.EZLNXID        :sec.none                      :disk.any
* :title.Allow a priviledged user to Secuser to a linux system.
* :version.1.0.0       :date.2002-11-08 :sco.VM
* :oname.                               :onode.        :ouser.
* :aname.IBM International Technical Support Organization/Poughkeepsie
* :support.none        :doc.n/a
* :req.n/a
* :lic.
* This program is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Public License as published by
* the Free Software Foundation; either version 2 of the License, or
* (at your option) any later version.
* This program is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
* GNU General Public License for more details.
**********************************************************************
* Linux fileserver support using secuser - files on z/VM system
      EZLNXID  EXEC    *  Main Linux interface exec.
      EZLNXID  HELPCMS *  Help
      EZLNXIDS XEDIT   *  Xedit Profile macro
      EZLNXID  XEDIT   *  macro to select userids from a list of ids
      EZLNXPFX XEDIT   *  macro to add prefix commands S and SS-SS
      EZLNXIDC XEDIT   *  macro to execute  command at cursor location
      EZZVM  EXEC      *  secuser to linux front end
* file on the Linux system
*     EZLINUX.REXX     *  Main Linux interface script
```

The variables you have to adjust for your system environment are at the beginning of the EZLNXID EXEC. Example 12 shows the relevant lines to customize.

Example 12 Variables for customizing the EZLNXID EXEC

```
....
/*******************************************************************/
/* Change the following to match your system and linux userid.       */
/*                                                                   */
system='LNXA'                          /* Linux system VM userid */       1
server='9.12.9.14'                     /* Linux system IP-address*/       2
share ='data'                          /* Linux system share name*/       3
tools ='/usr/local/ezlnxid/'           /* Linux ezlinix tool location */  4
ownertext='Residency -'                /* default Samba comment text*/    5

/* If you have more then one userid administering the linux file    */
/* system space, set up the VM secuser execs, log and userid lists  */
/* on a shared sfs directory or cms mini-disk.                      */
sfs='No'                    /* If Yes, VM secuser execs are on sfs */     6
directory=''                    /* sfs where secuser execs reside  */     7
/*sfs='No'*/                /* If No, VM secuser execs are on mini */
minidisk='LNXADMIN 191'     /* minidisk where secuser execs reside */     8
/*                                                                   */
/*******************************************************************/
....
```

The variables to customize are:

1. system

   The name of the z/VM user ID where your Linux system is running.

2. server

   The IP address of the Linux system.

3. share

   The file system directory where the Samba data resides.

4. tools

   The directory where the automation tool script, ezlinux.rexx, resides on the Linux system.

5. ownertext

   The default text for the comment area of the Samba share (optional).

The next set of variables specify the type of z/VM file system on which EXLNXID is stored:

6. sfs

   Specify `sfs=Yes` if the EZLNXID PACKAGE files are on an SFS directory.

7. directory

   Specify the directory path if SFS is used.

8. minidisk

   Specify the user ID and minidisk address of EZLNXID if a CMS minidisk is used. Otherwise, leave it blank.

Installation on the Linux guest

The EZLNXID script (or REXX exec) requires the Regina REXX interpreter, which is often not automatically installed. You can determine if Regina is installed with the **rpm -q** command:

```
# rpm -q regina
package regina is not installed
```

If it is not installed, you can again use the **rpm** command; or, on SuSE, you can use **yast** to do the install the Regina package:

1. Select **Package Management (Update, Installation, Queries)**

   Provide the package location information and press F10.

2. Choose **Change or create configuration**.

   Select **Development (C, C++, lisp, etc)** and choose **regina (The Regina REXX interpreter)**. Complete the selection by pressing F10.

3. Choose **Start installation**.

   Watch for the message: `INSTALLATION COMPLETE`.

4. Complete installation by selecting **Main Menu** and then selecting **Exit Yast**.

Now you can see that Regina is installed.

```
# rpm -q regina
regina-3.0-17
```

Now the EZLNXID package can be installed. Using **ftp**, download the ezlnxid.tar.gz package in binary and extract the Linux REXX execs using the **tar** command. This will create the ezlnxid subdirectory:

```
# cd /usr/local
# ftp <your FTP server>
ftp> cd <correct directory>
ftp> bin
ftp> get ezlnxid.tar.gz
ftp> quit
# tar -zxf lnxezid.tar.gz
# ls
ezlnxid/  ezlnxid.tar.gz
```

The EZLNXID back-end is now ready for customization.

Customizing the Linux back-end

In this example, the scripts are extracted, then copied to the /home/tools directory. The directory contains the ezlinuxid.rexx script. This calls REXX subroutines to execute the appropriate Linux commands. Customizations to the ezlinux.rexx script are shown in Example 13.

Example 13 The variables for customizing the ezlinux.rexx script

```
...
locshare = '/data/'                        /* directory for samba shares    */     1
locsmbpw = '/usr/local/samba/private/'     /* directory of smbpasswd file   */     2
locsmbconf = '/usr/local/samba/lib/'       /* directory of smb.conf file    */     3
...
```

Three variables are modified:

- locshare

  This points to the root directory containing Samba shared drives.
- locsmbpw

  This points to the smbpasswd file.
- locsmbconf

  This points to the directory where the smb.conf file is located.

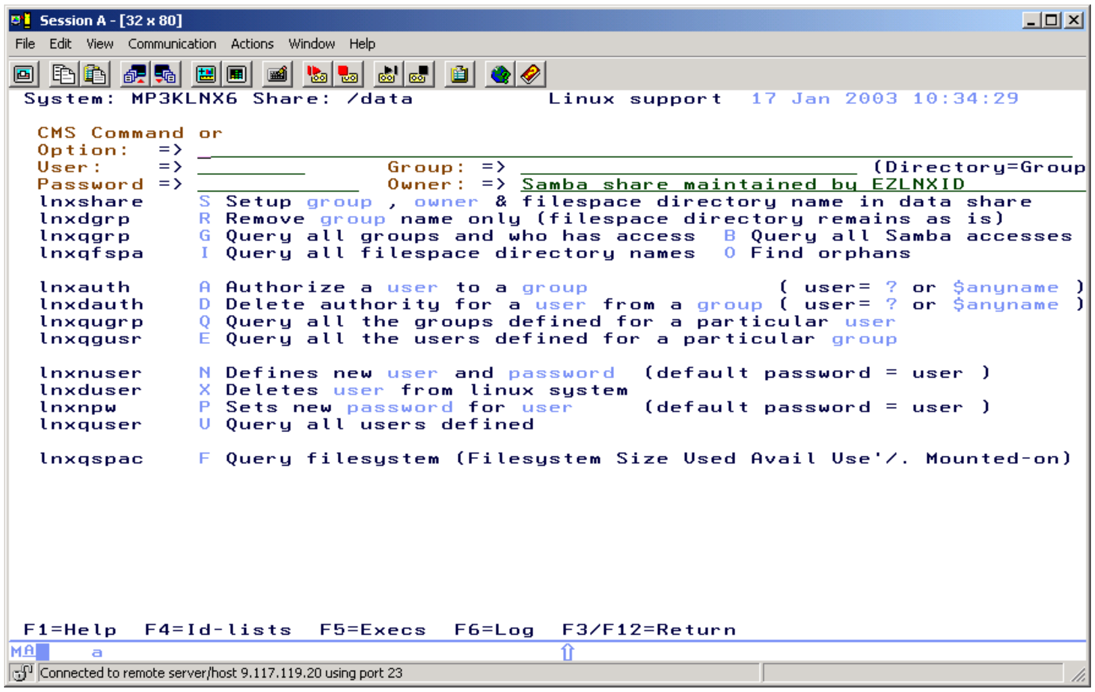
Customization is now complete.

Using EZLNXID

Now that both the CMS and Linux pieces of EZLNXID are in place and customized, you should be ready to run EZLNXID from CMS. Remember to have root logged on to the Linux console; it is okay if the machine is disconnected.

To invoke EZLNXID, log on to the LNXADMIN user ID and enter the **EZLNXID** command. If your system has IOS3270, the main panel should look like the one shown in Figure 6.

If commands are being sent to the console, be sure root is logged on to the console as described in "Automatic logon to Linux as root" on page 16 (it is okay if root is logged on and disconnected).



```
 System: MP3KLNX6 Share: /data           Linux support  17 Jan 2003 10:34:29

  CMS Command or
  Option:    => _
  User:      => __________       Group:  => _______________________________ (Directory=Group
  Password   => ______________   Owner:  => Samba share maintained by EZLNXID
  lnxshare      S Setup group , owner & filespace directory name in data share
  lnxdgrp       R Remove group name only (filespace directory remains as is)
  lnxqgrp       G Query all groups and who has access   B Query all Samba accesses
  lnxqfspa      I Query all filespace directory names   O Find orphans

  lnxauth       A Authorize a user to a group            ( user= ? or $anyname )
  lnxdauth      D Delete authority for a user from a group ( user= ? or $anyname )
  lnxqugrp      Q Query all the groups defined for a particular user
  lnxqgusr      E Query all the users defined for a particular group

  lnxnuser      N Defines new user and password   (default password = user )
  lnxduser      X Deletes user from linux system
  lnxnpw        P Sets new password for user      (default password = user )
  lnxquser      U Query all users defined

  lnxqspac      F Query filesystem (Filesystem Size Used Avail Use'/. Mounted-on)




 F1=Help  F4=Id-lists  F5=Execs  F6=Log  F3/F12=Return
```

Figure 6 EXLNXID main panel on system with IOS3270

If your system does *not* have IOS3270, the main panel should look like the one shown in Example 14 on page 29. Actions to perform are then selected from the EZLNXID main menu.

Example 14 The EZLNXID main menu

```
System: LNXA Share: /data  EZLNXID Linux support 17 Dec 2002 12:03:09
====>
Please enter an Option and hit enter.

Option:  => _                                                            1
User:    => ________                                                     2
Password => ____________                                                 3
Group:   => _________________________                                    4
Owner:   => _____________________________________________                5

* * * Top of File * * *
 lnxshare   S Setup "group" , "owner" & filespace directory name in  a share
 lnxdgrp    R Remove "group" name only (filespace directory remains as is)
 lnxqgrp    G Query all groups and who has access
            B Query all Samba accesses
 lnxqfspa   I Query all filespace directory names
            O Find orphans

 lnxauth    A Authorize a "user" to a "group"              ( user=? or $anyname
 lnxdauth   D Delete authority for a "user" from a "group" ( user=? or $anyname
 lnxqugrp   Q Query all the groups defined for a particular "user"
 lnxqgusr   E Query all the users defined for a particular "group"
 lnxnuser   N Defines new "user" and "password"  (default password = user )
 lnxduser   X Deletes "user" from linux system
 lnxnpw     P Sets new "password" for "user"     (default password = user )
 lnxquser   U Query all users defined
 lnxqspac   F Query filesystem (Filesystem Size Used Avail Use'/. Mounted-on)

 1= Help      2= .         3= QUIT      4= Id Lists  5= Lnx exec  6= Lnx Log
 7= BACKWARD  8= FORWARD   9= .        10= Lnx Msgs 11= All Msgs 12= QUIT
```

Parameters supplied to the EZLNXID tool are explained below:

1. `Option` indicates what action is to be performed.

   Options are specified using a single character chosen from the displayed list. Associated with each option is the name a REXX script used to execute the action.

2. **`User`** indicates a Linux user to operate on.
3. `Password` indicates a password to assign to a user.
4. `Group` indicates a Linux group name to operate on.
5. `Owner` indicates a comment used to identify the shared Samba file system.

A complete list of all actions, the REXX script name that implements the action, the parameters required by the script, and a brief description of the action is summarized in Table 4.

Table 4 EZLNXID actions and required parameters

Script	Option	Purpose	Parameters
lnxshare	S	Create a new Samba shared drive.	group
			owner
lnxdgrp	R	Delete a group	group

Script	Option	Purpose	Parameters
lnxqgrp	G	List all Linux groups	-
	B	List all Linux groups with access to any Samba shared drive.	-
lnxqfspa	I	List all Samba shared drives.	-
	O	List all ophan Samba shared drives.	-
lnxauth	A	Authorize a user to a Samba shared drive.	userid
			group
lnxdauth	D	Remove user access to a Samba shared drive.	userid
			group
lnxugrp	Q	List all Samba shared drives to which user is authorized.	userid
lnxgusr	E	List all users authorized to use a Samba shared drive.	group
lnxnusr	N	Create a new Samba user.	userid
			password
lnxdusr	X	Delete a Samba user.	userid
lnxnpw	P	Assign new password to user.	userid
			password
lnxquser	U	List all defined users.	-
lnxqspac	F	List all Samba shared drives.	-

The main panel of EZLNXID should appear similar to Example 14 on page 29. If you choose an option without the necessary parameters, you get a message that tells you what parameters are missing. The Samba-related functions are explained on the panel.

There are several function keys on the main panel.

- F1 = Help

  Shows a help panel where all the functions provided by this tool are described.
- F3 = QUIT

  With this key you leave the tool and go back to CMS.
- F4 = Id List

  Lists the user IDs that can be given authority to a group.
- F5 = Lnx exec

  Shows a list of all EXECs of this tool. Performs a `FILEL * *EXEC* <fm>` where `<fm>` is the file mode of the minidisk of this EXEC.
- F6 = Lnx Log

  Shows the log of activity on the Linux system that is stored on z/VM
- F7 = BACKWARD

  Show the previous panel.

- F8 = FORWARD

  Show the next panel.

- F10 = Lnx Msgs

  Shows only the returned Linux **SECUSER** messages that have a LNX prefix.

- F11 = All Msgs

  Shows all the returned Linux **SECUSER** messages.

- F12 = QUIT

  Same as F3.

- Function O = Find orphans

  With this function you can find potential “orphans” in the system. Since this Linux system is set up to have groups associated with Samba shares and file systems, this function checks to see if any one of those are not associated with the others. If orphans are found, text is displayed on how to resolve it.

- Selection from user ID list

  Within the “add and delete group authorization” for a user functions, rather than having to enter each user ID separately, you could select from a list of user IDs already defined to your Linux system.

  After filling in the option and group fields, enter a question mark (`?`) in the userid field. This will display an XEDIT screen listing all the user IDs. Select all the ones you want to include by placing an `s` to the left of the user IDs, then type: `quit` on the command line.

Managing Samba shared drives with EZLNXID

To illustrate how to manage Samba shared drives using EZLNXID, we use a simple example.

Creating a new Samba user

We first create a new Samba user, *klaus*, by using option `N` from the EZLNXID main menu. The User and Password parameters are required. The resultant /etc/passwd file is shown in Example 15.

Example 15 User definitions in /etc/passwd

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/bin/bash
daemon:x:2:2:daemon:/sbin:/bin/bash
.
.
.
klaus:x:510:100::/home/klaus:/bin/false
```

Creating a Samba shared drive

Next, we create a Samba shared drive named redp3604 using option `S` from the EZLNXID main menu. The Group parameter is assigned *redp3604*, and the Owner parameter is assigned *Residency redp3604*.

The resulting /etc/smb.conf file is shown in Example 16.

Example 16 Samba shared directory definition in /etc/smb.conf

```
[redp3604]                                  1
          comment = Residency redp3604      2
          path = /data/redp3604             3
          force group = +redp3604           4
          read only = No                    5
          create mask = 0770                6
          directory mask = 0770             7
          force directory mode = 02770      8
```

The example is explained below:

1. The Samba shared drive is named `redp3604`.

   The name is derived from the Group parameter presented on the EZLNXID main menu. Clients see the shared drive name when browsing the Samba server.
2. The comment field associates descriptive text to the shared directory.

   The comment is derived from the Owner parameter presented on the EZLNXID main menu. Clients can see the comment field when requesting details on the shared drive.
3. The path name of the shared directory is `/data/redp3604`.

   The Group parameter is used to name the subdirectory defined as a Samba shared drive.
4. The force group option explicitly sets the Linux group owner to be `redp3604`.

   File created by Samba clients in the redp3604 shared drive will have group ownership set to the redp3604 group.

   The `+redp3604` allows only users in **group redp3604 to hav**e access to this shared drive.
5. The `read only = No` allows Samba clients to create files in the shared drive.
6. The create mask option specifies the maximum allowable permissions when Samba creates files in the shared drive.

   Using the `0770` value creates files with permissions -rwxrwx---.
7. The directory mask option specifies the maximum allowable permissions when Samba creates directories in the shared drive.

   Using the `0770` value creates directories with permissions drwxrwx---.
8. The force directory mode option specifies bits that will be ORed with the directory mask option when creating directories in the shared drive.

   Using the `02770` value creates directories with permissions drwxrws---.

Note: The 02770 value turns on the permission s-bit when creating a subdirectory. This ensures files and directories in the subdirectories will inherit the shared drive group ownership.

Granting access to a Samba shared drive

Finally, we grant access to the redp3604 shared drive to user klaus using option `A` from the EZLNXID main menu. The User (klaus) and Group (redp3604) parameters are required.

The resultant /etc/group file is shown in Example 17.

Example 17 Group definitions in /etc/group

```
root:x:0:root
bin:x:1:root,bin,daemon
daemon:x:2:
.
.
.
users:x:100:roy,klaus
redp3604:x:101:klaus
```

In the example, we see the user klaus has been added to group redp3604.

The z/VM EXECs and Linux scripts

The z/VM EXECs reside on a specially created z/VM user ID (LNXADMIN), which has user class C to enable it to use the **SECUSER** command.

The functions you invoke from the main panel call subroutines of the EZLNXID exec. These subroutines use **CP SECUSER** and **CP SEND** to pass prompts to the Linux user ID, LNXA, to run subroutines in the ezlinux.rexx script.

Table 5 EZLNXID subroutines

z/VM routine	Linux routine called	Description
LNXUSER	mknewuser	Creates a new user and password. The parameters are <user> and <password>. The **useradd** command is invoked with the flag -s /bin/false, so that it is not possible to log on to this user ID. The **chpasswd** command sets the Linux password, then the **smbpasswd** command creates the Samba password for this user to the value that was given with the command.
LNXNPW	mknewpasswd	Changes the password for an existing user. The parameters are <user> and <password>. The **chpasswd** command resets the Linux password, and the **smbpasswd** resets the Samba password.
LNXSHARE	mknewshare or mksmbshare	Creates a file system and group for a new share. The parameters are <share name> and <owner>. The **groupadd** command creates a new Linux group with the name as the share given with the command. The **mkdir** command creates a new directory also with the same name. The **chmod** command sets the permission bits to 2770 (again, the leading 2 sets the “group setuid bit”). Finally, the share is appended to the smb.conf file.

z/VM routine	Linux routine called	Description
LNXAUTH	mkgroupauth	Authorizes a user to a group. The parameters are <user> and <group>. The command `usermod -G` adds the specified group to the user's list of groups.
LNXQUGRP	listusergroups	Lists group information for a user. The parameter is the <user>. The contents of the file /etc/group that contain the specified user is displayed.
LNXQGRP	listgroups	Lists group information for a user. There are no parameters passed. The contents of the file /etc/group is sorted and displayed.
LNXQSPAC	none	Lists file system usage. Just the output of the command `df -h` is displayed.
LNXQUSER	listusers	Lists defined Samba users. There are no parameters. All user information with UIDs greater than 100 is obtained from /etc/passwd and displayed.
LNXDAUTH	delgroupauth	Deletes user authorization to a group. The parameters are <user> and <group> to be removed. The `usermod -G` command is used to remove the specified group from the user's list of groups
LNXDUSER	deluser	Deletes a user from the system. The parameter is < user>. The `userdel` command deletes the specified user, and the `smbpasswd -x` command deletes the Samba password from the smbpasswd file.
LNXDGRP	delgroup	Deletes a group from system. The required parameter is <group>. The `groupdel` command deletes the group from the /etc/group file.
LNXQFSPA	listfile system	Lists all Samba file system. There are no parameters. The command `ls -l /data` is invoked.
LNXQGUSR	listgroupusers	Lists all users in a group. The required parameter is <group>. Information about all users in the specified group that are defined in the group.
LNXQSMB	listsmbshare	Lists Samba share definition. There are no required parameters. It contains information about all defined Samba shares is obtained from the smb.conf file.

Other considerations

There are other factors to consider that are beyond of the scope of this paper to examine in detail. They are listed here and briefly discussed for the sake of completeness.

- Security
- Data backup and recovery
- File locking
- Printing
- Performance and tuning

Security

There are also many factors to consider with respect to security. The following aspects of security are discussed here.

- Data encryption
- Client access
- The **CP SECUSER** command
- Access to the EXECs and scripts

Data encryption

Passwords are encrypted over the network; however, SMB data is not. SMB can be tunneled over SSH—which does encrypt the data, but also negatively affects performance. There are also technical issues with this solution when using Windows 95/98/ME clients; refer to the following URL for more information:

`http://hr.uoregon.edu/davidrl/samba.html#ssh`

Client access

In addition to user authorization, you can also specifically limit the clients who have access to your Samba server. This is done via the hosts allow and hosts deny smb.conf parameters. By default, all network users have access, based of course on authorization.

CP SECUSER command security

As mentioned in “Using the CP SEND command” on page 8, the LNXADMIN user is granted class CP class C privilege in order to execute the **SET SECUSER** command. This may raise security concerns:

- CP class C is designated for *systems programmer* use and allows updates or changes to system-wide parameters of the z/VM system.

  Giving your administration machine (LNXADMIN, in this case) class C privilege also allows the user to perform all other class C CP commands.

- It also exposes the possibility that other user IDs with class C privilege can issue the **SET SECUSER** command to your Linux server (and thereby manipulate its environment).

To address these concerns, we recommend assigning the **SECUSER** command a new CP class. z/VM enables this through use of the **MODIFY COMMAND** statement in the SYSTEM CONFIG file, or by using the CP **MODIFY COMMAND** command.

For example, a new CP class designated class L may be defined. The **SET SECUSER** command can then be assigned to class L. In order to issue the **SET SECUSER** command, a user must have class L privilege.

Note: Remember to grant the new class privilege to all users requiring access to the command.

To define a new command class and restrict access to a specific command, add the lines shown in Example 18 to the SYSTEM CONFIG file located on the MAINT CF1 disk.

Example 18   Definition of a new CP command class in the SYSTEM CONFIG file

```
......
MODIFY CMD SET SUBCMD SECUSER,  1
    IBMCLASS C,                 2
    PRIVCLASS L,                3
......
```

In Example 18, the CP **SET SECUSER** subcommand (as noted in line 1) assignment to privilege class C (as noted in line 2) is overridden and instead assigned to privilege class L (as noted in line 3). Note that class A users retain access to the command (only class C privilege is overridden). For complete details on using the MODIFY COMMAND statement, see *z/VM V4R3 CP Planning and Administration*, SC24-6043.

Note: This maintains the command modification across system IPLs. However, a system IPL is required to implement the change.

To effect the change immediately, use the CP **MODIFY COMMAND** command:

```
CP MODIFY COMMAND SET SUBCMD SECUSER IBMCLASS C PRIVCLASS L
```

To enable a user to issue the **SET SECUSER** command, modify the USER DIRECTORY entry, adding the new privilege class (shown highlighted) as in Example 19.

Example 19   Authorizing a user to an overridden privilege class

```
USER LNXADMIN LNXADMIN 32M 64M LG
 INCLUDE IBMDFLT
 MACH ESA
 IPL CMS
 MDISK 191 3390 3189 010 430W01  MR
```

Details on the **MODIFY COMMAND** command can be found in *z/VM V4R3 CP Command and Utility Reference*, SC24-6008.

Access to the EXECs and scripts

The automation EXECs on z/VM should reside on an administrator's disk that is not accessible to other users on the system. This is accomplished by not defining link passwords in the USER DIRECTORY for that disk. If you are using an external security manager, such as RACF, then access should be appropriately restricted. Normal z/VM logon password security should restrict logon access to the EXECs.

On the Linux system, the file system where the Linux REXX scripts are located should have restricted access through the Linux permissions. The directory and scripts should only allow access to root as owner. This can be done by issuing the Linux commands:

```
# chmod 0700 /usr/local/ezlnxid
# chmod 0700 /usr/local/ezlnxid/*
```

The results should appear as shown in Example 20 on page 37.

Example 20   Permissions for the EZLNXID Linux scripts

```
drwx------    3 root     root         4096 Nov 12 15:30 .
drwx------    6 root     root         4096 Nov  1 12:19 ..
-rwx------    1 root     root        15013 Nov 12 13:02 ezlinux.rexx
```

Data backup and recovery

When using a Linux on zSeries and S/390 as your Samba server, establishing a proper data backup process is important. This should be addressed at two levels:

- Disaster recovery - How do I restore my system if my entire site is wiped out?
- Incremental backup - How do I recover a file quickly if it is accidentally deleted or corrupted?

There are several products available for use with Linux on zSeries and S/390, including:

- UTS Global Backup and Restore (BAR)

  `http://www.utsglobal.com/linuxprod.html`

- Tivoli Storage Manager from IBM

  `http://www.tivoli.com/products/index/storage-mgr/`

- Innovation Data Processing FDR/UPSTREAM

  `http://www.innovationdp.fdr.com/ups.cfm`

- SecureAgent's SecureBackup

  `http://www.secureagent.com/securebackup/`

- Legato Networker

  `http://www.legato.com/products/networker/index.cfm`

- Open Source Software - Amanda

  `http://www.amanda.org/`

For further details, refer to *Linux on zSeries and S/390: Systems Management*, SG24-6820.

File locking

Samba does not do file locking. It is possible to use the lockd daemon on Linux, but this is not commonly used. The SMB file system, however, does have a locking mechanism.

Printing

Samba is often called a print server, but that really is a misnomer because a print server must first be established. Traditionally, the lpd package has been the print server on UNIX. Today there are two new packages vying for that de facto standard:

- lpr next generation (LPRng)

  LPRng follows the lpd architecture more closely, especially because it has one main configuration file called /etc/printcap.

- Common UNIX Printing System (CUPS)

  CUPS uses the new Internet Printing Protocol (IPP).

Performance and tuning

In this section, we include general observations and recommendations from the IBM Linux Scalability Center (LSC). Also included are recommendations derived from general Samba usage over time, as well as some unofficial performance tests. Keep in mind that “your mileage may vary”, so you may want to test performance yourself.

General observations by the LSC

Some observations are as follows:

- With a single Gbe OSA card, up to 25 guests with one concurrent request each and an aggregate throughput of 13.37 MB/second could be supported. Maximum OSA throughput was reached between 12-15 SMB processes.
- With a single guest server and a single OSA card, we were able to support up to 30 concurrent users at an aggregate throughput of 19.4 MB/second.
- Summary of native results vs. VM guest results. The cost in throughput between the 2.4.17 kernel in a native LPAR vs. running the timer change version of this kernel on z/VM is most significant with small numbers of guests.
  - Cost for the first 1 to 10 guests was 20 to 26% total.
  - Cost for 15 to 35s guest was only 9.7 to 16% total.
- SuSE SLES 7 throughput is not as good as with an internal kernel. At 35 connections, the 2.4.17 Timer kernel produced up to 125% improvement over SuSE SLES 7. However, it is likely that significant improvement would be seen in SuSE SLES 8 vs. SLES 7.

Recommendations by the LSC

Some Samba performance recommendations are as follows:

- Network configuration for communications

  It is recommended to use direct connections from OSA gigabit ethernet (or fast ethernet, if gigabit ethernet is not an option) cards to each Samba server guest. If the configuration must use connections to one or more Linux guests used as a router, the recommendations for routing are as follows:

  - Use VM guest LAN rather than VCTC to connect the guests to the Linux system that is providing routing capability, or
  - Use VM TCP/IP routing from the OSA card to the guest servers.

  Note: Having software routers in the path adds a degree of flexibility (such as the ability to add firewalls), but at a significant cost in throughput, CPU utilization, and increased response times.

- Recommended virtual memory size

  The recommended virtual memory size for Samba guests is 128 MB. If a large number of guests are to be implemented, the goal should be to keep the Samba server virtual memory size as small as possible, while avoiding Linux guest paging. Individual results, of course, depend on the user’s configuration.

- Real memory needed per guest

  To avoid paging, 128 MB per guest is recommended.

- Minidisk caching

  Minidisk caching of the data files used by Samba proved to be of little value and used a large amount of VM storage. It is recommended that you not using minidisk caching for Samba VM guests.

- SuSE SLES 7 QDIO

  The SuSE SLES 7 (2.4.7 kernel) QDIO communications provided significantly less throughput than the 2.4.17 kernel. With the changes provided in the 2.4.17 kernel and device drivers, the Internal throughput improved by up to 100% in a heavily loaded guest, and by approximately 15% in a lightly loaded multiple guest configuration.

- QETH and QDIO device drivers

  It is recommend that you download the latest QETH and QDIO device drivers supported for your kernel from IBM developerWorks to ensure that the installation is running at peak performance.

Other observations

The following smb.conf settings *may* give you better performance:

```
max xmit = 8192
socket options = TCP_NODELAY IPTOS_LOWDELAY SO_SNDBUF=14596 SO_RCVBUF=14596
dead time = 10
```

Additional material

This Redpaper refers to additional material which can be downloaded as described in the following sections.

Locating the Web Material

The Web material associated with this Redpaper is available in softcopy on the Internet from IBM Redbooks Web Server. Point your browser to:

```
ftp://www.redbooks.ibm.com/redbooks/REDP3604/
```

Alternatively you can go to the IBM Redbooks site at:

```
ibm.com/redbooks
```

Select Additional materials and open the directory that corresponds to Redpaper form number REDP3604.

Using the Web material

The additional material that accompanies this Redpaper includes:

- **ezlnxid.tar.gz** - the Linux scripts that comprise the EZLNXID back-end in zipped tar format
- **ezlnxid.vmarc** - the REXX scripts that comprise the EZLNXID front-end in VMARC format

System requirements for downloading the Web Material

The following system configuration is recommended:

Hard disk space:	1MEG
Operating System:	z/VM 4.3 and a Linux for zSeries distribution

To install the EXLNXID tool, follow the instructions outlined beginning in “Planning” on page 24.

References

ITSO publications

- *Building Linux Systems Under IBM VM*, REDP0120

  http://www.ibm.com/redbooks/abstracts/redp0120.html
- *Linux for S/390*, SG24-4987

  http://www.ibm.com/redbooks/abstracts/sg244987
- *Linux on IBM @server zSeries and S/390: ISP/ASP Solutions*, SG24-6299

  http://www.ibm.com/redbooks/abstracts/sg246299.html
- *Linux on IBM @server zSeries and S/390: Distributions*, SG24-6264

  http://www.ibm.com/redbooks/abstracts/sg246264.html
- *Linux on IBM @server zSeries and S/390: Large Scale Linux Deployment*, SG24-6824

  http://www.ibm.com/redbooks/abstracts/sg246824.html
- *zSeries HiperSockets*, SG24-6816

  http://www.ibm.com/redbooks/abstracts/sg246816.html
- *Linux on zSeries and S/390: Systems Management*, SG24-6820

  http://www.ibm.com/redbooks/abstracts/sg246820.html
- *Linux on IBM @server zSeries and S/390: TCP/IP broadcast on z/VM Guest LAN*, REDP3596

  http://www.ibm.com/redbooks/abstracts/redp3596.html

z/VM Documentation

- *z/VM V4R3 CP Command and Utility Reference*, SC24-6008
- *z/VM V4R3 CP Planning and Administration*, SC24-6043

Other resources

- *Using Samba*, Robert Eckstein, et al., O’Reilly Publishers, 1999, ISBN 1-56592-449-5
- *The Unofficial Samba HOWTO* by David Lechnyr, found at:

  http://hr.oregon.edu/davidlr/samba.html
- *Patching the Linux/390 kernel*, found at:

  http://linuxvm.org/Info/HOWTOs/krnpatch.html
- *LVM - Logical Volume Manager*, found at:

  http://www.suse.de/us/whitepapers/lvm/

Referenced Web sites

- Samba Web pages

http://www.samba.org

- UTS Global Backup and Restore (BAR)

  http://www.utsglobal.com/linuxprod.html

- Tivoli Storage Manager from IBM

  http://www.tivoli.com/products/index/storage-mgr/

- Innovation Data Processing FDR/UPSTREAM

  http://www.innovationdp.fdr.com/ups.cfm

- SecureAgent's SecureBackup

  http://www.secureagent.com/securebackup/

- Legato Networker

  http://www.legato.com/products/networker/index.cfm

- Open Source Software - Amanda

  http://www.amanda.org/

The team that wrote this Redpaper

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

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

Roy Costa is an Advisory Systems Programmer at the International Technical Support Organization, Poughkeepsie Center. He has 17 years of experience in VM systems programming. Roy has worked with Linux for S/390 for more than two years. He has provided technical advice and support to numerous redbooks for the past seven years.

Klaus Egeler is an IT System Management Specialist with IBM Global Services, Germany. He has more than 10 years of experience as a VSE and VM systems programmer. He has worked with Linux for S/390 for more than two years.

Michael MacIsaac is a Linux technical specialist in the zSeries New Technology Center (zNTC). He has written extensively on Linux on zSeries and z/OS UNIX Systems Services.

Acknowledgements

The authors would like to thank the following people for their invaluable contribution to this Redpaper:

Fred Borchers
International Technical Support Organization, Poughkeepsie Center

Romney White
IBM Endicott

Become a published author

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

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

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

`ibm.com/redbooks/residencies.html`

Comments welcome

Your comments are important to us!

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

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

  `ibm.com/redbooks`

- Send your comments in an Internet note to:

  `redbook@us.ibm.com`

- Mail your comments to:

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

Notices

This document created or updated on February 25, 2003.

Send us your comments in one of the following ways:

- Use the online **Contact us** review redbook form found at:
  ibm.com/redbooks
- Send your comments in an Internet note to:
  redbook@us.ibm.com
- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HYJ Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400 U.S.A.

Trademarks