The following paper was originally published in the
Proceedings of the Eleventh Systems Administration Conference (LISA '97)
San Diego, California, October 1997

For more information about USENIX Association contact:

1. Phone: 510 528-8649
2. FAX: 510 548-5738
3. Email: office@usenix.org
4. WWW URL: http://www.usenix.org

# Managing PC Operating Systems with a Revision Control System

*Gottfried Rudorfer* – Vienna University of Economics and Business Administration

## ABSTRACT

During the lifetime of a workstation the system administrator is faced with constant change in system configuration (updates, new software). The users of a workstation, too, may change the system configuration. We describe the necessary concepts for maintaining many similar configured PC clients in a lab environment. The main software component consists of *fsrcs*, a revision control system similar to *RCS* and *SCCS*. The revision control software is available without charge.

### Motivation

This paper was motivated by the need to reduce the huge administrative effort for running a computer training room at the Department of Applied Computer Science. It is the first attempt at our university of a fully automatic client installation and update solution using free software. The students can handle the following from the boot-prompt of the PC client without assistance of the system administrator:

1. Repair or install Linux software (operating system and applications) on the PC.
2. Repair or install Microsoft Windows 95 software (operating system and applications) on the PC.
3. The installation can be requested by the user at any time.
4. The user can decide to update both operating systems or just Windows 95 or just Linux.

Administration tasks are reduced because the system administrator installs new software only on one PC. After installation a set of programs help the system administrator to define a new master copy for all clients on the server.

### Comparison with other tools

There are many tools available for managing software environments [CW92, Fut95, Har94, Jam94, PS94, Rid94, VCV92]. A new draft standard for software administration tries to define interfaces and formats for the administration of a network of heterogeneous systems [Arc93]. Many tools use software packages and supply commands for the administration of these packages. Microsoft's Systems Management Server (SMS) distributes software packages (i.e., Microsoft Office) and runs unattended installation. Limited functionality is provided for automatic checks of an existing package on the PC. Installation scripts have to be written for local customizations, patches, and not SMS aware software. This is a very complex task.

Other approaches just extract a ZIP- or a compressed tar-archive. This approach is not well suited for checking the correct installation of the software. Files on the PC which are not part of the software installation might remain after the extraction of the archive. Unfortunately, many of the above tools only provide a partial solution for our software distribution problem. We often have a situation where the software is already installed, but then a user modifies some files of the installation. We need a system that checks and repairs an already existing installation as fast as possible. Users may not accept the system if this process is too time-consuming.

Our approach is based on the fact, that the same software packages installed on PCs result in a very similar set of files even if the PCs have a different hardware configuration. Compared to the above approaches, our approach tries to replicate already installed software to other PCs. The necessary configuration of the software is done afterwards. Our approach just distributes files without looking at the semantics, if possible. However, we have to look into files if they contain parameters to configure for proper operation of the software. The operating system Linux has many tools and concepts for software management of PCs available [Tro96]. Therefore, we decided to develop our programs under Linux.

### Implementation

The management software consists of three parts (see Figure 1):

- a set of programs for downloads from the server to the clients
- a set of programs for uploads from the clients to the server
- a revision control system for file trees

#### Implementation of the download programs

*On the PC Client Side*

The root file system of the installation program is loaded into the main memory to avoid any conflict with the hard disk. This is done by loading the file system as initial ram disk [AL96]. If the PC cannot boot from a prior installation, the same kernel and the initial ram disk are loaded from a floppy disk.

The normal usage is to load the kernel and the ram disk from the Linux file system of the already installed PC by entering the boot option "c" (update both Linux and Windows 95) or "d" (update Windows 95 only) or "e" (update Linux only) at the LILO [Alm96] prompt.

When the initial ram disk is mounted the script */linuxrc* is executed. The script configures the network interface with a *bootp* request. On success the NFS exported directories of the server are mounted read-only and other libraries and programs are made available.

Finally, the installing perl script is executed. This script does the following:

1. Get the system time from the server and write it to the CMOS clock.
2. Check the partition table and if it does not correspond to the sample table recreate the partition(s).
3. Repair the file systems if they are inconsistent.
4. Mount the file systems and enable swap.
5. Ask the server to update the client.
6. Detach the mounted file systems.
7. Write a new master boot record.
8. Do local customizations (i.e., add the hostname to the registry of Microsoft Windows 95)

There is no reboot necessary during installation.

*On the Server Side*

The regular login shell of the download user *pc7inst* is replaced by the program *toclient* which first changes the effective root directory (*chroot* ) to home of the master copy. The Set-user-ID permissions bit of the program is set to run the program as user *root*. All *root* users of the clients have access to this account via the *.rhosts* file. The name of the client and the mode of update (all operating systems or just Linux or Windows 95) are passed as command line arguments to the program *toclient*. Finally the program *rdist* [Coo92] is executed.

**Implementation of the Upload Programs**

The implementation of the upload programs tries to preserve the security on the server with an upload user *pc7adm*. This user has *root* privileges on the Linux operating system for the installation on the clients (is added to the *.rhosts* file). The core of the upload program is *rdist*, too. The regular login shell of the upload user *pc7adm* is replaced by the program *ask*:

1. If the system administrator logs in successfully at the server as *pc7adm* the program is called without arguments and asks the user which client to upload. When the client is found in the database, access of the *root* user from the client is granted. Finally the program starts a remote shell on the client with a slightly modified *rdist*.
2. The client program *rdist* connects via *rsh* to the upload user *pc7adm* and executes the Set-user-ID root program *toserver -S*. This program first changes the effective root directory to the root of the client master installation. Then the *rdist* server *rdistd -S* (which is statically linked) is executed. The files are installed with the *rdist* options *-onumchkgroup -onumchkowner* to use the numeric group and user ID for checking

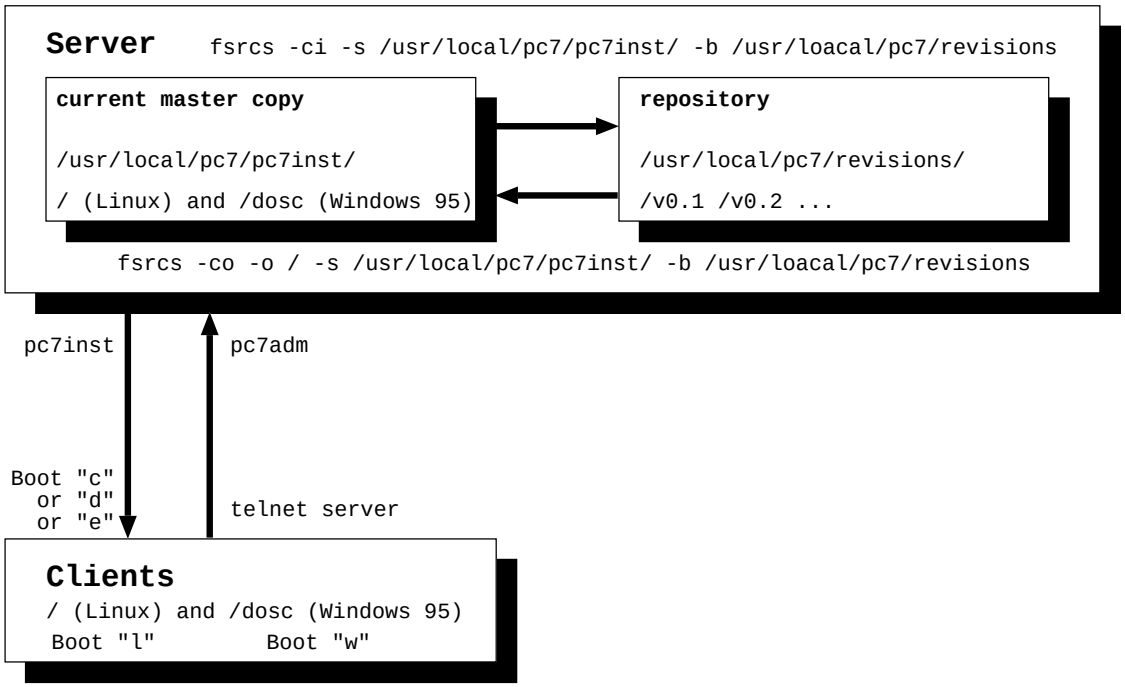


**Figure 1**: Relationship between clients, server and the repository.

group and user ownership because the group and user name might not exist on the server.

3. Finally, the access of the client is removed from the *.rhosts* file.

**Implementation of the Revision Control System**

The first version of our management software had no revision control system. After some testing we found that one version of the master installation has the disadvantage of no ability to downgrade after an update with errors.

We decided to store the master copy in the repository of our file system revision control system *fsrcs*. It can handle at least text and binary files, symbolic and hard links, character and block device files and directories with proper access permissions and ownership.

The current implementation stores the master copy on the server. A new version of the master copy is manually created when a major change in the software configuration of the clients occurred.

Multiple revisions can be managed by our system. The repository is implemented as a file-tree. Underneath the root there are directories with the name of each revision. New or changed files (and directories and links) are stored inside each version-directory. However, a symbolic link to the original file in the repository is created when there was no change. Symbolic and hard links are stored in separate directories for each version. The initial repository is created by copying the whole tree into the directory of the first version.

The program is implemented in the programming language *perl* [WCSP96].

*Check-In*

This section describes the updating process, done on the check-in of the new version. We distinguish a different behavior for files (plain files, character and block special files), symbolic links, hard links and directories. The procedure for directories is fairly obvious. If a directory is found, the check-in function is called with its name (recursion). Directories are created with the access rights and ownership of the original. If a directory does no longer exist in the new version, no reference to the previous version is made. Files are compared with the current version of the file in the repository if it exists. If the size, permissions, ownership and group are the same, this file is first marked as unchanged. The check-in function performs link optimization.

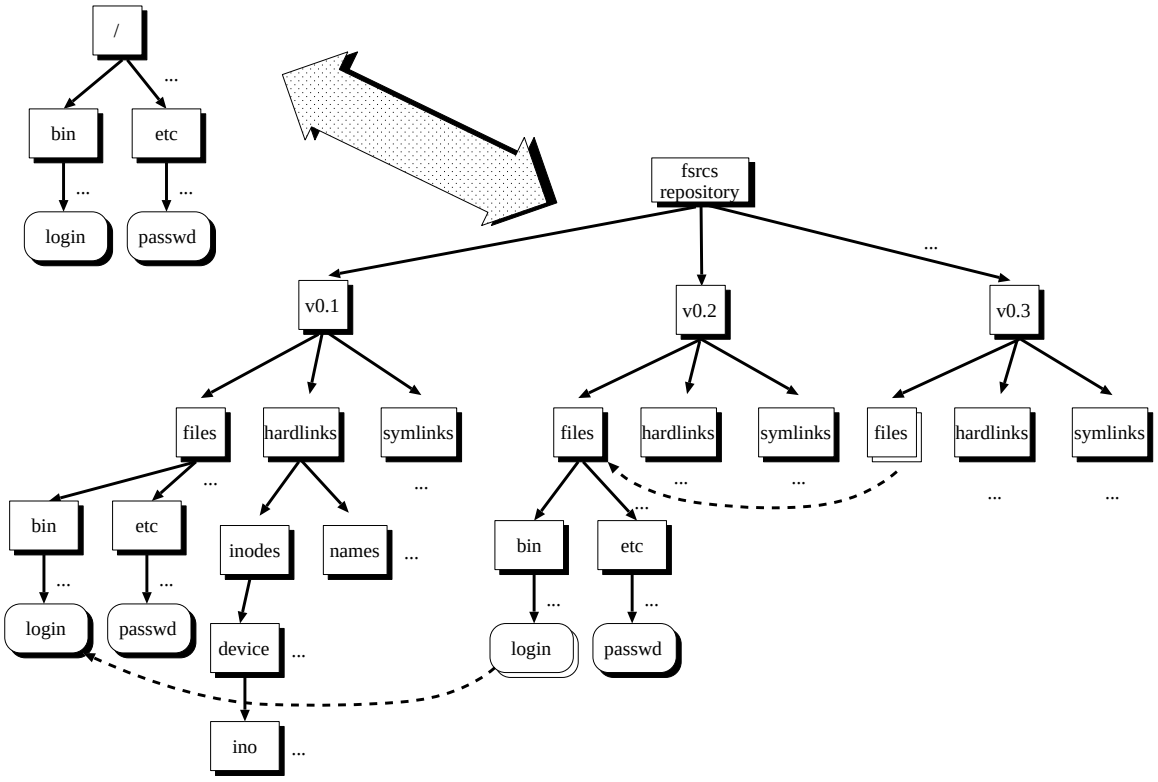


**Figure 2**: Internal representation of file system objects. At check-in time of version v0.2 the file `/bin/login` has not been modified and the file `/etc/passwd` has been changed compared to the first version. The check-in function made a reference to the previous version of `/bin/login`. All files of version v0.3 have not been changed at check-in time compared to the previous version v0.2. The check-in function made a reference at the highest possible level to the prior version with link optimization.

If a whole subtree is unchanged, a symbolic link is created at the highest possible level in the repository. If a reference to a prior version of an object within the repository is necessary, another link optimization is done. The check-in function does not simply create a symbolic link to the prior version, instead it creates a reference to the original object in the repository. These two levels of link optimization guarantee a minimum number of symbolic links in the repository. Compared with *RCS* and *SCCS* [BB95], the smallest piece is an entry of a directory. Your software does not bother about the differences between text or binary files.

*Check-Out*

On check-out, the program compares the entries of the given check-out directory with the entries in the repository. Missing entries are created and entries in the checkout-directory that are not in the repository are recursively removed (directories) or unlinked.

*Limitations*

The current version of *fsrcs* has no support for access control lists (ACL). Some UNIX operating systems support ACL which allows the file owner to permit or deny access to a list of users. However, this feature is neither used nor supported by Linux and Windows 95 installations. Another restriction exists for symbolic links. Many UNIX operating systems do not set the permissions for symbolic links correctly. On repeated checkouts symbolic links will probably be recreated even if they have not been changed.

*Availability*

Our file system revision control system is available from `ftp://ftpai.wu-wien.ac.at/pub/fsrcs/fsrcs.tar.gz`.

## Administration tasks

The time to administer a set of PCs is reduced to the installation or update of new software on a single PC. Our tools guarantee the correct installation of the software on all other PCs. If some files or partitions have been changed, our tool repairs the installation automatically and quickly without user interaction.

### Installing New Software on a PC

Administration sessions require the following steps to install or upgrade software:

1. Make a full update (boot with option "c").
2. Uninstall the prior version of the software, if required.
3. Install the new version of the software.
4. Upload the new installation to the server.

The new software is transferred to other clients when they make a full update (boot with option "c").

### Finding the Correct Configuration

The most complicated task is searching for mandatory customizations between different PCs. First, we tried to eliminate individual configuration parameters, if possible.

*Linux*

Linux software distributions often configure their network parameters statically. A networked Linux PC needs a unique IP address and hostname. First we installed the Linux distribution Redhat 4.0 (Colgate) and then replaced the file */etc/rc.d/init.d/network* with the file */etc/rc.d/init.d/rc.bootp* of the package *bootpc* [Haw96]. This package demands the network parameters with a *bootp* request from the server, configures the network interface and creates the files */etc/hosts* and */etc/resolv.conf*. After this change in the configuration of Linux, there are no individual configuration parameters left.

*Windows 95*

We configured IP networking to obtain the network parameters with a *DHCP* request from the server. Windows 95 uses the binary files *system.dat* and *user.dat* to store its configuration. These files are modified by the program *regedit* (see Table 1) of Windows 95 which is called from *dosemu* [LS97] after the file checking phase. Plug and play (PNP) cards need entries in the registry, too. Otherwise Windows 95 will wrongly detect new hardware and try to install new software. First we exported the registry as text files of two different PCs using *regedit*. Then we looked at the differences between the files with the program *diff*. We found that the necessary entries for our PNP sound cards are in `HKEY_LOCAL_MACHINE\Enum\ISAPNP\`. We generated a similar text file in which we replaced a generic string with the PNP-ID of the sound card.

```
REGEDIT4

[HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\VxD\VNETSUP]
"ComputerName"="hostname"
"Workgroup"="PC7"

[HKEY_LOCAL_MACHINE\System\CurrentControlSet\control\ComputerName\ComputerName]
"ComputerName"="hostname"
```

**Table 1**: The contents of a configuration file for the registry of Windows 95. First, the string hostname is replaced by the actual hostname of the PC. Finally *regedit* is called with the modified file to patch the registry.

*Periodical Checks*

The cron daemon of each PC is configured to run our management software each day early in the morning.

## Performance

For software distribution a UNIX server with Pentium processor, approx. 128 MByte main memory and a PCI based network interface will be enough to serve approx. 30 client PCs.

Our hardware consists of one Digital Alpha Server 4000 5/300 (2 × 300 MHz CPUs, 1 GByte main memory, 20 GByte hard disk, 2 × 100 Mbps full duplex ethernet cards), one Cisco Catalyst switching hub, 27 PCs (Pentium Pro 200 MHz CPU, 64 MByte main memory, 2 GByte hard disk, 10/100 Mbps ethernet card currently used in 10 Mbps mode).

The size of Windows 95 and applications is currently 202 MBytes (3413 files in 194 directories). The size of Linux is currently 707 MByte (38372 files, 2797 symbolic links, 735 character special files, 304 block special files, 706 hard links in 2171 directories)

Installation from scratch is done with the prior described kernel and root file system on a boot floppy which requires approx. 42 minutes for partitioning, formatting, checking and copying of both operating systems. After automatic reboot of the installation system, the two operating systems are available.

Checking of the whole installation (Linux and Microsoft Windows 95) with minor modifications requires approx. eight min. Checking our installation of Microsoft Windows 95 requires approx. two min.

Each of the 27 clients needs about 20 minutes when all of them check their whole installation at the same time against the server. In this case the server is the limiting resource. Currently, the server is not tuned for optimal performance.

## Conclusion

By using open operating systems for systems management, it was easy to develop a powerful tool for UNIX and Windows 95 operating systems. The users are greatly satisfied with the ability to repair the local installation by themselves if something goes wrong with the software. The time spent on troubleshooting has decreased by 90%.

## Future Work

This implementation proves, that Linux is a suitable operating system for automated installation and update of software.

This version of software requires PCs with an equal hardware configuration. We plan to extend the shown concept to handle different hardware installations by extending the revision control system to find the differences between two software installations. The program *rdist* seems to be the limiting factor on the server side. There is one independent *rdist* process for each client which scans the state of each file. This causes a heavy system load on the server when many clients perform an update at the same time. This problem could be solved by extending *rdist* using a state database. Another area of improvement is in adding revision control capabilities to *rdist* directly.

## Author Information

Gottfried Rudorfer works at the Department of Applied Computer Science at the University of Economics and Business Administration where he received his MBA. His research interests are centered on issues of distributed database systems, administration of heterogeneous computing environments, operations research and artificial intelligence. He has been a system administrator since 1993. Reach him via mail at University of Economics, Augasse 2-6, A-1090 Vienna, Austria; or electronically at Gottfried.Rudorfer@wu-wien.ac.at.

## Bibliography

[Al96] Werner Almesberger and Hans Lermen, "Using the initial RAM disk (initrd)," `ftp://ftp.funet.fi/pub/Linux/PEOPLE/Linus/v2.0/linux-2.0.30.tar.gz`, almesber@lrc.epfl.ch, lermen@elserv.ffm.fgan.de, 1996.

[Alm96] Werner Almesberger "Generic boot loader for Linux," `ftp://lrcftp.epfl.ch/pub/linux/local/lilo/lilo.19.tar.gz`, almesber@lrc.epfl.ch, 1996.

[Arc93] Barrie Archer, "Towards a POSIX Standard for Software Administration," *LISA,* pp. 67-79, November 1-5, 1993.

[BB95] Don Bolinger and Tan Bronson, *Applying RCS and SCCS*, O'Reilly & Associates, Inc., 103 Morris Street, Suite A, Sebastopol, CA 95472, 1995.

[Coo92] Michael A. Cooper, "Overhauling Rdist for the '90s," *LISA VI*, pp. 175-182, October 19-23, 1992.

[CW92] Wallace Colyer and Walter Wong, "Depot: A Tool for Managing Software Environments," *LISA VI*, pp. 153-162, October 19-23, 1992.

[Fut95] Atusushi Futakata, "Patch Control Mechanism for Large Scale Software," *LISA IX*, pp. 213-219, September 17-22, 1995.

[Har94] Magnus Harlander, "Central System Administration in a Heterogeneous Unix Environment: GeNUAdmin," *LISA*, pp. 1-8, September 19-23, 1994.

[Haw96] Charles Hawkins, "Linux Bootp Client," `http://www.damtp.cam.ac.uk/linux/bootpc/`, ceh@eng.cam.ac.uk, 1996.

[Jam94] Kevin Jameson, "Multi-Platform Code Management," O'Reilly Associates, Inc., 103 Morris Street, Suite A, Sebastopol, CA 95472, 1994.

[LS97] Matthias Lautner and Robert Sanders, "DOSEMU PC Emulator," `ftp://tsx-11.mit.edu/pub/linux/ALPHA/dosemu/`, linux-msdos@vger.rutgers.edu, 1997.

[PS94] Dieter Pukatzki and Johann Schuhmann, "AUTOLOAD: The Network Management System," *LISA*, pp. 9-17, September 19-23, 1994,

[Rid94] Paul Riddle, "Automated Upgrades in a Lab Environment," *LISA,* pp. 33-36, September 19-23, 1994.

[Tro96] Jim Trocki, "PC Administration Tools: Using Linux to Manage Personal Computers," *LISA X,* September 29-October 4, 1996.

[VCV92] Ram R. Vangala, Michael J. Cripps, Raj G. Varadarajan, "Software Distribution and Management in a Networked Environment," *LISA VI,* pp. 163-170, October 19-23, 1992,

[WCSP96] Larry Wall, Tom Christiansen, Randal L. Schwartz, and Stephan Potter, *Programming Perl,* Second Edition, O'Reilly Associates, Inc., 103 Morris Street, Suite A, Sebastopol, CA 95472, 1996.