Managing Computer Science Laboratories Using Open Software

John E. Howland
Department of Computer Science
Trinity University
715 Stadium Drive
San Antonio, Texas 78212-7200
Voice: (210) 999-7364
Fax: (210) 999-7477
E-mail: jhowland@Trinity.Edu
Web: http://www.cs.trinity.edu/~jhowland/

November 26, 2002

Abstract

An approach to installing and maintaining computer science department laboratory systems using open software is presented. Issues such as system security, updating and maintenance are addressed. Some details, such as programs and systems administration scripts are presented. [1]

Subject Areas: Computer Science Education, Computer Science Curriculum Computer Science Laboratories.
Keywords: GNU Software, Software Freedom, Open Software, Open Sources.

1 Introduction

The Trinity University Computer Science Department began to assume increasing responsibility for development and maintenance of its laboratory computers a number of years ago when it became clear that the university computer center

 This paper appears in the Journal of Computing in Small Colleges, Volume 16, Number 3, pages 117-126, March 2001. This paper was presented at the 12th annual South Central Conference of the Consortium for Computing in Small Colleges, Amarillo, Texas, April 20-21, 2001.

staff did not have the required expertise for the specialized needs of the Department. Such needs included Unix operating systems, compilers and interpreters for a wide range of languages, system security, frequent updating of software packages and remote access to systems. As the Department began to acquire a variety of Unix systems from Sun Microsystems, Silicon Graphics,Incorporated and Hewlett Packard it became clear that it was necessary to perform our own systems administration. Two computer science faculty volunteered to provide management for this effort, to train computer center staff, and perform much of the work themselves. Several attempts to involve students in these efforts proved unsuccessful and were later abandoned. As the systems administration workload increased, it became increasingly clear that automation of administrative tasks, where ever possible, would be useful.

As the Sun and HP Unix systems became outdated, they were replaced with higher-end Intel architecture machines running Linux. The use of Linux, because it is based on Open Source software, [DiB 1999, Ray 1997, Ra1 1997, Ray 1999] had the advantage of lowering the cost of each Unix workstation substantially, while, at the same time increasing the range of available software. As the number of Linux systems grew to 78 out of a total of 106 computer science machines, the two faculty responsible for the systems administration of these machines saw their workload increase substantially. To make matters worse, one faculty member retired leaving the entire task to the author.

To ease the systems workload, a number of decisions were made to simplify the organization of our systems and provide a similar software environment for each workstation independent of the brand of the system. These decisions included:

- Network Information System (NIS): provide a common password database to authenticate user logins.
- Network File System (NFS): provide user home directories from a common source.
- `/usr/local` file-system: provide a common shared file-system for optional installed software.
- Login Scripts: provide a system which identifies the system architecture and, where possible, gives a common user interface and installed software base.

Initially the NIS and NFS services were provided on dedicated Sun or HP server machines. When these machines were scheduled for replacement we used a single Intel based machine running Red Hat Linux. Later, when the Sun and HP workstations were replaced we used Intel processor workstations running Red Hat Linux. The Red Hat distribution was chosen primarily for its Red Hat Package Manager (RPM) [Bai 1997] which simplifies some software installation and maintenance tasks.

Nearly all of the software packages we use in our departmental laboratories are available in RPM form. However, the usual approach in package development is to put package binaries in `/usr/bin` or `/usr/X11R6/bin`, libraries in `/usr/lib`, documentation in `/usr/doc`, etc. This means that providing all of your installed software packages on an NFS `/usr/local` file-system was no longer practical. The Red Hat Package Manager also provides some facilities for automatic system updates from a server machine. This facility is used occasionally, but we have found that another approach is preferable to handle updates and occasionally the need to install systems from scratch when disk drives fail or new major releases of operating systems become available.

Linux vendors do supply systems, such as Kickstart,[Red 2000] which can be used to partially automate the installation of a vendor's system software. However, such systems do not seem to offer all the features for system installation and updating, particularly when a rather large base of additional software is to be provided on each machine. Such systems also require large amounts of disk space for each system to be configured. Finally, we desired a system for installation and updating which could be operated remotely, over the network, and perhaps be run automatically. Symantec markets a system, Ghost, for cloning Windows and Linux systems, however, the system would not properly install Linux filesystems during our testing.

2 Software Images

When managing the software for a number of computers it is necessary to develop techniques to manage the similarities and differences of the machines. Many vendors of proprietary software attempt to key operating systems and application programs to single machines even under circumstances where appropriate licenses have been negotiated for each computer. This usually means that a systems administrator must spend time in front of each system console installing operating systems and application software rather than making a single installation of the operating system and application programs and copying that installation to other machines. The labor requirement is significant, particularly when one considers the fact that in a CS laboratory setting, software must be periodically upgraded and re-installed. When re-installing, one often must re-install each application package on each machine.

Open Source software, because of the freedom provided by Open Source licenses, [Gnu 1989, Art 1998, Net 1998, SGI 1999, App 1999] allows the possibility of making a single installation of operating system and application software and then creation of a system image which can then be easily, even automatically, installed or re-installed on each system. Red Hat provides a system installation tool, Kickstart, which can be used to automate the Linux installation process from CDROM or NFS file-systems by using an installation script for each target machine. This system is not suitable for our site, since our Linux system occupies approximately 2.8G disk space and will not, therefore, fit on CDROM. NFS Kickstart installation requires machine console access (as does

CDROM installation) and for this reason was considered unsuitable.

A standard compressed tar file of all appropriate file-systems is created from a tested *master* system. This image is stored on an NFS file-system which is mountable on any system where the image may be loaded. Creating the system image would be simple if each laboratory machine were identical in configuration. However, differences in feature such as the number of processors, sound, video, network, and disk hardware, make image creation a bit more challenging. For example, different video adapters require different X11 configurations. Different disk drive types (SCSI or IDE) as well as different disk drive partitioning schemes require different LILO and `/etc/fstab` configurations. These differences are handled by including all configuration possibilities in the system image and selecting the appropriate configuration during the host configuration phase of installation by setting up *soft links* to appropriate configuration files.

3 Disk Partitioning

The key to allowing automated re-installation of updated system images is to provide a small Linux system which is run while loading the new system on a disk partition. Initially, this Linux system is booted from a floppy disk or a CDROM and is used to create initial disk partitions for Linux (swap, small installer Linux and regular Linux) and, if the system is a dual boot system, a file-system for Windows. The initial disk partitioning must be done in a traditional manner. We use the Linux `fdisk` program. Once the disk is partitioned, a small Linux system is installed on one of the two Linux partitions, `lilo` is run to prepare the system for booting and the system is rebooted. At this point, the system is ready to load the prepared Linux image and this part of system installation may be done remotely and automatically from an installation script running on another machine.

Following are the disk partitions used on a sample Pentium III workstation which has a 10G disk drive. Both Linux partitions are of equal size to allow the next Linux system to be loaded while running the current Linux system. If disk space is scarce, the second Linux system need only consume about 100M. This system would need to be booted before loading the next Linux system on the larger Linux partition.

```
[root@Xena00 root]# fdisk /dev/hda

The number of cylinders for this disk is set to 1244.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., LILO)
2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)

Command (m for help): p

Disk /dev/hda: 255 heads, 63 sectors, 1244 cylinders
Units = cylinders of 16065 * 512 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hda1             1       560   4498168+  83  Linux
```

```
/dev/hda2           561       597    297202+  83  Linux
/dev/hda3   *       598      1228   5068507+   c  Win95 FAT32 (LBA)
/dev/hda4          1229      1244    128520   82  Linux swap

Command (m for help): q
```

4 Loading System Images

Once disk partitions have been created and a small *installer* Linux has been installed and booted, it is possible to create scripts which login to the target Linux system, mount the NFS file-system which contains the Linux image to be installed and then run the installation script. Arguments to this script select such things as target disk partitions, etc. The installation scripts may be run from a remote machine and setup so that a new version of the operating system and application software may be installed overnight when the machines are not being used.

Following is a sample installation script:

```
#!/bin/bash
# A Script to load a Linux system from Sol:sysadmin/images
# This script resides on Sol:sysadmin which must be mounted
# on /sysadmin before running the script.
# The user should be in /root before running this script.
# The mounting and un-mounting of all other partitions is handled
# by the script.
#
# NB. Usage
#
#   load-linux <dest> <image>
#
# Typical usage (for JanusXX or XenaXX) :
#
# /sysadmin/load-linux hda1 janus-hda1.tgz
# /sysadmin/load-linux hda2 janus-hda2.tgz
#
# (for DwarfX or SnowWhite)
#
# /sysadmin/load-linux sdb1 Dwarf-sdb1.tgz
# /sysadmin/load-linux sdb2 Dwarf-sdb2.tgz
#
# This script removes the old (non running) linux system from /<dest> .
# The tar file containing the new linux system is assumed to reside on
# Sol:/sysadmin/images/ and the script argument <image> is the name of
# the gzipped tar file.
#
# After loading the tar file, the script runs lilo.  To activate the new
# Linux system, reboot the machine after running host configuration
# scripts.
#
# dest is the name of the destination partition
# mount point, for example, either /hda1 or /hda2 .  This partition
# must be described in the /etc/fstab so that the
# mount (below) will work.

dest=$1

# image is the name of the gzipped tar file in /sysadmin/images containing
# the system to be loaded into dest.

image=$2
```

```
mke2fs -b 1024 /dev/$dest > load-linux.out 2>&1

mount /$dest

date >> load-linux.out 2>&1
( cd /$dest ; tar xzf /sysadmin/images/$image ) >> load-linux.out 2>&1
lilo -r /$dest -v -v >> load-linux.out 2>&1
date >> load-linux.out 2>&1

umount /$dest
```

5 Host Configuration

After loading a new Linux system, a number of system dependent configuration parameters must be changed as they reflect the configuration of the system used to make the image. For example, the system image may have been made on a single processor machine, but the target machine may have 2 or 4 processors. This particular configuration is made by establishing `/etc/lilo.conf` to point to an appropriate `lilo` configuration file which boots the multiprocessor Linux kernel. Soft links are also used to setup appropriate `/etc/X11/XF86Config` configuration files for each target machine video adapter and `/etc/fstab` configuration files to select appropriate disk partitions and disk type (SCSI or IDE).

Red Hat Linux distributions contain as subsystem, called `kudzu` which detects and configures various hardware parameters. `kudzu` runs at boot-up and will detect and configure any hardware it finds in the system which has changed from what is found in its configuration file `/etc/sysconfig/hwconf`. If the target system has different hardware from the system used to make the image, then adjustments will be performed by `kudzu` during the first system boot.

Finally, system identity, network parameters (IP number, netmask, etc.) must be changed to reflect the identity of the target system. Our systems run OpenSSH (a secure shell daemon) and new host keys must be computed. This final configuration must be performed after a new target system is booted.

Script for setting system identity:

```
#!/bin/bash
# A simple script to perform CS Linux host configuration.
# We use Red Hat distributions, so we assume Red Hat file organization.

# Ordinarily this script will be run from Sol:/sysadmin which is usual
# mounted as /sysadmin on our systems, so one must first mount (as root)
#
# mount /sysadmin
#
# Then the script is run as (supposing the image being configured is
# Xena00 which has an IP of 131.194.131.90) and you wish to configure
# this machine as Janus00 which has an IP of 131.194.131.150 and the
# non-running partition is mounted on /hda1

# /sysadmin/host-config Xena00 90 Janus00 150 hda1

# The name of the host (unqualified) in CS.Trinity.Edu
sourcename=$1

# The ip number of the host (unqualified) in 131.194.131
sourceip=$2
```

```
# The new name of the host (unqualified) in CS.Trinity.Edu
newname=$3

# The ip number of the host (unqualified) in 131.194.131
newip=$4

# The name of mount point for the partition
mount=$5

# Setup the mount point so we edit the files in etc of the
# non-running Linux system
cd /$mount

# Edit etc/sysconfig/network
# to reflect identity of workstation being configured
sed -e "s/$sourcename/$newname/" etc/sysconfig/network > junk1
mv junk1 etc/sysconfig/network
chmod 644 etc/sysconfig/network
chown root:root etc/sysconfig/network

# Edit etc/sysconfig/network-scripts/ifcfg-eth0
sed -e "s/131\.$sourceip/131\.$newip/" etc/sysconfig/network-scripts/ifcfg-eth0 > junk1
mv junk1 etc/sysconfig/network-scripts/ifcfg-eth0
chmod 755 etc/sysconfig/network-scripts/ifcfg-eth0
chown root:root etc/sysconfig/network-scripts/ifcfg-eth0

# Edit etc/hosts
sed -e "s/131\.$sourceip/131\.$newip/" etc/hosts > junk1
mv junk1 etc/hosts
sed -e "s/$sourcename/$newname/g" etc/hosts > junk1
mv junk1 etc/hosts
chmod 644 etc/hosts
chown root:root etc/hosts

# Edit etc/HOSTNAME
sed -e "s/$sourcename/$newname/" etc/HOSTNAME > junk1
mv junk1 etc/HOSTNAME
chmod 644 etc/HOSTNAME
chown root:root etc/HOSTNAME

# Edit etc/http/conf/httpd.conf
sed -e "s/$sourcename/$newname/" etc/httpd/conf/httpd.conf > junk1
mv junk1 etc/httpd/conf/httpd.conf
chmod 644 etc/httpd/conf/httpd.conf
chown root:root etc/httpd/conf/httpd.conf

# Edit etc/resolv.conf
sed -e "s/131\.$sourceip/131\.$newip/" etc/resolv.conf > junk1
mv junk1 etc/resolv.conf
chmod 644 etc/resolv.conf
chown root:root etc/resolv.conf
```

Script for secure shell (`ssh`) configuration:

```
#!/bin/bash
# Do the Openssh host key configuration
rm /etc/ssh/ssh_host_dsa_key
rm /etc/ssh/ssh_host_dsa_key.pub
rm /etc/ssh/ssh_host_key
rm /etc/ssh/ssh_host_key.pub
ssh-keygen -b 1024 -f /etc/ssh/ssh_host_key -N ""
ssh-keygen -d -f /etc/ssh/ssh_host_dsa_key -N ""
```

6 Windows 2000 Software Images

Lab machines which are configured as *dual boot* machines have a FAT32 partition which is used to hold a Windows 2000 system. A FAT32 file-system, rather than an NTFS file-system, is used so that students may have read access to the Windows file-system from Linux. Admittedly, a FAT32 file-system provides less security than NTFS, but past experience has indicated that we need to allow students to have write access to large portions of a lab machine's file-system as a convenience. We have also found that the Windows operating system needs to be re-installed on a periodic basis. Windows operating systems are closely keyed to the hardware of the system on which they are installed and cannot be easily cloned to machines having different hardware features.

The Symantec Ghost program is used to make the initial installation of Windows 2000 on lab machines, using a separate image for each machine type. Of course these different images must be installed individually. We have found that having once installed a Ghost image that it is possible to re-install Windows 2000 systems from a Linux tar image of the Windows 2000 FAT32 file-system. Re-installation is required when it is necessary to restore a working Windows system or when newly installed software needs to be propagated to all machines in a lab.

The procedure is to update a master machine for each different hardware configuration. Then boot Linux and make a gzipped tar image of the Windows FAT32 file-system on our NFS image server. This system may then be loaded onto the FAT32 partition of each dual boot lab machine. Of course, just as when loading Linux images, this work may be performed remotely and automatically from scripts. After loading a new Windows image, however, the identity of each machine must be established manually and, if the machine is a part of a Windows domain, the machine must be deleted and re-added to the domain server.

7 Current Status of Laboratories

Trinity University [How 2000] has used Linux based systems in its CS laboratories for several years with excellent results. Linux was first used in mission critical server applications such as departmental Web servers and NFS file servers. The choice was made, not because of cost, but rather, because of our desire for reliable and secure server systems. Over a period of four years, these systems have experienced only a few hours of unscheduled down time. The NFS server initially functioned in a multi-platform environment providing home directories to SGI and HP Unix systems as well as to PowerPC and Intel based Linux systems. One year ago, a 23 machine laboratory of HP 712 workstations were replaced with 23 Pentium III machines running our own variant of Red Hat Linux, version 6.0. This summer a second lab of 22 Intel based Linux machines has been constructed, and, together with about 10 other dual processor Linux machines, all being served by the NFS file server, all machines run a common

Red Hat 6.2 based Linux image. These machines are connected to the Internet through a pair of Cisco 3500 series Ethernet switches so that each lab machine and the file server have a dedicated 100MB Ethernet line. The switches improve the performance of these machines when run as a 66 processor parallel processing system. The NFS/NIS server machine has been upgraded to a dual processor Pentium III machine which supplies an NIS database to all departmental Unix machines as well as home directories via NFS and Windows 9X/NT/2K access to home directories via Samba. This machine also functions as the departmental Web and FTP server.

The Computer Science Department maintains its own set of local changes to the current Red Hat distribution of Linux. These changes include security modifications, updates and installed software packages which are used in a variety of courses. We make this distribution available to students on a Departmental image server so that students can install and use on their own machines the same software environment which is available in the Computer Science Department labs. Since the software bears the GNU license, this may be done at no expense to the students.

8 Conclusions

Maintaining computer science laboratory systems has traditionally been a rather labor intensive activity. Operating systems need to be re-installed and upgraded on a regular basis. We have found that by using Open Source software solutions that it is possible to have lab machines which have a richer and more reliable software base than is the case with proprietary systems. In addition, we have found that it is possible to reduce system administration work-loads by using non-standard approaches for loading system software and some of this work may be automated.

References

[App 1999] Apple Computer, "Open Source Projects at Apple" http://www.publicsource.apple.com/apsl/, 1999.

[Art 1998] The Artistic License, http://language.perl.com/misc/Artistic.html, 1998.

[Bai 1997] Bailey, Edward C., *Maximum RPM*, Red Hat Software, Inc., Durham, NC, 1997.

[DiB 1999] *OPENSOURCES, Voices from the Open Source Revolution*, Edited by Chris DiBona, Sam Ockman and Mark Stone, O'Reilly & Associates, Sebastapol, California, 1999.

[Gnu 1989] GNU General Public License, http://www.gnu.org/copyleft/gpl.html, 1989.

[How 2000] Howland, John E., " Software Freedom, Open Software and the Undergraduate Computer Science Curriculum", Journal for Computing in Small Colleges, Vol. 15, No. 3, March 2000.

[Jen 1999] Jensen, Michael, "Information Technology at a Crossroads: Open Source Computer Programming", The Chronicle of Higher Education, October 29, 1999.

[Net 1998] Netscape, Mozilla & Netscape Public Licenses, http://www.mozilla.org/MPL/, January 1998.

[Ope 1999] "The Open Source Definition", http://www.kr.debian.org/OpenSource/osd.html, 1999.

[Ray 1997] Raymond, Eric, "The Cathedral and the Bazaar", http://www.tuxedo.org/~esr/writings/cathedral-bazaar/, 1997.

[Ra1 1997] Raymond, Eric, "Home steading the Noosphere", http://www.tuxedo.org/~esr/writings/homesteading/, 1997.

[Ray 1999] Raymond, Eric, "The Magic Cauldron", http://www.tuxedo.org/~esr/writings/magic-cauldron/, 1999.

[Red 2000] Red Hat, Inc., "Reference Guide for Red Hat Linux 6.2", Red Hat, Inc., Durham, NC, 2000.

[SGI 1999] Silicon Graphics "Linux", http://www.sgi.com/developers/technology/linux/, 1999.