USENIX Association

Proceedings of the 14th Systems Administration Conference (LISA 2000)

New Orleans, Louisiana, USA
December 3–8, 2000

Automating Dual Boot (Linux and NT) Installations

Rajeev Agrawala, Rob Fulmer, & Shaun Erickson – Lucent/Bell-Labs Research

ABSTRACT

This paper presents a solution for automating dual boot Linux and Windows NT 4.0 installations on a PC. The process is automated to the extent that one starts with a PC with a blank hard disk and ends up with a fully customized dual-boot PC. The process involves booting a PC with a floppy disk; choosing what kin of system is required from a menu and then, once the install begins, popping out the disk. The unattended PC goes through installing and customizing the Linux installation. It then automatically boots into DOS and installs Windows NT 4.0. The whole process takes about an hour or two, depending upon the type of installation and network speed. After the process is complete, the PC is ready, fully customized to the user's local environment. The solution presented in this paper can be used to do either Linux-only or NT-only installs in addition to dual-boot configurations.

Introduction

Automating the operating system's installation has always been a topic of interest to System Administrators. In the era of installing Windows 3.1 and applications, automation meant eliminating the process of feeding 30 floppy disks one by one, only to find out that the 27th disk had a bad sector error. Solaris provides JumpStart, with which one can automate OS installation. IRIX, in its latest release (6.5), provides Roboinst to automate the OS installation. We have automated Windows NT 4.0 installs [1]. Linux (Redhat) provides Kickstart to automate the OS installation [2].

All of these methods automate the process of one OS installation only. Our environment initially consisted of Unix workstations on desktops. Then the trend shifted towards PCs running Windows NT 4.0 on desktops and Unix servers for the backend processing. With the popularity of Linux, more and more users in our environment started asking for PCs running both Linux and NT with a choice to boot in either operating system. With this trend, the support group had to spend a lot of time on each PC to make it dual boot. Also, each installed PC was one of a kind, depending upon the way a particular administrator decided to customize and install it. The method described in this paper was developed so those system administrators could install PCs more efficiently and consistently.

Overview

The installation process uses a modified "bootnet" disk from RedHat Linux Kickstart. The PC is booted off this disk, and a menu is presented to the administrator. The menu typically consists of the choices for different ways of customizing the installation. The choices, for example, could be

- Both Linux and NT install customized for department A, or B, or C.
- Only Linux install for department A, or B, or C
- Only NT install for department A, or B, or C
- Generic Linux and NT install (No customization)
- Generic Linux install
- Generic NT install

Once the system administrator chooses the installation type from the menu, the installation starts. At this stage, he can pop the floppy disk out and walk away from the PC. From this point onwards, the installation goes unattended. When the installation is finished, the PC is ready with both operating systems and is usable. Since Redhat Linux does not support the latest video cards out of the box, we skip the installation of X altogether for consistency. Also, sound installation is not supported through kickstart. Therefore, with our current setup, one has to configure X and audio after the installation is done.

The steps involved in doing a dual boot installation are:

- Starting the first OS installation
- Customization of first operating system
- Preparing/repartitioning and formatting disk for second OS installation.
- Copying the second OS installation files to the partition prepared in previous step.
- Handing over control to second operating system install program.
- Second operating system installation and customization.
- Passing control back to first operating system.

The first step above is started manually by the administrator. The rest of it happens automatically.

Constraints

The seven steps mentioned above impose some restrictions on what can be chosen as the first operating system and what can be second.

The following problems need to be addressed to be able to do a successful install:

1. The installation mechanism should be capable of installing either operating system or both.
2. There is some customization required, based on departments, user requirements, etc. Therefore, it should be possible to specify the type of customization for both operating systems at the start of the installation process.
3. The first operating system should have knowledge about the filesystem for the second operating system and should be able to make the file system bootable for it.

We shall see later in this paper how the first two requirements are met. The third requirement makes Linux a natural choice for the first operating system, since Linux can create a DOS filesystem through

```
Default deptA
Prompt 1
Display dualboot.msg
F1 dualboot.msg
F2 other.msg
F3 expert.msg
F4 param.msg
F5 rescue.msg
F6 boot.msg
F7 snake.msg
Label linux
  Kernel vmlinuz
  Append initrd=initrd.img network
Label text
  Kernel vmlinuz
  Append initrd=initrd.img network text
Label expert
  Kernel vmlinuz
  Append expert initrd=initrd.img network
Label ks
  Kernel vmlinuz
  Append ks initrd=initrd.img network
Label deptA
  Kernel vmlinuz
  Append initrd=initrd.img network ks=nfs:serverIPAddress:/kickstart/ks.cfg profile=deptA
Label 1
  Kernel vmlinuz
  Append initrd=initrd.img network ks=nfs:serverIPAddress:/kickstart/ks.cfg profile=deptA
Label generic
  Kernel vmlinuz
  Append initrd=initrd.img network ks=nfs:serverIPAddress:/kickstart/ks.cfg profile=generic
Label 2
  Kernel vmlinuz
  Append initrd=initrd.img network ks=nfs:serverIPAddress:/kickstart/ks.cfg profile=generic
Label ntonly
  Kernel vmlinuz
  Append initrd=initrd.img network ks=nfs:serverIPAddress:/kickstart/ntonlyks.cfg profile=ntonly
Label 3
  Kernel vmlinuz
  Append initrd=initrd.img network ks=nfs:serverIPAddress:/kickstart/ntonlyks.cfg profile=ntonly
Label deptAlinux
  Kernel vmlinuz
  Append initrd=initrd.img network ks=nfs:serverIPAddress:/kickstart/linux-server.cfg profile=deptAlinux
Label 4
  Kernel vmlinuz
  Append initrd=initrd.img network ks=nfs:serverIPAddress:/kickstart/linux-server.cfg profile=deptAlinux
Label genericlinux
  Kernel vmlinuz
  Append initrd=initrd.img network ks=nfs:serverIPAddress:/kickstart/linux-server.cfg profile=genericlinux
Label 5
  Kernel vmlinuz
  Append initrd=initrd.img network ks=nfs:serverIPAddress:/kickstart/mathsummer.cfg profile=genericlinux
```

Figure 1: Modified SYSLINUX.CFG file.

available tools, and, with some effort, it is possible to make the DOS partition bootable.

The Details

First we place the Linux distribution on a Unix server. This can be downloaded from the RedHat Web site at http://www.redhat.com. This distribution should be exported read only through NFS.

Boot Floppy

We next create a boot floppy from the bootnet floppy image provided by the RedHat Linux distribution. We then modify the syslinux.cfg and create a file named dualboot.msg to include the menu to be presented to the administrator. Figure 1 shows part of modified syslinux.cfg file for the possible menu choices mentioned earlier. Figure 2 shows the dualboot.msg file, which is presented to the system administrator at startup.

Let's analyze the line shown in bold in Figure 1. This line is used by syslinux to invoke the kernel, when the administrator chooses 'dual boot install for department A'. This line tells the kernel to use kickstart to do the Linux installation and use the file ks.cfg from serverIPAddress as the kickstart configuration file. Here the administrator will need to replace the "serverIPAddress" with the IP address of the NFS Server where the ks.cfg file is stored. This line also specifies 'profile=deptA'. This is not a kickstart or kernel directive. The kernel is passed this parameter profile but ignores it. Later in the postinstall section, we retrieve this parameter from the kernel and use it to customize Linux and NT. Syslinux specifies two configuration files. One is the kickstart configuration file, used by kickstart. The second is a customization configuration file, which is used by the configuration control program ksconfig. Ksconfig is a perl script, which is the heart of the installation system. It takes control of the installation during the %post section of kickstart and then drives the rest of the installation process using the directives in the profile parameter.

Figure 2 shows the 'DUALBOOT.MSG' file. This file is displayed when syslinux starts from the boot floppy. The text in the file acts as the menu text for the administrator.

Note that by passing the profile parameter to the kernel, we solve the first constraint mentioned in earlier. As we will see later, the profile specifies the customization for both the operating systems.

Kickstart

We encourage the reader to read about kickstart in reference [2]. Here we give a brief description of kickstart, as it relates to this paper.

Figure 3 gives a typical ks.cfg file for used for Linux installation. Here we provide the anatomy of important lines and their significance. For the sake of clarity, we have added line numbers, but they are not part of the actual ks.cfg file.

Line 3 instructs kickstart to use network mode and use DHCP to get IP information.

Line 5 specifies /linux/redhat-6.2 as the distribution directory using NFS from kickstartServer.

Line 9 says to restore the standard MBR to the disk. This option didn't work for us for some reason, so we had to do it later in the customization section.

Line 11 instructs kickstart to delete all the partitions on the disk.

Line 13 creates a partition of type ext2 which will be mounted as /boot. The grow parameter on this line causes the partition to grow to a maximum. In reality, this will be constrained, as we will see later.

Line 14 creates a swap partition of size 256 MB.

Line 15 creates an ext2 type partition, which will be mounted as /. The grow parameter on this line causes it to grow to the maximum available disk space.

Since Line 13 and Line 15 both specify the grow option, both the partitions compete for available disk space and get roughly half of the total disk space, minus the swap partition. Kickstart also restricts the /boot partition to be a maximum of 1024 cylinders (due to boot restrictions of LILO), so the /boot partition gets a maximum of 8GB disk if the disk is larger than 16GB. Also, as of the writing of this paper, Kickstart will create a primary partition for /boot and put swap and / in an extended partition.

Line 17 sets the kickstart mode to install rather than upgrade.

```
                 Welcome to Red Hat Linux 6.1!

     Please choose from the following options
      1.  deptA      (Dual boot install linux customized for deptA)
      2.  generic  (Dual boot install linux not customized)
                   (use deliver script before delivery)
      3.  ntonly   (Finally only NT is left installed on the disk)
      4.  genericlo (only Linux installation - uncustomized)
     You can still type commands like ks profile=<someprofile> or
     Use other kickstart config file by specifying in DHCP configuration.
     You can still use other linux boot floppy commands.
           [F1-Main] [F2-Other] [F6-Linux Boot Screen]
```

Figure 2: DUALBOOT.MSG file.

Line 23 instructs kickstart to skip the X server installation. If we don't skip this, kickstart will fail for an unsupported video card.

Line 25 sets the root password for the machine.

Line 29 instructs LILO to install its boot sector on the /boot partition instead of in the MBR.

Line 31 sets up kickstart to reboot automatically, instead of waiting for the user to press return.

Line 34 specifies installation of the complete RedHat distribution.

In the %post section, Line 38 NFS mounts the directory which contains the ksconfig file.

Line 39 executes the script ksconfig. From this point onwards, the ksconfig script takes control of the installation. The complete functionality of this script is discussed later.

Ksconfig Script

During the %post section of kickstart, the ksconfig script is executed. Before the system's first reboot, this script installs itself to be run during runlevels 1 and 3. This script also changes the /etc/inittab file to set the default runlevel to 1, so once kickstart is done and the machine reboots, it goes into runlevel 1.

The ksconfig script is mostly profile driven. It extracts the name of the profile, which was passed to the kernel by syslinux at the start of installation. Although Linux does not provide any direct way of obtaining the invocation time parameters, it keeps them in the /proc/cmdline file. The ksconfig script uses

```
 1. lang en_US
 2. #
 3. network --bootproto dhcp
 4. #
 5. nfs --server kickstartServer --dir /linux/redhat-6.2
 6. #
 7. keyboard us
 8. #
 9. zerombr yes
10. #
11. clearpart -all
12. #
13. part /boot --size 1 -grow
14. part swap --size 256
15. part / --size 1 -grow
16. #
17. install
18. #
19. mouse logimmanps/2
20. #
21. timezone US/Eastern
22. #
23. skipx
24. #
25. rootpw !#%&wtqr
26. #
27. auth --useshadow --enablemd5
28. #
29. lilo --location partition
30. #
31. reboot
32. #
33. %packages
34. @ Everything.
35. #
36. %post
37. /bin/mkdir /mnt2
38. /bin/mount serverIPAddress:/kickstart /mnt2
39. /mnt2/ksconfig
40. /bin/umount /mnt2
41. /bin/rmdir /mnt2
```

Figure 3: A typical kickstart configuration file (ks.cfg) used for Linux installation.

this file to get the profile name. The script assumes that all the profiles are available in the directory from which it was run. It copies the profile to the local system. A sample profile is given in Figure 4. This profile is divided into sections. Each section header is enclosed within '%' characters.

The format of the section header is given below:

```
%SectionName  runlevel-validpasses%
```

SectionName is the name of the section. The valid section names are:

- Environment
- Disklayout
- Nt
- Linux
- Installapplications
- Prepnt
- Config
- Zerombr
- Removepackages
- Cleanup

The section names are case insensitive.

Runlevel-validpass is an optional control string. If present, runlevel is the Linux runlevel, during which this section is active. So, with a section header specifying a runlevel of 3, it will be processed during runlevel 3 only. During the %post section of kickstart (before the machine is rebooted), the runlevel is not set. To process a section during the %post section itself, runlevel should be specified as 'i' (signifying install runlevel).

Validpass is a combination of the following values:

- 0-9 – A number specifying the installation pass during a particular runlevel (e.g., 1 means first reboot in the specified runlevel, 2 means 2nd reboot in the same run level etc.).
- 'd' – The installation pass is set to 'd' when the ksconfig script is manually run to customize a machine after a generic installation. The letter 'd' signifies delivery pass (when the PC is to be finally delivered to the user).

For example, if the administrator decides to process a section during runlevel 1 and pass 2, he will specify 1-2 in the section header. When the ksconfig script first boots into runlevel 1, it sets the pass number to 1. Since the section is valid in pass 2, it will reboot the machine once again in run level 1 (after processing other applicable sections), increase the pass to 2 and then process the section. Specifying '1-1d' as the control will execute the section after first reboot in

```
%environment%
logFile=/var/log/ksconfiglog
dualboot=true
%endEnvironment%
%diskLayout 1-1%
#saves /boot partition on root
partrootdev1=saveOnPartrootdev6
#unmounts /boot partition
partrootdev1=umount
#deletes /boot partition
partrootdev1=delete
#Creates a 30M ext2 partition starting from cylinder 1
partrootdev1=30M,ext2,1
#Makes e2fs filesystem
partrootdev1=mke2fs
#Marks this partition active
partrootdev1=markactive
#mounts it on /boot
partrootdev1=mounton /boot
#restores the original /boot contents
partrootdev1=restoreFromPartrootdev6
#create a 2047M DOS Partition after /boot partition
partrootdev3=2047M,FAT16,afterpartrootdev1
%endDiskLayout%
%nt%
%diskLayout 3-1%
partrootdev3=mkdosfs
%endDiskLayout%
%installapplications 3-1%
acrobat
audix
exceed
gsview
mcafee
netscape
office97
reskit
security
ssh
winzip
%endInstallApplications%
%prepNt 3-1%
hostname=useDummy
preparent=rootdev3
%endPrepNt%
%endnt%
%linux%
%zerombr i-1%
rootdev
%endZerombr%
%removePackages 1-1d%
rpmRemCmd=/bin/rpm --quiet --allmatches --nodeps -e
cxhextris
gnome-games
gnome-games-devel
gnuchess
howto-chinese
kdegames
.
.
.
%endRemovePackages%

%config 1-1%
copyFile=functions
customFile=deptA.1-1
%endConfig%

%config 3-1d%
customFile=deptA.3-1
%endConfig%
%cleanup 3-1d%
/save
/etc/rc.d/init.d/ksconfig
%endLinux%
```

Figure 4: A sample profile 'deptA' for ksconfig.

runlevel 1, and again if the profile is used to manually customize a generic machine.

To better understand the use of pass 'd', consider a scenario where PCs are ordered in bulk for various departments and kept preinstalled. Since, at the time of installation, it is not known which department a particular PC will go to, all the PCs are initially installed with no customization. At the time of delivery, the ksconfig script can be run manually with the profile name for the particular department. At that time only sections marked with pass 'd' will be executed. So the same profile can be used either at the time of customized installation from the start or during manual customization.

The %endSectionName% directive marks the end of a section. Sections can be nested.

Let's now look at the sample profile 'deptA' given in Figure 4. The profile is read top to bottom by the ksconfig script. Any sections not valid for the current runlevel and pass are skipped.

Now we shall discuss the various sections. This discussion is not necessarily in the same order as it appears in the sample profile given, but in the sequence in which the sections are executed.

Environment Section

The first section name is 'environment'. There is no control string, so the section will be executed in all run levels and passes. The first command in this section is logFile=filename. Ksconfig will log the installation in this file. The only other command in this section is dualboot=true. This tells the ksconfig script that this machine is going to be a dual boot system. The other choices are ''linux' or 'nt'

Zerombr Section

The Zerombr section is executed in runlevel 'i'. This means it is executed in the %post section of kickstart. The only command in this section is 'rootdev'. This command instructs ksconfig to restore the Standard MBR to the root disk. For a discussion on what is a root disk, see the next section. This section was introduced as a workaround to Linux Kickstart's zerombr [2], as the command actually did not restore the standard MBR (as of RedHat 6.1). To do this, the ksconfig script dd's (the Unix dd command) 440 bytes of standard MBR to /dev/hda (or the root device).

DiskLayout Section

In the example profile, there are two DiskLayout Sections. The first section is processed in first pass of runlevel 1. This means it is done after kickstart is finished and the machine reboots the first time in single user mode. This section does the disk repartitioning for the installation of Windows NT. As you will recall, during kickstart, we created two ext2 type partitions, roughly the same in size, occupying half of the disk each. The first partition was mounted on /boot, and the 2nd partition was mounted on '/'. Now we will save the contents of the /boot partition, delete the /boot partition, create a 30M /boot partition and restore the /boot files saved earlier. In the rest of the empty space so created, we will make a 2047MB, FAT16 partition and make a DOS filesystem on it. For a dual boot installation, this assumes a minimum disk of size 5 GB. This value can be tweaked, based on the standard NT load. However, if the disk is larger in size and this partition is greater than 2GB, the rest of the partition is not wasted. The NT installation grows this partition to the available size. Figure 5 shows the disk partitioning as the installation proceeds.

The First command in the 'diskLayout' section is:

```
Partrootdev1 = saveOnPartRootdev6
```

This command instructs the ksconfig script to find out the base root device. This would be hda in normal IDE based systems, sda in SCSI based systems and hde if Ultra66 IDE card is used. The ksconfig script takes partition 1 of the root device and saves it as a tar file on partition 6 of the root device. As can be seen in Figure 5(b), partition 1 is the /boot partition and partition 6 is where ''/' is mounted

The rest of the commands in this section:

- Unmount the /boot partition.
- Delete the partition.
- Create a 30MB ext2 partition.
- Make an e2fs filesystem on this partiton.
- Mark the partition active.
- Mount the partition on /boot
- Restore the partition contents saved earlier.
- Create a 2047MB, FAT16 partition after the /boot partition.

The second DiskLayout section is executed in runlevel 3 and is ignored in runlevel 1. This section instructs ksconfig to create a DOS filesystem on partition 3. Partition 3 was already created as type FAT16, in the first DiskLayout section.

Removepackages Section

This section is executed in runlevel 1 and passes 1 and 'd'. This means that if this profile is used during the automatic dual boot install process, it will be processed the first time the machine boots in runlevel 1. Also, if this profile is used to manually customize a generic Linux installed system, this section will be executed. This section lists various RPMs which should be removed from the system. The ksconfig script reads one package name at a time and removes the package. By default 'rpm -quiet -e' is used to remove the package. If other packages have dependencies on the current package, it will not be removed. One can specify a command to remove the package by the directive rpmRemoveCommand=command. This way, if the administrator wants, he can remove the package regardless of dependencies.

Config Section

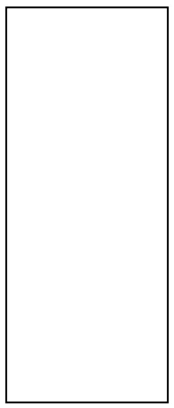
This section facilitates the execution of any script/program to customize the system. In the example profile, there are two Config sections. The first section is processed in runlevel 1 pass 1. The valid commands in this section are copyFile and customFile. The CopyFile command simply copies the file/directory from the NFS mounted directory to the local disk on the system. The CustomFile command copies the filename specified to local disk and executes it. Though the command execution is done in the runlevel/pass specified in the header, the actual copy operation is done during the %post section of the install itself. If networking is not available (as in single user mode), the commands can still be executed. This section is an exception, in the sense that ksconfig script does not completely ignore this section during the %post phase of the installation. For security reasons, one would like to execute a configuration script to set the root password and turn off certain services in runlevel 1, and then do the rest of the customization in runlevel 3 when all the networking is available, such as NIS, mounts etc.

InstallApplications Section

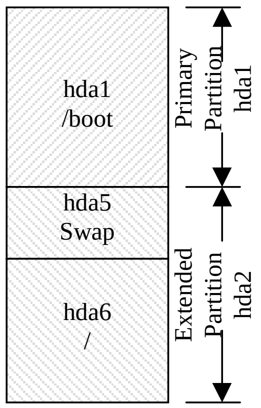
This section lists the applications that should be installed on NT. During this section the ksconfig script only makes a list of applications to be installed. This list is used during the prepNT section (discussed later), to generate a bat file (apps.bat) for NT to install these applications.

PrepNT Section

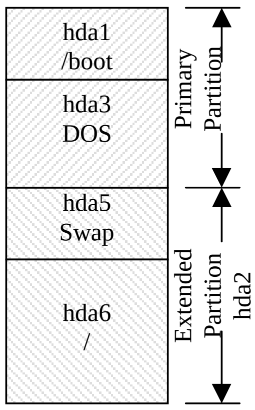
This section prepares for NT installation on the DOS partition created earlier. The only valid commands in this section are hostname and prepareNT. If



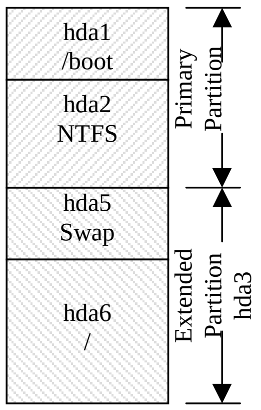
a) Empty Disk



b) Disk partition by linux install



c) Disk partition for NT install



d) Disk partition after NT install

Figure 5: Disk partition during installation process.

the hostname is specified to be 'useDummy', the script generates a dummy hostname and assigns it to the NT side. This is required because NT 4.0 does not use the hostname provided by DHCP. One of the items in my TODO list is to provide a better solution for setting the NT hostname.

The second command, prepareNT, specifies the partition that should be prepared for NT installation. A lot goes on behind the scenes to prepare the partition. A DOS filesystem was already created on the partition during the DiskLayout section.

Our automated procedure for loading NT relies on having a DOS partition pre-loaded with certain programs and data files. This is later converted into an NTFS partition. We use Linux tools to create and populate this partition. The entire process is described below:

Making Disk DOS Bootable

The first thing we need to do is to make it DOS bootable. For this, we will copy the necessary DOS files io.sys, msdos.sys and command.com to the partition. We will also copy the DOS boot sector to the first 512 bytes of the partition.

Setting Up Networking

Within Linux, we know about the Ethernet adapter, so we generate the necessary LANMAN configuration files and copy them to the DOS partition along with other LANMAN driver files for the Ethernet adapter. We keep all these files in a subdirectory under the same directory where ks.cfg (the kickstart configuration file) is kept.

Setting up to start installation

We next copy the config.sys and autoexec.bat files, which set up the MS LANMAN networking, map a directory from the network which contains the NT installation and other application installation files, and then runs the NT setup program to start NT installation. We also generate a file called apps.bat, which contains commands to install applications mentioned in 'installApplications' section (described earlier).

Passing control to DOS

After copying all the necessary files to the DOS file system, we need to pass control to DOS, to start NT installation. We do this by modifying the LILO.CONF file. LILO is set to boot into DOS by default and then the system is rebooted.

Once control is passed to DOS, the NT installation starts. The rest of the NT installation and customization process is described in detail in Autoinstall for NT [1].

Getting Control Back to Linux

During the installation, NT makes its own partition active. Therefore when the NT installation finishes, the system immediately boots NT, without going through LILO. We run a script as the last part of the NT installation to mark partition 1 active again. This makes the boot process go through LILO, giving the choice of booting into either NT or Linux, with NT being default.

Cleanup Section

The cleanup section removes the files that were copied onto the local disk during the installation process.

Installing Other Combinations of Operating Systems

We discussed the dual boot installation process above. Along similar lines, other combinations of OSes can be installed. The SYSLINUX.CFG file in Figure 1 shows the commands to invoke kickstart for other combinations of OSes. Notice that the configuration files for kickstart and ksconfig are different for each combination of OS. For example, to start a Linux-only system, we will use a kickstart file which will only have 30 MB for boot partitions to start with, instead of using the grow option. Also in the ksconfig profile, we will remove all the NT-related sections (DiskLayout & PrepNT etc.).

To install an NT-only system, we change the Kickstart configuration file to do a minimal Linux installation instead of everything (enough for our procedures to work), and in the disk layout section, we don't create a 30MB /boot partition. Instead, we create a DOS partition starting at cylinder 1. We also delete the extended partition (containing the Linux root and swap partitions) before rebooting to go into NT.

Linux and Device Drivers

There are some situations where Linux drivers for the PC hardware being installed are not available. If the drivers are not critical to the Linux boot process, then the installation will complete without any problem. For example, if Linux does not support the Video card or Sound card in the system, this will not affect the installation. However if Linux does not support the IDE controller (for the hard disk), such as an Ultra 66/100 IDE card, then the default RedHat installation will not work. This problem can be solved if there is a kernel patch available for the device in question. Using the patch, a new kernel can be compiled for the boot floppy. Also, new RPMs should be created for the updated kernel and should be added to the Linux distribution. Now the installation can be done on the system in the usual manner. Once you have created a new distribution and boot floppy, you can continue to use this for PCs which may not have the new hardware in them. This means that you only have to maintain one Linux distribution and boot floppy for installing systems. As you add support for new hardware, you are updating the distribution and floppy, which you can use on all systems. Only having to maintain a single boot floppy for many kinds of installations has turned out to be a good thing. In particular it reduces complexity, reduces the work involved in maintaining the system, and reduces confusion.

Alternative Install Techniques

Dual boot installation can be automated using other techniques, in addition to what is described in this paper. Those techniques include

- Cloning the hard disk.
- Copying the NT image during the %post section of kickstart.

In the first method both the operating systems and applications are installed on a "standard PC" in the desired manner. Once the install is done, the disk is kept as a master copy for the install. For any new PC install, a simple disk copy is done onto new PC disk and after host specific customizations installation is complete.

In the second method, kickstart is done pretty much the same way as described in this paper, but during the %post section, instead of installing NT, a previously made NT image is copied to the disk.

Both of these methods provide very good turnaround time for installing PCs in bulk.

However both of these methods have some disadvantages that made us decide to develop our current method. If there are many different hardware configurations, you may need an image per hardware configuration. If there are different types of software configurations required, again, an image per software configuration may be required. If there are many combinations of both software and hardware configurations, then the number of images required can multiply rapidly. If a change is needed, a patch or new software package or whatever, many of these images may require changes. The amount of work required to maintain this seemed too great to be worthwhile. In the case of true installation, one has to upgrade the operating system or application in the distribution only. The next time a new PC is installed, it automatically gets the upgraded OS or application.

The second issue with cloning or copying NT is that all the cloned systems will share the same SID. When these systems are put on the network, Local Administrator on one system will be treated as Administrator on all the cloned systems, because they share same SID, even though they may have different passwords for administrators. This is true for both Administrator and non-Administrator accounts. There is software available which can reset the SID to some random generated value after cloning the disk (like Norton Ghost [3]), but Microsoft does not support resetting the SID.

The advantage of using the method described in this paper is that both operating systems are installed using their native install procedures on the PC where they are going to be used.

Evaluation of the Technique

We have been using this technique to do installs for more than 6 months (as of the writing of this paper). Our PC install group is very satisfied with this method. We have not faced any major problems although we have found some unsupported hardware in Linux. We solve this problem by recompiling the boot kernel and upgrading the kernel in the distribution. We have been requesting the Linux Kickstart developer community to support a %pre section in the kickstart configuration file, similar to the %post section, with the difference that commands specified in %pre section would be executed before Linux installation begins. In that case we would be able to partition the disk the way we need it from the start, and would not have to jump through hoops to repartition the disk. It appears that a %pre section may be available in a future release of RedHat.

To Do List

Though the system described above works quite nicely, there is room for improvement. Some of the major items in our TODO list are mentioned below:

- Solving the NT hostname problem described earlier.
- Automatically installing/configuring X and the sound card during Linux installation.
- A more ambitious item is to extend the system to install any two (or possibly more) operating systems (especially to support Linux and Windows 2000 dual-boot systems).

Source

The source and complete profiles to do different combinations of operating systems (NT and RedHat Linux) are available from the author by sending email to dualbootinfo@research.bell-labs.com.

Author Information

Rajeev Agrwala is working as a System Administrator for Bell-labs Research for the last four years. He is currently reponsible for planning, deploying, and maintaining Unix Systems and process automation. Reach him via US mail at Lucent Technologies; Room 2t-406; 600 Mountain Ave.; Murray hill, NJ 07974. Reach him via electroninc mail at rajeeva@lucent.com.

Rob Fulmer has been an Systems Administrator for Bell Labs Research for the last six years. He is currently responsible for maintaining and building the NT infrastructure and for NT process automation. Reach him via US mail at Lucent Technologies; Room 2t-412; 600 Mountain Ave.; Murray hill, NJ 07974. Reach him via electroninc mail at rjf@lucent.com.

Shaun Erickson has worked at Lucent Technologies (previously AT&T), for 12 years, the last 5 as a System Administrator for Bell Labs, in Murray Hill, NJ. He can be reached via U.S. Mail at 600-700 Mountain Ave., Room 2C-533, Murray Hill, NJ, 07974-0636. He can be reached electronically at ste@research.bell-labs.com.

References

[1] Rober Fulmer and Alex Levine (Lucent Technologies, Bell Labs), ''Autoinstall for NT: Complete NT Installation over the Network,'' Usenix, Large Installation System Administration of Windows NT conference, *1998 proceedings*. This paper provides the method to install NT over the network. We use the same technique to install NT in later part of our install.

[2] RedHat's kickstart user guide available at http://www.redhat.com/kickstart explains all the kickstart commands.

[3] Information on Norton Ghost can be found at http://www.symantec.com/sabu/ghost/indexB.html .