HP SmartStart Scripting Toolkit
Linux Edition
Best Practices

August 2004 (First Edition)
Part Number 365443-001

The information contained herein is subject to change without notice. The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

HP SmartStart Scripting Toolkit Linux Edition Best Practices

August 2004 (First Edition)
Part Number 365443-001

Contents

About This Guide

Chapter 1
Introduction

Chapter 2
Booting the Linux Toolkit Environment

Chapter 3
Using the Linux Toolkit Environment

Appendix A
Editing the File System File

Index

About This Guide

This guide describes the implementation of the HP Smart Scripting Toolkit utilities with Linux.

This guide **does not** provide details about installing the Linux operating system.

Audience Assumptions

This guide is intended for experienced network administrators. Some of the utilities in the Toolkit can cause a loss of data if used incorrectly. Thus, a clear understanding of the system and these utilities is critical.

Where to Go for Additional Help

For issues or problems not addressed by this guide, refer to the following resources for more information.

- The SmartStart Scripting Toolkit website: http://www.hp.com/servers/sstoolkit
- The Red Hat Linux website: http://www.redhat.com/

Customer Feedback

HP offers standard e-mail support for the SmartStart Scripting Toolkit. Refer to http://www.hp.com/go/support, select your particular HP ProLiant server product, and use the online form to submit your question.

Telephone Numbers

For the name of your nearest HP authorized reseller:

- In the United States, call 1-800-345-1518.
- In Canada, call 1-800-263-5868.

For general HP technical support:

- In the United States and Canada, call 1-800-HP INVENT (1-800-474-6836).
- Outside the United States and Canada, refer to http://www.hp.com/.

1

Introduction

This document describes how to best utilize the Linux edition of the SmartStart Scripting Toolkit to configure HP ProLiant servers and contains information about using the Toolkit utilities in an unattended environment.

NOTE: This document does not include information about installing the operating system.

The best practices also includes suggestions about how to organize utilities and data files by following a set of operational procedures that standardize configuration procedures and help reduce errors.

△ **CAUTION:** Because of the potential risk of data loss, be sure that all necessary precautions are taken so that mission-critical systems are not disrupted if a failure occurs.

Toolkit Changes

Previous versions of the SmartStart Scripting Toolkit utilities were designed for the MS-DOS environment. However, limitations in MS-DOS have become a significant problem as hardware and software have evolved. In response to this issue, HP has migrated the SmartStart Scripting Toolkit to a Linux environment to provide better scripting and better hardware support.

The Linux edition of the Toolkit provides the same functionality as the earlier MS-DOS version. However, tools, arguments, and data files are different in the Linux edition of the Toolkit, so you must update your customized scripts to work in a Linux environment. Data files have been migrated to industry-standard XML format to provide improved extensibility. For more information about the Toolkit utilities, refer to the *HP SmartStart Scripting Toolkit Linux and Win32 Editions User Guide*.

Minimum Requirements

Before beginning the deployment process, be sure to have the following items available:

- *HP SmartStart Scripting Toolkit Linux and Win32 Editions User Guide*
- SmartStart Scripting Toolkit Linux Edition
- A Linux workstation (any Linux distribution)
- Syslinux package, downloaded from the Web

Overview

The Linux Toolkit environment is a small "multi-purposed" Linux environment that can be booted from various media, such as CD, network or Preboot eXecution Environment (PXE), or bootable USB device. After the Linux environment is booted, you can do almost anything that Linux is capable of doing, provided that the tools are available. This document discusses the following topics:

- Booting the environment
- Using the environment

The first section describes how to set up the pieces needed to boot the environment from CD or using PXE. The second section describes how to use the environment to perform certain functions, such as capturing configurations, flashing configurations, and setting up partitions for operating system installation.

Because the Toolkit is Linux-based, the operations described here are assumed to be performed under Linux. Unless specifically stated, a full Linux installation of any distribution of Linux released in the last two years will enable you to perform all the operations described in this document.

2

Booting the Linux Toolkit Environment

Overview

The three main components needed to boot any Linux environment are the bootloader, the Linux kernel, and the Linux file system. For the Linux edition of the SmartStart Scripting Toolkit, these components are:

- Bootloader: syslinux
- Kernel: vmlinuz, a kernel based on Red Hat 7.3
- File system: initrd.img, a reduced Red Hat 7.3 installation

In general, you must only modify the bootloader to boot the Toolkit in your environment.

Whether you boot from a CD or through a network, the following general steps must occur for the Toolkit environment to boot:

1. The system boots and performs a Power-On Self Test (POST).
2. The BIOS determines how the system boots, the BIOS settings, and the available media.
3. The BIOS retrieves the bootloader using the available media.
4. The bootloader (syslinux) loads the Linux file system into memory.
5. The bootloader boots the Linux kernel.
6. A startup script is executed to finish the environment setup.

Syslinux

Syslinux is a free third-party bootloader available at http://syslinux.zytor.com/index.php.

Syslinux consists of a suite of programs that perform various boot functions. The Toolkit uses the following two bootloader programs:

- isolinux.bin—This program enables you to boot from ISO media.
- pxelinux.0—This program enables you to boot using PXE protocol.

Both bootloaders require a configuration file to run:

- isolinux.cfg—This is the configuration file for booting from ISO media.
- default.cfg—This is the configuration file for booting using PXE.

The following is a sample configuration file:

```
default normal_boot
prompt 1
timeout 100

label normal_boot
   kernel vmlinuzA
   append initrd=initrd.img rw root=/dev/ram0 ramdisk_size=39567
         init=/first_script
label bash_prompt
   kernel vmlinuzB
   append initrd=initrd.img rw root=/dev/ram0 ramdisk_size=39567
         init=/bin/bash
```

In this example, there are two distinct boot directives: `normal_boot` and `bash_prompt`. The configuration file instructs the bootloader to prompt the user to pick a boot directive, wait 10 seconds, and then boot the default directive, `normal_boot`, if no user input is recorded.

The `normal_boot` directive instructs the bootloader to use vmlinuzA as the kernel. The append line specificies which parameters the bootloader passes to the kernel. The parameter `init=/first_script` instructs the kernel to pass control to the file `first_script`.

For more information about syslinux usage, refer to the syslinux documentation.

Booting from the CD

Figure 2-1 explains a Toolkit CD boot.

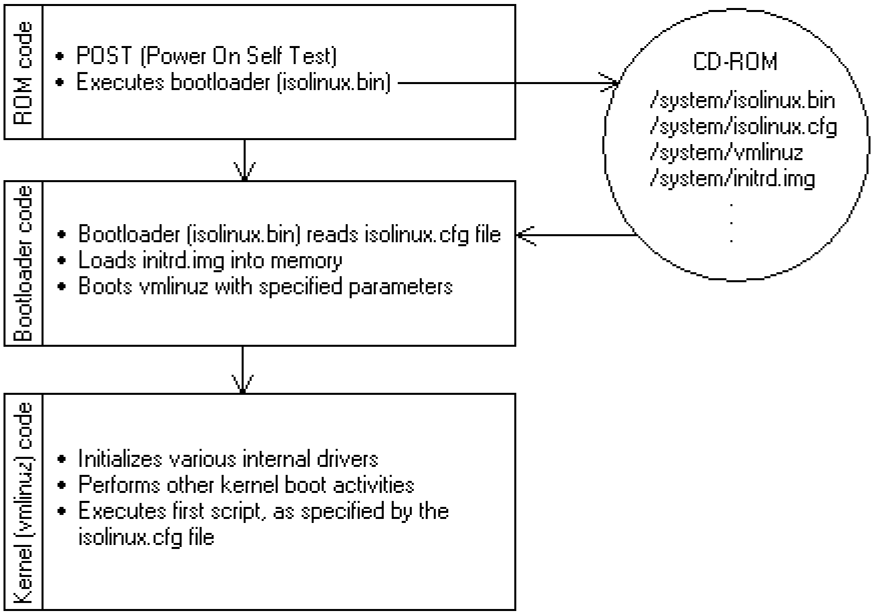
ROM code
• POST (Power On Self Test)
• Executes bootloader (isolinux.bin)
CD-ROM
/system/isolinux.bin
/system/isolinux.cfg
/system/vmlinuz
/system/initrd.img
Bootloader code
• Bootloader (isolinux.bin) reads isolinux.cfg file
• Loads initrd.img into memory
• Boots vmlinuz with specified parameters
Kernel (vmlinuz) code
• Initializes various internal drivers
• Performs other kernel boot activities
• Executes first script, as specified by the isolinux.cfg file

Figure 2-1: Toolkit CD boot process

Creating a Toolkit bootable CD requires the following general steps:

1. Create a CD build directory.
2. Create an ISO image to be burned to CD.

Creating a CD Build Directory

1. Create a directory on the Linux workstation; for example ./linuxbootCD.
2. Create a subdirectory on which to store boot files; for example ./linuxbootCD/system.
3. Copy the necessary boot files to the ./linuxbootCD/system directory:
   - isolinux.bin (the syslinux binary used for ISO media)
   - isolinux.cfg (the boot directive used by isolinux.bin)
   - initrd.img (the Linux file system)
   - vmlinuz (the Linux kernel)

If needed, copy additional files to ./linuxbootCD. These files might include Toolkit files, configuration files, or third-party tools. Alternately, refer to chapter 3, “Using the Linux Toolkit Environment,” for information about downloading the files from a network share.

Creating an ISO Image

The mkisofs command is used to create an ISO image. The following table explains some of the arguments used with this command.

Table 2-1: Mkisofs arguments

Argument	Description
`-o linuxbootCD.iso`	This argument is the output of the mkisofs command, the ISO file.
`-b system/isolinux.bin`	This argument sets isolinux.bin as the bootloader.
`-V LinuxBootCD`	This argument sets the volume label of the CD.
`./linuxbootCD`	This argument specifies the target directory that will be the root of the CD.

To create the ISO image, execute the following command at the shell prompt:

```
mkisofs  -J -iso-level 3 -R -L -o linuxbootCD.iso  \
         -b system/isolinux.bin -c system/boot.cat \
         -V LinuxBootCD                            \
         -no-emul-boot -boot-load-size 4           \
         -boot-info-table                          \
         ./linuxbootCD
```

The ISO file can now be burned to a CD.

Booting Using PXE

IMPORTANT: A basic understanding of Dynamic Host Configuration Protocol (DHCP), PXE, and Trivial File Transfer Protocol (TFTP) is required to perform the procedure described in this section.

Figure 2-2 explains a simplified Toolkit PXE boot.

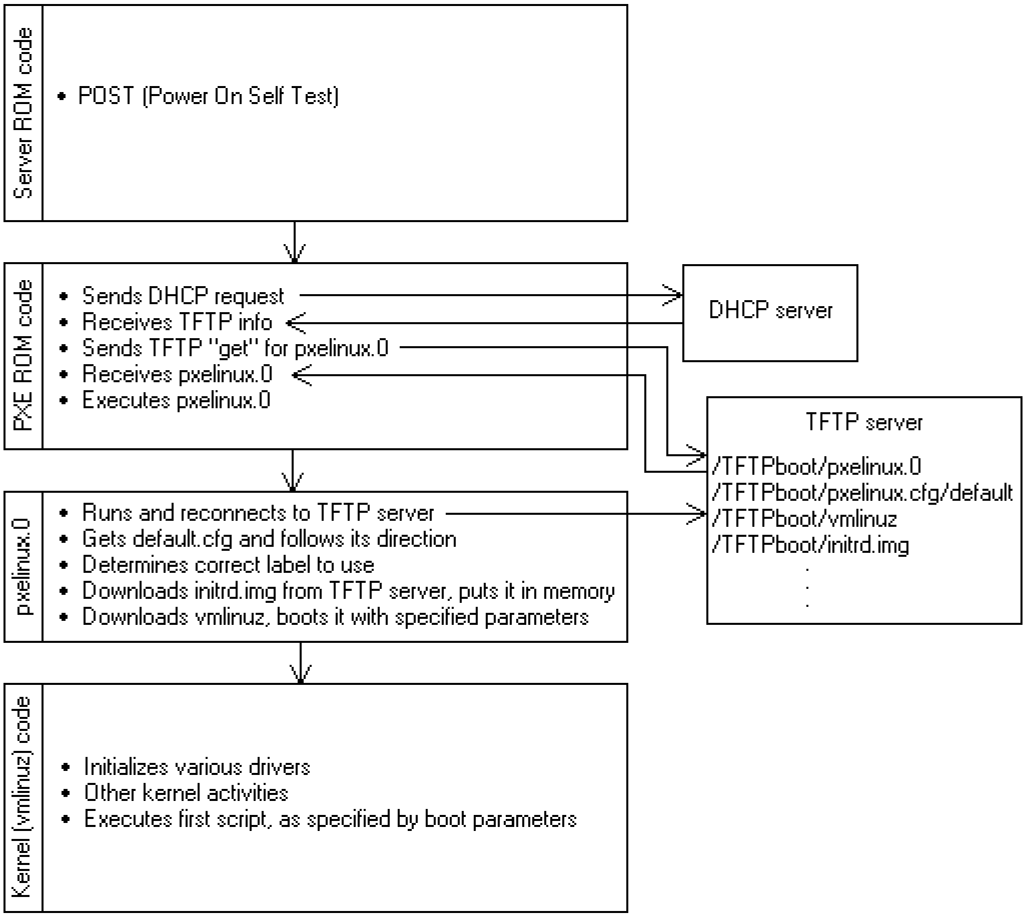
Server ROM code
• POST (Power On Self Test)
PXE ROM code
• Sends DHCP request
• Receives TFTP info
• Sends TFTP "get" for pxelinux.0
• Receives pxelinux.0
• Executes pxelinux.0
DHCP server
TFTP server
/TFTPboot/pxelinux.0
/TFTPboot/pxelinux.cfg/default
/TFTPboot/vmlinuz
/TFTPboot/initrd.img
pxelinux.0
• Runs and reconnects to TFTP server
• Gets default.cfg and follows its direction
• Determines correct label to use
• Downloads initrd.img from TFTP server, puts it in memory
• Downloads vmlinuz, boots it with specified parameters
Kernel (vmlinuz) code
• Initializes various drivers
• Other kernel activities
• Executes first script, as specified by boot parameters

Figure 2-2: Toolkit PXE boot process

Setting up a PXE boot environment requires the following general steps:

1. Set up a DHCP server with the appropriate options.
2. Set up a TFTP server with the appropriate options.
3. Populate the TFTP directory share with the Linux Toolkit boot components.

NOTE: These steps assume that a Linux workstation is used as the DHCP/TFTP server. You might need to download additional components and adapt the following instructions to suit your environment.

Setting Up a DHCP Server

To set up a DHCP server, you must create and edit a dhcpd.conf file in the /etc directory of your server. The following is a sample dhcpd.conf file:

```
allow booting;
allow bootp;
ddns-update-style ad-hoc;
```

[Insert the usual DHCP directives, IP ranges, subnet masks, and so on here.]

```
next-server IP_ADDRESS_OF_TFTP_SERVER;
filename "pxelinux.0";
```

The `next-server` command tells the DHCP client where to send the TFTP `get` request

The `filename` command tells the DHCP client which file to `get`. In this example, the file is pxelinux.0.

When you have finished creating the dhcpd.conf file, restart the dhcpd process:

```
/etc/init.d/dhcpd restart
```

Setting Up a TFTP Server

Most Linux installations include a TFTP server and an automated method of launching the server upon receiving a TFTP request. The parent process for detecting a TFTP request and launching the TFTP server is called xinetd. However, you might have to enable the TFTP service. The TFTP file is located in the /etc/xinetd.d/ directory. The following is a sample TFTP file:

```
# default: off
# description: The tftp server serves files using the \
#  trivial file transfer protocol. The tftp protocol is \
#  often used to boot diskless workstations, download \
#  configuration files to network-aware printers and to \
#  start the installation process for some operating systems.

service tftp
{
   socket_type = dgram
   protocol    = udp
   wait        = yes
   user        = root
   server      = /usr/sbin/in.tftpd
   server_args = -s /tftpboot
   disable     = yes
   per_source  = 11
   cps         = 100 2
}
```

In this example, “disabled” is the default setting, and /tftpboot is the root directory for all client access.

To enable the TFTP service on your server, edit the `disable` line to read:

```
disable           = no
```

Populating the TFTP Directory Share

Use the /tftpboot directory from the previous section, “Setting Up a TFTP Server,” as the directory from which TFTP clients will get their files. To populate the TFTP directory share:

1. Create a /tftpboot directory if needed.
2. Copy the necessary boot files to the /tftpboot directory:
   - pxelinux.0 (the syslinux binary used for PXE boot)
   - initrd.img (the Linux file system)
   - vmlinuz (the Linux kernel)
3. Create a pxelinux configuration file subdirectory called /tftpboot/pxelinux.cfg.
4. Copy the default.cfg file (the boot directive used by pxelinux.0) into the /tftpboot/pxelinux.cfg/ directory.

The tftpboot directory should now contain the following items:

```
/tftpboot/pxelinux.0

/tftpboot/initrd.img

/tftpboot/vmlinuz

/tftpboot/pxelinux.cfg/default.cfg
```

3

Using the Linux Toolkit Environment

Overview

This chapter describes the general tasks for setting up a target server for operating system installation:

- Basic Toolkit strategy
- Toolkit environment setup
- Operating system-independent configuration
- Windows pre-installation setup
- Linux pre-installation setup

You might need to adapt these tasks to suit your specific needs.

Basic Toolkit Strategy

The three primary components of a Toolkit Linux boot are the bootloader, the kernel, and the file system (initrd.img):

- The choice of bootloader depends on whether the boot process occurs using the CD media or over a network using PXE.
- The kernel is generally static and cannot be modified easily. HP recommends that you use the kernel that is shipped with the Toolkit because it has been tested on all servers supported by the Toolkit.
- The initrd.img must be created during the Toolkit setup process and is easily modified. For details on creating an initrd.img file, refer to Appendix A, “Editing the File System File.”

The initrd.img file enables a target machine to boot a Linux environment with network connectivity. You can edit the network settings in the packaged initrd.img file to be specific to your deployment environment. For more information, refer to Appendix A.

The bootstrap script in the initrd.img file copies the next script in the execution path and executes it, beginning the Toolkit process. Because this script, toolkit_redirect.sh, resides on a network share and not inside initrd.img, you can modify it as often as needed without needing to rebuild the initrd.img file.

The primary purpose of the toolkit_redirect.sh script is to connect to the main network share and copy the main_toolkit_launch.sh script. This redirection is necessary because the deployment environment can change over time. If the main network share crashes and a secondary network share takes its place, the toolkit_redirect.sh script can be changed to point to the new network share. The initrd.img file does not need to be revised.

The main_toolkit_launch.sh script copies the remaining toolkit utilities to the ramdrive created by the bootstrap script. It then determines the server ID of the target server and uses that information to copy the server-specific configuration scripts and data files.

The main_toolkit_launch.sh script then executes the server-specific script. The remaining steps are specific to each deployment situation.

Toolkit Environment Setup

To set up the Toolkit environment, the bootstrap script at the root of the initrd.img file system:

1. Mounts the /usr directory. This step makes the /usr directory available to the rest of the Linux environment.
2. Creates an expandable ramdrive to serve as the Toolkit work area.
3. Loads all common network drivers needed by HP Proliant servers. You can edit this script to add or remove network drivers.
4. Configures the network interface device (eth0). You **must** edit the bootstrap script to match your deployment environment. To assist you in configuring the eth0 device, the ifconfig and dhclient programs are provided in the /bin directory.
5. Mounts a network share on the server.
6. Copies the initial toolkit script (toolkit_redirect.sh) from the network share and executes it.

The Toolkit boot is complete.

NOTE: The commands are located in the toolkit_bootstrap.sh script in the /root/compaq/ directory. Steps 1 and 2 are required and should not be modified. Steps 3, 4, and 5 can be modified for your specific deployment environment.

Operating System-Independent Configuration

After the main network share is mounted, the main_toolkit_launch.sh script initiates Toolkit operations.

The sample main_toolkit_launch.sh script:

1. Copies all toolkit utilities from the network share to the target server.
2. Runs hardware discovery to determine server type.
3. Obtains server ID information from the hardware discovery file.
4. Copies server-specific configuration script and data files from the network share.
5. Runs the server-specific configuration script:
    a. Loads drivers for storage controllers and any other devices that must be configured.
    b. Runs the CONREP utility.
    c. Runs the CPQACUXE utility if needed. This is server specific.
    d. Runs any other configuration utilities.
    e. Runs hardware discovery to determine the boot controller.
    f. Obtains the device node of the boot controller. This information is required for the disk carving portion of the operating system setup, as described in the following two sections.
    g. Runs the operating system-specific setup script.

These steps can be adapted to your server deployment process.

Windows Pre-installation Setup

You can use the win_diskcarve.sh script to set up a DOS partition for Microsoft® Windows® Server 2000 or Windows 2003 Server setup. Run win_diskcarve.sh in its own directory to create a 4 GB bootable DOS partition on the system's boot controller.

After the bootable DOS partition is created, you can populate the partition with the Windows installation files (i386 directory and unattend.txt), HP drivers, and ProLiant Support Packs.

NOTE: This step is not specific to the Linux Toolkit environment. For more information on setting up a Windows unattended install, refer to Microsoft documentation.

Linux Pre-installation Setup

To set up a Red Hat kickstart install using the Toolkit Linux Edition:

1. Run the linux_diskcarve.sh script in its own directory. This script:
   a. Creates a FAT16 partition.
   b. Copies ldlinux.sys, which is the syslinux binary for booting off FAT partitions, to the partition.
2. Copy the Red Hat vmlinuz kernel to the partition.
3. Copy an initrd.img file that contains the necessary drivers for the install to the partition:
   a. Obtain a Red Hat bootnet.img diskette image from the Red Hat media or from the HP website (http://www.hp.com), where there is a downloadable bootnet.img file for each server ProLiant server that supports Red Hat Linux.
   b. Mount the image on your Linux system.
   c. Copy the vmlinuz and initrd.img files from the diskette image to your network share.

   NOTE: The initrd.img from the bootnet.img file is **not** equivalent to the initrd.img delivered with this toolkit.
4. Copy a syslinux.cfg file that specifies a kickstart install to the partition.
5. Reboot to the FAT16 partition.

The following is a sample syslinux.cfg file:

```
default ks
prompt 1
timeout 6
display boot.msg
F1 boot.msg
label linux
kernel vmlinuz
  append initrd=initrd.img lang= devfs=nomount ramdisk_size=9216
label ks
  kernel vmlinuz
  append ks=nfs:10.7.90.120:/STORAGE/examples/ks_rh80.cfg
initrd=initrd.img lang= devfs=nomount ramdisk_size=9216 network
```

The following line points the Red Hat install to a Network File System (NFS) share to obtain the kickstart file ks_rh80.cfg:

```
append ks=nfs:10.7.90.120:/STORAGE/examples/ks_rh80.cfg
```

You must modify this line to match your deployment environment.

Most recent Red Hat installations provide a kickstart file, called anaconda-ks.cfg, in the /root directory. Modify this file, and put it on your network share for subsequent installations.

To perform a Red Hat unattended install, modify the following lines:

```
network --device eth0 --bootproto dhcp
```

The preceding line specifies which device to configure and what protocol to use.

```
nfs --server 10.7.90.120 --dir /STORAGE/rh80
```

The preceding line specifies that NFS be used as the file sharing protocol. You must set up the NFS share with the Red Hat installation tree. For more information about how to set up the installation tree, refer to the README file on the Red Hat CD media.

```
clearpart --all --drives=cciss/c0d0
```

The preceding line erases everything from the install drive, including the FAT16 partition that was created in step 1 of the “Linux Pre-installation Setup” section in this chapter. HP recommends erasing the FAT16 partition.

Modify the rest of the ks.cfg file as needed.

A

Editing the File System File

Overview

The ability to edit the file system file (initrd.img) enables you to adapt the Linux toolkit to various deployment environments.

Most users are familiar with the concept of a file system located on a disk partition. The initrd.img file is a file system located in a file. In this case, a large file (initrd.img) is created and formatted as a file system. The file acts as a complete disk partition.

In Linux and UNIX®-like systems, partitions on a hard drive must be mounted at a mount point before they can be accessed. The initrd.img is no exception. However, because initrd.img is a file, it must be mounted as a loopback device. When mounted, the initrd.img can be manipulated like any other mounted partition. Files can be deleted from it and added to it.

NOTE: After the initrd.img file is created, its size cannot be changed.

Editing the initrd.img File

NOTE: All steps described in this appendix must be performed on a Linux system as the root user.

△ **CAUTION:** Improper editing of the initrd.img file can result in an unusable file. Be sure to back up your work before proceeding.

The following basic steps are required to edit the initrd.img file:

1. Uncompress the initrd.img file.
2. Mount the initrd.img.uncompressed file.
3. Edit the initrd.img.uncompressed file.
4. Unmount the initrd.img.uncompressed file.
5. Compress the initrd.img file.

These steps are described in detail in the following sections.

Uncompressing the initrd.img File

The initrd.img file was compressed with the gzip utility. To uncompress initrd.img, use the gunzip utility.

1. Rename the packaged initrd.img file with a file name that has the extension `.gz`:

```
mv (or cp) initrd.img initrd.img.uncompressed.gz
```

2. Uncompress the renamed file.

```
gunzip (or gzip -u) initrd.img.uncompressed.gz
```

This produces a new file called initrd.img.uncompressed. Notice that the size of this file is different from the original initrd.img file.

Mounting the initrd.img.uncompressed File

Linux enables you to mount file systems that exist in a file (as opposed to file systems on a disk partition) by using a loopback device. Discussion of loopback devices is outside the scope of this document.

Mount the initrd.img.uncompressed file:

```
mount -o loop initrd.img.uncompressed mnt_point/
```

After mounting the file system, you can navigate to it using the mount point.

Editing the initrd.img.uncompressed File

When the initrd.img.uncompressed file is mounted, you can make any desired changes to it by using the mount point, just as you would edit files on a mounted partition.

NOTE: The initrd.img file shipped with the Toolkit has approximately 5 MB of free space. Linux requires a small amount of writeable free space on the filesystem to operate properly, so be sure not to fill the entire free space of the initrd.img file.

The bootstrap script that begins the toolkit execution process is located at the root of the file system. You might need to make changes to this script to suit your specific needs.

Unmounting the initrd.img.uncompressed File

After editing the initrd.img.uncompressed file, you must unmount it:

```
umount mnt_point/
```

Compressing the initrd.img.uncompressed File

1. After unmounting the file, compress it:

   ```
   gzip -v -9 initrd.img.uncompressed
   ```

   The resulting file is called initrd.img.uncompressed.gz.

2. Rename initrd.img.uncompressed.gz with the original name:

   ```
   mv (or cp) initrd.img.uncompressed.gz initrd.img
   ```

The new initrd.img is ready to be used.

Index