Automated Generic Operating System Installation and Maintenance

Joel D. Martin, jmartin@css.tayloru.edu
Compaq Computer Corporation
Aaron D. Brooks, abrooks@css.tayloru.edu
Taylor University CSS Department

Abstract

Microsoft Windows NT deployment and maintenance is one of the most time consuming tasks for systems administrators. The primary motivation for the JACAL project is to streamline this process by creating a set of free and open tools and guidelines for automating the installation and maintenance of one or more operating systems in large computing environments.[1]

It is well known that Windows NT workstations have subtle and potentially serious problems when they are duplicated from a disk image. These problems leave IT administrators with the task of manually setting up NT workstations or cobbling together a number of different automation methods which, in the end, still require the administrators to manually fill in the parts which the various automation tools leave out. Most large scale installation and maintenance tools do not provide solutions that solve the problem completely from boot to a completed build with minimal administrator intervention.

The JACAL project was developed at Taylor University's Computing and System Sciences Department. Our lab and classroom environment consists of 40 workstations that dual−boot Windows NT and Linux. We also have 5 dedicated Linux workstations, 12 dedicated NT workstations, and 7 faculty workstations. Each Windows NT setup has nearly 80 applications that are available to every user. Nearly 1 GB of application data is installed to the local hard drive and over 7 GB of application data is served to the workstations from the application server. Manual installation of these applications takes 25+ hours on a single machine. JACAL allows us to fully rebuild a workstation in under 2 hours with only a few minutes of administrator intervention.

1 As we have developed this paper the emphasis has gradually shifted. Most of the time spent in developing JACAL was spent getting around architectural and common practice problems with Windows NT. Likewise, this paper focuses on the work that we did on the Windows NT side of the JACAL project. A more appropriate title for this paper might be "Applying Linux/UNIX Tools and Methodologies to Aid in Windows NT Installation and Maintenance."

Requirements

Although our environment is not especially large in terms of the number of seats, it is very complex in terms of the number and variety of applications available from each workstation. Our requirements for the JACAL project include the following:

- JACAL should provide a quick and flexible way to rebuild all of our workstations with minimal administrator intervention. This system should be applicable to other environments.
- The tools used should be freely available so that JACAL can be used in organizations with small budgets.
- The tools used should be Open Source where possible to allow for third party customization and extension.
- JACAL should allow Windows NT applications to be installed so that most components of the application can be run from the application server.
- JACAL should provide a simple way to add and repair applications on the workstation.
- JACAL should support the installation of other operating systems.
- JACAL should be able to install multiple operating systems on the same workstation (multi−boot)
- Application images created with the JACAL system should not be order dependent. They should be installable in any order.
- JACAL should resolve the problem known as "DLL Hell". In other words, JACAL should provide a method for resolving shared file conflicts.

The Problem

According to Microsoft, Windows NT 4.0 should not be duplicated from a disk image because of the resulting security issues [SID]. While this is true, most of the security issues can be cleaned up by a SID (System ID) changer such as the one available from sysinternals.com [SYS]. The real problem with "ghosting" an NT workstation is that the NT setup process ties itself very tightly to the particular hardware configuration that it is installed on. This means that a "disk image" must be created for each hardware configuration on the network.

JACAL uses NT's built−in unattended setup process which uses an answer file[2] to automate the process [BELL]. The unattended solution is less than ideal because the process is much slower and less flexible than being able to perform the necessary steps directly from a JACAL script. However, this appears to be the only acceptable solution and the future does not look much brighter. The tools included with Windows 2000 appear to promote the same unattended methodology included in NT 4.0 [UNA1].

The lack of flexibility of disk duplication[3] is not acceptable for our diverse environment because we depend on the ability to swap similar but different components between our many different workstations. JACAL allows us to rebuild our NT workstations very quickly on new hardware configurations without sacrificing flexibility.

Even more serious than the problems with Windows NT setup is the installation and maintenance of NT applications. Gomberg, Evard, and Stacey in their paper [ANL] outline many of the problems related to large scale application installations on Windows NT. These problems include:

- NT applications are more complex and monolithic than typical UNIX applications.
- NT applications are tightly integrated with the OS.
- NT software is oriented to a single−machine and single−user environment (non−networked).
- NT installation programs are GUI based making software installation more difficult to extend and automate.
- Each software installation tunes itself to the particular NT setup, registry and hardware configuration.

DLL Hell

One of the largest problems that JACAL solves is what Microsoft refers to as "DLL Hell"[HELL]. The problem stems from the monolithic nature of Windows applications and the tight integration between the applications and the OS. New applications tend to carry shared file "baggage"[4] that can easily replace shared system files. This can potentially break other applications that depend on certain versions of those shared files.

2 unattend.txt

3 Tools that accomplish this include Ghost and ImageCast

4 This baggage consists of shared files that the software company did not create but included with the application installation. Typically these shared files originate at Microsoft.

UNIX systems also have administrative issues in relation to shared files. However, typical UNIX applications do not actually carry this shared file baggage and therefore do not have the potential to break other applications. Instead of breaking other applications, the new application will simply not work with the current shared files. These problems can usually be resolved with symbolic links.

The Options

Microsoft's System Management Server

The problem of large automated installs has a long and rich history. Microsoft has many pages of online documentation relating to the automation of large NT installations. Microsoft suggests using System Management Server (SMS) which provides remote administration of NT machines including installation of software. We do not consider SMS a viable option for several reasons. SMS is prohibitively expensive for our environment [SMS]. SMS does not provide the ability to build NT (or other operating systems) from scratch. SMS and the other solutions that we looked at do not perform the shared file conflict resolution that JACAL performs. In addition, JACAL now has the ability to do large−scale remote maintenance of NT workstations with a combination of our "wintel.pl" script and the NT Telnet Server "NDTelnet" [NDT]. This is described in more depth in the Detailed System Description.

Bell Lab's AutoInstall for NT

In a paper by Fulmer and Levine of Bell Labs[BELL], they describe a system that is somewhat similar to our JACAL project. This paper was written in 1998 and their system has probably developed considerably since that time. There are many differences between JACAL and the system that they described. The bootdisk system at Bell labs used a MS−DOS bootdisk to initiate the setup process. One of the challenges that they faced was getting all the necessary tools to fit on a single floppy. We have avoided this problem by having only a Linux kernel on our bootdisk which mounts its root filesystem via NFS. Because of this our tools and scripts are not limited by the size of a floppy disk. We are also able to use the JACAL bootdisk to support the installation of other operating systems.

UBS AG's State Driven Solution

One of the more promising solutions was a state driven system recently documented by Martin Sjolin [STATE]. This state driven system stores application configuration information in a central repository which is queried by a client side component whenever an update or installation is performed. This state driven system corresponds to the second phase of our JACAL

system. It assumes that the client workstations have a complete NT setup. Their state driven solution quite possibly more flexible than our system and provides more administrative maintenance tools. However, it is unclear whether shared file conflict resolution could be easily accomplished using this system. In the future we may incorporate some of the state driven features of the system into the JACAL project.

The Solution

To meet our requirements for JACAL, we developed an internal system at Taylor University's Computing and System Sciences Department which allows us to quickly rebuild, from scratch, every one of our lab and classroom machines with both Linux and Windows NT (dual−boot) and a full set of applications on both.

Many of the past solutions focus on using the installation tools that are preferred for each application such as InstallShield, a custom setup program for that application or the new Windows Installer. However, in our environment, this is unacceptable because we need more control over the configuration of the target machines. We have approximately 80 applications installed on each workstation. Without our conflict resolution system, the sheer volume of applications installed resulted in constant conflicts as new applications replaced shared files.

We have developed a system based on Microsoft's "sysdiff" program[5] that adds the capability to resolve shared file conflicts. The addition of this functionality has made the program suitable for nearly all of our applications. This conflict resolution capability is one of the biggest differences between our solution and other solutions that have been implemented in the past.

NT Service Packs[6] are the main exception to our conflict resolution system. Microsoft service releases make changes that are too dynamic and far reaching to be properly replicated by a system difference tool such as sysdiff.

JACAL Overview

There are two phases to the JACAL system: the JACAL loader phase and the individual OS and applications install phase. We currently implement the **first** phase as a single bootable floppy with a Linux kernel that NFS[7] mounts its filesystem. The JACAL loader scripts are then executed. At this point the administrator is asked for the machine name, and is given a choice of several different standard machine setups. The options define partition sizes, applications to install, and whether to make the machine multi−boot or dedicated, or to simply refresh the existing machine setup.

Setting up the local Linux partition(s) is a simple matter of using Andrew Tridgell's popular rsync tool [RSYNC] to mirror our existing Linux filesystem. The Linux configuration is then localized completing the Linux installation. The loader phase also copies the Windows NT install files to the workstation in preparation for the second phase.

Since NT cannot be fully setup from the loader phase, the **second** phase is specifically related to Windows NT. This phase could easily be used to install any other OS that has "ghosting" difficulties and has the ability to perform a scripted installation. The second phase begins when the loader phase reboots the machine. The Windows NT unattended setup is executed at this point.

When Windows NT has completed its setup, a string of Perls scripts are executed that install applications, Service Packs, and drivers that cannot automatically be specified in the Windows NT unattended setup answer file.

The unattended installation capability of Windows and its service packs are built into the software. We do not install regular applications in this "unattended" method. Instead, we use Microsoft's sysdiff utility to capture the changes that an application makes to a bare system. We then do some semi−automated processing of this "diff" output. First, we change the registry and Windows shortcut paths to allow the application to run from a network server. Then we process the captured directory structure and resolve shared file conflicts, such as DLLs, using a script that allows us to symlink[8] these files to the chosen version of the file. When the application is installed the symlinks are dereferenced and the chosen shared file is used.

At the end of the installation process the administrator is only required to change the local administrative password to complete the workstation setup. JACAL also provides post−setup maintenance tools[9].

---

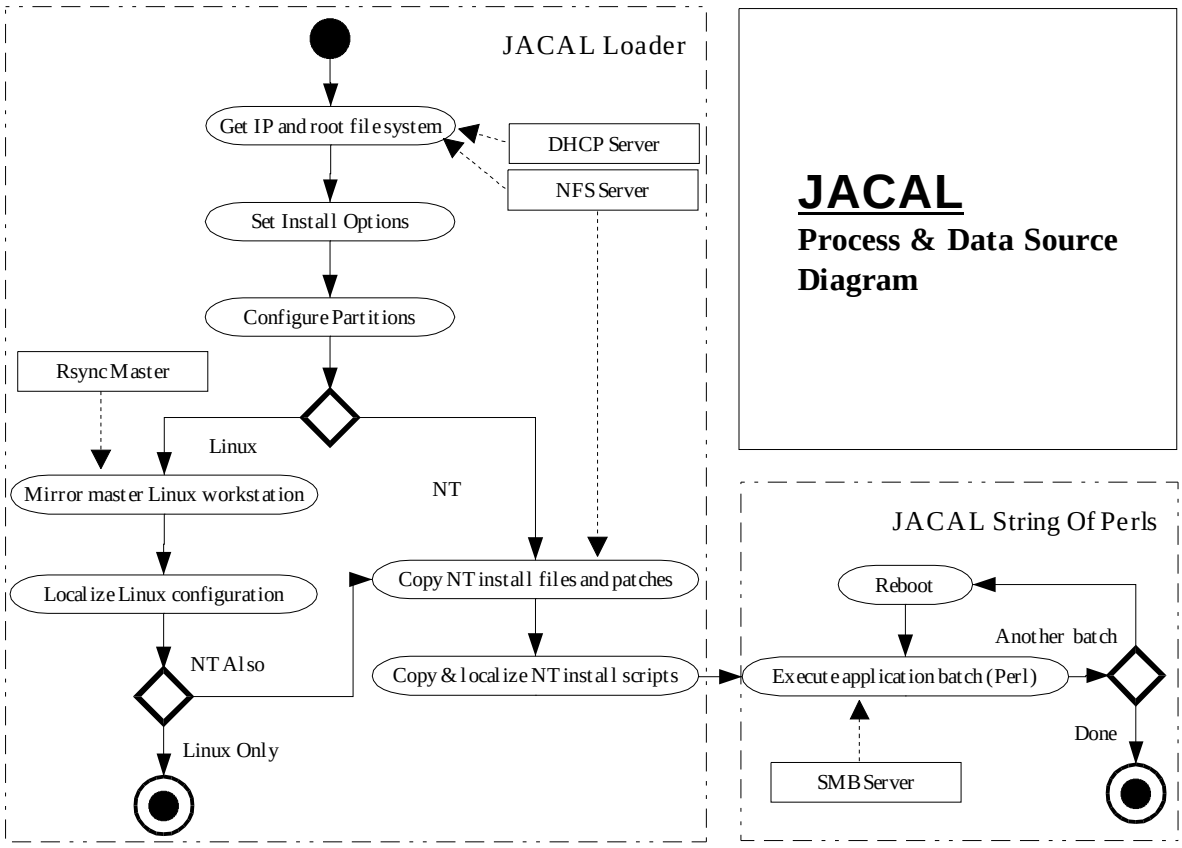
5 This tool is now called "Discover" under Windows 2000

6 We consider Microsoft Internet Explorer to be equivalent to a Windows Service Pack

7 Network File System common on UN*X networks

8 UN*X platforms allow pseudo−files, called symlinks, which point to the original file. This referencing is transparent unlike Microsoft ".lnk" files

9 These tools include the Perl based scripts "wintel.pl" and "reghack.pl"



Detailed System Description

(See above diagram)

JACAL Loader

1. (25 Seconds) Boots Linux kernel off of floppy disk
   - The 3½" floppy is ext2fs formatted with only a Linux kernel booted by lilo
   - The kernel is compiled with the following
     - NFS root
     - NFS client
     - Kernel auto−configuration
     - DHCP/Bootp[10] client
     - As many NIC[11] cards compiled in as the administrator would like[12]
     - This disk is all that is needed to start the setup on the client end
2. (20 Seconds) Runs Linux startup environment
   - Gets IP address from DHCP
   - Mounts linuxroot export from NFS Server
   - Runs modified init scripts
3. (0−5 Seconds) Automatically detects or queries administrator for machine identity
   - Can be fully automated
     - Database lookup of MAC ID from network card or similar hardware ID (CPUID)
     - If DHCP is setup to assign the same address to the same machine, DNS[13] can be used to figure out machine name
   - Administrator can be manually queried for
     - Machine name
     - Configuration (select single menu item)
   - Target drive is partitioned if need be
     - Partitioning can be relative based on hard drive size
     - Swap partitions can be created dynamically based on the amount of RAM
4. (25 Minutes *optional*) Linux Install[14]
   - Format ext2fs partition (if needed)
   - Rsync filesystem from server
   - Modify configuration files to reflect machine identity
5. (5 Minutes *optional*) NT Partition Preparation
   - Format FAT16 partition (Convert to NTFS later if desired.)
   - Copy NT installation files

10 Dynamic Host Configuration Protocol/Boot Protocol
11 Network Interface Card
12 In our case, all ethernet cards
13 Domain Name System
14 25 minutes is based on a "from scratch" Linux installation. If the partition already has linux rsync will often take less than 5 minutes

- Copy NT installation patches
- Parse and customize NT installation scripts and configuration files
- Run spp client to reset domain machine account password on the SaMBa server

JACAL String Of Perls (Windows NT Installation)

In order to run this step the JACAL Loader (above) must have prepared the NT target partition. Quite possibly the most frustrating problem for an administrator is how often Windows NT 4.0 requires a reboot of the system. Events like changing the cache directory for Internet Explorer 5 to another directory requires a subsequent shutdown and reboot.

When automating the NT installation process, it became apparent that the first thing we would have to [conquer] would be constant administrator intervention to start the next phase of the install after a reboot.

The "String of Perls" in this phase of the installation refers to a customizable reboot management system built on RunOnce[15] registry keys, batch scripts and Perl scripts. (Initially there were only Perl scripts, now most of the .pl's are .cmd's which are faster, lighter and still do the job.) These scripts insert the RunOnce key for the following section of the install and then run the next step in the installation process which requires a reboot of the machine upon completion.

The NT SOP install takes 45 minutes on a single machine.

1. (5 Minutes) NT unattended install
2. (5 Minutes) NT Service Pack 6
3. (5 Minutes) Internet Explorer 5
4. (25 Minutes) Application installations
5. (2 Minutes) NT SMP kernel installation
6. (2 Minutes) Display adapter installation
7. (1 Minute) Cleanup scripts

The last step is only included here for accuracy. At the time of this writing the cleanup scripts run after a reboot which is entirely unnecessary. This step can be merged with the preceding step. Cleanup includes removing installation files and directories from the target drive and changing the password for the "Administrator" account. At the current time the administrator password must be set manually. This is the only human intervention required after the initial 45 second floppy disk boot.

The total sum of human interaction is listed below:

1. Place floppy disk in drive and power up machine
2. Remove floppy disk after floppy access completes
3. If install process is not set up to automatically detect machine identity, type in machine name and select machine configuration from numbered menu.
4. Play Quake Arena[16] for 25 to 75 minutes depending on install[17]
5. If NT installation, change administrator password

Application Image Creation

The process used to create an application image is described in detail on the JACAL website. The basic steps that we use to create NT application install images and where the steps are performed are as follows:

1. (NT) Snapshot a clean system using "sysdiff.exe /snap"
2. (NT) Install the application locally
3. (NT) Capture changes made by the install using "sysdiff.exe /diff"
4. (NT) Extract the "difference" files to the SMB install share using "sysdiff.exe /inf /m"[18]
5. (NT) Rename the "difference" files to use long filenames using "parse−rename.pl"
6. (NT) Repath Windows shortcuts (*.lnk) using "repath−shortcuts.pl"
7. (UNIX) Repath the registry settings file (.inf) and Windows .ini files using regular expressions
8. (UNIX) Create the SMB execution share and move the main application directory to the share
9. (UNIX) Resolve shared file conflicts using "mvdlls.pl" and "filelink.pl"
10. (UNIX) Zip up install files for distribution
11. (NT) Apply an application

Step 6, 7 and 8 may be eliminated if the entire application is going to be installed locally to every workstation. There are also some rare and intelligent applications that can be installed directly to a UNC[19] share. This makes the application image creation process much simpler. To perform this type of installation, step 8 is done first and steps 6 and 7 can be eliminated. When the application is setup it should be installed to the execution UNC share.

Application DLL Resolution

The most significant contribution of the JACAL project to NT software installation is the shared file

15 HKEY_LOCAL_MACHINE\SOFTWARE\...
...Microsoft\Windows\CurrentVersion\RunOnce

16 Or Solitare if you are concerned about network bandwidth

17 Up to 120 minutes if many (25+) machines are running in parallel (These times are based on builds run on our 100Mb ethernet network)

18 According to Microsoft documentation "/m" is the "mandatory option"

19 Universal Naming Convention

conflict resolution system. In Taylor University's computer science labs we run approximately 80 applications on all of our NT workstations. Before we implemented conflicting file resolution we had significant problems with over 2600 files in conflict. The conflict resolving software is composed of two Perl scripts, "mvdlls.pl" and "filelink.pl."

These two scripts detect files which are in common between applications and resolve these conflicts. These scripts are run server side as they make UN*X[20] symlinks which are transparent to the SMB protocol. For all practical purposes, these tools necessitate running SaMBa [21]on a UN*X server.

Applications are stored in separate directories. Each of these directories is the result of a "sysdiff /inf" command. The sysdiff program creates "C" and "$$" directories which represent "C:\" and "C:\winnt\"[22]. Each of these special directories beneath the application directories are recursively searched by "mvdlls.pl" for files that conflict with other applications. When a conflicting file is found,[23] the file is moved to a common _shared_ directory structure. When each conflicting file is placed in the _shared_ structure it is placed in a subdirectory path relative to the C:\ drive. The file is also renamed so that any internal file version[24] is added as a component of the file name as well as the application that the file came from. This convention prevents files from being overwritten by colliding versions in other applications.

Once all conflicts have been detected by "mvdlls.pl" and moved out into the common _shared_ directory, "filelink.pl" is used to select which version of each conflicting file will be part of the final install.

The actual file resolution is achieved by creating a symlink with the original file name which points to the desired version of the file which the administrator has determined should be deployed.

Most file resolution is fairly simple. The largest portion of conflicting files on our site are fonts shared between different Microsoft publishing programs. The key resolutions reconcile different DLL, EXE and OCX versions. Usually in these cases, the most recent version of the executable or library is the best choice.

20 UN*X commonly refers to all variations of Unix
21 SaMBa is a UN*X implementation of Microsoft's Service Message Block (SMB) protocol
22 Actually the directory is the environment variable %SystemRoot%, typically "C:\winnt\"
23 Not just DLLs as the name "mvdlls.pl" suggests
24 File versions are grabbed from internal information from DLLs, EXEs and OCXs

However, this is not always the case.

An example of the resolution process between Visual Studio 97 and Windows Scripting Host would look like this:

Conflicting files:
VisualStudio/$$/system32/scrrun.dll
WSH/$$/system32/scrrun.dll

Moved to files in the _shared_ directory:
shared/$$/system32/scrrun.dll.4.0.0.2926.VisualStudio
shared/$$/system32/scrrun.dll.3.1.0.2430.WSH

Since both files are from Microsoft we create a symlink:
shared/$$/system32/scrrun.dll
Which points to the most recent version:
shared/$$/system32/scrrun.dll.4.0.0.2926.VisualStudio

"filelink.pl" currently has two modes. The first prompts the administrator for file conflicts which have not been resolved. The second mode prompts the administrator to resolve all conflicting files. In the future we will be adding the capability to provide a list of applications for "filelink.pl" to resolve rather than redoing all file resolutions.

Application Installation

The applications or portions of applications that are targeted for the local workstation are stored in separate directories. Each directory is structured like the flat file inf generated by sysdiff. There is an ".inf" file which contains all of the registry and ini modifications. There are also the standard "C" and "$$" directories. However, within these directories are zip files containing the subdirectories and files for the applications. Compressing the applications in this manner reduces the copy time and network bandwidth utilization while adding checksums to the file transfer process.

The installation program copies down the zip file from each of the inf directories. After the archive is on the local disk, files are extracted to their final locations. If the file is read only (i.e. already in use by another program) it is detected from the output of the unzipping program. The file is then placed in the C:\temp directory and "MOVEEX.EXE" is run to place the appropriate registry key into the kernel so NT will move the file on the next reboot. Since applications already have their conflicting files resolved, multiple calls to "MOVEEX.EXE" do not cause any problems.

The applications are all installed in a serial fashion without rebooting. Although at our site we install all

applications, it would be very easy to modify the installation serializing script, "zipinst.pl," to check a file that lists applications that should be installed on the workstation. This method results in an incredibly fast and flexible installation of applications with high reliability and complete conflicting file resolution.

Workstation Maintenance

For performing maintenance of workstations that have already been rebuilt we created a Perl script, "wintel.pl", that allows us to execute functions on a large number of workstations in parallel. This script is built to take advantage of Nicolas Deschatrettes's NT Telnet Server. We use it primarily as a way of applying applications and applications fixes in batch. The "wintel.pl" script takes the following parameters: a file that has the list of commands to run on the target machines and a file that has a list of the machines to execute the commands on.

Design Philosophy

Eliminate administrator intervention

Perhaps one of the primary forces driving the computing and information revolution is the belief that repetitive actions performed by people should be handled by Information Technology. It follows that administrators should not be required to manually build multiple workstations that are nearly identical. JACAL is our attempt to eliminate repetition from standard NT and Linux workstation deployment.

Use the best tools for the job

Linux is optimal for manipulating and moving file data around in a heterogeneous environment. Linux is also good at automatically detecting hardware configurations. Auto−detection of hardware is one of the issues that the Bell Labs team [BELL Section 7.4] had trouble performing under NT with their AutoInstall. Linux allows us to easily retrieve and parse information about the hardware. After hardware information is gathered it is used to modify the course of the install process. We currently only use this to detect whether NT should support multiple processors and to detect which video card drivers to use. This auto−detection system could be extended to account for much greater variability in hardware configurations.

Linux does not have tools available to install NT, so we use NT's "SETUP.EXE" for this stage. We use Perl as our primary scripting language, instead of NT batch scripting, because it is very flexible and cross−platform. Eventually we plan to eliminate all the UNIX shell scripts that are still part of JACAL and replace them with Perl scripts.

Use as few development tools as possible

The JACAL project attempts to minimize complexity by limiting the number of development tools. This philosophy keeps JACAL slim and enhances the project's portability, extensibility and maintainability. There is a trade off between using the best tools for the job and using as few tools as possible. We have leaned toward using the best tool but have intentionally designed the system so that the different tools can be replaced or eliminated with a corresponding loss of flexibility or functionality. JACAL currently relies on the following development tools: Perl on both Windows and Linux, UNIX shell scripting, NT batch language, Microsoft's sysdiff utility, Microsoft's ScriptIt tool, limited C.

In keeping with our philosophy of using as few development tools as possible we have designed JACAL without any dependence on NT/2000 Server. In some environments, including ours, Windows NT is only used for desktop workstations. This design philosophy will allow JACAL to be used on other networks like ours and also on networks that contain NT Servers.

Rely on as few network services as possible

Our setup of JACAL requires the following network services: DHCP, NFS, and SMB, and a Linux rsync server. Each one of these services could be eliminated but at the cost of flexibility. DHCP could be eliminated by having a fixed IP address for each build disk. This would require a system to keep track of and give each boot disk a unique IP address.

NFS could be eliminated by storing the JACAL loader and the NT install files on a larger boot media such as a CD−ROM or on a permanent partition on each machine's hard drive. This would have the added benefit of increased build speed but changes to the loader or the install files would be more difficult and time consuming. The NT install files could also be moved from the NFS server to the SMB server.

The application files and install images served from the SMB server could likewise be moved to a local media but the result would be that applications would have to be installed to run locally. This would also limit changes to the applications and would limit the number and size of the applications that can be installed.

Use tools that are flexible and can be automated

We use the criteria of flexibility and automation in evaluating tools for inclusion in the JACAL system.

As an example, one of the major tools that we choose to use for JACAL was Microsoft's sysdiff utility. Although sysdiff itself is not very flexible, it uses a flat file output format that has allowed us to extend the sysdiff system to fit our needs very nicely.

One note of concern in this area is that Microsoft has replaced the sysdiff system in Windows NT 4.0 with VERITAS WinINSTALL LE included in Windows 2000 [WINST]. Sysdiff allowed us to directly access and manipulate the information that was captured during the snapshot process. This has allowed us to eliminate shared file conflicts and to be able to properly maintain and change the application image data as our environment changes. Hopefully we will be able to find a similar method of extracting information from Microsoft's .MSI install files which are created with VERITAS WinINSTALL LE.

Future Goals

Eliminate the JACAL Bootdisk

The most ambitious goal of the project is to move away from the boot disk based model. The JACAL loader would be placed on a small partition on each machine. At the reboot the script would check the network to see if that machine's rebuild flag had been set and then proceed to either re−install the entire setup (specified in a network configuration file) or simply continue to boot the machine. This would increase automation and allow the systems to be locked down by turning off floppy disk boot capability. A bootdisk would only be necessary on a new machine or to update the JACAL boot partition.

Server Controlled Rebuilds

A related goal is a server based command would give the systems administrator power to schedule automatic machine builds. The syntax would be something like:

jacal *date−time* <*machinename*>...

At the scheduled time the flags for each of the machines in the list would be set for a rebuild and the machines would then be remotely rebooted.

The BeeHive Project

One of the headaches that we have had to deal with in creating application images is that the user registry hive settings must be entered into the global login script. This complicates the application image creation process and slows down user logins. We have started an independent project named BeeHive to create a cross−platform NT/2000 registry editor. This will allow us to edit, repair and optimize roaming profiles from our Linux user space server [BEE].

Rewrite Sysdiff

Another future goal is to write our own system difference tool so that we can simplify the application image creation process and also perform conflict resolution of registry entries and capture incremental changes automatically. We may also write a component which allows us to extract information from Microsoft's new ".MSI" install file format.

FCP (File Cast Protocol)

One project that our team has embarked on is not directly part of the JACAL project but will complement it greatly. This task is to create a cross−platform multi−cast FTP style system so that a large number of workstations can be rebuilt simultaneously with minimal network traffic.

Conclusions

The art of large, automated installation is in a state of rapid change. Both the UNIX and Windows world seem to be converging in the way that large installation and maintenance is carried out. Many UNIX applications are becoming more dependent on local machine resources and as such have become more complicated in their installation process. Microsoft is moving towards a more modularized and protected installation scheme with the Windows Installer under Windows 2000. Both of these trends will certainly continue as UNIX moves toward the desktop and Windows moves toward the server.

JACAL was developed in a very heterogeneous environment and has picked the best parts of each platform in an attempt to create a more manageable network. JACAL is continuing to develop. One of the upcoming challenges will be coexisting with the new Windows Installer system. Another area of growth will be in further development of multi−boot functionality. This is made more important with the growth of viable x86 based operating systems such as BeOS, the BSD series, Solaris, Linux, etc.

The components of JACAL have been released as an Open Source project in order to spur continued growth and to give back to the community which made JACAL possible. Hopefully, JACAL will be a helpful tool for network administrators in many different environments.

JACAL has a permanent home thanks to VA Linux's SourceForge site. At this site you can find the latest documentation and also download the latest distribution of JACAL:

http://jacal.sourceforge.net

Works Cited:

[ANL] A Comparison of Large−Scale Software Installation Methods on NT and UNIX, Michail Gomberg, Remy Evard and Craig Stacey, Mathematics and Computer Science Division, Argonne National Laboratory
- http://www−fp.mcs.anl.gov/~stace/Papers/NTLisa1998/ntapps.html
- http://www−fp.mcs.anl.gov/~stace/Papers/NTLisa1998/ (LISA Presentation)

[BELL] AutoInstall for NT: Complete NT Installation Over the Network, Robert Fulmer and Alex Levine, Lucent Technologies, Bell Labs
- http://www.usenix.org/publications/library/proceedings/lisa−nt98/fulmer.html

[STATE] State Driven Software Installation for Windows NT, Martin Sjolin, Warburg Dillon Read
- http://www.usenix.org/events/lisa−nt99/sjolin.html

[HELL] The End of DLL Hell, Rick Anderson. MSDN Library, January 2000
- http://msdn.microsoft.com/library/techart/dlldanger1.htm

[SID] Windows NT Duplication Problems
- http://support.microsoft.com/support/kb/articles/q162/0/01.asp
- http://support.microsoft.com/support/kb/articles/Q183/2/53.asp

[SYS] Sysinternals − Advanced utilities, technical information and source code related to Windows NT/2K
- http://www.sysinternals.com/

[UNA1] Windows 2000 Professional Automated Deployment Options: An Introduction
- http://www.microsoft.com/windows2000/library/planning/client/autodeploy.asp

[SMS] Microsoft Announces Availability of Systems Management Server 2.0
- http://www.microsoft.com/presspass/press/1999/feb99/smspr.asp

[NDT] NT Telnet Server (NDTelnet)
- http://hem.passagen.se/deschatr/ndtelnet.htm

[WINST] Windows Installer
- http://www.microsoft.com/windows2000/library/howitworks/management/installer.asp

[WINST] Step−by−Step Guide to Creating Windows Installer Packages
- http://www.microsoft.com/windows2000/library/planning/management/veritas.asp

[BEE] BeeHive − Cross−platform registry editor
- http://sourceforge.net/project/?group_id=1987

[RSYNC] rsync is an open sourceutility that provides fast incremental file transfer
- http://rsync.samba.org/

References:

MS Windows NT Workstation Deployment Guide − Automating Windows NT Setup
- http://www.microsoft.com/TechNet/winnt/winntas/technote/implemntintegra/gdautset.asp

An Unattended Windows NT Workstation Deployment
- http://www.microsoft.com/TechNet/winnt/ntwrkstn/technote/ntinstal.asp

Windows NT 4.0 FAQs: Deployment and Unattended Setup Questions
- http://www.microsoft.com/TechNet/winnt/winntas/technote/troubleshooting/topntqa1.asp

Easier Windows NT Workstation 4.0 Deployment with Disk Image Copying and the MS System Preparation Tool
- http://www.microsoft.com/TechNet/winnt/ntwrkstn/prodfact/sysprep.asp

Chapter 11 − Windows NT Workstation Unattended Modular Build
- http://www.microsoft.com/TechNet/winnt/winntas/technote/implemntintegra/manntnet/ntnfch11.asp

Automating Windows NT Setup Deployment Guide Supplement (Sysdiff)
- http://www.microsoft.com/TechNet/winnt/winntas/technote/implemntintegra/advsysdf.asp

Microsoft's IE Administration Kit
- http://www.microsoft.com/windows/ieak/

Microsoft's Office 2000 Resource Kit
- http://www.microsoft.com/office/ork/2000/

ActiveState Perl win32 FAQ
- http://www.activestate.com/ActivePerl/docs/perlwin32/perlwin32faq.html

The MS ScriptIt Utility
- http://www.microsoft.com/TechNet/winnt/Winntas/tools/scriptit.asp

Top Ten Windows NT Support Issues
- http://www.microsoft.com/TechNet/maintain/topsup.asp

Customizing the Windows NT 4.0 Upgrade Process
- http://www.microsoft.com/TechNet/winnt/winntas/technote/planning/nt4cusup.asp