Maintaining Windows in an Open Multi-User Laboratory Environment

K. Scott Rowe – New Mexico Institute of Mining and Technology

July 27, 1999

Abstract

Over the past couple of years, Windows95 usage has steadily increased. This has posed tough administration problems, especially when used in an open multi-user laboratory environment such as in a college computer center. Windows' inherent problems handling multiple users, coupled with its lack of filesystem security and large installation support have forced many system administrators to develop their own methods of keeping Windows machines up and reliable.

Gutinteg is a system that solves many of these problems by utilizing the UNIX operating system to pull a Windows filesystem to a client and configure it as necessary. It also manages individual packages of third party software, and installs only the software needed by certain machines.

Background

In 1996 the New Mexico Tech Computer Center (TCC) adopted *Gutinteg* to maintain its UNIX clients. *Gutinteg* was originally designed and written by Michael Fisk who worked for the TCC at the time. It was simple to use, and could be run by our student Help Desk employees. Over the years, it has been upgraded and retooled to support more features and more operating systems. It now supports SunOS, Solaris, Linux and Windows95.

In the summer of 1997, the TCC began moving from a Windows3.11/Novell environment to a Windows95/Samba environment. The TCC had been using a program called dos-integ to maintain its DOS clients. This program simply copied from a networked drive, the needed DOS-6.22 files and enough networking files to access the Windows3.11 installation on the Novell server. It would essentially reinstall a DOS machine in a few minutes with a minimum of intervention. We wanted to maintain this level of simple installation and maintenance when we moved to Windows95. *Gutinteg* seemed the logical answer.

What is Gutinteg

Concepts

The idea is that the operating system and software running on a computer is merely a collection of files, and that these files can be broken down and separated into *packages* of files that are independent of each other. These packages, with some minor adjustments, can then be copied to each machine, or to a collection of machines producing a lab of computers all with the same software and operating system setup. This requires a machine database, *Machdb*[2], that can keep track of the packages installed and specific configurations required by each machine. Finally, UNIX is used to provide a bootstrap-like environment where files, especially Windows system level files, can be written to the computer without intervention of the Windows operating system.

How It Works

All computers at the TCC that run Windows95, are dual-boot machines capable of running either Windows95 or Linux. It is this Linux installation that provides the bootstrapping needed to write all the files that make up a Windows95 machine. When a computer needs to have its Windows95 partition completely re-installed, or if the Windows95 partition is out of sync with what it should be, the machine is rebooted into Linux and *Gutinteg* is run.

What *Gutinteg* does is use *Machdb* to look up what packages and specific configurations the computer needs. Next it formats the Windows95 partition with *dosemu*, which is an application that enables the Linux operating system to run many DOS programs[3]. Finally, it copies the required files using *packagelink*[2].

Packages

A package is a directory structure consisting of the files needed by a certain piece of software, along with the registry entries needed to utilize that software. The directory structure is built relative to `C:\` on the installed machine. For example; a package for a certain ethernet card would contain installation files in `C:\Windows\Inf`, drivers in `C:\Windows\System`, and a registry file in `C:\Tmp` for folding into the system registry after installation.

There are two programs used to create a Windows package. RegView is a Registry editing and System restoring utility[1] much like Windows regedit, but it can also track changes in the filesystem and INI files. *RegView* is used to save the registry, INI file, and filesystem information, then once the software package is installed, RegView is run again to record the differences in the registry, INI files and filesystem made by the software install procedure. However, RegView outputs a difference file instead of a normal registry file. Because of this, a program called *Wholewheat* has been developed at the TCC. *Wholewheat* is a Perl script that converts RegView output into a registry file that Windows regedit can import. This registry file is then put in a Tmp directory in the package so that Windows can import it using regedit when it first boots.

Packages are also made for the operating system and the architecture. Windows95 is itself a package, installed with the minimal amount of options and additional software so that it can be used on many different architectures. There is also a registry file created for each architecture (486, 586, 686, etc.). The registry for a particular architecture, once created, is maintained by hand with additions for new machines of the same class. For example, most of the computers at the TCC are Pentium class machines, but there are several different types of motherboard and processors in that class. Each new type of machine usually has a few differences that need to be folded into the architecture class registry. These usually consist of different address schemes for the PCI bus or different entries for serial ports. Since Windows only checks detected hardware against what is in the registry, it is not a problem to have multiple entries for similar devices. Windows will simply use the entry for the device it detected.

Machdb

Machdb is a program which retrieves information from a database for a particular computer. It is keyed by hostname and contains elements such as IP address, nameserver, as well as a list of software packages to be installed. The database is maintained in an object oriented fashion allowing classes of machines to be constructed. One such class, for example, could be for a room of similar machines:

```
speare9:  dot4 dec21140
   PRINTER=speare9
```

Here, the class `speare9` inherits the classes `dot4` and `dec21140`, and also sets the variable `PRINTER` to `speare9` which in this case is the name of the network printer in the speare9 lab. The `dot4` class:

```
dot4:
   DOMAIN=nmt.edu
   GATEWAY=129.138.4.1
   NAMESERVER=129.138.4.216
```

sets variables used in configuring the network setup of the machine and then `dec21140` class:

```
dec21140:
   PACKAGES+=dec21140
```

adds the package `dec21140`, which is a package for the DEC 21140 Tulip chipset ethernet card, to the list of software packages to be installed. All these variables will be used later when copying files to the disk. Finally, there would be an entry for the name of the machine, such as `jupiter`, that inherits the `speare9` class.

```
jupiter:  speare9
   IP=129.138.4.110
```

Packagelink

Packagelink is a Perl program originally designed to build networked filesystems for UNIX servers from a list of software packages. It has since been expanded to creating filesystems on local workstations as well. It uses a "compare-before-copy" method, if a file currently on the disk is just like the file in the image, there's no need to copy. Packagelink also has the ability to preprocess files, replacing certain tokens with variables defined in the *Machdb*. A `.plprocess` extension is used to denote files that need preprocessing. This is most useful in the registry file for a package where tokens such as `##DOMAIN##` are replaced with the value of the `DOMAIN` variable defined in the *Machdb* database.

For example, the registry file for a network card package might contain entries like this...

```
   "Domain"="##DOMAIN##"
   "DefaultGateway"="##GATEWAY##"
   "IPAddress"="##IP##"
   "NameServer"="##NAMESERVER##"
```

Gutinteg

Gutinteg is the software that ties everything into one, easy to use, command. It first looks up all the necessary information about the to be installed client from the *Machdb* database. It then runs *dosemu* to format the Windows95 partition with system files (command.com, msdos.sys, io.sys). Next, it calls *Packagelink* with the needed software packages. In this case, Windows95 is itself a package, it just happens to be first in the order in which the packages are written to the disk. This way, a base operating system is layed down, followed by additional software packages and system configurations.

Once all the packages have been installed to the local workstation, *Gutinteg* runs LILO, a boot loading program for Linux, in such a way that when the computer reboots, it will boot directly into Windows. The default `autoexec.bat` runs a BAT file called `bootup.bat` that loads a specific system registry depending on the architecture (486, 586, 686, etc.), then loads the registry files in `C:\Tmp`. When this process has finished, the `autoexec.bat` removes `bootup.bat` so that it does not run again until *Gutinteg* is run again. Here is the text for `bootup.bat`:

```
@echo off
cd \Tmp
regedit /L:\windows\system.dat /C ##CPU##.reg
for %%x in (*.reg) do regedit %%x
```

Down The Road

There are still many improvements to be made to *Gutinteg*. The current method of building the registry the first time the machine boots into Windows takes upwards of five minutes. This is time that users could be doing their work on the machine rather then waiting for the registry to finish building so they can log in. Solutions to this problem could be to use *dosemu* to build the registry. Another could be to use the *Regutils*[4], maintained by Michael Rendell, to build the Windows registry. Both solutions build the registry before booting into Windows, thus removing the wait that users see the first time a machine boots into Windows.

Another improvement is a companion program called *dos-cache* also developed at the TCC. This is a very simple script that makes an image of the Windows partition and stores it on the Linux partition of the same machine. The *dos-cache* program is then run nightly, or as needed, to rebuild the Windows partition. Since the image is kept locally on each machine, there is no network traffic and the procedure is much faster then running *Gutinteg*. Also, since the image is kept on the Linux partition, it is protected from malicious users because Linux supports true filesystem security.

A Final improvement is a program currently being developed at the TCC that would combine the features of *RegView* and *Wholewheat*. This would greatly simplify the package making process, as everything needed could be generated and compiled in one environment with little intervention required.

Conclusions

Gutinteg has proven very useful for installing and maintaining UNIX machines, and has also exposed many administration problems in Windows. Most of these problems have been overcome, but there is still room for improvement. It does, however, show that a large installation of Windows machines can be maintained with freely available solutions.

Availability

Gutinteg, *Packagelink*, and *Machdb* are freely available programs, but are not written with portability in mind. It would be quite possible for an experienced systems administrator to adapt them to his/her system, and the author is willing to offer assistance in such an endeavor.

Acknowledgements

The author is grateful to Michael Fish for envisioning and developing *Gutinteg*, Jason Trowbridge for his continued support of *Wholewheat* and insight in to Windows95, and to Mike Topliff for supporting such experimental projects.

Author Information

K. Scott Rowe was involved with the original *Gutinteg* project at New Mexico Institute of Mining and Technology Computer Center, and retooled it to work with Windows95. He has since graduated from that institution and now is the Manager of Engineering Systems at the TCC. He continues to maintain and improve *Gutinteg*. He can be reached via email at *kscott@mailhost.nmt.edu* or by postal mail at *NMIMT, Speare Box 2, Socorro NM 87801*.

References

[1] Vincent Chiu. RegView for Windows 95/98/NT. *http://www.xnet.com/~vchiu/*, 1997.

[2] Michael Fisk. “Automating the Administration of Heterogeneous LANs”. In *USENIX LISA X Proceedings*, 1996.

[3] Hans Lermen. Dosemu. *http://www.dosemu.org/*, 1999.

[4] Michael Rendel. Regutils - win9x registry & ini file manipulation tools for unix. *http://www.cs.mun.ca/~michael/regutils/*, 1998.