

# Sharing and maintaining RHEL 5.3 Linux under z/VM

This paper is divided into the following sections:

- ▶ 1.1, “History” on page 2 - a history of the different versions of this paper
- ▶ 1.2, “Overview” on page 3 - an overview of the paper
- ▶ 1.3, “Background of read-only root Linux” on page 4 - describes the shared root file structure and the maintenance system
- ▶ 1.4, “Summary of virtual machines” on page 18 - summarizes the z/VM virtual machines (or *virtual servers*) that comprise the system
- ▶ 1.5, “Building a read-write maintenance system” on page 19 - describes how to start with *cloning*, by first building a conventional read-write Linux system that can be copied to other virtual servers
- ▶ 1.6, “Building a read-only root system” on page 41 - describes how to move to cloning *read-only root* systems once the conventional read-write cloning is working
- ▶ 1.7, “Maintaining systems” on page 51 - discusses how to maintain cloned Linux images
- ▶ 1.8, “Other considerations” on page 61 - discusses other advanced techniques for improving performance and maintainability of systems
- ▶ 1.9, “Contents of tar file” on page 65 - describes the contents of the file associated with this paper
- ▶ 1.10, “Linux code” on page 66 - lists the code of the Linux shell scripts
- ▶ 1.11, “z/VM code” on page 76 - lists the code of the z/VM REXX EXECs
- ▶ 1.12, “The team that wrote this paper” on page 87 - describes all who were involved in making this paper possible

This paper is based on z/VM version 5.4 and Red Hat Enterprise Linux (RHEL) version 5.3.

## 1.1 History

This paper was originally published as the IBM Redpaper *Sharing and maintaining Linux under z/VM*, largely based on input from architects and system administrators from Nationwide Insurance, published in February of 2008, available on the Web at:

<http://www.redbooks.ibm.com/abstracts/redp4322.html>

### Second paper

In July of 2009, the original paper was updated, with most input coming from system administrators at Penn State University. This second paper is available on the Web at:

<http://linuxvm.org/present/ro-root-S10.pdf>

The changes made in the second paper are as follows:

- ▶ It was based on z/VM 5.4 (previously 5.3).
- ▶ Linux was based on SLES 10 SP2.
- ▶ Linux scripts and z/VM REXX EXECs were updated or added.
- ▶ The Linux script to create a read-only system was renamed to **mnt2rog1d.sh** (formerly **mkror.sh**) to better follow the script naming convention.
- ▶ Disk space of each Linux system was increased: previously a read-write system occupied 3338 cylinders, or a single 3390-3. In this paper, a read-write system occupies 5008 cylinders, or half of a 3390-9. Also a read-only system has been increased to 1669 cylinders, or half of a 3390-3.
- ▶ A more detailed section on maintaining Linux was added.

### Third paper

This is the third paper with most updates coming from collaboration between IBM and Red Hat. The majority of the changes were in moving from SLES to RHEL.

- ▶ Linux is based on Red Hat Enterprise Linux (RHEL) 5.3
- ▶ No modified initialization script is necessary because Red Hat's `/etc/rc.d/rc.sysinit` allows for read-only root file systems.
- ▶ All read-write file systems on a read-only root system are maintained under `/local/`. In the previous papers, `/var/` was on its own minidisk.

## 1.2 Overview

Large operating systems, such as z/OS®, have, for several decades, leveraged shared file structures. The benefits are reduced disk space, simplified maintenance and simplified systems management. This paper describes how to create a Linux® solution with shared file systems on IBM® System z™ hardware (the mainframe) running under z/VM®. It also describes a maintenance system where the same Linux image exists on a test, maintenance and *golden* virtual servers (The *golden* disks are those that are copied when cloning). The benefits of such a system are the following:

- ▶ Extremely efficient resource sharing: which maximizes the business value of running Linux on System z
- ▶ Staff productivity: fewer people are needed to manage a large-scale virtual server environment running on z/VM
- ▶ Operational flexibility: companies can leverage and utilize their IT infrastructure to enhance their business results

A word of caution and a disclaimer are necessary. The techniques described in this paper are not simple to implement. Both z/VM and Linux on System z skills are needed. It is not guaranteed that such a system would be supported. Check with your Linux distributor and your support company to verify the changes described in this paper will be supported. This paper is based on a system that has been implemented and is in production at Nationwide Insurance and at Penn State. While they are based on Novell/Suse's SLES distribution, the examples in this paper are based on Red Hat's RHEL.

### 1.2.1 Conventions

The following font conventions are used in this paper:

**Monospace and bold** Commands entered by the user on the command line

<value> Value inside angle brackets is to be replaced with a value specific to your environment

monospace File, directory and user ID names

The following command conventions are used in this book:

- ▶ z/VM commands are prefixed with ==>
- ▶ z/VM XEDIT subcommands are prefixed with =====>
- ▶ Linux commands running as root are prefixed with hostname: #

## 1.3 Background of read-only root Linux

The system is called *read-only root* because the root file system (/) is mounted with read permission, not read-write.

### 1.3.1 Why a read-only root?

By creating a *read-only root* file structure, the basic Linux system code can be shared among many virtual Linux servers. This helps with Linux standardization. Many Linux servers will share exactly the same version of Linux operating system.

This environment makes maintenance much simpler. It becomes possible to roll out a new version of Linux by updating the master copy of the shared root file system, and not each of the *read-only root* systems that uses it.

### 1.3.2 Overview of the system

The root file system is read-only because it does not have to be read-write. Actually, four directories are chosen to be read-write: `/etc/`, `/root/`, `/srv/` and `/var/`. In addition, the `/tmp/` directory is read-write, but is in-memory and built at boot time. The `/proc/` and `/sys/` pseudo-directories are abstractions of kernel control blocks and the permissions are not under the control of the systems administrator.

During the boot process, read-write copies of the directories `/etc/`, `/root/`, `/srv/` and `/var/` are bind-mounted from `/local/` over the read-only copies. There is a background discussion on bind-mounts in section 1.3.5, "Overview of bind mounts" on page 14.

Figure 1-1 shows a block diagram of the entire system. The boxes above the dashed line represent a maintenance system for conventional Linux servers with most file systems being read-write. The boxes below the line represent the read-only root portion of the solution.

You may not understand the system the first time you see this figure, but read on, then come back to it. You may also want to look at Figure 1-11 on page 20 which shows the file system layout.

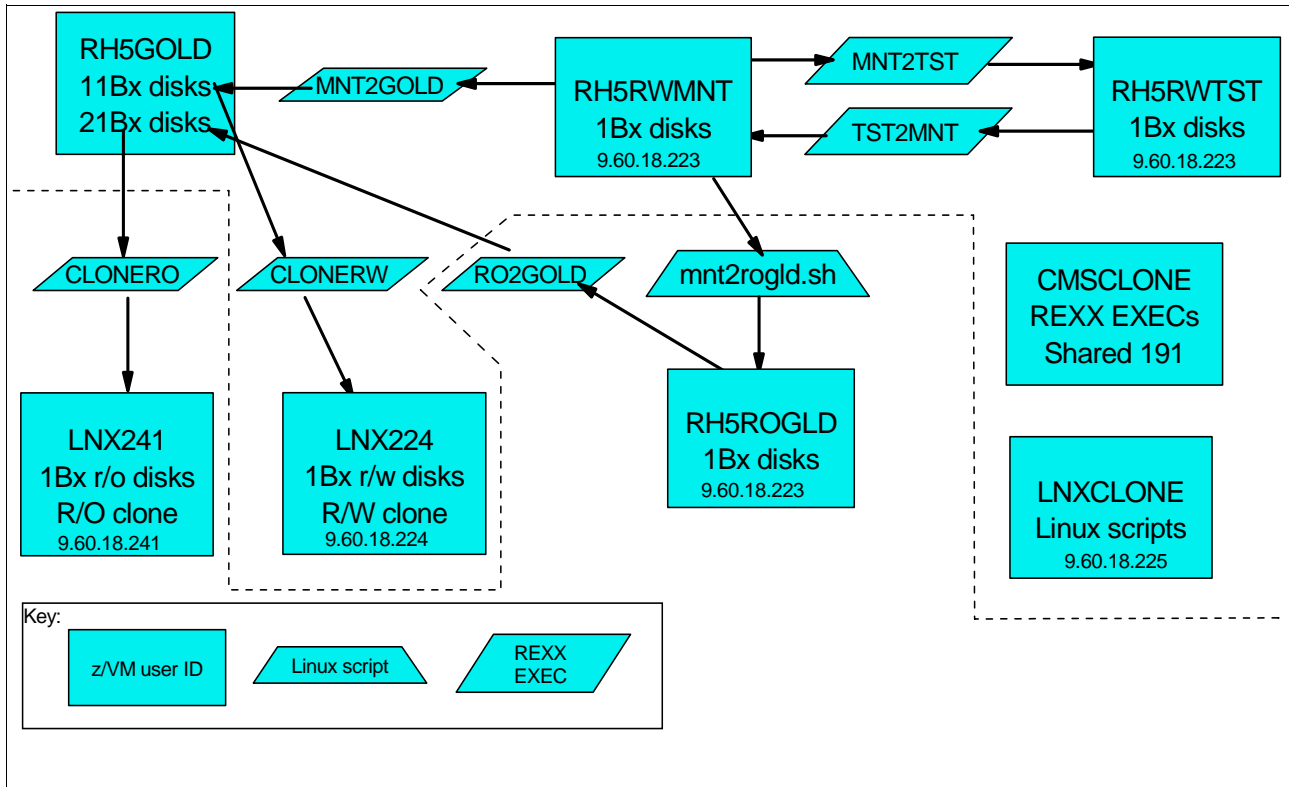


Figure 1-1 Block diagram of read-only root system

The names of the z/VM user IDs are chosen to convey their function:

RH5GOLD	The golden systems are stored on this virtual machines disks but never booted.
RH5RWMNT	The read-write system on which maintenance is done
RH5RWTST	The read-write system on which testing is done
RH5ROGLD	The virtual machine with a <i>hybrid</i> read-only system
LNXC241	A user ID to run a Linux script to clone a read-only root system
CMSCLONE	A user ID to run REXX cloning EXECs (CMS files)
LNXC224	The first read-write Linux that is cloned
LNXC225	The first read-only root Linux that is cloned

A more detailed summary of these systems is in section 1.4, “Summary of virtual machines” on page 19.

### 1.3.3 High level approach of read-only root system

In the section that follows, the read-only root solution is described at a high level. In addition to the *read-only root* environment, a maintenance process is also established. Details on implementing the solution begin in section 1.5, “Building a read-write maintenance system” on page 20.

RHEL 5.3 Linux is installed on an initial virtual server with the z/VM user ID RH5RWMNT (Figure 1-2). Packages to create a minimal Linux system are selected, security

hardening done and other configuration changes are applied. This server has the minidisks described in Table 1-1 on page 16.

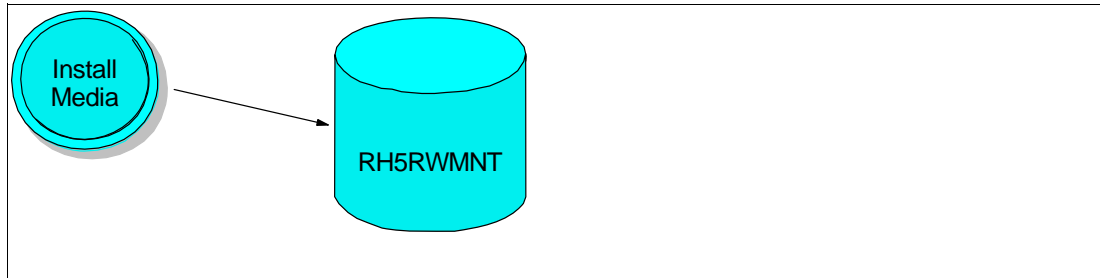


Figure 1-2 Installing the first Linux system

Once Linux is installed onto the initial server (RH5RWMNT), it is shutdown and copied to a test server, RH5RWTST (Figure 1-3). The Linux system running on RH5RWTST is configured then all features and functions are tested. When the system is validated, it is copied in the reverse direction back to RH5RWMNT. If changes on the test system are not valid, they can be discarded by recopying the maintenance system.

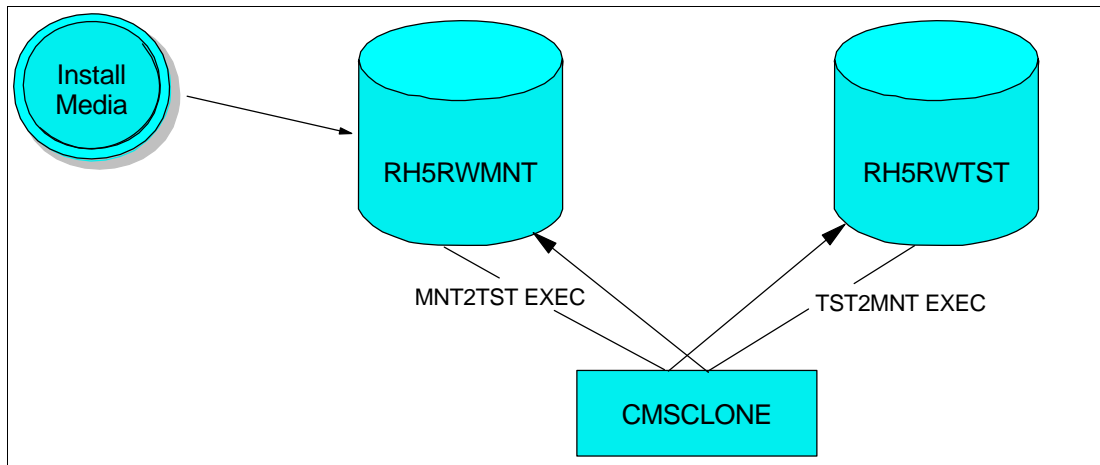


Figure 1-3 Cloning the first Linux system

Once configuration and testing are completed, the read-write Linux system is frozen to become the source disks for Linux cloning. This copy is called the *golden* version of the Linux minidisks. This whole system is shown in Figure 1-4. A Linux script (`cloneprep.sh`) is written to clean up the system before cloning. After running the cleanup script, the system is shutdown, then the disks are copied to the golden user ID (RH5GOLD) by means of a REXX EXEC.

At Nationwide, there are three versions of the golden minidisks maintained: *old*, *current*, and *next*. This paper only addresses a single copy: *current*. To add additional versions such as *old* and *next*, you would simply create more sets of minidisks on the golden user ID, with an agreed upon device numbering convention.

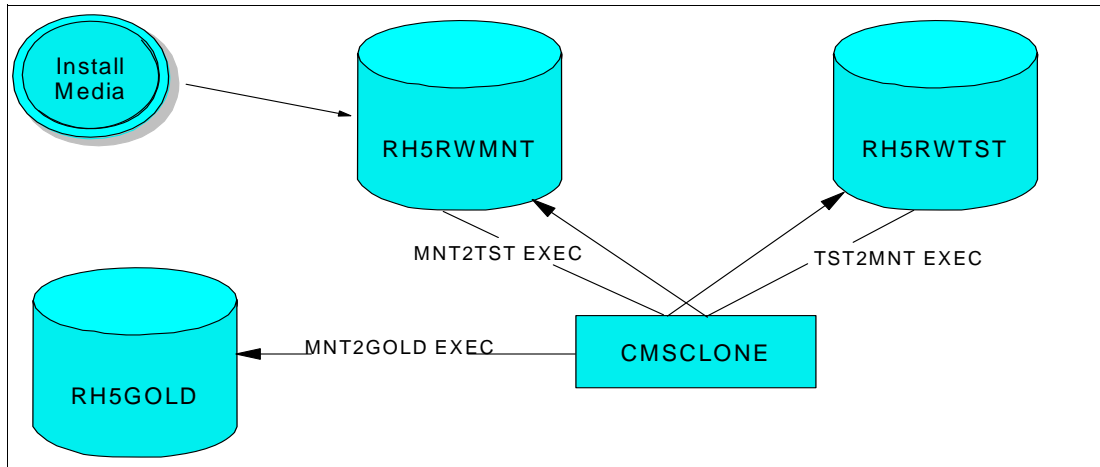


Figure 1-4 Freezing a golden copy of a Linux system

When a new Linux server needs to be cloned or provisioned with conventional read-write disks, it is copied from the *golden* minidisks. A REXX EXEC is run from the CMS virtual machine named CMSCLONE that copies from the *golden* minidisks to target Linux user IDs.

A script named `/etc/init.d/boot.findself` is also copied to the golden image. It is designed to run once at boot time to uniquely configure the new Linux server. It modifies the IP address and host name. This script will not modify the IP address and host name of the predefined RH5xxxx user IDs, so they will all have to share the same values. As such, only one of these user IDs can have Linux IPLed at any given time.

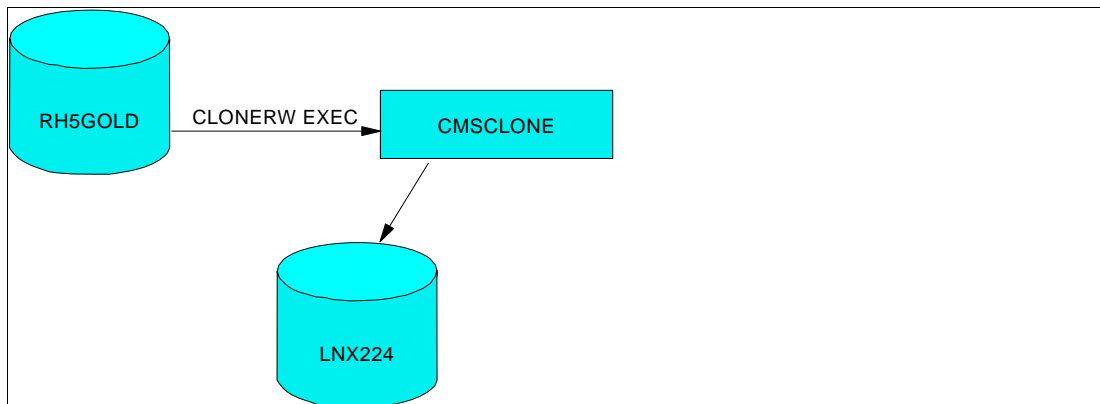


Figure 1-5 Cloning a read-write virtual server

After a read-write system has been created on RH5RWMNT, a read-only root system can be created from it. Another Linux system running on LNXCLONE is used to build the read-only root system. A Linux script (`mnt2rogld.sh`) is provided to create the first read-only root system on the RH5ROGLD user ID.

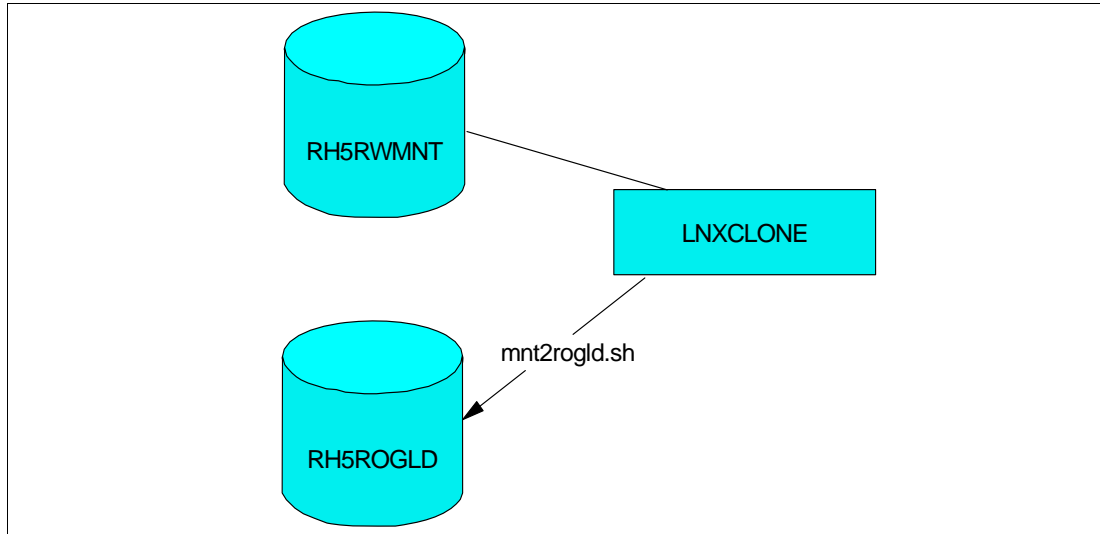


Figure 1-6 Creating a read-only root system with the `mnt2roglid.sh` script

Then the cleanup script is run on the first read-only root Linux system and the system is shut down. Now the **RO2GOLD EXEC** is executed to copy from RH5ROGLD to the 21Bx minidisks on RH5GOLD. These are now the *golden* read-only root disks.

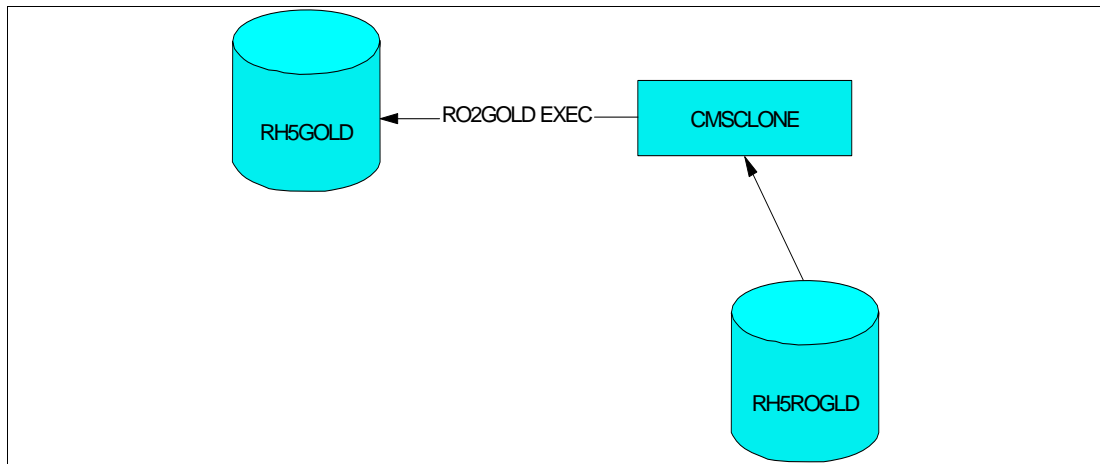


Figure 1-7 Freezing the read-only root system to RH5GOLD

To clone a read-only root system, the process is similar, except that the target user ID is defined to have two read-write minidisks and three read-only links to the *golden* disks. Therefore, the **CLONERO EXEC** only has to copy two minidisks, not file.

### 1.3.4 Directory structure of the read-only root system

The following section describes the directory structure used in this paper. The information was adapted from the following resources:

- ▶ The Linux File Hierarchy Standard (FHS):  
<http://tldp.org/LDP/Linux-Filesystem-Hierarchy/html/Linux-Filesystem-Hierarchy.html>
- ▶ The Linux Standards Base (LSB). See the following Web site:

<http://www.linux-foundation.org/en/LSB>

Sections below are copied verbatim in accordance with their license.

To comply with the FHS the following directories, or symbolic links to directories, are required in /.

/bin	Essential command binaries
/boot	Static files of the boot loader
/dev	Device files
/etc	Host-specific system configuration
/lib	Essential shared libraries and kernel modules
/media	Mount point for removable media
/mnt	Mount point for mounting a file system temporarily
/opt	Add-on application software packages
/sbin	Essential system binaries
/srv	Data for services provided by this system
/tmp	Temporary files
/usr	Secondary hierarchy
/var	Variable data

The following directories, or symbolic links to directories, must be in /, if the corresponding subsystem is installed:

/home	User home directories (optional)
/lib<qual>	Alternate format essential shared libraries (optional)
/root	Home directory for the root user (optional)

Each of these directories is addressed in the section that follows.

## **/sbin**

The /sbin directory is primarily used privileged commands such as shutdown, ifconfig, reboot and others. The commands in this directory are mostly intended for system administrators and usually require root access. The /sbin/ directory is in root's PATH, but not in the PATH of non-root users.

## **/bin**

The FHS states:

“Unlike /sbin, the /bin directory contains several useful commands that are of use to both the system administrator as well as non-privileged users. It usually contains the shells like bash, csh, etc.... and commonly used commands like cp, mv, rm, cat, ls. For this reason and in contrast to /usr/bin, the binaries in this directory are considered to be essential. The reason for this is that it contains essential system programs that must be available even if only the partition containing the root directory is mounted. This situation may arise should you need to repair other partitions but have no access to shared directories (i.e. you are in single user mode and hence have no network access). It also contains programs which boot scripts may require to be present. “

## **/boot**

The FHS states:

“This directory contains everything required for the boot process except for configuration files not needed at boot time (the most notable of those being those that belong to the GRUB boot-loader, not used in System z Linux) and the map installer. Thus, the /boot directory stores data that is used before the kernel begins executing user-mode programs. This may include redundant (back-up) master boot records, sector/system map files, the

kernel and other important boot files and data that is not directly edited by hand. Programs necessary to arrange for the boot loader to be able to boot a file are placed in `/sbin`. Configuration files for boot loaders are placed in `/etc`. The system kernel is located in either `/` or `/boot` (or as under Debian in `/boot` but is actually a symbolically linked at `/` in accordance with the FHS). The boot loader for IBM System z also resides in this directory.“

In this paper, `/boot/` is read-only, however, there may be times when a read-write copy is necessary.

## **`/dev`**

This directory highlights one important characteristic of the Linux file system - almost everything is a file or a directory. If you look at this directory and you should see `dasda1`, `dasda2` etc, which represent the various partitions on the disk drive of the system. The entries beginning with `sda*` are SCSI devices. Each logical minidisk would be represented as `dasda` for the first, `dasdb` for the second, etc...

## **`/etc`**

The FHS states:

“This is the nerve center of your system, it contains all system related configuration files (or in subdirectories). A “configuration file” is defined as a local file used to control the operation of a program; it must be static and cannot be an executable binary. For this reason, it’s a good idea to backup this directory regularly. Normally, no binaries should be or are located here.”

## **`/home`**

The LSB states that:

“Linux is a multi-user environment so each user is also assigned a specific directory which is accessible only to them and the system administrator. These are the user home directories, which can be found under `/home/<username>`. This directory also contains the user specific settings for programs like IRC, X etc.”

The FHS states that:

`/home` is a fairly standard concept, but it is clearly a site-specific file system. Different people prefer to place user accounts in a variety of places. This section describes only a suggested placement for user home directories; nevertheless we recommend that all FHS-compliant distributions use this as the default location for home directories. On small systems, each user’s directory is typically one of the many subdirectories of `/home` such as `/home/smith`, `/home/torvalds`, `/home/operator`, etc. On large systems (especially when the `/home` directories are shared amongst many hosts using NFS) it is useful to subdivide user home directories. Subdivision may be accomplished by using subdirectories such as `/home/staff`, `/home/guests`, `/home/students`, etc.

The setup will differ from host to host. Therefore, no program should rely on this location.

If you want to find out a user’s home directory, you should use the `getpwent(3)` library function rather than relying on `/etc/passwd` because user information may be stored remotely using systems such as NIS.

User specific configuration files for applications are stored in the user’s home directory in a file that starts with the `.'` character (a “dot file”). If an application needs to create more than one dot file then they should be placed in a subdirectory with a name starting with a `.'` character, (a “dot directory”). In this case the configuration files should not start with the `.'` character.

It is recommended that apart from autosave and lock files programs should refrain from creating non dot files or directories in a home directory without user intervention.”

In this paper, /home/ is not addressed in detail, though there is a short discussion of using automount, NFS and LDAP in 1.9.2, “Implementing /home/ with automount, NFS and LDAP” on page 65.

## **/lib**

The LSB states that:

“The /lib directory contains kernel modules and those shared library images (the C programming code library) needed to boot the system and run the commands in the root file system, i.e. by binaries in /bin and /sbin. Libraries are readily identifiable through their filename extension of \*.so. Windows® equivalent to a shared library would be a DLL (dynamically linked library) file. They are essential for basic system functionality. Kernel modules (drivers) are in the subdirectory /lib/modules/'kernel-version'. To ensure proper module compilation you should ensure that /lib/modules/'kernel-version'/kernel/build points to /usr/src/'kernel-version' or ensure that the Makefile knows where the kernel source itself are located.”

## **/lost+found**

The LSB states that:

“Linux should always go through a proper shutdown. Sometimes your system might crash or a power failure might take the machine down. Either way, at the next boot, a lengthy file system check using fsck will be done. Fsck will go through the system and try to recover any corrupt files that it finds. The result of this recovery operation will be placed in this directory. The files recovered are not likely to be complete or make much sense but there always is a chance that something worthwhile is recovered.”

## **/media**

The LSB states that:

“Amid much controversy and consternation on the part of system and network administrators a directory containing mount points for removable media has now been created. Funnily enough, it has been named /media.”

## **/mnt**

The LSB states that:

“This is a generic mount point under which you mount your file systems or devices. Mounting is the process by which you make a file system available to the system. After mounting your files will be accessible under the mount-point.”

The FHS v2.3 has changed the purpose of this directory:

“This directory is provided so that the system administrator may temporarily mount a file system as needed. The content of this directory is a local issue and should not affect the manner in which any program is run.

This directory must not be used by installation programs: a suitable temporary directory not in use by the system must be used instead.”

For this paper, the root directory (/) will be read-only and /mnt/ is not a separate file system. Therefore mount points under /mnt/ may only be created by file system design.

## **/opt**

The LSB states that:

“This directory is reserved for all the software and add-on packages that are not part of the default installation. To comply with the FHS, all third party applications should be installed in this directory. Any package to be installed here must locate its static files (i.e. extra fonts, clipart, database files) in a separate /opt/package' or /opt/provider' directory tree (similar to the way in which Windows will install new software to its own directory tree C:\Windows\Program Files\“Program Name”), where 'package' is a name that describes the software package and 'provider' is the provider's LANANA registered name.”

“Although most distributions neglect to create the directories /opt/bin, /opt/doc, /opt/include, /opt/info, /opt/lib, and /opt/man they are reserved for local system administrator use. Packages may provide “front-end” files intended to be placed in (by linking or copying) these reserved directories by the system administrator, but must function normally in the absence of these reserved directories. Programs to be invoked by users are located in the directory /opt/package/bin. If the package includes UNIX® manual pages, they are located in /opt/package/man and the same substructure as /usr/share/man must be used. Package files that are variable must be installed in /var/opt. Host-specific configuration files are installed in /etc/opt.”

In the read-only root system at Nationwide, DB2® is installed under /opt/IBM/db2, MQ Series is installed under /opt/mqm, with links into /usr/bin/ and /usr/lib/. So these mount points are created early, while /opt/ is still read/write.

## **/proc**

The LSB states that:

“/proc is very special in that it is also a virtual file system. It's sometimes referred to as a process information pseudo-file system. It doesn't contain 'real' files but runtime system information (e.g. system memory, devices mounted, hardware configuration, etc). For this reason it can be regarded as a control and information center for the kernel. In fact, quite a lot of system utilities are simply calls to files in this directory.”

## **/root**

The LSB states that:

“This is the home directory of the System Administrator, 'root'. This may be somewhat confusing ('root on root') but in former days, '/' was root's home directory (hence the name of the Administrator account). To keep things tidier, 'root' got his own home directory. Why not in '/home'? Because '/home' is often located on a different partition or even on another system and would thus be inaccessible to 'root' when - for some reason - only '/' is mounted.”

The FHS states that:

“If the home directory of the root account is not stored on the root partition it will be necessary to make certain it will default to / if it can not be located.

We recommend against using the root account for tasks that can be performed as an unprivileged user, and that it be used solely for system administration. For this reason, we recommend that subdirectories for mail and other applications not appear in the root account's home directory, and that mail for administration roles such as root, postmaster, and web master be forwarded to an appropriate user.”

## **/tmp**

The LSB states that:

“This directory contains mostly files that are required temporarily. Many programs use this to create lock files and for temporary storage of data. Do not remove files from this directory unless you know exactly what you are doing! Many of these files are important

for currently running programs and deleting them may result in a system crash. Usually it won't contain more than a few KB anyway. On most systems, this directory is cleared out at boot or at shutdown by the local system. The basis for this was historical precedent and common practice. However, it was not made a requirement because system administration is not within the scope of the FHS. For this reason people and programs must not assume that any files or directories in `/tmp` are preserved between invocations of the program. The reasoning behind this is for compliance with IEEE standard P1003.2 (POSIX, part 2)."

In the read-only root system `/tmp` is actually an in-memory only virtual file system as well, for more information see `tmpfs` (also known as `SHMFS`).

## **`/usr`**

The LSB states that:

"`/usr` usually contains by far the largest share of data on a system. Hence, this is one of the most important directories in the system as it contains all the user binaries, their documentation, libraries, header files, etc.... `X` and its supporting libraries can be found here. User programs like `telnet`, `ftp`, etc.... are also placed here. In the original Unix implementations, `/usr` was where the home directories of the users were placed (that is to say, `/usr/someone` was then the directory now known as `/home/someone`). In current Unices, `/usr` is where user-land programs and data (as opposed to 'system land' programs and data) are. The name hasn't changed, but it's meaning has narrowed and lengthened from "everything user related" to "user usable programs and data". As such, some people may now refer to this directory as meaning 'User System Resources' and not 'user' as was originally intended."

The FHS states:

"`/usr` is shareable, read-only data. That means that `/usr` should be shareable between various FHS-compliant hosts and must not be written to. Any information that is host-specific or varies with time is stored elsewhere.

Large software packages must not use a direct subdirectory under the `/usr` hierarchy."

## **`/var`**

The LSB states that:

"(`/var`) Contains variable data like system logging files, mail and printer spool directories, and transient and temporary files. Some portions of `/var` are not shareable between different systems. For instance, `/var/log`, `/var/lock`, and `/var/run`. Other portions may be shared, notably `/var/mail`, `/var/cache/man`, `/var/cache/fonts`, and `/var/spool/news`. '`/var`' contains variable data, i.e. files and directories the system must be able to write to during operation.

The FHS states:

"If `/var` cannot be made a separate partition, it is often preferable to move `/var` out of the root partition and into the `/usr` partition. (This is sometimes done to reduce the size of the root partition or when space runs low in the root partition.) However, `/var` must not be linked to `/usr` because this makes separation of `/usr` and `/var` more difficult and is likely to create a naming conflict. Instead, link `/var` to `/usr/var`.

Applications must generally not add directories to the top level of `/var`. Such directories should only be added if they have some system-wide implication, and in consultation with the FHS mailing list."

## **/srv**

The FHS states:

“/srv contains site-specific data which is served by this system. This main purpose of specifying this is so that users may find the location of the data files for particular service, and so that services which require a single tree for readonly data, writable data and scripts (such as cgi scripts) can be reasonably placed. Data that is only of interest to a specific user should go in that users' home directory.

The methodology used to name subdirectories of /srv is unspecified as there is currently no consensus on how this should be done. One method for structuring data under /srv is by protocol, eg. ftp, rsync, www, and cvs.

On large systems it can be useful to structure /srv by administrative context, such as /srv/physics/www, /srv/compsci/cvs, etc. This setup will differ from host to host. Therefore, no program should rely on a specific subdirectory structure of /srv existing or data necessarily being stored in /srv. However /srv should always exist on FHS compliant systems and should be used as the default location for such data.

Distributions must take care not to remove locally placed files in these directories without administrator permission. This is particularly important as these areas will often contain both files initially installed by the distributor, and those added by the administrator.”

## **One other directory**

In addition to the directories defined by the Linux Standard Base, Nationwide has included the directory /local/. It contains local content specific to this server. The subdirectories for local copies of /etc/, /root/, /srv/ and /var/ are located here, with bind mounts to the real directories.

### **1.3.5 Overview of bind mounts**

The /etc/, /root/, /srv/ and /var/ directories are implemented on the read-only root systems by means of bind mounts. A background is given on these types of mounts to help explain how the solution will work and text was taken from the IBM Redbook *Linux on IBM eServer zSeries and S/390: Large Scale Linux Deployment*, SG24-6824. This redbook is online at:

<http://w3.itso.ibm.com/abstracts/sg246824.html?0pen>

A bind mount expands the functionality of the device file system mount. Using bind mounts, it is possible to graft a directory sub-tree from one part of the global file system to another. Bind mounts differ from device mounts in that the source is the global file system itself - not a block device. The Linux kernel maintains coherence and consistency whichever name is used.

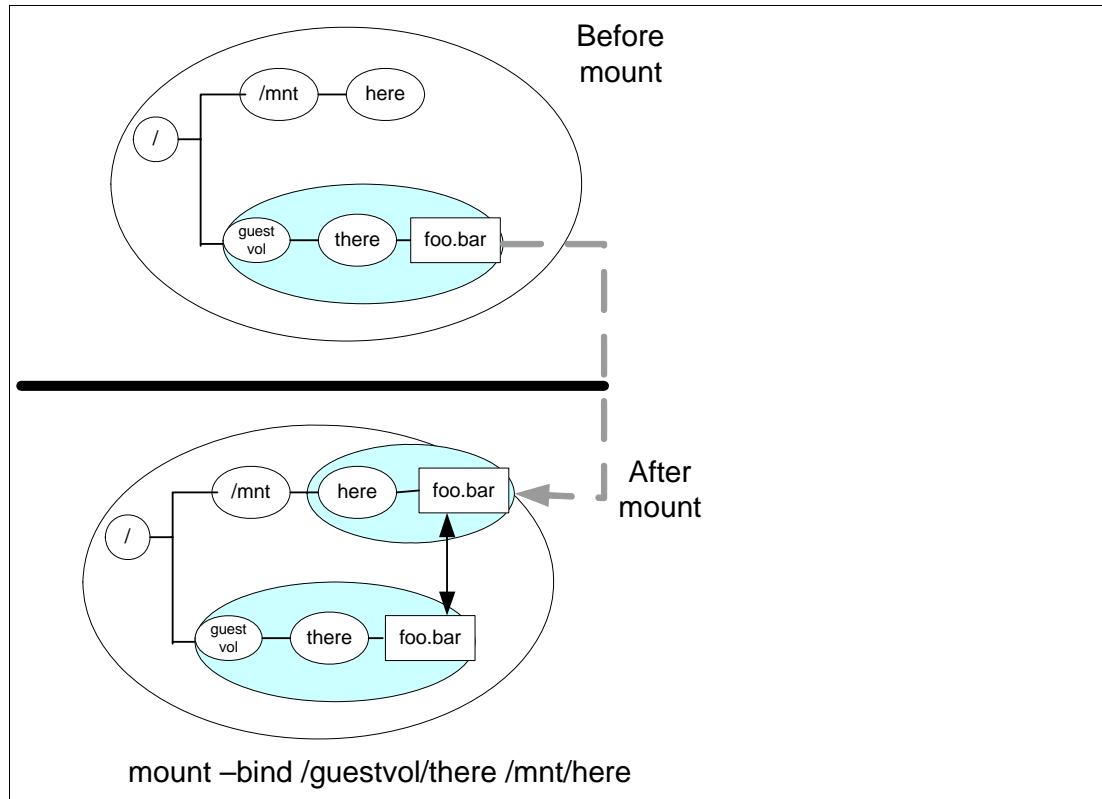


Figure 1-8 Overview of bind mounts

As an example, consider a directory `/guestvol/there` containing a file named `foo.bar`, as shown in Figure 1-8. An additional directory `/mnt/here` exists in the global name space. The following command will bind-mount `/guestvol/there/` over `/mnt/here/`:

```
# mount --bind /guestvol/there /mnt/here
```

Now the same file, `foo.abc`, can be referenced by two path names: `/guestvol/there/foo.abc`, the original path name, and `/mnt/here/foo.abc`, the bind mount path name. Both names refer to the same underlying file.

Figure 1-11 on page 20 also shows an example of a bind mount.

### 1.3.6 Summary of read-only root file systems

Table 1-1 on page 16 summarizes the file systems in the read-only root system. Note that file systems which will be read-only use a type `ext2`, because a journal cannot be written to a read-only file system. Read-write file systems use a type of `ext3` which is more conventional.

Table 1-1 Summary of file systems and swap spaces

Directory	FS type	Attributes	Device	Vaddr	Notes
/	ext2	R/O	/dev/dasdb1	1B1	read-only root, 3200 cylinder (~2.2 GB) minidisk
/bin/	ext2	R/O			Part of root file system
/boot/	ext2	R/O	/dev/dasda1	1B0	60 cylinder (~41 MB) minidisk
/dev/	udev	R/W			The device file system
/etc/	ext3	R/W			Bind mounted from /local/etc/ to /etc/
/home/	automount	R/W			Discussed in "Implementing /home/ with automount, NFS and LDAP" on page 65
/lib/, /lib64/	ext2	R/O			Part of the root file system
/local	ext3	R/W	/dev/dasdf1	1B5	1119 cylinder minidisk (~706MB) - contains R/W /etc/, /root/, /srv/ and /var/
/mnt/	ext2	R/O			R/W directory can be mounted over R/O
/opt/	ext2	R/O			Part of the root file system
/proc/	procfs	R/W			In memory kernel file system
/root/	ext3	R/W			Bind mounted from /local/root/ to /root/
/sbin/	ext2	R/O			Part of root file system
/srv/	ext3	R/W			Bind mounted from /local/srv/ to /srv/
/sys/	sysfs	R/W			In memory file system
/tmp/	tmpfs	R/W			In memory file system - contents are lost at shutdown
/usr/	ext2	R/O			Part of the root file system
/var/	ext3	R/W			Bind mounted from /local/var/ to /var/
/var/lib/rpm/	ext2	R/O	/dev/dasdg1	1B6	Mounted read-only over read-write /var/
swap 1	swap	R/W	/dev/dasdc1	1B2	64 MB in memory VDISK
swap 2	swap	R/W	/dev/dasdd1	1B3	128 MB in memory VDISK
swap 3	swap	R/W	/dev/dasde1	1B4	550 cylinder minidisk (~384MB)

### 1.3.7 The modified boot process

During the normal Linux boot process, the root file system is initially mounted read-only, and then later mounted read-write. In the read-only root system, it is not remounted read-write. Figure 9 on page 17 shows a block diagram of the System z Linux boot process.

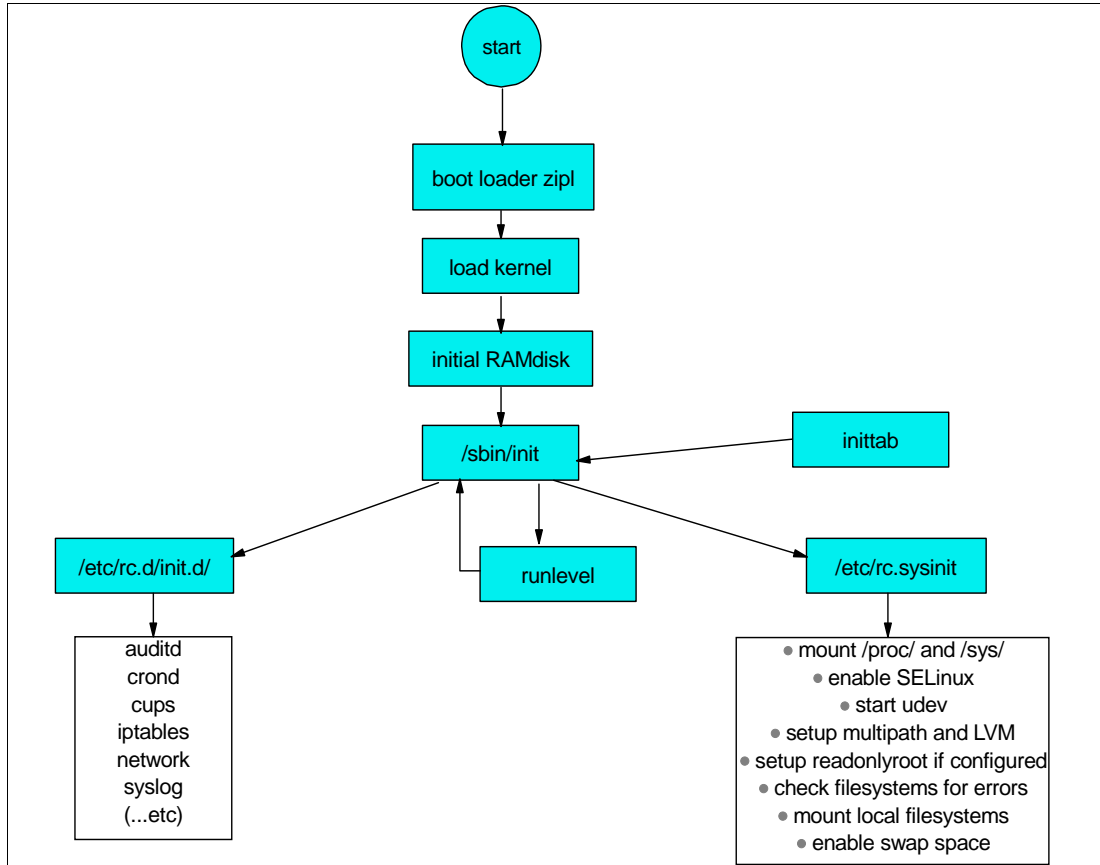


Figure 9 Modified boot process

Because the script `/etc/rc.sysinit` is passed in the parameter `readonlyroot`, the root file system is not re-mounted read-write.

## 1.4 Summary of virtual machines

Following is a summary of the function of the virtual machines in the system.

### CMSCLONE

This user ID only runs CMS, Linux is never IPLed. It has a 191 disk where the REXX EXECs for copying systems are stored, and it has a 192 disk which will become the Linux user ID's 191 disk.

### LNXCONE

A Linux system is run on this user ID. It contains all of the tools used to create the RH5ROGLD machine i.e. all shared root images. It also contains the DVD images of all shared software for other Linux guests to mount through NFS.

Following are the important files in `/usr/local/sbin/` on this system:

<code>boot.findself</code>	On first boot, this script is run to set up the proper host name and IP of the new Linux guest. The parameter file for this guest must exist on CMSCLONE first.
<code>cloneprep.sh</code>	A backup of the script that resides on RH5RWTST and other machines. Gets rid of extraneous files unique to a machine.
<code>mnt2rogld.sh</code>	Creates a <i>hybrid</i> read-only system on the RH5ROGLD user ID.
<code>offliner.sh</code>	Makes sure all minidisks are offline and not linked before running <code>mnt2rogld.sh</code> (useful after script failure).

### RH5RWMNT

The backup/mirror of RH5RWTST. This machine's binaries are used to create shared-root golden disks as well as read-write golden disks, in conjunction with RH5RWTST.

### RH5RWTST

All new RPMs and other changes are tested on this machine. Once tests have been made, the `cloneprep.sh` script is run and the system is copied to RH5RWMNT using the `TST2MNT EXEC` on CMSCLONE. If there is a need to *roll-back* the system from RH5RWMNT because a change is not desired, that can be accomplished using the `MNT2TST EXEC` on CMSCLONE.

### RH5ROGLD

This machine is created by the `mnt2rogld.sh` script residing on LNXCONE. It is essentially a read-only Linux guest. Here you can login before cloning a real read-only guest to see if the read-only system is running properly. Testing new read-only binaries and applications should be done from this virtual machine.

Once the binaries for read-only machines are good (stable), they should be copied to RH5GOLD using the `R02GOLD EXEC` on CMSCLONE.

### RH5GOLD

This virtual machine just defines two series of minidisks. Because its function is solely to store disks of Linux *golden images*, it cannot be logged on to.

11Bx	Read-write machines are cloned from this series of minidisks. These minidisks were populated and updated from RH5RWMNT.
21Bx	Read-only machines are linking these minidisks as RR. Consider this the production binaries for shared-root.

## 1.5 Building a read-write maintenance system

Before building a read-only root system, a system for maintaining and cloning conventional read-write Linux systems is described in this section. The read-write system is created with a maintenance plan in mind and is shown in the boxes shown above the dashed line in Figure 1-1 on page 4. When the read-write system is tested and working, the second phase is to create a read-only root system.

### Disk planning

A read-write system occupies 5008 cylinders, or half of a 3390-9. A read-only system requires 1669 cylinders, or half of a 3390-3. The CMS worker machine, CMSCLONE is given 130 cylinders for two CMS disks.

Four 3390-9s and one 3390-3 are reserved for this environment. The initial disk layout is shown in Figure 1-10:

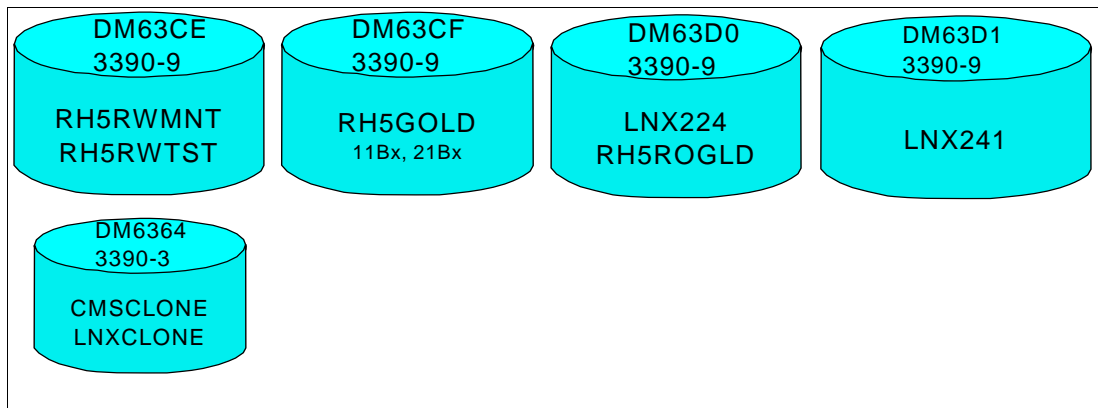


Figure 1-10 Disk planning with four 3390-9s and one 3390-3

### File system planning

Ignoring the swap spaces, there are four minidisks for file systems:

Table 1-2 Minidisks for file systems

Minidisk virtual device	File system	Attributes
1B0	/boot/	Read-only
1B1	/	Read-only
1B5	/local/	Read-write - contains four directories that are bind mounted: /etc/, /root/, /srv/ and /var/
1B6	/var/lib/rpm	Read-only

Table 1-11 on page 20 shows the hierarchy. A bind mount is shown from /var/ which is read-only to /local/var/ which is read-write, but for simplicity, it is not shown for the other three file systems that are also bind-mounted.

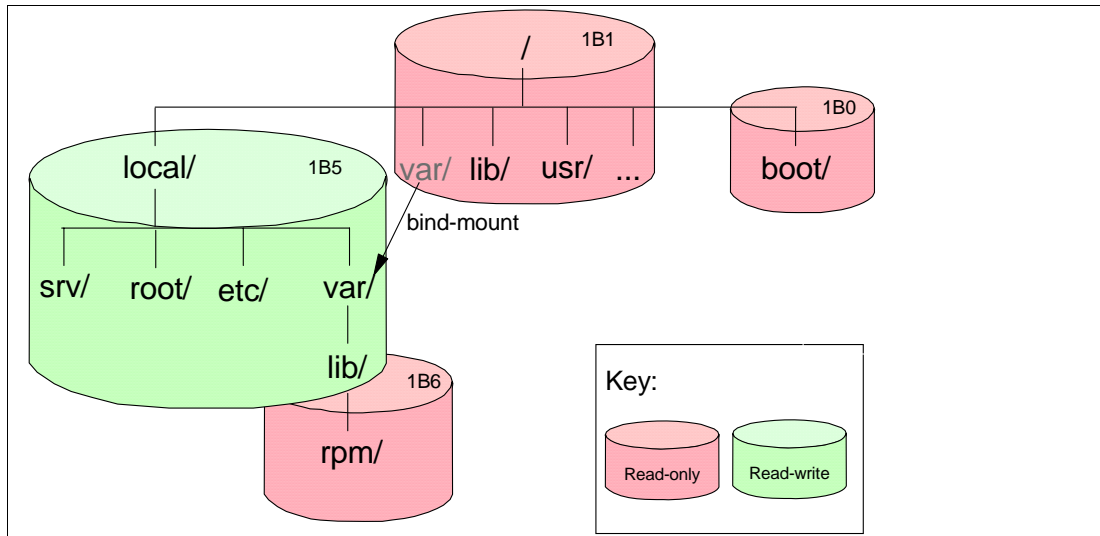


Figure 1-11 File system layout

Following are the steps to create the read-write system:

1. "Creating the first z/VM user IDs" on page 23
2. "Populating CMS disks on CMSCLONE" on page 23
3. "Installing RHEL 5.3 onto RH5RWMNT" on page 25
4. "Installing and customizing RHEL 5.3 onto LNXCLONE" on page 29
5. "Copying Linux from the maintenance system to the test system" on page 30
6. "Customizing Linux on RH5RWTST" on page 31
7. "Copying the golden image from RH5RWTST to RH5RWMNT" on page 35
8. "Copying read-write Linux to the golden disks" on page 36
9. "Cloning a read-write Linux system" on page 37

## 1.5.1 Creating the first z/VM user IDs

Five z/VM user IDs are defined to get started: CMSCLONE, RH5RWMNT, RH5RWTST, RH5GOLD and LNXCLONE. Two user directory profiles are also defined - LNXDFLT for read-write systems and RORDFLT for read-only root systems.

### Defining two user directory profiles

A profile named LNXDFLT is added that will be common to all read-write Linux user IDs. It has the NICDEF statement which creates a virtual OSA connection to the system VSWITCH named VSW1. Think of this statement as creating a virtual Network Interface Card (NIC) that is created and *plugged in* to the system VSWITCH when the user ID is logged on. The LINK statement to the CMSCLONE 192 disk enables the other user IDs to share a common read-only 191 disk:

```

PROFILE LNXDFLT
  IPL CMS
  MACHINE ESA 4
  CPU 00 BASE
  CPU 01
  NICDEF 0600 TYPE QDIO LAN SYSTEM VSW1

```

```

SPOOL 000C 2540 READER *
SPOOL 000D 2540 PUNCH A
SPOOL 000E 1403 A
CONSOLE 009 3215 T
LINK MAINT 0190 0190 RR
LINK MAINT 019D 019D RR
LINK MAINT 019E 019E RR
LINK CMSCLONE 192 191 RR
LINK TCPMAINT 592 592 RR

```

In addition, a profile name RORDFLT is created for read-only root Linux systems. It is identical to the LNXDFLT profile except that it adds three LINK statements to the three disks that will become read-only file systems (1B0 for /boot/, 1B1 for / and 1B6 for /var/lib/rpm/):

```

PROFILE RORDFLT
  IPL CMS
  MACHINE ESA 4
  CPU 00 BASE
  CPU 01
  NICDEF 0600 TYPE QDIO LAN SYSTEM VSW1
  SPOOL 000C 2540 READER *
  SPOOL 000D 2540 PUNCH A
  SPOOL 000E 1403 A
  CONSOLE 009 3215 T
  LINK MAINT 0190 0190 RR
  LINK MAINT 019D 019D RR
  LINK MAINT 019E 019E RR
  LINK CMSCLONE 192 191 RR
  LINK TCPMAINT 592 592 RR
  LINK RH5GOLD 21B0 01B0 RR
  LINK RH5GOLD 21B1 01B1 RR
  LINK RH5GOLD 21B6 01B6 RR

```

## Defining five virtual machines

A user ID CMSCLONE is defined with the following directory definition. It is given two minidisks:

- 191      A small 30 cylinder minidisk for storing the cloning EXECs.
- 192      A 100 cylinder common disk that will become the Linux user ID's read-only 191 disk. It will store kernels, RAM disks, parameter files, the **SWAPGEN EXEC**, install EXECs, etc. The **ALL** keyword is used for the read password in the MDISK statement so that any user ID can link to the disk without a password.

The option **LNKNOPAS** allows this user ID to link to all minidisks without the need for a password. Class B permission allows the **FLASHCOPY** command to be invoked.

Following is the directory definition for the CMSCLONE user ID:

```

USER CMSCLONE PASSWD 64M 1G BG
INCLUDE IBMDFLT
OPTION APPLMON LNKNOPAS
IPL CMS
MACHINE ESA 4
MDISK 0191 3390 0001 0030 DM6364 MR PASSWD PASSWD PASSWD
MDISK 0192 3390 0031 0100 DM6364 MR ALL PASSWD PASSWD

```

The user ID RH5RWMNT is defined. The first Linux image is installed into this ID. Refer to Table 1-1 on page 16 and Table 1-2 on page 19 to see which minidisks are for which file systems. This definition utilizes half of a 3390-9 (5008 cylinders) for all disks. You may want to use more disk space for each read-write Linux system or you may want to implement logical volumes on some file systems, such as /var/, to allow for growth. Logical volumes

have not been implemented in this paper, but there is a brief discussion on the topic in the section 1.9.1, "Utilizing logical volumes" on page 64.

Following is the directory definition for the user ID RH5RWMNT:

```
USER RH5RWMNT PASSWD 256M 1G G
INCLUDE LNXDFLT
OPTION APPLMON
MDISK 01B0 3390 0001 0060 DM63CE MR PASSWD PASSWD PASSWD
MDISK 01B1 3390 0061 3200 DM63CE MR PASSWD PASSWD PASSWD
MDISK 01B4 3390 3261 0550 DM63CE MR PASSWD PASSWD PASSWD
MDISK 01B5 3390 3811 1119 DM63CE MR PASSWD PASSWD PASSWD
MDISK 01B6 3390 4930 0079 DM63CE MR PASSWD PASSWD PASSWD
```

A user ID RH5RWTST is created with the following directory definition. This is where changes to the golden image are tested. In this example the second half of the 3390-3 with label DM63CE is used.

```
USER RH5RWTST PASSWD 256M 1G G
INCLUDE LNXDFLT
OPTION APPLMON
MDISK 01B0 3390 5009 0060 DM63CE MR PASSWD PASSWD PASSWD
MDISK 01B1 3390 5069 3200 DM63CE MR PASSWD PASSWD PASSWD
MDISK 01B4 3390 8269 0550 DM63CE MR PASSWD PASSWD PASSWD
MDISK 01B5 3390 8819 1119 DM63CE MR PASSWD PASSWD PASSWD
MDISK 01B6 3390 9938 0079 DM63CE MR PASSWD PASSWD PASSWD
```

The user ID RH5GOLD, is created with the following directory definition. This user ID should never be logged on to, so the password is set to NOLOG. The minidisks 11Bx are the read-write golden disks and 21Bx are the read-only golden disks. It requires the space of a complete 3390-9 because it stores two read-write systems.

```
USER RH5GOLD NOLOG 64M 1G G
INCLUDE LNXDFLT
OPTION APPLMON
MDISK 11B0 3390 0001 0060 DM63CF MR PASSWD PASSWD PASSWD
MDISK 11B1 3390 0061 3200 DM63CF MR PASSWD PASSWD PASSWD
MDISK 11B4 3390 3261 0550 DM63CF MR PASSWD PASSWD PASSWD
MDISK 11B5 3390 3811 1119 DM63CF MR PASSWD PASSWD PASSWD
MDISK 11B6 3390 4930 0079 DM63CF MR PASSWD PASSWD PASSWD
MDISK 21B0 3390 5009 0060 DM63CF MR PASSWD PASSWD PASSWD
MDISK 21B1 3390 5069 3200 DM63CF MR PASSWD PASSWD PASSWD
MDISK 21B4 3390 8269 0550 DM63CF MR PASSWD PASSWD PASSWD
MDISK 21B5 3390 8819 1119 DM63CF MR PASSWD PASSWD PASSWD
MDISK 21B6 3390 9938 0079 DM63CF MR PASSWD PASSWD PASSWD
```

Finally, a user ID LNXCLONE is created with the following directory definition. A minimal Linux image will be installed onto the 1B0 disk so the `mnt2rog1d.sh` script can be run from `/usr/local/sbin/`. The option `LNKNOPAS` is included so that all minidisks can be linked with no password.

```
USER LNXCLONE PASSWD 256M 1G BG
INCLUDE RORDFLT
OPTION APPLMON LNKNOPAS
MDISK 01B0 3390 0131 3208 DM6364 MR PASSWD PASSWD PASSWD
```

The minidisk layout is verified and the changes are brought online.

## Allowing access to the system VSWITCH

The RH5RWMNT, RH5RWTST, RH5ROGLD and LNXCLONE user IDs are given access to system's VSWITCH. The RH5GOLD user ID does not need access to the VSWITCH as no Linux system will ever be IPLed from it. The following statements are put in AUTOLOG1's PROFILE EXEC:

```
'cp set vswitch vsw1 grant rh5rwmnt'  
'cp set vswitch vsw1 grant rh5rwtst'  
'cp set vswitch vsw1 grant rh5roglD'  
'cp set vswitch vsw1 grant lnxclone'
```

These commands are also run interactively from the command line for the current IPL.

## 1.5.2 Downloading the associated tar file

The tar file associated with this paper is available at:

<http://linuxvm.org/present/misc/ro-root-RH5.tgz>

Download it to a Linux or UNIX machine and un-tar it. In this example, it is downloaded to /tmp/:

```
# cd /tmp  
# ... download the file ...  
# tar xzf ro-root-RH5.tgz  
README.txt  
sbin/  
sbin/boot.findself  
sbin/cloneprep.sh  
sbin/mnt2roglD.sh  
sbin/offliner.sh  
vm191/  
vm191/CLONERO.EXEC  
vm191/CLONERW.EXEC  
vm191/COPYMDSK.EXEC  
vm191/MNT2GOLD.EXEC  
vm191/MNT2TST.EXEC  
vm191/RO2GOLD.EXEC  
vm191/RO2GOLD2.EXEC  
vm191/TST2MNT.EXEC  
vm191/LINKED.EXEC  
vm192/  
vm192/PROFILE.EXEC  
vm192/PROFILE.XEDIT  
vm192/SAMPLE.CONF-RH5  
vm192/SAMPLE.PARM-RH5  
vm192/SWAPGEN.EXEC  
vm192/RHEL53.EXEC
```

## 1.5.3 Populating CMS disks on CMSCLONE

The CMSCLONE user ID has two CMS disks:

- 191 To store the CLONE EXECs - these are part of the tar file associated with this paper.
- 192 A common CMS disk that will be linked by other Linux systems can share that disk read-only as their 191 disk.

First the CMS disks are formatted using the **FORMAT** command.

## Populating the CMSCLONE 191 disk

The following files are copied to the CMSCLONE 192 disk or created on it:

CLONERO EXEC	An EXEC to clone read-only Linux systems
CLONERW EXEC	An EXEC to clone read-write Linux systems
COPYMSDK EXEC	An EXEC to copy a minidisk first trying to use <b>FLASHCOPY</b> then <b>DDR</b>
LINKED EXEC	An EXEC to list which Linux systems are linked to which golden disks
MNT2GOLD EXEC	An EXEC to copy the maintenance golden image on RH5RWMNT to the golden read-write disks on RH5GOLD
MNT2TST EXEC	An EXEC to copy the maintenance golden image on RH5RWMNT to the test machine on RH5RWTST
RO2GOLD EXEC	An EXEC to copy the read-only golden image from RH5ROGLD to the golden read-only disks on RH5GOLD
RO2GOLD2 EXEC	An EXEC to copy the read-only golden image from RH5ROGLD to the golden read-only disks on RH5GOLD2
TST2MNT EXEC	An EXEC to copy the test golden image on RH5RWTST to the test machine on RH5RWMNT

## Populating the CMSCLONE 192 disk

The following files are copied to the CMSCLONE 192 disk or created on it. These files will be available to each Linux virtual machine as its 191 or A disk

PROFILE EXEC	An initialization file for each Linux system to boot it from minidisk 1B0
PROFILE XEDIT	An XEDIT initialization file similar to that on the MAINT 191 disk
RH5RWTST PARM-RH5	The parameter file for the user ID RH5RWTST. A sample RHEL 5 parameter file, <code>SAMPLE PARM-RH5</code> , is included in the tar file associated with this paper.
RH5RWTST CONF-RH5	The configuration file for the user ID RH5RWTST. A sample RHEL 5 parameter file, <code>SAMPLE CONF-RH5</code> , is included in the tar file associated with this paper.
RHEL53 EXEC	The EXEC to invoke the RHEL 5.3 installation. It is included in the tar file or it can be copied from the section 1.11.10, "RHEL53 EXEC".
RHEL53 KERNEL	The RHEL 5.3 kernel. This is available from the <code>images/</code> directory of the RHEL 5.3 install media, where it is named <code>kernel.img</code> .
RHEL53 INITRD	The RHEL 5.3 initial RAMdisk. This is also available from the <code>images/</code> directory of the RHEL 5.3 install media, where it is named <code>initrd.img</code> .
SWAPGEN EXEC	The EXEC to create VDISK swap spaces. It is included in the tar file for convenience. The latest copy of this EXEC is available from the <i>Sine Nomine Associates</i> Web page: <a href="http://download.sinenomine.net/swapgen/">http://download.sinenomine.net/swapgen/</a>

**Important:** Downloading and setting up **SWAPGEN** from [sinenomine.net](http://www.vm.ibm.com/download/) can be a bit tricky. There is a “mailable” format, but people have reported problems using this. There is also a VMARC (VM archive) format, which is like a tar file or zip file. However, VMARC is not standard with z/VM. To obtain and use **VMARC** command, see the section *How to download something* on the z/VM download Web site:

<http://www.vm.ibm.com/download/>

The **SWAPGEN EXEC** is included in the tar file for convenience.

Be sure the CMSCONE user ID is logged off. Following is a sample FTP session shown moving the files from the associated tar file to the CMSCONE user ID using the FTP subcommand **mput**. In this example the IP address of the z/VM system is 9.60.18.218:

```
# cd /tmp/vm191
# ftp 9.60.18.218
Name (9.60.18.218:root): cmsclone
Password:
230 CMSCONE logged in; working directory = CMSCONE 191
Remote system type is z/VM.
ftp> mput *
mput CLONERO.EXEC [anpqy?]? a
125 Storing file 'CLONERO.EXEC'
...
ftp> lcd ../vm192
Local directory now /tmp/vm191
ftp> cd cmsclone.192
250 Working directory is CMSCONE 192
ftp> mput *
mput PROFILE.EXEC [anpqy?]? a
...
ftp> quit
```

## 1.5.4 Installing RHEL 5.3 onto RH5RWMNT

RHEL 5.3 is installed twice:

1. Onto the golden image (RH5RWMNT) which will be the system that is cloned
2. Onto a worker system (LNXCLONE) which will be used for running Linux scripts

This section does not supply every detail on installing Linux. For more details, see the Red Hat System z Install Guide starting at:

<http://www.redhat.com/docs/manuals/enterprise/>

Additionally, the IBM Redbook *z/VM and Linux on IBM System z The Virtualization Cookbook for RHEL 5.2*, SG24-7492 might be a good resource. It is on the Web at:

<http://www.redbooks.ibm.com/abstracts/sg247492.html>

Following are high-level steps:

- ▶ An install server is set up on another Linux system where the ISO image of the first DVD is mounted loopback and exported by NFS. Following is the name of the mounted ISO image on the install server:

```
# mount | grep RHEL5.3
/nfs/rhe15.3/RHEL5.3-Server-20090106.0-s390x-DVD.iso on /nfs/rhe15.3/dvd type iso9660
```

- ▶ The kernel (kernel.img) and the initial RAMdisk (initrd.img) are copied from the images/ directory by means of FTP to the CMSCLONE 192 disk. Do not forget to transfer them in binary mode, with fixed-record 80 byte blocks. If you are FTPing from Linux to CMS, this can be accomplished by the FTP subcommands **bin** and **site fix 80**.
- ▶ The RH5RWMNT PARM-RH5 file points to the RH5RWMNT CONF-RH5 configuration file which is populated with the correct IP and DNS information. There are files SAMPLE PARM-RH5 and RH5RWMNT CONF-RH5 for your convenience. Following are examples of the parameter and configuration files used in this paper:

```

==> type rh5rwmnt parm-rh5
ramdisk_size=40000 root=/dev/ram0 ro ip=off
CMSDASD=191 CMSCONFIGFILE=RH5RWMNT.CONF-RH5
vnc vncpassword=lnx4vm
method=nfs:9.60.18.133:/nfs/rhe15.3/dvd
==> type rh5rwmnt conf-rh5
DASD=1b0-1bf,2b0-2bf,320-33f
HOSTNAME=gpok223.endicott.ibm.com
NETTYPE=qeth
IPADDR=9.60.18.223
SUBCHANNELS=0.0.0600,0.0.0601,0.0.0602
NETWORK=9.60.18.128
NETMASK=255.255.255.128
SEARCHDNS=endicott.ibm.com
BROADCAST=9.60.18.255
GATEWAY=9.60.18.129
DNS=9.0.3.1
MTU=1500
PORTNAME=DONTCARE
PORTNO=0
LAYER2=0

```

**Tip:** While installing RHEL 5.3 over NFS, you may encounter poor performance. You might see warning messages with the text:

```
nfs: server <myserver> not responding, still trying ...
```

If this happens, you can try adding NFS client options to the parameter file such as the following:

```

==> type rh5rwmnt parm-rh5
ramdisk_size=40000 root=/dev/ram0 ro ip=off
CMSDASD=191 CMSCONFIGFILE=RH5RWMNT.CONF-RH5
vnc vncpassword=lnx4vm
method=nfs:tcp,rsize=8192,wsiz=8192,timeo=20:9.60.18.133:/nfs/rhe15.3/dvd

```

- ▶ When you logon to the first user ID that Linux is installed onto, RH5RWMNT, you should see a virtual NIC being created and two VDISK swap spaces being created by the **SWAPGEN EXEC**:

```

LOGON RH5RWMNT
00: NIC 0600 is created; devices 0600-0602 defined
00: z/VM Version 5 Release 4.0, Service Level 0801 (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 13:29:48 EDT MONDAY 08/10/09
z/VM V5.4.0 2008-12-05 12:25

```

```
DMSACP723I A (191) R/0
```

DMSACP723I C (592) R/O

**DIAG swap disk defined at virtual address 1B2** (16245 4K pages of swap space)

**DIAG swap disk defined at virtual address 1B3** (32493 4K pages of swap space)

Do you want to IPL Linux from DASD 1B0? y/n

n

- ▶ Set the memory size to 1GB because the default of 256MB is not sufficient memory with which to install RHEL. The CP command **DEF STOR 1G** is issued then CMS is re-IPLed.
- ▶ Start the install process with the **RHEL53 EXEC**.
- ▶ Start an SSH session as root to the in-memory install system.
- ▶ Start a VNC session to the graphical installer, Anaconda.
- ▶ For each minidisk, you should see a warning message “Would you like to initialize this drive, erasing all data?”. Click **Yes** as this will be your only opportunity to format the minidisks for Linux file systems.

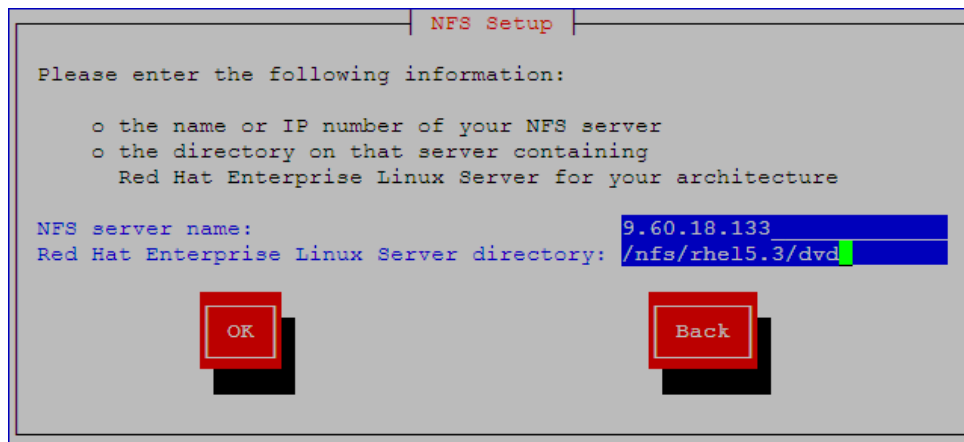


Figure 1-12 Setting NFS install server parameters

- ▶ In the top dropdown box, choose **Create custom layout** as shown in Figure 1-13 on page 28. Click **Next**.

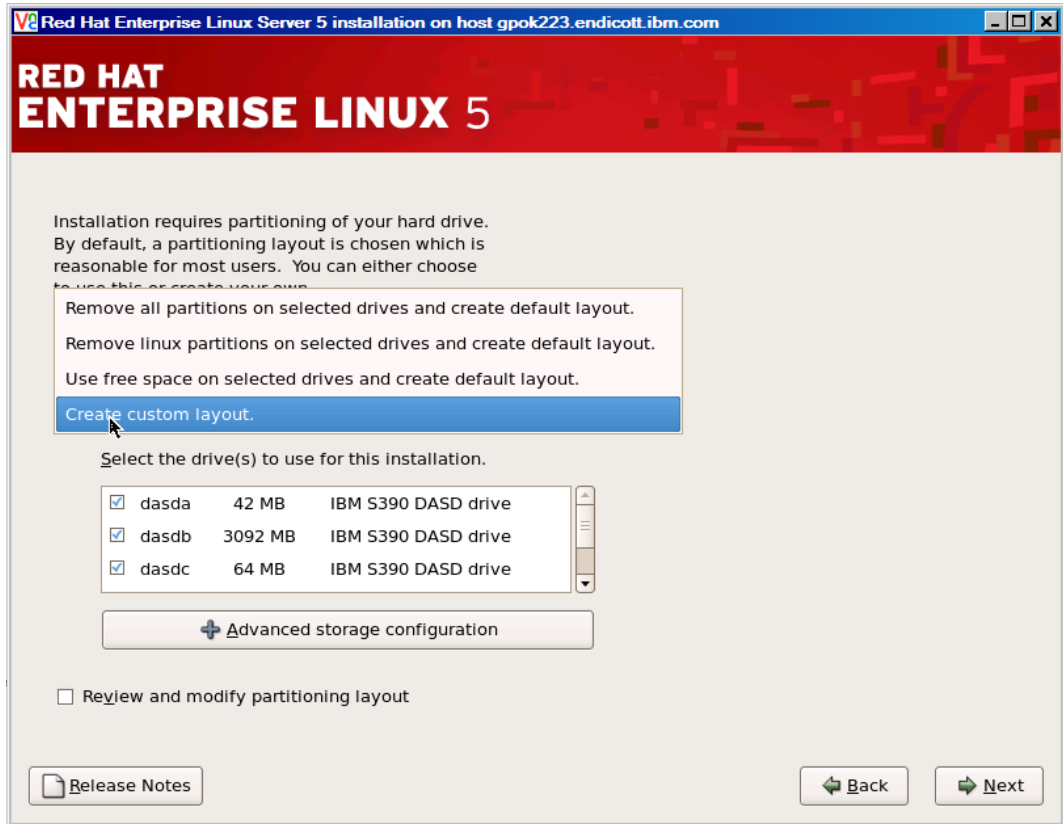


Figure 1-13 Choosing custom partition layout

- The partitions are allocated as described in Figure 1-14 on page 28.

Hard Drives						
▼ /dev/dasda						
/dev/dasda1	/boot	ext2	✓	42	2	480
▼ /dev/dasdb						
/dev/dasdb1	/	ext2	✓	2249	2	25600
▼ /dev/dasdc						
/dev/dasdc1		swap	✓	63	1	64
▼ /dev/dasdd						
/dev/dasdd1		swap	✓	127	1	128
▼ /dev/dasde						
/dev/dasde1		swap	✓	386	2	4400
▼ /dev/dasdf						
/dev/dasdf1	/local	ext3	✓	786	2	8952
▼ /dev/dasdg						
/dev/dasdg1	/var/lib/rpm	ext3	✓	55	2	632

Figure 1-14 File system and swap space layout

- A screen for software customization should appear. Click **Customize now**.

- ▶ All package groups are deselected except **Base system** => **Base** as shown in Figure 1-15.

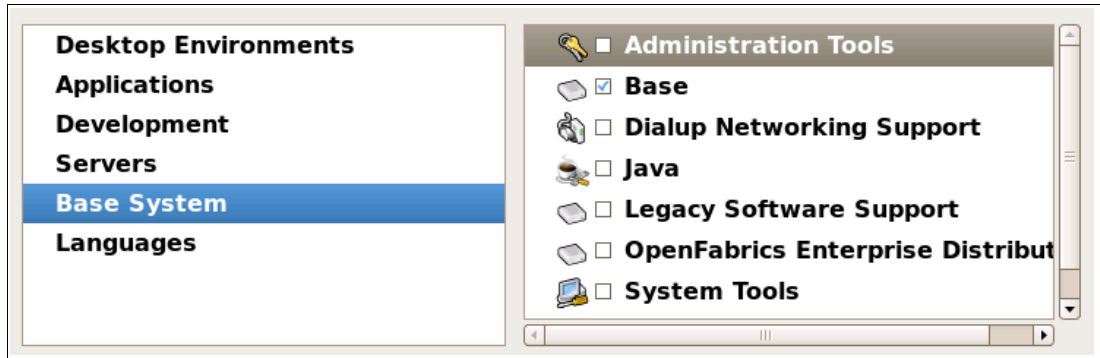


Figure 1-15 Deselecting all package groups except Base

- ▶ Complete the installation.

When the installation is complete, an SSH session as root is started. The file systems are queried with the `df` command:

```
gpok222:~ # df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/dasdb1     2.2G  766M  1.3G  38% /
/dev/dasdg1      54M   14M   38M  26% /var/lib/rpm
/dev/dasdf1     763M   17M  706M   3% /local
/dev/dasda1      41M   7.3M   32M  19% /boot
tmpfs           502M    0   502M   0% /dev/shm
```

The contents of the `/etc/fstab` file is viewed:

```
gpok222:~ # cat /etc/fstab
LABEL=/ / ext2 defaults 1 1
LABEL=/var/lib/rpm /var/lib/rpm ext2 defaults 1 2
LABEL=/local /local ext3 defaults 1 2
LABEL=/boot /boot ext2 defaults 1 2
tmpfs /dev/shm tmpfs defaults 0 0
devpts /dev/pts devpts gid=5,mode=620 0 0
sysfs /sys sysfs defaults 0 0
proc /proc proc defaults 0 0
LABEL=SWAP-dasdc1 swap swap defaults 0 0
LABEL=SWAP-dasdd1 swap swap defaults 0 0
LABEL=SWAP-dasde1 swap swap defaults 0 0
```

This Linux system will be the basis for both the read-write and the read-only systems. You can now shut down the Linux system running on RHRWMNT and log off of the user ID.

```
# shutdown -h now
...
```

This system will be customized later after it is copied to RH5RWTST.

## 1.5.5 Installing and customizing RHEL 5.3 onto LNXCLONE

A minimal system is also installed onto the LNXCLONE user ID. No cloning is attempted because it has just a single 3390-3 minidisk at 1B0 for a root file system.

The main function of this system is to be able to create a read-only root system, with the `mnt2rogld.sh` script, with both the source and target systems shutdown and their user IDs logged off from z/VM.

To customize the system the following modifications are made:

- ▶ The tar file associated with this paper is downloaded and untarred.
- ▶ The `cmm` and `vmcp` modules are set to load at boot time.

### Downloading the files associated with this paper

The tar file `ro-root-RH5.tgz` is copied to `/usr/local/` then untarred.

```
# cd /usr/local
...Copy the tar file ...
# tar xzf ro-root-RH5.tgz
```

Because this file is now copied to a system running Linux under z/VM, it is no longer needed on the staging Linux or UNIX system used earlier.

### Setting the `cmm` and `vmcp` module to be loaded

The `vmcp` module allows CP commands to be issued from Linux. When the `cmm` module is loaded, in conjunction with configuration changes on z/VM, significant performance gains are possible. Collaborative Memory Management and VMRM are discussed in more detail in 1.9.3, “Enabling Collaborative Memory Management (CMM)” on page 66.

Loading of the `cmm` and `vmcp` modules with the `modprobe` command are added to the file `/etc/rc.d/rc.local`. This will cause these modules to be loaded at boot time:

```
# cd /etc/rc.d
# vi rc.local
#!/bin/sh
#
# This script will be executed *after* all the other init scripts.
# You can put your own initialization stuff in here if you don't
# want to do the full Sys V style init stuff.

touch /var/lock/subsys/local
modprobe vmcp
modprobe cmm
```

The system is rebooted to verify the change. When the system comes back you should see the modules loaded:

```
# lsmod | egrep "cmm|vmcp"
cmm                44032  0
smsgiucv           36368  1 cmm
vmcp                36872  0
```

The Linux system running on the LNXCLONE user ID should now be configured.

## 1.5.6 Copying Linux from the maintenance system to the test system

The Linux running on RH5RWMNT should now be shut down. The system is copied from the newly installed RH5RWMNT to the corresponding minidisks on RH5RWTST. Linux is IPLed on RH5RWTST. Then modifications are made to the Linux on RH5RWTST. In this fashion, there is a backup copy of Linux. If tests are not successful on the test system, a fresh copy of Linux can quickly be *rolled back* from the maintenance system.

The **MNT2TST EXEC** run from the **CMSCLONE** user ID tries to use **FLASHCOPY** to copy the minidisks quickly. If this command is not supported or fails, it falls back to using the **DDR** command. See section “**MNT2TST EXEC**” on page 86 for a complete listing of the source code.

```
==> mnt2tst
Checking that source and target user IDs are logged off
HCPCQU045E RH5RWMNT not logged on
HCPCQU045E RH5RWTST not logged on
Do you want to copy R/W disks from RH5RWMNT to RH5RWTST? y/n
y

Copying minidisk 01B0 to 11B0 ...
Command complete: FLASHCOPY 01B0 0 END TO 11B0 0 END
Return value = 0

Copying minidisk 01B1 to 11B1 ...
Command complete: FLASHCOPY 01B1 0 END TO 11B1 0 END
Return value = 0

Copying minidisk 01B4 to 11B4 ...
Command complete: FLASHCOPY 01B4 0 END TO 11B4 0 END
Return value = 0

Copying minidisk 01B5 to 11B5 ...
Command complete: FLASHCOPY 01B5 0 END TO 11B5 0 END
Return value = 0

Copying minidisk 01B6 to 11B6 ...
Command complete: FLASHCOPY 01B6 0 END TO 11B6 0 END
Return value = 0

Cleaning up ...
01B0 01B1 01B4 01B5 01B6 11B0 11B1 11B4 DETACHED
11B5 11B6 DETACHED
```

The system has now been copied from **RH5RWMNT** to **RH5RWTST**.

## 1.5.7 Customizing Linux on **RH5RWTST**

With the system from **RH5RWMNT** now copied to **RH5RWTST**, you can customize the golden image. The following steps are recommended. Many of them are required for the **mnt2rogld.sh** script to work:

- ▶ “Modifying **zipl.conf**” on page 32
- ▶ “Verifying the **dasd** driver parameters” on page 33
- ▶ “Inserting the **cmm** and **vmcp** modules” on page 33
- ▶ “Modifying the **/etc/inittab** file” on page 33
- ▶ “Copying the **cloneprep.sh** and **boot.findself** scripts” on page 33
- ▶ “Configuring **YUM**” on page 34
- ▶ “Creating empty mount points under **/opt**” on page 35

Start a **3270** session to **RH5RWTST**. You should see a virtual Network Interface Card (NIC) defined at virtual addresses 600-602. With the sample **PROFILE EXEC** and **SWAPGEN EXEC**, you should see the two **VDISK** swap spaces get created at virtual addresses 1B2 and 1B3. You should then be prompted to IPL Linux from 1B0:

```
LOGON RH5RWTST
00: NIC 0600 is created; devices 0600-0602 defined
00: z/VM Version 5 Release 4.0, Service Level 0801 (64-bit),
```

```
00: built on IBM Virtualization Technology
00: There is no logmsg data
00: FILES: 0001 RDR, NO PRT, NO PUN
00: LOGON AT 14:15:02 EDT MONDAY 06/01/09
z/VM V5.4.0 2008-12-05 12:25
```

```
DMSACP723I A (191) R/O
DMSACP723I C (592) R/O
DIAG swap disk defined at virtual address 1B2 (16245 4K pages of swap space)
DIAG swap disk defined at virtual address 1B3 (32493 4K pages of swap space)
Do you want to IPL Linux from DASD 1B0? y/n
y
```

Linux should then boot:

```
00: zIPL v1.5.3 interactive boot menu
00:
00: 0. default (linux)
00:
00: 1. linux
00:
00: Note: VM users please use '#cp vi vmsg <input>'
00:
00: Please choose (default will boot in 15 seconds):
00: Booting default (linux)...
Linux version 2.6.18-128.el5 (mockbuild@spud.z900.redhat.com) (gcc version 4.1.2
20080704 (Red Hat 4.1.2-44)) #1 SMP Wed Dec 17 11:45:02 EST 2008
We are running under VM (64 bit mode)
...
```

If Linux does not boot, verify that all the disks were successfully copied.

## Modifying zipl.conf

The parameter line is modified in `/etc/zipl.conf` as follows:

- ▶ The menu time-out (**timeout**) is set to 3 seconds so Linux IPLs more quickly with no user input. The default is to wait for 15 seconds.
- ▶ The parameters `vmppoff=LOGOFF` and `vmhalt=LOGOFF` are added so that VM user IDs are logged off after Linux is shut down.

Back up the original `zipl.conf` then make the following changes:

```
# cd /etc
# cp zipl.conf zipl.conf.orig
# vi zipl.conf
[defaultboot]
default=linux
target=/boot/
timeout=3
[linux]
image=/boot/vmlinuz-2.6.18-128.el5
ramdisk=/boot/initrd-2.6.18-128.el5.img
parameters="root=LABEL=/ vmppoff=LOGOFF vmhalt=LOGOFF"
```

Run `zipl` to write the changes to the `/boot/` directory:

```
# zipl
Using config file '/etc/zipl.conf'
Building bootmap in '/boot/'
Building menu 'rh-automatic-menu'
Adding #1: IPL section 'linux' (default)
```

```
Preparing boot device: dasda (01b0).
Done.
```

## Verifying the dasd driver parameters

The parameters passed to the dasd drivers are set in `/etc/modprobe.conf`. The values should have been obtained from the `LNXCONE CONF-RH5` file on the `LNXCONE 192 (RH5RWMNT 191)` disk. Verify the last line of `/etc/modprobe.conf` contains the additional DASD ranges:

```
# cat /etc/modprobe.conf
alias eth0 qeth
options dasd_mod dasd=1b0-1bf,2b0-2bf,320-33f
```

These extra DASD addresses allow room for growth.

## Inserting the cmm and vmcp modules

The `cmm` and `vmcp` modules are added to the file `/etc/rc.d/rc.local` as they were on `LNXCONE`:

```
# cd /etc/rc.d
# vi rc.local
#!/bin/sh
#
# This script will be executed *after* all the other init scripts.
# You can put your own initialization stuff in here if you don't
# want to do the full Sys V style init stuff.

touch /var/lock/subsys/local
modprobe vmcp
modprobe cmm
```

## Modifying the /etc/inittab file

The `/etc/inittab` file is modified so the system is set to halt on **SIGNAL SHUTDOWN**, not reboot. Rather than rebooting, (`shutdown -r`), the system is set to halt (`shutdown -h`) when the shutdown signal is trapped as a Ctrl-Alt-Del signal:

```
# cd /etc
# vi inittab // change shutdown -r to shutdown -h
...
# Trap CTRL-ALT-DELETE
ca::ctrlaltdel:/sbin/shutdown -t3 -h now
...
```

## Copying the cloneprep.sh and boot.findself scripts

The script `cloneprep.sh` is copied to `/usr/local/sbin/` so it can be used to clean up files just before cloning. The script `boot.findself` is copied to `/etc/init.d/` so it can be run one time at first boot.

The scripts are copied with the `scp` command. In this example, the IP address of the Linux running on `LNXCONE` is `9.60.18.225`.

```
# cd /usr/local/sbin
# scp 9.60.18.225:/usr/local/sbin/cloneprep.sh .
The authenticity of host '9.60.18.225 (9.60.18.225)' can't be established.
RSA key fingerprint is e2:20:51:93:1b:47:25:83:86:08:3a:92:d1:24:e9:9b.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '9.60.18.225' (RSA) to the list of known hosts.
Password:
cloneprep.sh                               100% 2280      2.2KB/s   00:00
# cd /etc/init.d
```

```
# scp 9.60.18.225:/usr/local/sbin/boot.findself .
Password:
boot.findself          100% 6307    6.2KB/s   00:00
```

The `boot.findself` script is set to run at boot time with the `chkconfig` command and the `--add` parameter:

```
# chkconfig --add boot.findself
```

The change is verified:

```
# chkconfig --list | grep boot.findself
boot.findself 0:off 1:on 2:on 3:on 4:on 5:on 6:off
```

See “The `boot.findself` script” on page 69 for a listing of the code and a brief description of the logic.

## Configuring YUM

The `yum` tool is a wrapper around `rpm` that allows easy resolution of dependencies. It is convenient to have `yum` *pointing at* the installation source. To accomplish this, perform the following steps:

1. Create a mount point for the installation source:

```
# mkdir /mnt/rhe15.3
```

2. Create a file named `rhe15.3.repo` in the `/etc/yum.repos.d/` directory with the following four lines:

```
# cd /etc/yum.repos.d/
# vi rhe15.3.repo
[RHEL5.3]
name=RHEL 5.3
baseurl=file:///mnt/rhe15.3/Server
gpgcheck=0
```

This points `yum` to the NFS mount point of the installation source.

3. Back up and modify the `/etc/fstab` file to add an NFS mount of the install source (the line wraps in the example, but it is only a single line):

```
# cp fstab fstab.good
# vi fstab
LABEL=/                /                    ext2    defaults    1 1
LABEL=/var/lib/rpm     /var/lib/rpm        ext2    defaults    1 2
LABEL=/local           /local              ext3    defaults    1 2
LABEL=/boot            /boot               ext2    defaults    1 2
tmpfs                  /dev/shm            tmpfs   defaults    0 0
devpts                 /dev/pts            devpts  gid=5,mode=620 0 0
sysfs                  /sys                sysfs   defaults    0 0
proc                   /proc              proc    defaults    0 0
LABEL=SWAP-dasdd1     swap                swap    defaults    0 0
LABEL=SWAP-dasdc1     swap                swap    defaults    0 0
LABEL=SWAP-dasde1     swap                swap    defaults    0 0
9.60.18.133:/nfs/rhe15.3/dvd /mnt/rhe15.3        nfs     tcp,rsize=8192,wsiz=819
2,timeo=20 0 0
```

4. Mount the installation source and list the RPM files in the `Server/` directory:

```
# mount /mnt/rhe15.3
# ls /mnt/rhe15.3/Server
...
zsh-4.2.6-1.s390x.rpm
zsh-html-4.2.6-1.s390x.rpm
```

This mounts the installation source and effectively tests the addition to the `/etc/fstab` file. Listing the server directory shows the set of installation RPMs.

The `yum` tool should now be configured. You may wish to test this now. However, to show an example of maintaining systems, `yum` is used to install the NTP RPM in section 1.7.4, “Performing maintenance to the golden image” on page 54.

### Creating empty mount points under `/opt/`

Mounting middleware binaries read-only is beyond the scope of this paper. However, if there is a possibility that you may run WebSphere® Application Server, DB2 UDB, or MQ Series, you may wish to create the following, or other, empty mount points:

- ▶ `/opt/IBM/WebSphere/`
- ▶ `/opt/mqm/`
- ▶ `/opt/IBM/db2/`

In this fashion, all cloned servers will have empty mount points for possibly mounting software.

```
# cd /opt
# mkdir mqm IBM
# cd IBM
# mkdir WebSphere db2
```

### Other modifications

These are only a few modifications to the base system. You may choose many other modifications to the golden image.

### Running `cloneprep.sh`

Now the `cloneprep.sh` script can be run. The output should be similar to the following:

```
# cloneprep.sh
rm: cannot remove `~/var/log/*.gz': No such file or directory
System should be ready for shutdown and cloning
```

## 1.5.8 Testing the golden image

The next step is to perform all tests that are necessary and appropriate for your environment. When the golden image has been sufficiently tested, shutdown the system. The user ID should be automatically logged off.

```
# shutdown -h now
...
```

The system should now be ready to clone.

## 1.5.9 Copying the golden image from RH5RWTST to RH5RWMNT

Once the golden image has been thoroughly tested, it can be copied back to the maintenance system with the `TST2MNT EXEC` from the `CMSCONE` user ID:

```
==> tst2mnt
Checking that source and target user IDs are logged off
HCPCQU045E RH5RWTST not logged on
HCPCQU045E RH5RWMNT not logged on

Do you want to copy R/W disks from RH5RWTST to RH5RWMNT? y/n
```

y

```
Copying minidisk 01B0 to 11B0 ...
Command complete: FLASHCOPY 01B0 0 END TO 11B0 0 END
Return value = 0
```

```
Copying minidisk 01B1 to 11B1 ...
Command complete: FLASHCOPY 01B1 0 END TO 11B1 0 END
Return value = 0
```

```
Copying minidisk 01B4 to 11B4 ...
Command complete: FLASHCOPY 01B4 0 END TO 11B4 0 END
Return value = 0
```

```
Copying minidisk 01B5 to 11B5 ...
Command complete: FLASHCOPY 01B5 0 END TO 11B5 0 END
Return value = 0
```

```
Copying minidisk 01B6 to 11B6 ...
Command complete: FLASHCOPY 01B6 0 END TO 11B6 0 END
Return value = 0
```

```
Cleaning up ...
01B0 01B1 01B4 01B5 01B6 11B0 11B1 11B4 DETACHED
11B5 11B6 DETACHED
```

The test and the maintenance systems are now the same. The next step is to create the first *golden* read-write copy.

## 1.5.10 Copying read-write Linux to the golden disks

Again from CMSCLONE copy the maintenance system to the 11Bx disks on the golden system, RH5GOLD with the **MNT2GOLD EXEC**:

```
==> mnt2gold
Checking that source and target user IDs are logged off
HCPCQU045E RH5RWMNT not logged on
HCPCQU045E RH5GOLD not logged on
Do you want to copy R/W disks from RH5RWMNT to RH5GOLD? y/n
y

Copying minidisk 01B0 to 11B0 ...
Command complete: FLASHCOPY 01B0 0 END TO 11B0 0 END
Return value = 0

Copying minidisk 01B1 to 11B1 ...
HCPCMM296E Status is not as required - 01B1; an unexpected condition
HCPCMM296E occurred while executing a FLASHCOPY command, code = A7.
FLASHCOPY failed, falling back to DDR ...
z/VM DASD DUMP/RESTORE PROGRAM
HCPDDR696I VOLID READ IS 0X01B1
HCPDDR696I VOLID READ IS 0X01B1
COPYING 0X01B1
COPYING DATA 08/24/09 AT 18.26.32 GMT FROM 0X01B1 TO 0X01B1
INPUT CYLINDER EXTENTS OUTPUT CYLINDER EXTENTS
START STOP START STOP
...
Cleaning up ...
```

You may see **FLASHCOPY** fail as shown in the example above. This is not unexpected as the copying of the data from the previous EXEC had not completed in the background of the disk subsystem. The **COPYMDSK EXEC** falls back to the **DDR** command which copies data a byte at a time.

Now the same golden image should exist on RH5RWMNT, RH5RWTST and RH5GOLD.

### 1.5.11 Cloning a read-write Linux system

You should now be ready to clone the first read-write system. The **CLONERW EXEC** copies from the RH5GOLD 11Bx disks to 1Bx disks of the target user ID which must be specified.

1. Create a new user ID with the same size and numbered minidisks as the other IDs. In this example it is named LNX224 because the last octet of the IP address will be 224:

```
USER LNX224 PASSWD 256M 1G G
INCLUDE RORDFLT
OPTION APPLMON
MDISK 01B0 3390 0001 0060 DM63D0 MR PASSWD PASSWD PASSWD
MDISK 01B1 3390 0061 3200 DM63D0 MR PASSWD PASSWD PASSWD
MDISK 01B4 3390 3261 0550 DM63D0 MR PASSWD PASSWD PASSWD
MDISK 01B5 3390 3811 1119 DM63D0 MR PASSWD PASSWD PASSWD
MDISK 01B6 3390 4930 0079 DM63D0 MR PASSWD PASSWD PASSWD
```

Bring the changes online, either with the **DIRECTXA** command, or with the appropriate “add” command if you are using a directory maintenance product.

2. Create a parameter file and a configuration file on the CMSCLONE 192 disk (which is the Linux user ID’s read-only 191 disk). By default CMS accesses the 192 disk as D. So the RH5RWMNT parameter file on CMSCLONE’s D disk is copied as a template, and the file name of the configuration file is modified:

```
===> copy rh5rwmnt parm-rh5 d lnx224 = =
===> x lnx224 parm-rh5 d
ramdisk_size=40000 root=/dev/ram0 ro ip=off
CMSDASD=191 CMSCONFFILE=LNX224.CONF-RH5
vnc vncpassword=lnx4vm
```

Now the configuration file is copied and the IP address and host name are modified:

```
===> x lnx224 conf-rh5 d
DASD=1b0-1bf,2b0-2bf,320-33f
HOSTNAME=gpok224.endicott.ibm.com
NETTYPE=qeth
IPADDR=9.60.18.224
SUBCHANNELS=0.0.0600,0.0.0601,0.0.0602
NETWORK=9.60.18.128
NETMASK=255.255.255.128
SEARCHDNS=endicott.ibm.com
BROADCAST=9.60.18.255
GATEWAY=9.60.18.129
DNS=9.0.3.1
MTU=1500
PORTNAME=DONTCARE
PORTNO=0
LAYER2=0
```

3. Grant the new user ID access to the VSWITCH. The following statement is put in AUTOLOG1’s PROFILE EXEC:

```
'cp set vswitch vsw1 grant lnx224'
```

This command is also run interactively from the command line for the current IPL.

By using the RHEL parameter and configuration files to maintain the IP address and host name, there is a side effect. If for some reason you need to install Linux manually, or even use the install process as a rescue system, this file will be available and the correct IP/DNS information will be used.

The read-write Linux system is cloned to the LNX224 user ID with the **CLONERW EXEC**:

```
==> clonerw lnx224
Checking that source and target user IDs are logged off
HCPCQU045E RH5GOLD not logged on
HCPCQU045E LNX224 not logged on
Do you want to copy R/W system from RH5GOLD to LNX224? y/n
y

Copying minidisk 11B0 to 01B0 ...
Command complete: FLASHCOPY 11B0 0 END TO 01B0 0 END
Return value = 0

Copying minidisk 11B1 to 01B1 ...
Command complete: FLASHCOPY 11B1 0 END TO 01B1 0 END
Return value = 0

Copying minidisk 11B4 to 01B4 ...
Command complete: FLASHCOPY 11B4 0 END TO 01B4 0 END
Return value = 0

Copying minidisk 11B5 to 01B5 ...
Command complete: FLASHCOPY 11B5 0 END TO 01B5 0 END
Return value = 0

Copying minidisk 11B6 to 01B6 ...
Command complete: FLASHCOPY 11B6 0 END TO 01B6 0 END
Return value = 0

Cleaning up ...
11B0 11B1 11B4 11B5 11B6 01B0 01B1 01B4 DETACHED
01B5 01B6 DETACHED
```

You should now have four identical systems, three of which can be IPLed: RH5RWMNT, RH5RWTST and LNX226. All of these have identical IP and DNS information. For the first three *system* user IDs, this is acceptable knowing that only one will be booted at a time. However, the target systems will naturally need unique IP and DNS values. The **boot.findself** script sets these values.

You can now log on to LNX224 and IPL from 1B0. At the initial logon, be sure there are no errors related to minidisks nor VSWITCH access:

```
LOGON LNX224
00: NIC 0600 is created; devices 0600-0602 defined
00: z/VM Version 5 Release 4.0, Service Level 0801 (64-bit),
00: built on IBM Virtualization Technology
00: There is no logmsg data
00: FILES: 0001 RDR, NO PRT, NO PUN
00: LOGON AT 09:49:40 EDT TUESDAY 06/02/09
z/VM V5.4.0 2008-12-05 12:25

DMSACP723I A (191) R/O
DMSACP723I C (592) R/O
DIAG swap disk defined at virtual address 1B2 (16245 4K pages of swap space)
DIAG swap disk defined at virtual address 1B3 (32493 4K pages of swap space)
Do you want to IPL Linux from DASD 1B0? y/n
```

Choose to boot Linux from 1B0, and you should notice the modified default boot time of 3 seconds and the modified parameter line:

```
Do you want to IPL Linux from DASD 1B0? y/n
00: zIPL v1.5.3 interactive boot menu
00:
00: 0. default (linux)
00:
00: 1. linux
00:
00: Note: VM users please use '#cp vi vmsg <input>'
00:
00: Please choose (default will boot in 3 seconds):
00: Booting default (linux)...
Linux version 2.6.18-128.el5 (mockbuild@spud.z900.redhat.com) (gcc version 4.1.2
 20080704 (Red Hat 4.1.2-44)) #1 SMP Wed Dec 17 11:45:02 EST 2008
We are running under VM (64 bit mode)
Detected 2 CPU's
Boot cpu address 0
Built 1 zonelists. Total pages: 65536
Kernel command line: root=LABEL=/ vmpoff=LOGOFF vmhalt=LOGOFF BOOT_IMAGE=0
...
```

The script `/etc/init.d/boot.findself` later run in the boot sequence. It should access the 191 disk (CMSCONE 192), read the configuration file and set the TCP/IP address and host name. It does this by modifying the files `/etc/hosts`, and the `eth0` configuration file under `/etc/sysconfig/network-scripts/`. Note that the gateway, DNS server and broadcast information are not modified. The script assumes this information is the same for all Linux virtual servers. If this information is different, you will need to modify the script `/etc/init.d/boot.findself`. See 1.11.1, “The boot.findself script” on page 69 for a complete listing of the source code.

You should see informational messages similar to the following:

```
...
INIT: Entering runlevel: 3
Entering non-interactive startup
Starting boot.findself:
/etc/rc3.d/S01boot.findself: changing (escaped) gpok223\endicott.ibm.com to
gpok224.endicott.ibm.com in /etc/hosts
/etc/rc3.d/S01boot.findself: changing gpok223 to gpok224 and IP address in /etc/
hosts
/etc/rc3.d/S01boot.findself: changing (escaped) 9.60.18.223 to 9.60.18.224 in
/etc/sysconfig/network-scripts/ifcfg-eth0
Y OK ..
...
```

These messages show that the `boot.findself` script ran and modified the IP address and host name. You should now be able to start an SSH session with the cloned system at the updated IP address.

Start an SSH session as root. Verify that the IP address has been reset and that the `boot.findself` script has turned itself off:

```
# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 02:00:04:00:00:11
          inet addr:9.60.18.224  Bcast:9.60.18.255  Mask:255.255.255.128
...
# chkconfig --list boot.findself
service boot.findself supports chkconfig, but is not referenced in any runlevel (run
'chkconfig --add boot.findself')
```

You should now have a conventional read-write Linux system cloned. It is now time to move on to creating a read-only root system.

## 1.6 Building a read-only root system

You should now have a read-write Linux system with tools to maintain copies of Linux for test, maintenance and cloning. It is now time to tackle creating a read-only root system.

As summary, the steps are as follow:

1. "Defining user ID for first read-only root system"
2. "Creating a prototype read-only root Linux" on page 41
3. "Copying read-only Linux to golden disks" on page 47
4. "Cloning a read-only Linux" on page 48

### 1.6.1 Defining user ID for first read-only root system

A user ID **RH5ROGLD** is created with the following directory definition. This is where the read-only system will be created. Note that all minidisks are read-write. This is necessary as the process of moving from a read-write to a read-only system requires access to various disks as read-write for certain periods of time.

```
USER RH5ROGLD PASSWD 256M 1G G
INCLUDE LNXDFLT
OPTION APPLMON
MDISK 01B0 3390 5009 0060 DM63D0 MR PASSWD PASSWD PASSWD
MDISK 01B1 3390 5069 3200 DM63D0 MR PASSWD PASSWD PASSWD
MDISK 01B4 3390 8269 0550 DM63D0 MR PASSWD PASSWD PASSWD
MDISK 01B5 3390 8819 1119 DM63D0 MR PASSWD PASSWD PASSWD
MDISK 01B6 3390 9938 0079 DM63D0 MR PASSWD PASSWD PASSWD
```

The directory changes are brought online and the user ID is given access to system's VSWITCH.

### 1.6.2 Creating a prototype read-only root Linux

The script **mnt2rogld.sh** was written to help facilitate creating a read-only root system from a conventional read-write Linux system. It is neither supported nor well-tested. Check with your Linux distributor and/or support company to be sure that such a system will be supported. If you implement it, test everything well: remember, you are on the *bleeding* edge.

The global variables and functions calls are at the bottom of the script. Here are the global variables:

```
sourceID="RH5RWMNT"
targetID="RH5ROGLD"
```

The source user ID is RH5RWMNT and the target user ID is RH5ROGLD.

The function calls are as follow:

```
checkID $sourceID
checkID $targetID
linkSourceDisks
linkTargetDisks
enableSourceDisks
enableTargetDisks
copySystem
mountSourceRoot
mountTargetDisks
```

```

modifySystem
cleanUp
echo "Exiting with rc 0!"
exit 0

```

See 1.11.3, “The mnt2rogl.sh script” on page 70 for a complete listing of script. Here is a high level description of the functions.

- ▶ The first two function calls to **checkID()** verify that the source and target user IDs are logged off.
- ▶ The next two function calls, **linkSourceDisks()** and **linkTargetDisks()** utilize **vmcp** to link to the RH5RWMNT and RH5ROGLD minidisks.
- ▶ The next two function calls, **enableSourceDisks()** and **enableTargetDisks()**, utilize the Linux **chccwdev** command to enable the source and target disks.
- ▶ The next function **copySystem()**, calls **copyDisk()** to copy all minidisks.
- ▶ The next two function calls **mountSourceRoot()** and **mountTargetDisks()** mount the source and target disks following the Linux file system hierarchy over the directories **/mnt/source/** and **/mnt/target/**.
- ▶ The function **modifySystem()** is where some magic occurs, so it is analyzed in detail. Here is the code:

```

#+-----+
function modifySystem()
# 1) Modify /etc/modprobe.conf and run mkinitrd on target system
# 2) Copy source r/w directories to target /local/
# 3) Configure target system for read-only /
#+-----+
{
  TGT="/mnt/target"
  echo ""
  echo "Backing up and modifying /etc/modprobe.conf file ..."
  cp $TGT/etc/modprobe.conf $TGT/etc/modprobe.conf.orig
  if [ "$?" != 0 ]; then exit 38; fi
  cat $TGT/etc/modprobe.conf.orig | \
    sed -e 's/1b0-1bf/1b0(ro),1b1(ro),1b2-1b5,1b6(ro),1b7-1bf/g' > \
    $TGT/etc/modprobe.conf

```

Above the **(ro)** parameter is added to minidisks 1B0 (/boot/), 1B1 (/) and 1B6 (/var/lib/rpm/) so that Linux knows these disks are read-only.

```

echo ""
echo "Backing up and modifying /etc/fstab file ..."
cp $TGT/etc/fstab $TGT/etc/fstab.orig
if [ "$?" != 0 ]; then exit 39; fi
echo "tmpfs          /tmp                tmpfs    defaults    0 0" >>
$TGT/etc/fstab

```

The code above adds a line to **/etc/fstab** so that **/tmp/** is an in-memory (tmpfs) file system.

```

echo ""
echo "Running mkinitrd in target environment ..."
# yes, this is ugly code:
ramdisk=`ls /mnt/target/boot/initrd-*`
unameimg=${ramdisk##/mnt/target/boot/initrd-}
unamer=${unameimg%.img}
chroot $TGT mkinitrd --with dasd_eckd_mod --with dasd_fba_mod -f
/boot/initrd-${unamer}.img $unamer
if [ "$?" != 0 ]; then exit 40; fi

```

The code above runs the `mkinitrd` command in the target environment to create an updated initial RAMdisk.

```
echo ""
echo "Backing up and modifying /etc/zipl.conf file ..."
cp $TGT/etc/zipl.conf $TGT/etc/zipl.conf.orig
if [ "$?" != 0 ]; then exit 41; fi
cat $TGT/etc/zipl.conf.orig | \
  sed -e 's/vmpoff=LOGOFF/vmpoff=LOGOFF readonlyroot/g' > $TGT/etc/zipl.conf
echo "Running zipl in target environment ..."
chroot $TGT zipl
if [ "$?" != 0 ]; then exit 42; fi
```

The code above runs the `zipl` command in the target “chroot”ed environment.

```
echo ""
echo "Backing up and modifying /etc/sysconfig/readonly-root file ..."
cp $TGT/etc/sysconfig/readonly-root $TGT/etc/sysconfig/readonly-root.orig
sed -i 's/READONLY=no/READONLY=yes/' $TGT/etc/sysconfig/readonly-root
if [ "$?" != 0 ]; then exit 43; fi
sed -i 's/STATE_MOUNT=\/.snapshot/STATE_MOUNT=\/local/'
$TGT/etc/sysconfig/readonly-root
if [ "$?" != 0 ]; then exit 44; fi
```

The code above modifies the `/etc/sysconfig/readonly-root` configuration file.

```
# Add code to /etc/rc.d/rc.local on target system to set hostname
# This is to work around issue with the host name being set *before*
# /local/etc is bind-mounted over /etc)
grepCmd="grep HOSTNAME /etc/sysconfig/network|awk -F= '{print \$2}'"
echo "tgtHost=\`$grepCmd\`" >> $TGT/etc/rc.d/rc.local
echo 'hostname $tgtHost' >> $TGT/etc/rc.d/rc.local
if [ "$?" != 0 ]; then exit 45; fi
```

The code above puts two commands at the end of `/etc/rc.d/rc.local` on the target read-only root golden image. These two lines of code will reset the host name at boot time. The reason this code is necessary is because the host name is set early in the boot sequence: before the directory `/local/etc/` is bind-mounted over `/etc/`. Therefore, the host name on all read-only root clones is set to the golden image’s host name.

```
echo ""
echo "Copying source /local/etc,root,srv,var/ to target /local/ ..."
cp -a $TGT/etc $TGT/local 2>/dev/null
if [ "$?" != 0 ]; then exit 46; fi
cp -a $TGT/root $TGT/local 2>/dev/null
if [ "$?" != 0 ]; then exit 47; fi
cp -a $TGT/srv $TGT/local 2>/dev/null
if [ "$?" != 0 ]; then exit 48; fi
cp -a $TGT/var $TGT/local 2>/dev/null
if [ "$?" != 0 ]; then exit 49; fi
echo /etc > $TGT/local/files
echo /var >> $TGT/local/files
echo /srv >> $TGT/local/files
echo /root >> $TGT/local/files
}
```

Finally, the function `cleanup()` is called to unmount all mounted file systems, disable all devices and then **DETACH** the minidisks.

## Running the `mnt2rogld.sh` script

Now it is time to run the `mnt2rogld.sh` script from the Linux system running on the LNXCLONE user ID. This script takes about six minutes to run. Following is a summary of the output:

```
# mnt2rogld.sh
Invoking CP command: QUERY RH5RWMNT
HCPCQU045E RH5RWMNT not logged on
Error: non-zero CP response for command 'QUERY RH5RWMNT': #45
Invoking CP command: QUERY RH5ROGLD
HCPCQU045E RH5ROGLD not logged on
Error: non-zero CP response for command 'QUERY RH5ROGLD': #45
```

The previous output is from the two calls to the `checkID()` function.

```
Linking source disks ...
Invoking CP command: link RH5RWMNT 1b1 11b1 rr
Invoking CP command: link RH5RWMNT 1b0 11b0 rr
Invoking CP command: link RH5RWMNT 1b4 11b4 rr
Invoking CP command: link RH5RWMNT 1b5 11b5 rr
Invoking CP command: link RH5RWMNT 1b6 11b6 rr

Linking target disks ...
Invoking CP command: link RH5ROGLD 1b1 21b1 mr
Invoking CP command: link RH5ROGLD 1b0 21b0 mr
Invoking CP command: link RH5ROGLD 1b4 21b4 mr
Invoking CP command: link RH5ROGLD 1b5 21b5 mr
Invoking CP command: link RH5ROGLD 1b6 21b6 mr
```

The previous output is from the `linkSourceDisks()` and `linkTargetDisks()` functions.

```
Enabling source disks ...
Setting device 0.0.11b1 online
Done
Setting device 0.0.11b0 online
Done
Setting device 0.0.11b4 online
Done
Setting device 0.0.11b5 online
Done
Setting device 0.0.11b6 online
Done

Enabling target disks ...
Setting device 0.0.21b1 online
Done
Setting device 0.0.21b0 online
Done
Setting device 0.0.21b4 online
Done
Setting device 0.0.21b5 online
Done
Setting device 0.0.21b6 online
Done
```

The previous output is from the `enableSourceDisks()` and `enableTargetDisks()` functions.

```
Copying root file system ...
...
Copying /boot/ ...
...
Copying minidisk swap space ...
...
```

```
Copying /local/ ...
...
Copying /var/lib/rpm/ ...
...
```

The previous output is from the two calls to the **copySystem()** function.

```
Making source mount point ...

Mounting source root file system over /mnt/source ...

Mounting target file systems over /mnt/target ...
mkdir: cannot create directory '/mnt/target': File exists

Making target mount points ...
Mounting memory file systems ...
```

The previous output is from the **mountSourceRoot()** and **mountTargetDisks()** functions.

```
Backing up and modifying /etc/modprobe.conf file ...

Backing up and modifying /etc/fstab file ...

Running mkinitrd in target environment ...

Backing up and modifying /etc/zipl.conf file ...
Running zipl in target environment ...
Using config file '/etc/zipl.conf'
Building bootmap in '/boot/'
Building menu 'rh-automatic-menu'
Adding #1: IPL section 'linux' (default)
Preparing boot device: dasdbs (21b0).
Done.

Backing up and modifying /etc/sysconfig/readonly-root file ...

Copying target /etc,root,srv,var/ to target /local/ ...
```

The previous output is from the two calls to the **modifySystem()** function.

```
Cleaning up ...
Setting device 0.0.21b0 offline
Done
Setting device 0.0.21b1 offline
Done
Setting device 0.0.21b4 offline
Done
Setting device 0.0.21b5 offline
Done
Setting device 0.0.21b6 offline
Done
DASD 21B0 DETACHED
DASD 21B1 DETACHED
DASD 21B4 DETACHED
DASD 21B5 DETACHED
DASD 21B6 DETACHED
Setting device 0.0.11b0 offline
Done
Setting device 0.0.11b1 offline
Done
Setting device 0.0.11b4 offline
Done
```

```

Setting device 0.0.11b5 offline
Done
Setting device 0.0.11b6 offline
Done
DASD 11B0 DETACHED
DASD 11B1 DETACHED
DASD 11B4 DETACHED
DASD 11B5 DETACHED
DASD 11B6 DETACHED

```

The previous output is from the two calls to the `cleanup()` function.

```
Exiting with rc 0!
```

The target file systems are unmounted and minidisks detached so you should now be able to logon to RH5ROGLD and try the new system.

**Tip:** If the script fails, you can immediately issue the command `echo $?` to get the return code. That value should isolate the line in the script that is failing.

Also, if you run the script a second time after failing, you may get an error that a file system is already mounted over `/mnt/source/` or `/mnt/target/`. A helper script has been written named `offliner.sh`. If the `mnt2rogld.sh` fails, run `offliner.sh` to clean up any remaining linked disks.

You should now have a *hybrid* read-only root system copied and modified to the RH5ROGLD user ID. It is a hybrid in that it has all conventional minidisks, not links to minidisks. However, the file systems are mounted read-only. After testing this hybrid system, you will create the first read-only root clone.

## Testing the newly created system

Logon to RH5ROGLD. You should be prompted to boot Linux from 1B0. Linux should start to boot. Boot the system. Note the `readonlyroot` in the parameter line - this is added by the `mnt2rogld.sh` script which makes the assumption that the string `vmpoff=LOGOFF` exists in the parameter line. This parameter tells the kernel to mount the root file system read-only.

```

LOGON RH5ROGLD
00: NIC 0600 is created; devices 0600-0602 defined
00: z/VM Version 5 Release 4.0, Service Level 0801 (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 11:07:24 EDT TUESDAY 08/25/09
z/VM V5.4.0   2008-10-22 15:36

DMSACP723I A (191) R/O
DMSACP723I C (592) R/O
DIAG swap disk defined at virtual address 1B2 (16245 4K pages of swap space)
DIAG swap disk defined at virtual address 1B3 (32493 4K pages of swap space)
Do you want to IPL Linux from DASD 1B0? y/n
y
00: zIPL v1.5.3 interactive boot menu
00:
00: 0. default (linux)

```

```

00:
00: 1. linux
00:
00: Note: VM users please use '#cp vi vmsg <input>'
00:
00: Please choose (default will boot in 3 seconds):
00: Booting default (linux)...
Linux version 2.6.18-128.el5 (mockbuild@spud.z900.redhat.com) (gcc version 4.1.2
 20080704 (Red Hat 4.1.2-44)) #1 SMP Wed Dec 17 11:45:02 EST 2008
We are running under VM (64 bit mode)
Detected 2 CPU's
Boot cpu address 0
Built 1 zonelists. Total pages: 65536
Kernel command line: root=LABEL=/ vmpoff=LOGOFF readonlyroot vmhalt=LOGOFF BOOT_
IMAGE=0
...

```

The system should complete booting. Start an SSH session as root to the new system.

Even though this system has many read-write minidisks, it should be configured with most of the directories linked read-only. You may choose to verify that certain disks are read-only while others are read-write. Try the following command to create an empty file named foo in many different directories:

```

# for i in /etc/foo /var/foo /root/foo /srv/foo /tmp/foo /foo /var/lib/rpm/foo
> do
>   echo "touch $i"
>   touch $i
> done
touch /etc/foo
touch /var/foo
touch /root/foo
touch /srv/foo
touch /tmp/foo
touch /foo
touch: cannot touch `~/foo': Read-only file system
touch /var/lib/rpm/foo
touch: cannot touch `~/var/lib/rpm/foo': Read-only file system

```

The first five commands should succeed because those are the read-write directories. The last two commands should fail because those directories are accessed read-only. After that is verified you may want to delete the empty files so they don't get cloned.

```

# rm -f /etc/foo /var/foo /root/foo /srv/foo /tmp/foo

```

### 1.6.3 Copying read-only Linux to golden disks

Shutdown the read-only root system on RH5R0GLD and log off.

```

# shutdown -h now
...
Unmounting pipe file systems:
Unmounting file systems: umount2: Invalid argument
umount: /var/lib/stateless/writable: not mounted
umount2: Device or resource busy
umount: /etc: device is busy

Unmounting file systems (retry): umount2: Device or resource busy
umount: /etc: device is busy

```

```
Unmounting file systems (retry): umount2: Device or resource busy
umount: /etc: device is busy
```

Logon to CMSCLONE and use the **R02GOLD EXEC** to copy the contents to the 21Bx minidisks on RH5GOLD.

```
==> ro2gold
Checking that source and target user IDs are logged off
HCPCQU045E RH5ROGLD not logged on
HCPCQU045E RH5GOLD not logged on

Do you want to copy disks from RH5ROGLD to RH5GOLD? y/n
y

Copying minidisk 01B0 to 21B0 ...
Command complete: FLASHCOPY 01B0 0 END TO 21B0 0 END
Return value = 0

Copying minidisk 01B1 to 21B1 ...
Command complete: FLASHCOPY 01B1 0 END TO 21B1 0 END
Return value = 0

Copying minidisk 01B4 to 21B4 ...
Command complete: FLASHCOPY 01B4 0 END TO 21B4 0 END
Return value = 0

Copying minidisk 01B5 to 21B5 ...
Command complete: FLASHCOPY 01B5 0 END TO 21B5 0 END
Return value = 0

Copying minidisk 01B6 to 21B6 ...
Command complete: FLASHCOPY 01B6 0 END TO 21B6 0 END
Return value = 0

Cleaning up ...
01B0 01B1 01B4 01B5 01B6 21B0 21B1 21B4 DETACHED
21B5 21B6 DETACHED
```

The modified read-only version of the golden read-write system (on the 21Bx disks) should now be “alongside” the read-write version of the system (on the 11Bx disks) on RH5GOLD.

## 1.6.4 Cloning a read-only Linux

You should now be able to clone the first read-only root Linux system. A user ID, LNX241, is created to clone the system to. Only the 1B4 (swap) and 1B5 (/local/) minidisks are needed to be read-write. The other disks are read-only links to the RH5GOLD 21Bx minidisks as defined in the RORDFLT user directory profile.

The directory entry for the LNX241 virtual machine is as follows:

```
USER LNX241 PASSWD 256M 1G G
INCLUDE RORDFLT
OPTION APPLMON
MDISK 01B4 3390 0001 0550 DM63D1 MR PASSWD PASSWD PASSWD
MDISK 01B5 3390 0551 1119 DM63D1 MR PASSWD PASSWD PASSWD
```

Again a parameter and a configuration file must be created on the CMSCLONE 192 disk. The RH5RWMNT parameter file is copied as a template, and the host name and IP address are modified:

```

==> copy rh5rwmnt parm-rh5 d lnx241 = =
==> x lnx241 parm-rh5 d
ramdisk_size=40000 root=/dev/ram0 ro ip=off
CMSDASD=191 CMSCONFFILE=LNx241.CONF-RH5
vnc vncpassword=lnx4vm
method=nfs:tcp,rsiz=8192,wsiz=8192,timeo=20:9.60.18.133:/nfs/rhe15.3/dvd
==> copy rh5rwmnt conf-rh5 d lnx241 = =
==> x lnx241 conf-rh5 d
DASD=1b0-1bf,2b0-2bf,320-33f
HOSTNAME=gpok241.endicott.ibm.com
NETTYPE=qeth
IPADDR=9.60.18.241
SUBCHANNELS=0.0.0600,0.0.0601,0.0.0602
NETWORK=9.60.18.128
NETMASK=255.255.255.128
SEARCHDNS=endicott.ibm.com
BROADCAST=9.60.18.255
GATEWAY=9.60.18.129
DNS=9.0.3.1
MTU=1500
PORTNAME=DONTCARE
PORTNO=0
LAYER2=0

```

Be sure to give the new user ID access to the system VSWITCH. You should now be ready to clone a read-only Linux system with the **CLONERO EXEC**:

```

==> clonero lnx241
clonero lnx241
Checking that source and target user IDs are logged off
HCPCQU045E RH5GOLD not logged on
HCPCQU045E LN241 not logged on

Do you want to copy R/O system from RH5GOLD to LN241 y/n
y
...

```

Log off of CMSCLONE and logon to the newly created read-only root clone, LN241 in this example. Boot the new Linux system. Again, watch the messages to see **boot.findself** modify the host name and IP address:

```

...
/etc/rc3.d/S01boot.findself: changing (escaped) gpok223\endicott.ibm.com to gpok241.endicott.ibm.com in /etc/hosts
/etc/rc3.d/S01boot.findself: changing gpok223 to gpok241 and IP address in /etc/hosts
/etc/rc3.d/S01boot.findself: changing gpok223 to gpok241 /etc/sysconfig/network

/etc/rc3.d/S01boot.findself: changing (escaped) 9\60\18\223 to 9.60.18.241 in /etc/sysconfig/network-scripts/ifcfg-eth0
...
Red Hat Enterprise Linux Server release 5.3 (Tikanga)
Kernel 2.6.18-128.el5 on an s390x

```

**gpok241** login:

Start an SSH session to the new system as root. Issue the **mount** command:

```

# mount
/dev/dasdb1 on / type ext2 (rw)
proc on /proc type proc (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)

```

```
sunrpc on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw)
/dev/dasdg1 on /var/lib/rpm type ext2 (ro)
/dev/dasda1 on /boot type ext2 (ro)
tmpfs on /dev/shm type tmpfs (rw)
tmpfs on /tmp type tmpfs (rw)
```

Note that the `/boot/` and `/var/lib/rpm/` file systems are shown as ext2 and being mounted read-only (**ro**). The root file system is shown as ext2, but is shown erroneously as being read-write (**rw**).

Also note that the read-write mount of `/local/` and the bind mounts of `/etc/`, `/root/`, `/srv/` and `/var/` are not shown. This is probably because the `/etc/mtab` file is not accessible as read-write early enough in the boot process. For more details on this, see:

[https://bugzilla.redhat.com/show\\_bug.cgi?id=214891](https://bugzilla.redhat.com/show_bug.cgi?id=214891)

You should be able to see that these file systems are mounted in the file `/proc/mounts`:

```
# cat /proc/mounts
rootfs / rootfs rw 0 0
/dev/root / ext2 ro 0 0
/dev /dev tmpfs rw 0 0
/proc /proc proc rw 0 0
/sys /sys sysfs rw 0 0
none /selinux selinuxfs rw 0 0
devpts /dev/pts devpts rw 0 0
none /var/lib/stateless/writable tmpfs rw 0 0
/dev/dasdf1 /local ext3 rw,data=ordered 0 0
/dev/dasdf1 /etc ext3 rw,data=ordered 0 0
/dev/dasdf1 /var ext3 rw,data=ordered 0 0
/dev/dasdf1 /srv ext3 rw,data=ordered 0 0
/dev/dasdf1 /root ext3 rw,data=ordered 0 0
...
```

You should now have a read-only root clone of the golden image running.

## 1.7 Maintaining systems

In the original IBM Redpaper, the maintenance of the read-only and read-write systems was described only at a high level. In the updated paper, based on SLES 10 SP2, a methodology set up at Penn State was described. It adds a second user ID, `S10GOLD2`, which contains a second set of minidisks for read-only golden Linux image. This second user ID stores the next Linux system that is to be rolled out while the current system remains stored on the first user ID.

This paper has dropped the text of the original methodology and follows the updated paper. The new user ID is named `RH5GOLD2`. Figure 1-16 shows the block diagram of the updated methodology.

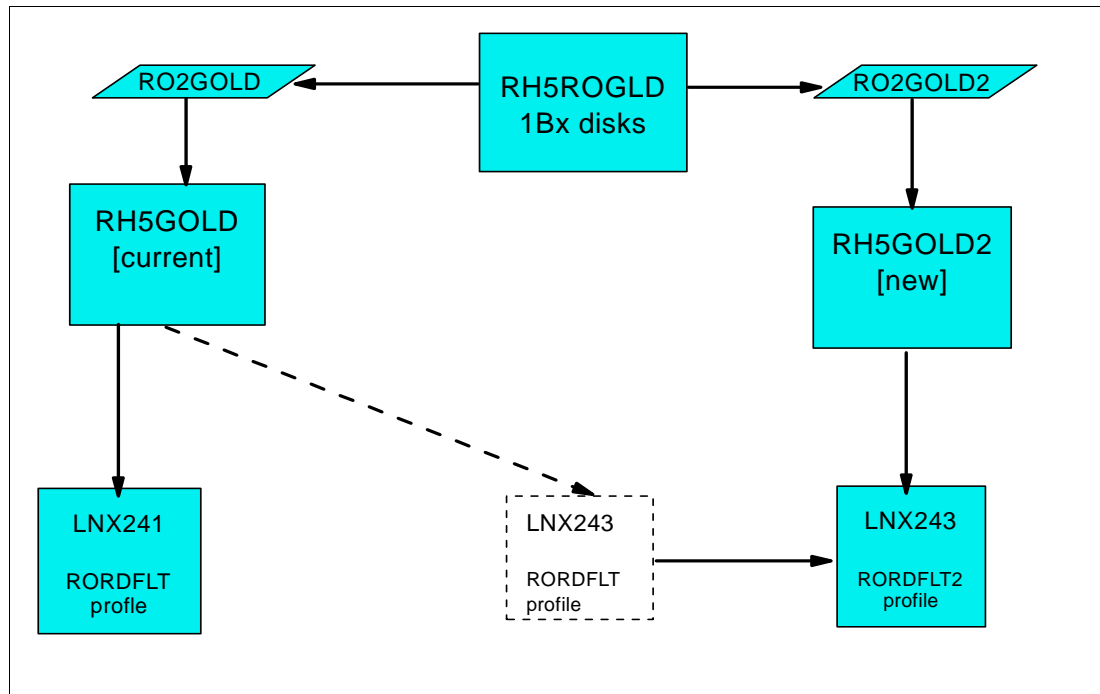


Figure 1-16 Adding a second user ID, `ROGOLD2`, to store the updated golden image

When read-only Linux clones are running and linked to the read-only golden disks, these disks cannot be updated until all virtual machines are either shutdown or migrated (changing the disks while Linux systems are running against them will likely lead to Linux crashing in strange ways).

The virtual machine shown in the lower left, `LNK241`, is running using links the current golden read-only disks. A second read-only system, `LNK243`, is cloned and also set to point to the same read-only root system.

When an updated Linux system is needed, a new golden image is created and tested on `RH5RWMNT` and `RH5RWTST` as described previously. A new read-only root hybrid system is again created on `RH5ROGLD` with the `mnt2rog1d.sh` script. This updated system is copied to the new `RH5GOLD2` user ID by means of a new `R02GOLD2 EXEC`.

Now the read-only root clones can be shut down, and migrated to the updated system. This is done by changing the `LINK` statements in the directory entry of the system being migrated. It is simply set to point to the 21Bx minidisks on `RH5GOLD2` rather than `RH5GOLD`. To make changing the link statements easier, a new user directory `PROFILE` is created.

The following sections demonstrate how to implement this maintenance system:

- ▶ “Disk planning for maintenance system”
- ▶ “Preparing for the new maintenance system” on page 52
- ▶ “Cloning a new read-only Linux system” on page 53
- ▶ “Performing maintenance to the golden image” on page 54
- ▶ “Copying test system to maintenance system and golden disks” on page 56
- ▶ “Converting the updated golden image to a read-only system” on page 56
- ▶ “Copying the new golden image to the second golden virtual machine” on page 56
- ▶ “Updating the directory of one read-only Linux” on page 57
- ▶ “Testing the updated golden image” on page 57
- ▶ “Observing the status of the system” on page 58

## 1.7.1 Disk planning for maintenance system

Refer to Figure 1-10 on page 19 for the current disk layout. The only user ID utilizing the disk with a label of DM63D1 is LNX241. It uses the first 1668 cylinders. A second user ID for storing golden disks, RH5GOLD2, and a second user ID for a read-only root clone, LNX243 are created using the space on the DM63D1 disk. Figure 1-17 on page 52 shows the updated layout.

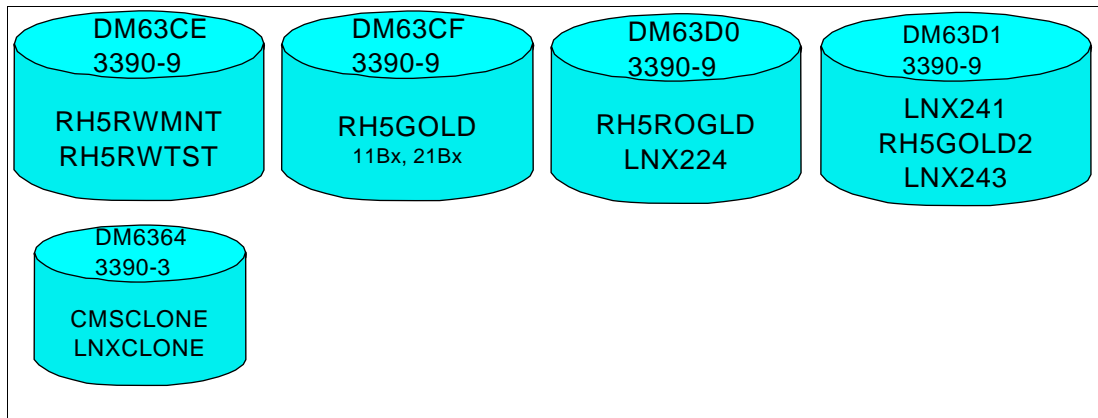


Figure 1-17 Disk planning with four 3390-9s

Next the new profile and user IDs are created.

## 1.7.2 Preparing for the new maintenance system

Create new systems by performing the following steps:

1. Create a RH5GOLD2 user ID that cannot be logged on to with a set of 21Bx minidisks, but not 11Bx (a second set of 11Bx disks is not needed because the read/write Linux systems stand on their own - they never link to golden disks):

```

USER RH5GOLD2 NOLOG 64M 1G G
INCLUDE LNXDFLT
OPTION APPLMON
MDISK 21B0 3390 1670 0060 DM63D1 MR PASSWD PASSWD PASSWD
MDISK 21B1 3390 1730 3200 DM63D1 MR PASSWD PASSWD PASSWD
MDISK 21B4 3390 4930 0550 DM63D1 MR PASSWD PASSWD PASSWD
MDISK 21B5 3390 5480 1119 DM63D1 MR PASSWD PASSWD PASSWD
MDISK 21B6 3390 6599 0079 DM63D1 MR PASSWD PASSWD PASSWD

```

2. Create a new user directory profile. This is based on the profile named RORDFLT defined previously, however, they also include the LINK statements to the read-only disks:

```

PROFILE RORDFLT2
  IPL CMS
  MACHINE ESA 4
  CPU 00 BASE
  CPU 01
  NICDEF 0600 TYPE QDIO LAN SYSTEM VSW1
  SPOOL 000C 2540 READER *
  SPOOL 000D 2540 PUNCH A
  SPOOL 000E 1403 A
  CONSOLE 009 3215 T
  LINK MAINT 0190 0190 RR
  LINK MAINT 019D 019D RR
  LINK MAINT 019E 019E RR
  LINK CMSCLONE 192 191 RR
  LINK TCPMAINT 592 592 RR
  LINK RH5GOLD2 21B0 01B0 RR
  LINK RH5GOLD2 21B1 01B1 RR
  LINK RH5GOLD2 21B6 01B6 RR

```

3. A directory entry for a new read-only root clone is created, LNX243 in this example. It is initially set to point to the first read-only root system on RH5GOLD:

```

USER LNX243 PASSWD 256M 1G G
  INCLUDE RORDFLT
  OPTION APPLMON
  MDISK 01B4 3390 6678 0550 DM63D1 MR PASSWD PASSWD PASSWD
  MDISK 01B5 3390 7228 1119 DM63D1 MR PASSWD PASSWD PASSWD

```

4. Bring the directory changes online for the new profile and the two new user IDs.

There is now a new directory profile, RORDFLT2, a new user ID for a second read-only root system, RH5GOLD2, and a new user ID for a read-only clone, LNX243.

### 1.7.3 Cloning a new read-only Linux system

The next step is to create a second read-only clone.

1. Logoff of MAINT and logon to CMSCLONE. Create a new configuration file on the CMSCLONE 192 disk. Following is an example of the LNX243 CONF-RH5 file. The IP address and the DNS host name are set correctly:

```

DASD=1b0-1bf,2b0-2bf,320-33f
HOSTNAME=gpok243.endicott.ibm.com
NETTYPE=qeth
IPADDR=9.60.18.243
SUBCHANNELS=0.0.0600,0.0.0601,0.0.0602
NETWORK=9.60.18.128
NETMASK=255.255.255.128
SEARCHDNS=endicott.ibm.com
BROADCAST=9.60.18.255
GATEWAY=9.60.18.129
DNS=9.0.3.1
MTU=1500
PORTNAME=DONTCARE
PORTNO=0
LAYER2=0

```

2. Create a new parameter file that points to the new configuration file (this may not actually be needed as it is used in a manual install, but it is good to have it for completeness):

```

ramdisk_size=40000 root=/dev/ram0 ro ip=off
CMSDASD=191 CMSCONFFILE=LNX243.CONF-RH5

```

```
vnc vncpassword=lnx4vm
method=nfs:tcp,rsize=8192,wsize=8192,timeo=20:9.60.18.133:/nfs/rhel5.3
```

3. Grant permission for the new system to attach to the VSWITCH. The following line is added to the PROFILE EXEC on the AUTOLOG 191 disk. Also, the command is run interactively to grant the permissions immediately:

```
'cp set vswitch vsw1 grant lnx243'
```

4. Clone the new read-only system with the **CLONERO EXEC**. Following is an example of cloning to LNX243:

```
==> clonero lnx243
```

```
Checking that source and target user IDs are logged off
HCPCQU045E RH5GOLD not logged on
HCPCQU045E LNX243 not logged on
```

```
Do you want to copy R/O system from RH5GOLD to LNX243 y/n
```

```
y
```

```
...
```

5. Logon to the new clone and start it. Again you should see **boot.findself** run later in the boot process and modify the IP address and host name:

```
...
```

```
Setting hostname gpok223.endicott.ibm.com: Y OK ..
```

```
No devices found
```

```
Setting up Logical Volume Management: Y OK ..
```

```
Mounting local filesystems: Y OK ..
```

```
Enabling local filesystem quotas: Y OK ..
```

```
rm: cannot remove directory `~/tmp/.ICE-unix': Read-only file system
```

```
chown: changing ownership of `~/tmp/.ICE-unix': Read-only file system
```

```
Enabling /etc/fstab swaps: Y OK ..
```

```
INIT: Entering runlevel: 3
```

```
Entering non-interactive startup
```

```
Starting boot.findself:
```

```
/etc/rc3.d/S01boot.findself: changing (escaped) gpok223\endicott\ibm\com to gpok243.endicott.ibm.com in /etc/hosts
```

```
/etc/rc3.d/S01boot.findself: changing gpok223 to gpok243 and IP address in /etc/hosts
```

```
/etc/rc3.d/S01boot.findself: changing (escaped) 9\60\18\223 to 9.60.18.243 in /etc/sysconfig/network-scripts/ifcfg-eth0
```

```
Y OK ..
```

```
...
```

The new read-only Linux system has been created, cloned and started. You should now have two identical read-only root clones. In this example, they are LNX241 and LNX243.

## 1.7.4 Performing maintenance to the golden image

Now an update to the golden image is made. In this simple example the maintenance of the golden image performed is to add the NTP (Network Time Protocol) client RPM. This RPM allows Linux to update its software clock very accurately using a reliable NTP server.

1. Log on to the user ID RH5RWTST and boot Linux.
2. Start an SSH session as root to the newly booted Linux.
3. Show that NTP is not installed by issuing the **ntpddate** command:

```
# ntpddate
-bash: ntpdate: command not found
```

4. Install the NTP client RPM with the `yum install ntp` command. The `yum` command should have been configured in “Configuring YUM” on page 34:

```
# yum install ntp
Loaded plugins: rhnplugin, security
This system is not registered with RHN.
RHN support will be disabled.
Setting up Install Process
Parsing package install arguments
Resolving Dependencies
--> Running transaction check
--> Package ntp.s390x 0:4.2.2p1-9.e15 set to be updated
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package      Arch          Version           Repository        Size
=====
Installing:
ntp          s390x         4.2.2p1-9.e15    RHEL5.3          1.4 M

Transaction Summary
=====
Install      1 Package(s)
Update      0 Package(s)
Remove      0 Package(s)

Total download size: 1.4 M
Is this ok [y/N]: y
Downloading Packages:
Running rpm_check_debug
Running Transaction Test
Finished Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing      : ntp                      [1/1]

Installed: ntp.s390x 0:4.2.2p1-9.e15
Complete!
```

5. Show that the NTP client RPM is installed:

```
# ntpdate
26 Aug 10:24:17 ntpdate[936]: no servers can be used, exiting

The ntpdate command is found and this shows that the NTP RPM was successfully installed (no NTP servers can be found because the Internet is not accessible from these systems).
```

6. Shut down the system.

```
# halt

Broadcast message from root (pts/0) (Wed Aug 26 11:26:37 2009):

The system is going down for system halt NOW!

It should shut down and automatically log off.
```

Maintenance to the golden image on RH5RWTST and a brief test are now complete.

## 1.7.5 Copying test system to maintenance system and golden disks

Now the new golden image has been updated and tested on the read-write test system (RH5RWTST), it is copied to back to the maintenance system (RH5RWMNT) with the **TST2MNT EXEC**

```
==> tst2mnt
Checking that source and target user IDs are logged off
HCPCQU045E RH5RWTST not logged on
HCPCQU045E RH5RWMNT not logged on

Do you want to copy R/W disks from RH5RWTST to RH5RWMNT? y/n
y
...
```

Now it is copied from maintenance system (RH5RWMNT) to the read-write golden disks (RH5GOLD 11Bx) with the **MNT2GOLD EXEC**:

```
==> mnt2gold
Checking that source and target user IDs are logged off
HCPCQU045E RH5RWMNT not logged on
HCPCQU045E RH5GOLD not logged on
Do you want to copy R/W disks from RH5RWMNT to RH5GOLD? y/n
y
...
```

The change to the test system has now been copied back to the maintenance system and read-write golden disks.

## 1.7.6 Converting the updated golden image to a read-only system

From the Linux system running on LNXCLONE, the read-write system on RH5RWMNT is converted into a hybrid read-only system on RH5ROGLD. This step can be done concurrently with the previous step since both scripts read from RH5RWMNT disks:

```
==> mnt2rogld.sh
...
```

When the script completes, log on to RH5ROGLD and boot the hybrid read-only root system. Again you should find the **ntpddate** command in root's PATH. When you are sure the updated system is correct, shut it down.

## 1.7.7 Copying the new golden image to the second golden virtual machine

Now the new golden image is copied to RH5GOLD2 so as to not affect the other read-only systems linked to the read-only disks on RH5GOLD (LNX241 and LNX243 in this example). To populate the new set of disks on RH5GOLD2, a **R02GOLD2 EXEC** is written (included in the associated tar file). It is almost identical to the **R02GOLD** script, except that it populates the disks on RH5GOLD2.

```
==> ro2gold2
1
DASD 21B1 DETACHED
Do you want to copy disks from RH5ROGLD to RH5GOLD2? y/n
y

Copying minidisk 01B0 to 21B0 ...
Command complete: FLASHCOPY 01B0 0 END TO 21B0 0 END
Return value = 0
```

```
Copying minidisk 01B1 to 21B1 ...
Command complete: FLASHCOPY 01B1 0 END TO 21B1 0 END
Return value = 0
```

```
Copying minidisk 01B4 to 21B4 ...
Command complete: FLASHCOPY 01B4 0 END TO 21B4 0 END
Return value = 0
```

```
Copying minidisk 01B5 to 21B5 ...
Command complete: FLASHCOPY 01B5 0 END TO 21B5 0 END
Return value = 0
```

```
Cleaning up ...
01B0 01B1 01B4 01B5 21B0 21B1 21B4 21B5 DETACHED
Setting current Gold Disk Variable ...
```

Now the golden image has been updated and its read-only contents copied to RH5GOLD2. The current system still exists on RH5GOLD.

## 1.7.8 Updating the directory of one read-only Linux

The LNX243 user ID can now be migrated to the updated golden read-only image. This is done by simply changing to the RORDFLT2 profile from RORDFLT. The change results in the system linking to the read-only disks on RH5GOLD2 rather than RH5GOLD.

```
USER LNX243 PASSWD 256M 1G G
INCLUDE RORDFLT2
  OPTION APPLMON
  MDISK 01B4 3390 1670 0550 DM6368 MR PASSWD PASSWD PASSWD
  MDISK 01B5 3390 2220 1119 DM6368 MR PASSWD PASSWD PASSWD
```

Bring the directory change online the appropriate command.

## 1.7.9 Testing the updated golden image

To update LNX243 with the updated directory entry, Linux must be shut down. Because of the kernel parameters `vmppoff=LOGOFF`, the virtual machine should automatically log off:

```
# shutdown -h now
...
```

**Important:** You will probably see error messages when shutting the system down of the following form:

```
...
Unmounting file systems: umount2: Invalid argument
umount: /var/tmp: not mounted
umount2: Invalid argument
umount: /var/run: not mounted
umount2: Invalid argument
umount: /var/log: not mounted
umount2: Invalid argument
umount: /var/lock: not mounted
umount2: Invalid argument
umount: /var/lib/stateless/writable: not mounted
...
```

This is a known issue. The system does eventually shut down. This issue will hopefully be fixed with the remedy to Red Hat Bugzilla but number 214891. See:

[https://bugzilla.redhat.com/show\\_bug.cgi?id=214891](https://bugzilla.redhat.com/show_bug.cgi?id=214891)

Log back on to LNX243 and IPL Linux. Login as root. Try the ntpdate command:

```
# ntpdate
4 Jun 10:21:28 ntpdate[1704]: no servers can be used, exiting
```

This shows that the read-only root system on LNX243 has picked up the NTP RPM from the updated golden image read-only disks copied to RH5GOLD2.

## 1.7.10 Observing the status of the system

The tricky part is keeping track of your system as more and more machines begin sharing the golden disks. To assist in keeping track of golden disks, the **LINKED EXEC** was written and is included in the associated tar file. Following is an example of using it:

```
==> linked

RH5ROGLD LINKS: TEST

RH5GOLD LINKS: GOLD 1
LNX241

RH5GOLD2 LINKS: [[GOLD 2]]
LNX243
```

The brackets around GOLD 2 indicate that this is the most current golden disks, meaning **ro2gold2** was the last script to be executed. Both **ro2gold** and **ro2gold2** write a variable to the file **CURGOLD FILE A** on **CMSCLONE** after execution to keep track of maintenance progress. In addition, the **CLONERO** exec also uses the **CURGOLD FILE A** in order to make sure the most recently updated golden disks are used when creating new read-only Linux guests.

## 1.7.11 Cloning a read-write Linux

A second test is run on the read-write gold disk. A new read-write Linux system is re-cloned to LNX224

```
==> clonerw 1nx224  
...
```

The new clone picks up the NTP RPM:

```
gpok224:~ # ntpdate  
3 Sep 07:52:38 ntpdate[3887]: no servers can be used, exiting
```

## 1.7.12 A final consideration

A final word of caution: this example of maintenance shows adding a simple RPM. The change to the RPM database in `/var/lib/rpm/` is addressed, but there can be other changes, especially to `/etc/`.

Consider the following sequence of commands:

```
# yum erase ntp  
...  
# reboot  
... // start a new SSH session after system comes back up  
# find /etc -newer /proc/1  
/etc  
/etc/sysconfig/hwconf  
/etc/mtab  
/etc/printcap  
/etc/blkid  
/etc/blkid/blkid.tab  
/etc/blkid/blkid.tab.old  
/etc/lvm/cache  
/etc/lvm/cache/.cache  
/etc/aliases.db
```

This shows the files and directories that have changed in `/etc/` since the system was booted (since `/proc/1/` is the directory corresponding to the Linux `init` script). These changes to `/etc/` can be disregarded since they are a normal part of the boot process (but it could be argued that they *all* belong in `/var/`).

Now the `ntp` RPM is added back and set to be started:

```
# yum install ntp  
...  
# chkconfig ntpd on
```

Now look at the changes after the `ntp` RPM is added:

```
# find /etc -newer /proc/1  
/etc  
/etc/sysconfig  
/etc/sysconfig/hwconf  
/etc/mtab  
/etc/printcap  
/etc/rc.d/init.d  
/etc/rc.d/rc0.d  
/etc/rc.d/rc0.d/K74ntpd  
/etc/rc.d/rc1.d  
/etc/rc.d/rc1.d/K74ntpd  
/etc/rc.d/rc2.d  
/etc/rc.d/rc2.d/S58ntpd  
/etc/rc.d/rc3.d  
/etc/rc.d/rc3.d/S58ntpd  
/etc/rc.d/rc4.d
```

```
/etc/rc.d/rc4.d/S58ntpd
/etc/rc.d/rc5.d
/etc/rc.d/rc5.d/S58ntpd
/etc/rc.d/rc6.d
/etc/rc.d/rc6.d/K74ntpd
/etc/blkid
/etc/blkid/blkid.tab
/etc/blkid/blkid.tab.old
/etc/lvm/cache
/etc/lvm/cache/.cache
/etc/aliases.db
/etc/ntp
```

This output shows that `/etc/rc.d/` has been modified. The read-only clones have a read-write `/etc/` so they will not pick up those changes to the golden image. So the **chkconfig** command will probably have to be run on each of the read-only clones to get them in sync. Ideally, `/etc/` should be read-only to address this issue, but that introduces new challenges. Each time maintenance is done, some additional work may have to be done on each of the read-write clones.

## 1.8 Other considerations

This paper describes a single solution. You may consider other changes such as:

- ▶ “Utilizing logical volumes” on page 61
- ▶ “Implementing /home/ with automount, NFS and LDAP” on page 61
- ▶ “Enabling Collaborative Memory Management (CMM)” on page 62

### 1.8.1 Utilizing logical volumes

Using Linux Logical Volume Manager (LVM) would make a lot of sense for file systems that are likely to grow large such as /var/ and perhaps /srv/. They were not implemented in this paper for simplicity and due to time constraints.

If you decide to implement your solution using logical volumes, and you plan to use the `mnt2rog1d.sh` script, it would have to be modified accordingly.

Following is a sample function to create logical volumes on the target system that was lightly tested. It targets device 2B0 which is linked read-write at virtual address 22B0 and the device file is set in the variable `dev22b0`:

```
function makeLVs
{
    echo ""
    echo "Making logical volumes ..."
    fdasd -a $dev22b0
    pvcreate "$dev22b0"1
    vgcreate rwVG "$dev22b0"1
    lvcreate -L 200M -n varLV rwVG
    lvcreate -L 1.2G -n srvLV rwVG
    mke2fs /dev/rwVG/varLV > /dev/null
    mke2fs /dev/rwVG/srvLV > /dev/null
    echo ""
    echo "Mounting logical volumes ..."
    vgscan
    vgchange -ay
    mount /dev/rw_vg/srv_lv /mnt/target/srv
    mount /dev/rw_vg/var_lv /mnt/target/var
}
```

### 1.8.2 Implementing /home/ with automount, NFS and LDAP

It is convenient for system administrators and developers to keep their work in their own home directory under /home/. With tens or even hundreds of Linux systems you would normally have tens or even hundreds of home directories. The size requirement for this file system would vary widely from server to server.

It is easier to manage a single large /home/ directory that “follows users around” This can be implemented with the automount service, NFS and LDAP. LDAP allows for central authentication and authorization of users and resources. Following is a block diagram:

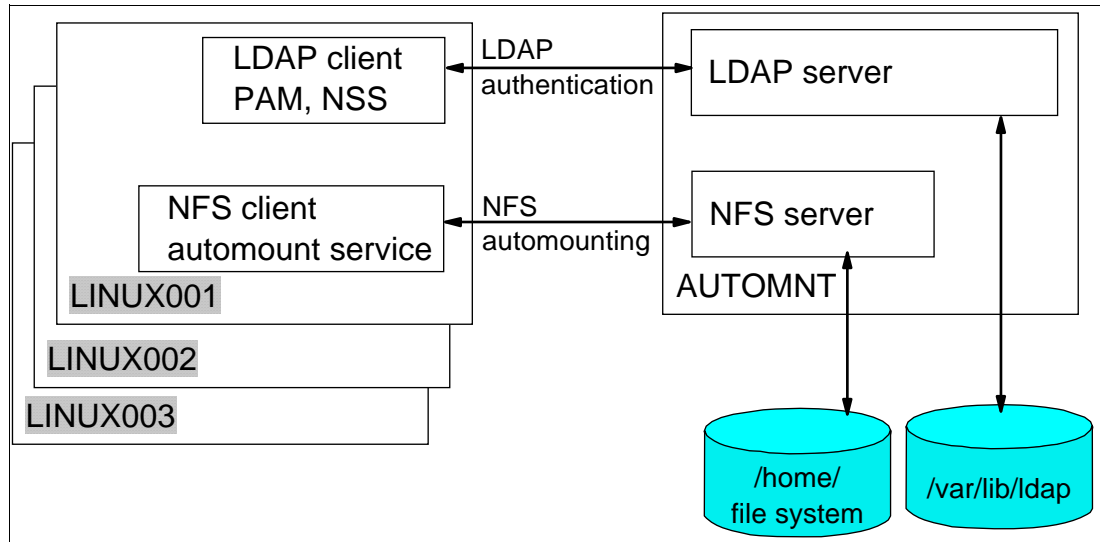


Figure 1-18 Automounted /home/ block diagram

For details on how to implement this, see the IBM Redbook *z/VM and Linux on IBM System z The Virtualization Cookbook for RHEL 5.2*, SG24-7492, on the Web at:

<http://www.redbooks.ibm.com/abstracts/sg247492.html>

### 1.8.3 Enabling Collaborative Memory Management (CMM)

VMRM is a z/VM tool that provides functions to dynamically tune the z/VM system. Groups of virtual machines can be defined to be part of a workload. The workloads can then be managed by VMRM to goals that are also defined

Cooperative Memory Management (CMM1) is a facility which provides z/VM with a way to indirectly control the amount of memory that Linux uses. This technique is known as ballooning, and is implemented by a special driver in Linux. When VMRM determines that the system is becoming too constrained by memory, it will direct the driver to allocate and pin memory within Linux, which forces Linux to relinquish or page out some of the memory it had been using. In turn, the driver informs z/VM of the pages it has pinned. Since the driver was designed never to use these pages, z/VM knows they are now available for it to use, and so can reclaim them, thus reducing overall system memory pressure.

On the other hand, when VMRM determines that the system is no longer under memory pressure, it will direct the ballooning driver to reduce the number of pages it has pinned, thus allowing Linux to expand its memory usage once more.

More details about CMM1 can be found at:

<http://www.vm.ibm.com/sysman/vmrm/vmrmcmm.html>

#### Collaborative Memory Management Assist

Collaborative Memory Management Assist (CMMA) consists of changes in hardware, z/VM 5.3, and Linux. The hardware facility includes a new instruction (EXTRACT AND SET STORAGE ATTRIBUTES - ESSA), which can be simulated by z/VM on machines where the facility is not available (although, due to simulation overhead, it is generally not recommended to do so).

CMMA introduces new usage and state information for each memory page, thus enabling Linux and z/VM to employ more efficient memory management algorithms which do not conflict with one another.

More details about CMMA can be found in the manual *z/VM V5R3.0 CP Programming Services* (SC24-6084), at:

<http://publibz.boulder.ibm.com/epubs/pdf/hcse5b21.pdf>

In both CMM flavors, the benefit is proportional to the potential. On systems where the guests are poorly configured, or over-sized, you will see the most benefit. There was an early study done by a group in IBM, *z/VM Large Memory - Linux on System z*, available at:

<http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101151>

CMMA is mentioned for completeness, but this function may become deprecated.

## Configuring CMM1

CMM1 must be enabled on both the Linux and z/VM sides for it to function. To enable it for Linux, the `cmm` module must be loaded at boot time. See “Inserting the `cmm` and `vmcp` modules” on page 32.

To enable z/VM for CMM1, perform the following steps:

1. Log on to the user ID `VMRMSVM`
2. Create a new or edit the existing configuration file, `VMRM CONFIG A`. Add two lines as follows:

```
==> x vmrm config
ADMIN MSGUSER VMRMADMN
NOTIFY MEMORY LNX* S10* RH5*
```

This example will apply to z/VM user IDs whose names begin with `LINUX`, `S10` or `RH5`.

3. Edit the `PROFILE EXEC` on `AUTOLOG1` so that `VMRMSVM` is autologged at IPL time. Logon to `AUTOLOG1`.
4. Before pressing Enter at the VM READ prompt, type `acc (noprof` so that the `PROFILE EXEC` is not run. Then add

```
LOGON AUTOLOG1
z/VM Version 5 Release 4.0, Service Level 0801 (64-bit),
built on IBM Virtualization Technology
There is no logmsg data
FILES: NO RDR, NO PRT, NO PUN
LOGON AT 09:07:43 EDT SUNDAY 05/25/08
DMSIND2015W Unable to access the Y-disk. Filemode Y (19E) not accessed
==> acc (noprof
==> x profile exec
/*****/
/* Autolog1 Profile Exec */
/*****/
'cp xautolog tcpip' /* start up TCPIP */
'CP XAUTOLOG VMSERVS'
'CP XAUTOLOG VMSERVU'
'CP XAUTOLOG VMSERV'
'CP XAUTOLOG DTCVSW1'
'CP XAUTOLOG DTCVSW2'
'cp xautolog vmrmsvw' /* start VMRM service virtual machine */
'cp set pf12 ret' /* set the retrieve key */
...
```

5. Start the user ID for this IPL of z/VM:

```
==> xautolog vmrmsvm
Command accepted
AUTO LOGON ***          VMRMSVM  USERS = 29
Ready; T=0.01/0.01 14:08:09
HCPCLS6056I XAUTOLOG information for VMRMSVM: The IPL command is verified by the
IPL command processor.
```

### **How to observe that CMM1 is working**

The answer to the question: “How do I know CMM1 is working?” is not entirely obvious.

From the z/VM perspective you should see resident pages and total pages for the CMM1-managed virtual machines drop when the system comes under constraint. For Performance Toolkit, this would be seen in UPAGE report (FCX113).

The ballooning driver releases the pages it has pinned by using Diagnose x'10'. You can see the diagnose 10 counts in PRIVOP report (FCX104).

Some other items of interest may be users in page wait, see USTAT (FCX114).

From within Linux, you can check the size of the pool of pages put out of play using the following commands:

```
# cat /proc/sys/vm/cmm_pages
# cat /proc/sys/vm/cmm_timed_pages
```

## 1.9 Contents of tar file

This file `ro-root-09.tgz` is a compressed tar file that contains the following files and directories:

<code>README.TXT</code>	The README file
<code>sbin/</code>	The Linux code - see Appendix 1.10, "Linux code" on page 66. The subdirectory is named <code>sbin/</code> so you can untar it from <code>/usr/local/</code> and have the scripts in your default <code>PATH</code> .
<code>vm191/</code>	The z/VM code for the CMSCLONE 191 disk - see Appendix 1.11, "z/VM code" on page 76
<code>vm192/</code>	The z/VM code for the CMSCLONE 192 disk - see Appendix 1.11, "z/VM code" on page 76

## 1.10 Linux code

This section of the appendix lists code that will run on Linux and one modified configuration file.

<code>boot.findself</code>	Script to obtain IP info on first boot of a cloned system
<code>boot.rootfsck.diffs</code>	Differences to apply to <code>/etc/init.d/boot.rootfsck</code>
<code>fstab.ror</code>	Modified <code>/etc/fstab</code> file for read-only root systems
<code>mnt2rogl1d.sh</code>	Script to create a read-only root system from a read-write system

Most of the scripts have the following disclaimer. It is not repeated, but only shown once here:

```
# IBM DOES NOT WARRANT OR REPRESENT THAT THE CODE PROVIDED IS COMPLETE
# OR UP-TO-DATE.  IBM DOES NOT WARRANT, REPRESENT OR IMPLY RELIABILITY,
# SERVICEABILITY OR FUNCTION OF THE CODE.  IBM IS UNDER NO OBLIGATION TO
# UPDATE CONTENT NOR PROVIDE FURTHER SUPPORT.
# ALL CODE IS PROVIDED "AS IS," WITH NO WARRANTIES OR GUARANTEES WHATSOEVER.
# IBM EXPRESSLY DISCLAIMS TO THE FULLEST EXTENT PERMITTED BY LAW ALL EXPRESS,
# IMPLIED, STATUTORY AND OTHER WARRANTIES, GUARANTEES, OR REPRESENTATIONS,
# INCLUDING, WITHOUT LIMITATION, THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR
# A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF PROPRIETARY AND INTELLECTUAL
# PROPERTY RIGHTS.  YOU UNDERSTAND AND AGREE THAT YOU USE THESE MATERIALS,
# INFORMATION, PRODUCTS, SOFTWARE, PROGRAMS, AND SERVICES, AT YOUR OWN
# DISCRETION AND RISK AND THAT YOU WILL BE SOLELY RESPONSIBLE FOR ANY DAMAGES
# THAT MAY RESULT, INCLUDING LOSS OF DATA OR DAMAGE TO YOUR COMPUTER SYSTEM.
# IN NO EVENT WILL IBM BE LIABLE TO ANY PARTY FOR ANY DIRECT, INDIRECT,
# INCIDENTAL, SPECIAL, EXEMPLARY OR CONSEQUENTIAL DAMAGES OF ANY TYPE
# WHATSOEVER RELATED TO OR ARISING FROM USE OF THE CODE FOUND HEREIN, WITHOUT
# LIMITATION, ANY LOST PROFITS, BUSINESS INTERRUPTION, LOST SAVINGS, LOSS OF
# PROGRAMS OR OTHER DATA, EVEN IF IBM IS EXPRESSLY ADVISED OF THE POSSIBILITY
# OF SUCH DAMAGES.  THIS EXCLUSION AND WAIVER OF LIABILITY APPLIES TO ALL
# CAUSES OF ACTION, WHETHER BASED ON CONTRACT, WARRANTY, TORT OR ANY OTHER
# LEGAL THEORIES.
```

### 1.10.1 The `boot.findself` script

This script is invoked once the first time a clone Linux system is booted. It does not take any action on the user ID's `RH5RWMNT`, `RH5RWTST` or `RH5ROGLD`.

```
#!/bin/bash
#
# /etc/init.d/boot.findself
#
# chkconfig: 12345 01 99
### BEGIN INIT INFO
# Provides:          boot.findself
# Description:       upon first boot find/modify IP@ + hostname, gen SSH keys
### END INIT INFO
#
# This script requires two RHEL 5 parameter files to exist on the user ID's
# 191 disk: (1) the file RH5RWMNT PARM-RH5 and (2) $userid PARM-RH5 where
# $userid is the ID of the user that is running the script. It then modifies
# the IP address, Host name and fully qualified domain name in three
# configuration files that contain this info. It also regenerates SSH keys.
# The script then turns itself off via "chkconfig" so it only runs once.
#
#+-----+
function findID()
# Get my VM user ID - don't find self on RH5RWMNT, RH5RWTST or RH5RWGLD
```

```

#+-----+
{
targetID=$(cat /proc/sysinfo | grep "VM00 Name" | awk '{print $3}')
if [ $targetID = "RH5RWMNT" ] || [ $targetID = "RH5RWTST" ] || \
  [ $targetID = "RH5ROGLD" ]; then # don't do anything on these three IDs
  exit
fi
}

#+-----+
function enableAdisk()
# Enable my 191 (A) disk
#+-----+
{
chccwdev -e 191 > /dev/null 2>&1
rc=$?
if [ $rc != 0 ]; then # unable to enable 191 disk
  echo "$0: Unable to enable 191, rc from chccwdev = $rc"
  exit 1
fi
sleep 1# wait a sec to be sure disk is ready
Adisk=/dev/$(egrep '^0.0.0191' /proc/dasd/devices | awk '{print $7}')
}

#+-----+
function findSourceIP()
# Get the source IP address and hostName
#+-----+
{
sourceConf="$sourceID.$confType"
cmsfs1st -d $Adisk | grep $sourceID | grep $confType > /dev/null
rc=$?
if [ $rc != 0 ]; then
  echo "$0: $sourceConf not found on 191 minidisk. Exiting"
  exit 2
fi
export local $(cmsfscat -a -d $Adisk $sourceConf)
# set global variable names escaping any dots (.) in the strings
sourceName=$(echo "$HOSTNAME" | sed -e 's:\.\:\.\.:g')
sourceHost=${HOSTNAME%%.*} # Chop domain name off to leave host name
sourceIP=$(echo "$IPADDR" | sed -e 's:\.\:\.\.:g')
}

#+-----+
function findTargetIP()
# Get my new IP address and hostname
#+-----+
{
targetParm="$targetID.$confType"
cmsfs1st -d $Adisk | grep $targetID | grep $confType > /dev/null
rc=$?
if [ $rc != 0 ]; then
  echo "$0: $targetParm not found on 191 minidisk. Exiting"
  exit 3
fi
export local $(cmsfscat -a -d $Adisk $targetParm)
targetName=$HOSTNAME
targetHost=${HOSTNAME%%.*} # Chop domain name off to leave host name
targetIP=$IPADDR
}

```

```

#+-----+
function modifyIP()
# Modify IP address and host name in /etc/hosts, /etc/sysconfig/network and
# /etc/sysconfig/network-scripts/ifcfg-eth0
#+-----+
{
# TODO: this function should also modify, DNS, Gateway, broadcast, etc.
eth0file="/etc/sysconfig/network-scripts/ifcfg-eth0"
echo ""
IPLine=`grep IPADDR $eth0file`
currentIP=${IPLine##IPADDR=} # chop "IPADDR=" off to leave IP address
if [ "$currentIP" != "$targetIP" ]; then # IP address has not been set
echo "$0: changing (escaped) $sourceName to $targetName in /etc/hosts"
sed --in-place -e "s/$sourceName/$targetName/g" /etc/hosts
echo "$0: changing $sourceHost to $targetHost and IP address in /etc/hosts"
sed --in-place -e "s/$sourceHost/$targetHost/g" \
-e "s/$sourceIP/$targetIP/g" /etc/hosts
sed --in-place -e "s/$sourceHost/$targetHost/g" /etc/sysconfig/network
echo "$0: changing (escaped) $sourceIP to $targetIP in $eth0file"
sed --in-place -e "s/$sourceIP/$targetIP/g" $eth0file
fi
# Need to always set hostname because golden image host name appears to be
# getting set before modified /etc/ is being bind mounted (how to fix?)
echo "$0: setting host name to $targetHost"
hostname $targetHost
}

#+-----+
function genSSHkeys()
# Regenerate a set of SSH keys
#+-----+
{
rm /etc/ssh/ssh_host_*
ssh-keygen -t rsa -N "" -q -f /etc/ssh/ssh_host_rsa_key
ssh-keygen -t dsa -N "" -q -f /etc/ssh/ssh_host_dsa_key
ssh-keygen -t rsa1 -N "" -q -f /etc/ssh/ssh_host_key
}

# main()
# global variables
sourceID="RH5RWMNT" # VM user ID where first Linux was installed
confType="CONF-RH5" # File type of parameter file on 191 disk

# function calls
findID
enableAdisk
findSourceIP
findTargetIP
modifyIP
genSSHkeys
# Need to always set hostname because golden image host name appears to be
# getting set before modified /etc/ is being bind mounted (how to fix?)
# So - cannot chkconfig this script off
# chkconfig --del boot.findself # run only once => turn self off

```

## 1.10.2 The cloneprep.sh script

This script prepares the Linux system on RH5RWMNT to be cloned.

```

#!/bin/bash
# ... disclaimer ...
# Script to clean up files before cloning
#+-----+
function cleanFile()
# delete file, create empty file and set permission mode
# arg 1: file to delete and create
# arg 2: mode to set empty file to
#+-----+
{
    if [ -f $1 ]; then
        rm $1
    fi
    touch $1
    chmod $2 $1
}

# main()
# clean up certain files in /var/log
rm /var/log/*.gz
cleanFile /var/log/authlog 600
cleanFile /var/log/faillog 600
cleanFile /var/log/lastlog 644
cleanFile /var/log/secure 600
cleanFile /var/log/secure 600
cleanFile /root/.bash_history 600

echo "System should be ready for shutdown and cloning"

```

### 1.10.3 The mnt2roglid.sh script

This script clones from RH5RWMNT to RH5ROGLD.

```

#!/bin/sh
# mnt2roglid.sh - script to create a read-only root system on target user ID
# Hard-coded virtual device addresses:
# 1b0 - /boot
# 1b1 - /
# 1b2 - swap (VDISK)
# 1b3 - swap (VDISK)
# 1b4 - swap
# 1b5 - /local
# 1b6 - /var/lib/rpm
#
# Source disks are linked as 1xxx
# Target disks are linked as 2xxx
#
# ... disclaimer ...
#+-----+
function CPcmd()
# Run 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 $@ 2>&1)
    if [ "$output" != "X" ]; then # there is some output - echo it
        echo "$output"
    fi
}

```

```

fi
retVal=0
retVal=$(echo $output | grep "Error: non-zero CP" | awk -F# '{print $2}')
return $retVal
}

#+-----+
function checkID()
# Arg 1: User ID to verify
# Verify user ID exists and is logged off
#+-----+
{
  userID=$1
  CPcmd QUERY $userID
  rc=$?
  case $rc in
    0) # user ID is logged on or disconnected
      echo "Error: $userID user ID must be logged off"
      exit 2
      ;;
    3) # user ID does not exist
      echo "Error: $ID user ID does not exist"
      exit 3
      ;;
    45) # user ID is logged off - this is correct - fall through
      ;;
    *) # unexpected
      echo "Unexpected rc from QUERY: $rc"
      echo "$targetID user ID must exist and be logged off"
      exit 4
  esac
}

#+-----+
function linkSourceDisks()
# Link source disks as 1xxx
#+-----+
{
  echo ""
  echo "Linking source disks ..."
  CPcmd link $sourceID 1b1 11b1 rr
  if [ $? != 0 ]; then exit 5; fi
  CPcmd link $sourceID 1b0 11b0 rr
  if [ $? != 0 ]; then exit 6; fi
  CPcmd link $sourceID 1b4 11b4 rr
  if [ $? != 0 ]; then exit 7; fi
  CPcmd link $sourceID 1b5 11b5 rr
  if [ $? != 0 ]; then exit 8; fi
  CPcmd link $sourceID 1b6 11b6 rr
  if [ $? != 0 ]; then exit 9; fi
}

#+-----+
function linkTargetDisks()
# Link the target disks as 2xxx
#+-----+
{
  echo ""
  echo "Linking target disks ..."
  CPcmd link $targetID 1b1 21b1 mr
}

```

```

if [ $? != 0 ]; then exit 10; fi
CPcmd link $targetID 1b0 21b0 mr
if [ $? != 0 ]; then exit 11; fi
CPcmd link $targetID 1b4 21b4 mr
if [ $? != 0 ]; then exit 12; fi
CPcmd link $targetID 1b5 21b5 mr
if [ $? != 0 ]; then exit 13; fi
CPcmd link $targetID 1b6 21b6 mr
if [ $? != 0 ]; then exit 14; fi
}

#+-----+
function enableSourceDisks()
# Enable the source disks
#+-----+
{
echo ""
echo "Enabling source disks ..."
chccwdev -e 11b1
if [ $? != 0 ]; then exit 15; fi
chccwdev -e 11b0
if [ $? != 0 ]; then exit 16; fi
chccwdev -e 11b4
if [ $? != 0 ]; then exit 17; fi
chccwdev -e 11b5
if [ $? != 0 ]; then exit 18; fi
chccwdev -e 11b6
if [ $? != 0 ]; then exit 19; fi
udevsettle
}

#+-----+
function enableTargetDisks()
# Enable the target disks
#+-----+
{
echo ""
echo "Enabling target disks ..."
chccwdev -e 21b1
if [ $? != 0 ]; then exit 20; fi
chccwdev -e 21b0
if [ $? != 0 ]; then exit 21; fi
chccwdev -e 21b4
if [ $? != 0 ]; then exit 22; fi
chccwdev -e 21b5
if [ $? != 0 ]; then exit 23; fi
chccwdev -e 21b6
if [ $? != 0 ]; then exit 24; fi
udevsettle
}

#+-----+
function copyDisk()
# Copy a disk via dasdfmt then dd
# Arg 1: Source vaddr
# Arg 2: Target vaddr
#+-----+
{
source=$1
target=$2

```

```

echo ""
echo "copying $source to $target ..."
sDev=/dev/$(egrep ^0.0.$source /proc/dasd/devices | awk '{ print $7 }')
if [ "$?" != 0 ]; then exit 25; fi
tDev=/dev/$(egrep ^0.0.$target /proc/dasd/devices | awk '{ print $7 }')
if [ "$?" != 0 ]; then exit 26; fi
echo ""
echo "dasdfmt-ing $tDev ..."
dasdfmt -d cdl -y -b 4096 -p -F -f $tDev
if [ "$?" != 0 ]; then exit 27; fi
echo ""
echo "dd-ing $sDev to $tDev ..."
dd bs=4096 if=$sDev of=$tDev
if [ "$?" != 0 ]; then exit 28; fi
echo ""
echo "disabling and re-enabling $target ..."
blockdev --rereadpt $tDev
if [ $? != 0 ]; then exit 29; fi
sync
if [ $? != 0 ]; then exit 30; fi
udevsettle
if [ $? != 0 ]; then exit 31; fi
}

#+-----+
function copySystem()
# Copy /mnt/source to /mnt/target
#+-----+
{
echo ""
echo "Copying root file system ..."
copyDisk 11b1 21b1

echo ""
echo "Copying /boot/ ..."
copyDisk 11b0 21b0

echo ""
echo "Copying minidisk swap space ..."
copyDisk 11b4 21b4

echo ""
echo "Copying /local/ ..."
copyDisk 11b5 21b5

echo ""
echo "Copying /var/lib/rpm/ ..."
copyDisk 11b6 21b6
}

#+-----+
function mountSourceRoot()
# Mount disk at 11b1 over /mnt/source
#+-----+
{
echo ""
echo "Making source mount point ..."
if [ ! -d /mnt/source ]; then
mkdir /mnt/source
if [ "$?" != 0 ]; then exit 32; fi

```

```

fi
echo ""
echo "Mounting source root file system over /mnt/source ..."
dev11b1=/dev/$(egrep '^0.0.11b1' /proc/dasd/devices | awk '{ print $7 }')1
mount -o ro $dev11b1 /mnt/source
}

#+-----+
function mountTargetDisks()
# Mount disk at 21b1 over /mnt/target, then make mount points.
# Then mount disks at 21b0 over /boot and 21b5 over /local
#+-----+
{
dev21b0=/dev/$(egrep '^0.0.21b0' /proc/dasd/devices | awk '{ print $7 }')
dev21b1=/dev/$(egrep '^0.0.21b1' /proc/dasd/devices | awk '{ print $7 }')
dev21b5=/dev/$(egrep '^0.0.21b5' /proc/dasd/devices | awk '{ print $7 }')
dev21b6=/dev/$(egrep '^0.0.21b6' /proc/dasd/devices | awk '{ print $7 }')

echo ""
echo "Mounting target file systems over /mnt/target ..."
mkdir /mnt/target
mount "$dev21b1"1 /mnt/target
if [ "$?" != 0 ]; then exit 33; fi
echo ""
echo "Making target mount points ..."
mkdir -p /mnt/target/{boot,usr,var,opt,local,sys,proc}
mount "$dev21b0"1 /mnt/target/boot
if [ "$?" != 0 ]; then exit 34; fi
mount "$dev21b5"1 /mnt/target/local
if [ "$?" != 0 ]; then exit 35; fi
# label disk
e2label "$dev21b5"1 stateless-state
echo "Mounting memory file systems ..."
mount -t sysfs sysfs /mnt/target/sys
if [ "$?" != 0 ]; then exit 36; fi
mount -t proc proc /mnt/target/proc
if [ "$?" != 0 ]; then exit 37; fi
}

#+-----+
function modifySystem()
# 1) Modify /etc/modprobe.conf and run mkinitrd on target system
# 2) Copy source r/w directories to target /local/
# 3) Configure target system for read-only /
#+-----+
{
TGT="/mnt/target"
echo ""
echo "Backing up and modifying /etc/modprobe.conf file ..."
cp $TGT/etc/modprobe.conf $TGT/etc/modprobe.conf.orig
if [ "$?" != 0 ]; then exit 38; fi
cat $TGT/etc/modprobe.conf.orig | \
sed -e 's/1b0-1bf/1b0(ro),1b1(ro),1b2-1b5,1b6(ro),1b7-1bf/g' > \
$TGT/etc/modprobe.conf

echo ""
echo "Backing up and modifying /etc/fstab file ..."
cp $TGT/etc/fstab $TGT/etc/fstab.orig
if [ "$?" != 0 ]; then exit 40; fi
}

```

```

    echo "tmpfs                /tmp                tmpfs defaults    0 0" >>
$TGT/etc/fstab

echo ""
echo "Backing up and modifying /etc/zipl.conf file ..."
cp $TGT/etc/zipl.conf $TGT/etc/zipl.conf.orig
if [ "$?" != 0 ]; then exit 42; fi
cat $TGT/etc/zipl.conf.orig | \
    sed -e 's/vmpoff=LOGOFF/vmpoff=LOGOFF readonlyroot/g' > $TGT/etc/zipl.conf
echo "Running zipl in target environment ..."
chroot $TGT zipl
if [ "$?" != 0 ]; then exit 43; fi

echo ""
echo "Copying source /local/etc,root,srv,var/ to target /local/ ..."
cp -a $TGT/etc $TGT/local 2>/dev/null
if [ "$?" != 0 ]; then exit 44; fi
cp -a $TGT/root $TGT/local 2>/dev/null
if [ "$?" != 0 ]; then exit 45; fi
cp -a $TGT/srv $TGT/local 2>/dev/null
if [ "$?" != 0 ]; then exit 46; fi
cp -a $TGT/var $TGT/local 2>/dev/null
if [ "$?" != 0 ]; then exit 47; fi
echo /etc > $TGT/local/files
echo /var >> $TGT/local/files
echo /srv >> $TGT/local/files
echo /root >> $TGT/local/files

echo ""
echo "Backing up and modifying /etc/sysconfig/readonly-root file ..."
cp $TGT/etc/sysconfig/readonly-root $TGT/etc/sysconfig/readonly-root.orig
sed -i 's/READONLY=no/READONLY=yes/' $TGT/etc/sysconfig/readonly-root
sed -i 's/STATE_MOUNT=\/.snapshot/STATE_MOUNT=\/local/'
$TGT/etc/sysconfig/readonly-root
}

#+-----+
function cleanUp()
# Unmount source and target file systems and detach minidisks
#+-----+
{
    echo ""
    echo "Cleaning up ..."
    # clean up target disks
    umount /mnt/target/local
    umount /mnt/target/boot
    umount /mnt/target/proc
    umount /mnt/target/sys
    umount /mnt/target
    chccwdev -d 21b0
    chccwdev -d 21b1
    chccwdev -d 21b4
    chccwdev -d 21b5
    chccwdev -d 21b6
    vmcp det 21b0
    vmcp det 21b1
    vmcp det 21b4
    vmcp det 21b5
    vmcp det 21b6
    # clean up source disks
}

```

```
umount /mnt/source
chccwdev -d 11b0
chccwdev -d 11b1
chccwdev -d 11b4
chccwdev -d 11b5
chccwdev -d 11b6
vmcp det 11b0
vmcp det 11b1
vmcp det 11b4
vmcp det 11b5
vmcp det 11b6
}

# main()
# global variables
sourceID="RH5RWMNT"
targetID="RH5ROGLD"
fstabFile="fstab.RH5"

# function calls
checkID $sourceID
checkID $targetID
linkSourceDisks
linkTargetDisks
enableSourceDisks
enableTargetDisks
copySystem
mountSourceRoot
mountTargetDisks
modifySystem
cleanUp
echo "Exiting with rc 0!"
exit 0
```

## 1.11 z/VM code

This section contains EXECs that run on z/VM.

COPYMDSK.EXEC	EXEC to copy a minidisk (used by clone EXECs)
CLONERO.EXEC	EXEC to clone from RH5GOLD read-only disks to target Linux
CLONERW.EXEC	EXEC to clone from RH5GOLD read-write disks to target Linux
MNT2GOLD.EXEC	EXEC to clone from RH5RWMNT to RH5GOLD read-write disks
MNT2TST.EXEC	EXEC to clone from to RH5RWMNT to RH5RWTST
PROFILE.EXEC	EXEC to IPL to create VDISK swap spaces and IPL from 1B0
RO2GOLD.EXEC	EXEC to clone from RH5ROGLD to RH5GOLD read-only disks
TST2MNT.EXEC	EXEC to clone from to RH5RWTST to RH5RWMNT
SAMPLE.PARM-RH5	Sample SLES 10 SP2 parameter file
RHEL53.EXEC	EXEC to kick off a RHEL 5.3 installation

Most of the EXECs start with the following disclaimer. It is only shown once here:

```
/*-----*/
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
-----*/
```

### 1.11.1 COPYMDSK EXEC

The **COPYMDSK EXEC** tries to copy a file with **FLASHCOPY**. If that fails, it falls back to **DDR**.

```
/*-----*/
/* COPYMDSK EXEC - copy minidisk w/FLASHCOPY, if it fails, try DDR */
/* Parm 1: vaddr of source minidisk */
/* Parm 2: vaddr of target minidisk */
Address 'COMMAND'
Parse Arg source target .
Say
Say 'Copying minidisk' source 'to' target '...'
'CP FLASHCOPY' source '0 END' target '0 END'
If (rc \= 0) Then Do /* Fallback to DDR */
  Say 'FLASHCOPY failed, falling back to DDR ...'
  /* Queue up DDR commands */
  Queue 'SYSPRINT CONS' /* Don't print to file */
  Queue 'PROMPTS OFF' /* Don't ask 'Are you sure?' */
  Queue 'IN' source '3390' /* Input minidisk */
  Queue 'OUT' target '3390' /* Output minidisk */
  Queue 'COPY ALL' /* Copy all contents */
  Queue ' ' /* Empty record ends DDR */
  'DDR'
  retVal = rc
End
Else retVal = rc
Say 'Return value =' retVal
Return retVal
```

## 1.11.2 CLONERO EXEC

The **CLONERO EXEC** clones from the RH5GOLD read-only 11Bx disks to a target Linux user ID:

```
/* Clone read-only Linux from gold ID to target ID */
Address 'COMMAND'
source = 'RH5GOLD'
Parse Upper Arg target .
If (target = '') Then
Do
    Say 'Target user ID is a required parameter'
    Say
    Say 'Syntax: CLONERO targetID'
    Exit 1
End

rw_addr = '01B4 01B5'
ro_addr = '21B4 21B5'

Say 'Checking that source and target user IDs are logged off'
Call CheckLoggedOff source
Call CheckLoggedOff target

Say 'Do you want to copy R/O system from' source 'to' target 'y/n'
Parse Upper Pull answer .
If (answer \= 'Y') Then Exit 1

/* link target disks R/W */
Do a = 1 to Words(rw_addr)
    addr = Word(rw_addr,a)
    'CP LINK' target addr addr 'MR'
    If (rc \= 0) Then Call CleanUp 100+a
End

/* link source disks R/O */
Do a = 1 to Words(ro_addr)
    addr = Word(ro_addr,a)
    'CP LINK' source addr addr 'RR'
    If (rc \= 0) Then Call CleanUp 200+a
End

/* copy disks */
Do a = 1 to Words(ro_addr)
    source_addr = Word(ro_addr,a)
    target_addr = Word(rw_addr,a)
    'EXEC COPYMSK' source_addr target_addr
    If (rc \= 0) Then Call CleanUp 300+a
End

/* cleanup */
Call CleanUp 0

/*+-----+*/
CheckLoggedOff: Procedure
/*| Verify that a user ID is logged off |*/
/*| parm 1: User ID to check |*/
/*+-----+*/
Parse arg virt_machine .
'CP QUERY' virt_machine
Select
```

```

When (rc = 0) Then Do /* user ID is logged on or disconnected */
  Say "Error:" virt_machine "user ID must be logged off"
  Exit 2
End
When (rc = 3) Then Do /* user ID does not exist */
  Say "Error:" virt_machine "user ID does not exist"
  Exit 3
End
When (rc = 45) Then /* user ID is logged off - this is correct */
  Return 0
Otherwise Do /* unexpected */
  Say "Error:" virt_machine "user ID must exist and be logged off"
  Exit 4
End
End

/*+-----+*/
CleanUp: Procedure expose ro_addrs rw_addrs
/*| Detach all disks before exiting |*/
/*| parm 1: Exit code |*/
/*+-----+*/
Parse Arg retVal
Say
Say 'Cleaning up ...'
CP DETACH ro_addrs rw_addrs
Exit retVal

```

### 1.11.3 CLONERW EXEC

The **CLONERW EXEC** clones from the RH5GOLD read-write 21Bx disks to a target Linux user ID:

```

/* Clone read-write Linux from gold ID to target ID */
Address 'COMMAND'
source = 'RH5GOLD'
Parse Upper Arg target .
If (target = '') Then Do
  Say 'Target user ID is a required parameter'
  Say
  Say 'Syntax: CLONERW targetID'
  Exit 1
End

rw_addrs = '01B0 01B1 01B4 01B5 01B6'
ro_addrs = '11B0 11B1 11B4 11B5 11B6'

Say 'Checking that source and target user IDs are logged off'
Call CheckLoggedOff source
Call CheckLoggedOff target

Say 'Do you want to copy R/W system from' source 'to' target||'? y/n'
Parse Upper Pull answer .
If (answer \= 'Y') Then Exit 1

/* link target disks R/W */
Do a = 1 to Words(rw_addrs)
  addr = Word(rw_addrs,a)
  'CP LINK' target addr addr 'MR'
  If (rc \= 0) Then Call CleanUp 100+a
End

```

```

/* link source disks R/O */
Do a = 1 to Words(ro_addrs)
  addr = Word(ro_addrs,a)
  'CP LINK' source addr addr 'RR'
  If (rc \= 0) Then Call CleanUp 200+a
End

/* copy disks */
Do a = 1 to Words(ro_addrs)
  source_addr = Word(ro_addrs,a)
  target_addr = Word(rw_addrs,a)
  'EXEC COPYMDSK' source_addr target_addr
  If (rc \= 0) Then Call CleanUp 300+a
End

/* cleanup */
Call CleanUp 0

/*+-----+*/
CheckLoggedOff: Procedure
/*| Verify that a user ID is logged off |*/
/*| parm 1: User ID to check |*/
/*+-----+*/
Parse arg virt_machine .
'CP QUERY' virt_machine
Select
  When (rc = 0) Then Do /* user ID is logged on or disconnected */
    Say "Error:" virt_machine "user ID must be logged off"
    Exit 2
  End
  When (rc = 3) Then Do /* user ID does not exist */
    Say "Error:" virt_machine "user ID does not exist"
    Exit 3
  End
  When (rc = 45) Then /* user ID is logged off - this is correct */
    Return 0
  Otherwise Do /* unexpected */
    Say "Error:" virt_machine "user ID must exist and be logged off"
    Exit 4
  End
End

/*+-----+*/
CleanUp: Procedure Expose ro_addrs rw_addrs
/*| Detach all disks before exiting |*/
/*| parm 1: Exit code |*/
/*+-----+*/
Parse Arg retVal
Say
Say 'Cleaning up ...'
'CP DETACH' ro_addrs rw_addrs
Exit retVal

```

## 1.11.4 MNT2GOLD EXEC

The **MNT2GOLD EXEC** copies the RH5RWMNT 1Bx read-write disks to the RH5GOLD read-write 21Bx disks:

```

/* Copy Linux from maintenance ID to gold ID's read-write disks */
Address 'COMMAND'

```

```

source = 'RH5RWMNT'
target = 'RH5GOLD'

ro_addrs = '01B0 01B1 01B4 01B5 01B6'
rw_addrs = '11B0 11B1 11B4 11B5 11B6'

Say 'Checking that source and target user IDs are logged off'
Call CheckLoggedOff source
Call CheckLoggedOff target

Say 'Do you want to copy R/W disks from' source 'to' target||'? y/n'
Parse Upper Pull answer .
If (answer \= 'Y') Then Exit 1

/* link target disks R/W */
Do a = 1 to Words(rw_addrs)
  addr = Word(rw_addrs,a)
  'CP LINK' target addr addr 'MR'
  If (rc \= 0) Then Call CleanUp 100+a
End

/* link source disks R/O */
Do a = 1 to Words(ro_addrs)
  addr = Word(ro_addrs,a)
  'CP LINK' source addr addr 'RR'
  If (rc \= 0) Then Call CleanUp 200+a
End

/* copy disks */
Do a = 1 to Words(ro_addrs)
  source_addr = Word(ro_addrs,a)
  target_addr = Word(rw_addrs,a)
  'EXEC COPYMSK' source_addr target_addr
  If (rc \= 0) Then Call CleanUp 300+a
End

/* CleanUp */
Call CleanUp 0

/*+-----+*/
CheckLoggedOff: Procedure
/*| Verify that a user ID is logged off |*/
/*| parm 1: User ID to check |*/
/*+-----+*/
Parse arg virt_machine .
'CP QUERY' virt_machine
Select
  When (rc = 0) Then Do /* user ID is logged on or disconnected */
    Say "Error:" virt_machine "user ID must be logged off"
    Exit 2
  End
  When (rc = 3) Then Do /* user ID does not exist */
    Say "Error:" virt_machine "user ID does not exist"
    Exit 3
  End
  When (rc = 45) Then /* user ID is logged off - this is correct */
    Return 0
  Otherwise Do /* unexpected */
    Say "Error:" virt_machine "user ID must exist and be logged off"
    Exit 4
End

```

```

        End
    End

/*+-----+*/
CleanUp: Procedure Expose ro_addrs rw_addrs
/*| Detach all disks before exiting |*/
/*| parm 1: Exit code |*/
/*+-----+*/
Parse Arg retVal
Say
Say 'Cleaning up ...'
Call Diag 8,'DETACH' ro_addrs rw_addrs
Exit retVal

```

### 1.11.5 MNT2TST EXEC

The **MNT2TST EXEC** copies the RH5RWMNT read-write disks to the RH5RWTST read-write disks:

```

/* Copy Linux from maintenance ID to test ID */
Address 'COMMAND'
source = 'RH5RWMNT'
target = 'RH5RWTST'

ro_addrs = '01B0 01B1 01B4 01B5 01B6'
rw_addrs = '11B0 11B1 11B4 11B5 11B6'

Say 'Checking that source and target user IDs are logged off'
Call CheckLoggedOff source
Call CheckLoggedOff target

Say 'Do you want to copy R/W disks from' source 'to' target'? y/n'
Parse Upper Pull answer .
If (answer \= 'Y') Then Exit 1

/* link target disks R/W */
Do a = 1 to Words(rw_addrs)
    addr = Word(rw_addrs,a)
    'CP LINK' target Right(addr,3) addr 'MR'
    If (rc \= 0) Then Call CleanUp 100+a
End

/* link source disks R/O */
Do a = 1 to Words(ro_addrs)
    addr = Word(ro_addrs,a)
    'CP LINK' source addr addr 'RR'
    If (rc \= 0) Then Call CleanUp 200+a
End

/* copy disks */
Do a = 1 to Words(ro_addrs)
    source_addr = Word(ro_addrs,a)
    target_addr = Word(rw_addrs,a)
    'EXEC COPYMDSK' source_addr target_addr
    If (rc \= 0) Then Call CleanUp 300+a
End

/* cleanup */
Call CleanUp 0

/*+-----+*/

```

```

CheckLoggedOff: Procedure
/*| Verify that a user ID is logged off |*/
/*| parm 1: User ID to check |*/
/*+-----+*/
Parse arg virt_machine .
'CP QUERY' virt_machine
Select
  When (rc = 0) Then Do /* user ID is logged on or disconnected */
    Say "Error:" virt_machine "user ID must be logged off"
    Exit 2
    End
  When (rc = 3) Then Do /* user ID does not exist */
    Say "Error:" virt_machine "user ID does not exist"
    Exit 3
    End
  When (rc = 45) Then /* user ID is logged off - this is correct */
    Return 0
  Otherwise Do /* unexpected */
    Say "Error:" virt_machine "user ID must exist and be logged off"
    Exit 4
    End
End

/*+-----+*/
CleanUp: Procedure Expose ro_addr rw_addr
/*| Detach all disks before exiting |*/
/*| parm 1: Exit code |*/
/*+-----+*/
Parse Arg retVal
Say
Say 'Cleaning up ...'
'CP DETACH' ro_addr rw_addr
Exit retVal

```

## 1.11.6 PROFILE EXEC

The **PROFILE EXEC** is a sample logon profile that will be on the Linux user ID's 191 disk.

```

/* PROFILE EXEC for Linux virtual servers */
Address 'COMMAND'
'CP SET RUN ON'
'CP SET PF11 RETRIEVE FORWARD'
'CP SET PF12 RETRIEVE'
'ACCESS 592 C'
'EXEC SWAPGEN 1B2 131072' /* create a 64M VDISK disk swap space */
'EXEC SWAPGEN 1B3 262144' /* create a 128M VDISK disk swap space */
'PIPE CP QUERY' Userid() | VAR USER'
Parse Value user With id . dsc .
iplDisk = 1B0 /* /boot/ is 1B0 */
If (dsc = 'DSC') Then /* user is disconnected */
  'CP IPL' iplDisk
Else Do /* user is interactive -> prompt */
  Say 'Do you want to IPL Linux from DASD' iplDisk'? y/n'
  Parse Upper Pull answer .
  If (answer = 'Y') Then
    'CP IPL' iplDisk
  End
Exit

```

## 1.11.7 RO2GOLD EXEC

The RO2GOLD EXEC copies the disks that will become the read-only root system from RH5ROGLD to the 21Bx disks of RH5GOLD:

```
/* Copy Linux from read-only ID to Gold ID */
'pipe cp link RH5GOLD2 21b7 21b7 rr'
'pipe cp 10000 q links 21b7 '|,
    'split /,/|',
    'strip|',
    'strip /,/|',
    'strip|',
    'count lines |',
    'var howmany |',
    'console|',
    'stem names.'
'cp det 21b7'
If (howmany > 1 ) Then Do
    Say 'WARNING WARNING WARNING WARNING!'
    Say 'You have Linux guests currently reading the RH5GOLD2'
    Say 'disks. Running this script would cause problems!'
    Say 'WARNING WARNING WARNING WARNING!'
    Exit 1
End

Address 'COMMAND'
source = 'RH5ROGLD'
target = 'RH5GOLD'

ro_addr = '01B0 01B1 01B4 01B5 01B6'
rw_addr = '21B0 21B1 21B4 21B5 21B6'

Say 'Checking that source and target user IDs are logged off'
Call CheckLoggedOff source
Call CheckLoggedOff target

Say 'Do you want to copy disks from' source 'to' target||'? y/n'
Parse Upper Pull answer .
If (answer \= 'Y') Then Exit 1

/* link target disks R/W */
Do a = 1 to Words(rw_addr)
    addr = Word(rw_addr,a)
    'CP LINK' target addr addr 'MR'
    If (rc \= 0) Then Call CleanUp 100+a
End

/* link source disks R/O */
Do a = 1 to Words(ro_addr)
    addr = Word(ro_addr,a)
    'CP LINK' source addr addr 'RR'
    If (rc \= 0) Then Call CleanUp 200+a
End

/* copy disks */
Do a = 1 to Words(ro_addr)
    source_addr = Word(ro_addr,a)
    target_addr = Word(rw_addr,a)
    'EXEC COPYMSK' source_addr target_addr
    If (rc \= 0) Then Call CleanUp 300+a
```

```

End

/* cleanup */
Call CleanUp 0

/*+-----+*/
CheckLoggedOff: Procedure
/*| Verify that a user ID is logged off |*/
/*| parm 1: User ID to check |*/
/*+-----+*/
Parse arg virt_machine .
'CP QUERY' virt_machine
Select
  When (rc = 0) Then Do /* user ID is logged on or disconnected */
    Say "Error:" virt_machine "user ID must be logged off"
    Exit 2
  End
  When (rc = 3) Then Do /* user ID does not exist */
    Say "Error:" virt_machine "user ID does not exist"
    Exit 3
  End
  When (rc = 45) Then /* user ID is logged off - this is correct */
    Return 0
  Otherwise Do /* unexpected */
    Say "Error:" virt_machine "user ID must exist and be logged off"
    Exit 4
  End
End

/*+-----+*/
CleanUp: Procedure Expose ro_addrs rw_addrs
/*| Detach all disks before exiting |*/
/*| parm 1: Exit code |*/
/*+-----+*/
Parse Arg retVal
Say
Say 'Cleaning up ...'
'CP DETACH' ro_addrs rw_addrs
Exit retVal

```

### 1.11.8 TST2MNT EXEC

The **TST2MNT EXEC** copies the read-write disks from RH5RWTST to RH5RWMNT:

```

/* Copy Linux from maintenance ID to test ID */
Address 'COMMAND'
source = 'RH5RWTST'
target = 'RH5RWMNT'

ro_addrs = '01B0 01B1 01B4 01B5 01B6'
rw_addrs = '11B0 11B1 11B4 11B5 11B6'

Say 'Checking that source and target user IDs are logged off'
Call CheckLoggedOff source
Call CheckLoggedOff target

Say ' '
Say 'Do you want to copy R/W disks from' source 'to' target'? y/n'
Parse Upper Pull answer .
If (answer \= 'Y') Then Exit 1

```

```

/* link target disks R/W */
Do a = 1 to Words(rw_addrs)
  addr = Word(rw_addrs,a)
  'CP LINK' target Right(addr,3) addr 'MR'
  If (rc \= 0) Then Call CleanUp 100+a
End

/* link source disks R/O */
Do a = 1 to Words(ro_addrs)
  addr = Word(ro_addrs,a)
  'CP LINK' source addr addr 'RR'
  If (rc \= 0) Then Call CleanUp 200+a
End

/* copy disks */
Do a = 1 to Words(ro_addrs)
  source_addr = Word(ro_addrs,a)
  target_addr = Word(rw_addrs,a)
  'EXEC COPYMDSK' source_addr target_addr
  If (rc \= 0) Then Call CleanUp 300+a
End

/* CleanUp */
Call CleanUp 0

/*+-----+*/
CheckLoggedOff: Procedure
/*| Verify that a user ID is logged off |*/
/*| parm 1: User ID to check |*/
/*+-----+*/
Parse arg virt_machine .
'CP QUERY' virt_machine
Select
  When (rc = 0) Then Do /* user ID is logged on or disconnected */
    Say "Error:" virt_machine "user ID must be logged off"
    Exit 2
  End
  When (rc = 3) Then Do /* user ID does not exist */
    Say "Error:" virt_machine "user ID does not exist"
    Exit 3
  End
  When (rc = 45) Then /* user ID is logged off - this is correct */
    Return 0
  Otherwise Do /* unexpected */
    Say "Error:" virt_machine "user ID must exist and be logged off"
    Exit 4
  End
End

/*+-----+*/
CleanUp: Procedure Expose ro_addrs rw_addrs
/*| Detach all disks before exiting |*/
/*| parm 1: Exit code |*/
/*+-----+*/
Parse Arg retVal
Say
Say 'Cleaning up ...'
'CP DETACH' ro_addrs rw_addrs
Exit retVal

```

### 1.11.9 SAMPLE configuration and parameter files

Following is the sample configuration file for a RHEL 5.3 install:

```
ramdisk_size=40000 root=/dev/ram0 ro ip=off
CMSDASD=191 CMSCONFFILE=SAMPLE.CONF-RH5
vnc vncpassword=1nx4vm
```

Following is the sample parameter file for a RHEL 5.3 install:

```
DASD=1b0-1bf,2b0-2bf,320-33f
HOSTNAME=myhost.mycompany.com
NETTYPE=qeth
IPADDR=n.n.n.n
SUBCHANNELS=0.0.0600,0.0.0601,0.0.0602
NETWORK=n.n.n.n
NETMASK=255.255.255.0
SEARCHDNS=mycompany.com
BROADCAST=n.n.n.n
GATEWAY=n.n.n.n
DNS=n.n.n.n
MTU=1500
PORTNAME=DONTCARE
PORTNO=0
LAYER2=0
```

### 1.11.10 RHEL53 EXEC

The **RHEL53 EXEC** punches the kernel, parameter file and RAMdisk to the reader and invoke the RHEL 5.3 installation process:

```
/* PROFILE EXEC for Linux virtual servers */
Address 'COMMAND'
'CP SET RUN ON'
'CP SET PF11 RETRIEVE FORWARD'
'CP SET PF12 RETRIEVE'
'ACCESS 592 C'
'EXEC SWAPGEN 1B2 131072' /* create a 64M VDISK disk swap space */
'EXEC SWAPGEN 1B3 262144' /* create a 128M VDISK disk swap space */
'PIPE CP QUERY' Userid() '| VAR USER'
Parse Value user With id . dsc .
iplDisk = 1B0 /* /boot/ is 1B0 */
If (dsc = 'DSC') Then /* user is disconnected */
  'CP IPL' iplDisk
Else Do /* user is interactive -> prompt */
  Say 'Do you want to IPL Linux from DASD' iplDisk'? y/n'
  Parse Upper Pull answer .
  If (answer = 'Y') Then
    'CP IPL' iplDisk
  End
Exit
```

## 1.12 The team that wrote this paper

This paper was updated for RHEL 5.3 in Poughkeepsie in 2009 by the following authors:

**Brad Hinson** works at Red Hat in Raleigh, NC. He primarily works on System z bug fixes, but in his spare time likes to test and document new functionality like Fedora, Cobbler, and of course, *read-only root*.

**Michael Maclsaac** works at IBM in Poughkeepsie NY with the z/VM team. He was way too busy to help create another *flavor* of this paper, but still found the time because it was much more fun than the day-to-day insanity at IBM.

### 1.12.1 Thanks

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

Lydia Parziale  
International Technical Support Organization, Poughkeepsie Center

Ernie Horn, Christian Paro, Nick (Jeng-Fang) Wang  
IBM Poughkeepsie

Mike Nardone, Cortez Crosby  
Nationwide Insurance

Mark Post  
Novell

Marian Gasparovic  
IBM Slovakia

A special thanks to Steve Womer and Rick Troth for writing the majority of the original paper.

A special thanks to Carlos Ordonez of IBM Poughkeepsie, for being willing to sit through high level architecture discussions and always asking “Why can’t that be done more simply?”

A special thanks to Jim Vincent of Nationwide for converting *willy-nilly* REXX code into something a bit more reputable.

A special thanks to Brian France, Kyle Black and Dominic Depasquale of Penn State for reaching out and “giving back” to make the update to the SLES paper in 2009 possible.

