Backup system for Skolelinux

—

Computer systems administration, depth study

Morten Werner Olsen

werner@skolelinux.no

Teaching supervisor (NTNU)	:	Anders Christensen
Teaching supervisor (Skolelinux)	:	Frode Jemtland

Department of Computer and Information Science, NTNU

November 28th, 2003

Preface

This is the report in the project part of "Computer systems administration, depth study". The project is accomplished by Morten Werner Olsen, Master of Technology student at the Department of Computer and Information Science, NTNU.

This project is accomplished for The Skolelinux Project. Skolelinux is a custom Debian GNU/Linux distribution for schools.

The project deals with backup and its main task is to adapt and document a backup solution for Skolelinux.

Many thanks to everyone who helped me out with this project. Thanks to Anders Christensen (NTNU) and Frode Jemtland (Skolelinux) for doing great jobs as my supervisors. Thanks to Petter Reinholdtsen (Skolelinux) for outstanding technical support. Thanks to Knut Yrvin (Skolelinux) for proof-reading my report. Thanks to everyone on the Skolelinux IRC channel and email-lists for great support. Thanks to Klaus Ade Johnstad (Skolelinux), Marius Kotsbak (Skolelinux), and Per Harald Westby (Skolelinux) for testing the product and reporting bugs for me to fix.

This report will be available in electronic form as well as source code for download on the projects web page.

Project web page:
`http://developer.skolelinux.no/info/studentgrupper/2003-backup/`

Trondheim, November 28th, 2003

Morten Werner Olsen
werner@skolelinux.no

Contents

List of Figures

Summary

This project is about backup, an important service in a computer network. The customer, Skolelinux, didn't have a working and pre-configured backup solution with a graphical interface that system administrators at the schools could use. A "release critical bug" was filed, as Skolelinux wanted a working backup solution installed and configured "out of the box" before releasing version 1.0 of their custom Debian GNU/Linux distribution.

The project was divided in two main parts. The first part was to find, evaluate, and test the most relevant available backup solutions for Linux and find one that will suit Skolelinux needs. The second part was to adopt and document this backup software for Skolelinux.

The software that was choosen as the underlying backup software in Skolelinux' backup solution was *rdiff-backup*. *rdiff-backup* was choosen because that it fulfills all the requirements for backup software in Skolelinux, and for its Debian packaging, disk usage, network bandwidth usage, documentation, activity, and support.

The product that this project has lead in to is called *slbackup*[1] and is using *rdiff-backup* as the underlying backup software. *slbackup* consists of two packages, one that handles the backups and one including a web interface that handles the configuration and restore parts. These software packages are released as Open Source software under the GPL license.

slbackup is included in the Skolelinux distribution and will be automatically installed and configured when a Skolelinux installation is completed.

[1]slbackup - Skolelinux Backup

Chapter 1

Introduction

This project is a component of the course "Computer systems administration, depth study" in the 9th term (of 10) in the Master of Technology study at the Department of Computer and Information Science, NTNU. The project assignment is as follows:

> Find, evaluate, and for the most relevant choises test available backup solutions for Linux and adapt and document one of them for Skolelinux.

1.1 The Skolelinux Project

The customer for this project is The Skolelinux Project, from now on called Skolelinux. Skolelinux is developing a Custom Debian GNU/Linux distribution for schools. Their aim is to make it simple to install and maintain. The Skolelinux Project is mainly based on people working in their spare time at no charge. In a community like Skolelinux, which practise a so called do-o-cracy (described in [Yrv01] (Norwegian)), the people who actually do something are the people that decides.

The main communication channels within Skolelinux is e-mail lists and IRC, but there are also organized developer gatherings about four times a year where the developers can meet face to face. On these gatherings discussions, bug squashing and testing are prioritized. When the distribution is going to be used in schools then Open Source[Ray01], free of charge and not requiring expensive hardware are unalterable requirements. When having Skolelinux as the customer, Skolelinux requires that the student(s) actively use the communication channels that are established and used by the other developers. The student(s) are also expected to use CVS when developing software and documentation.

The ideal student project for Skolelinux is a project that not dies when the project period is over, but that the code/solutions developed are maintained also after the student(s) have finished the project. The maintainer could be the student(s) or someone else that is interested in the project.

Skolelinux web site: http://www.skolelinux.no/

1.2 The backup system

The Skolelinux distribution has a goal that is to offer all services that is needed in a computer network on one CD, with a simple installation procedure. In August 2003 Skolelinux was still missing a autoconfigured and working backup solution in their distribution. The goal of this project is to supply Skolelinux with a configured backup system which is working "out of the box".

Writing new backup software is not an option, as there exists a lot of Open Source backup software that should suit Skolelinux needs. Skolelinux has a lot of requirements for a backup system, that are dealt with in detail in chapter 2. The main requirement is that the system shall be automatically configured in the Skolelinux installation and that it shall be easy for the system administrator to use. As Webmin is widely used for most configuring in Skolelinux, a Webmin-module for configuring the backup system is preferred.

1.3 Language

This report is written in English because a lot of the new Skolelinux developers do not speak Norwegian. The system is has users in Norway, German, France, and Brazil. Skolelinux wants the Users Manual to be written in Norwegian, because Skolelinux are putting together a System Administration handbook for Norwegian teachers and schools. They have decided that all documentation produced by student projects shall be written in Norwegian.

1.4 Licenses

All software created in this project shall be released under the General Public License (GPL).

Chapter 2

Requirement specification

The requirement specification for the backup system in this project was prepared as a separate document in the beginning of the project. It has been discussed on two of the mailinglists for developers and users of Skolelinux. Even if the requirement specification is not officially adopted, the lack of loud protests on the mailinglists is as sort of approval. The rest of this chapter is the requirement specification that was presented on the Skolelinux mailinglists.

2.1 Introduction

The Skolelinux project are developing a Linux distribution for schools. Their aim is to make it simple to install and maintain. Furthermore, to be of real use to primary school and above, it should be available in as many local languages as possible. For Norway, that means both the official standards of Norwegian (Bokmål and Nynorsk) as well as northern Sami.

Skolelinux currently does not have a working configuration for backup. This requirement specification, RS, describes Skolelinux's requirements for a backup solution.

The word "backup service" is defined as the product this requirement specification describes and "user" is defined as the user of the backup service, which can be both the system administrator, SA, and the end users, teachers, and pupils.

2.2 Primary system description

A Skolelinux-network mainly contains a file server, a LTSP-server and thin clients. The backup service is supposed to back up home-directories on the file server, the user- and configuration database (LDAP) on the file server and configuration files on the file server and LTSP-server(s).

Skolelinux wants a backup service that is working "out of the box", and of course is a full backup service where the SA easily can restore files and directories in case of an accident. The operator interface shall be as easy as possible.

A main goal with the system is to eliminate the pro-active and routine tasks of backup. This is important as the teachers that have the responsibility for doing the system administration task do not have very much time for this (three hours a week, depending on the size of the school). The system shall inform the SA when it need maintenance.

The backup service shall be so general that other distributions than Skolelinux can reuse the design and implementation.

2.3 General requirements

This chapter specifies the general requirements with respect to the backup service. The most important requirements are:

- must be able to be distributed with the Skolelinux CD (GPL, BSD or similar license)
- must support the architecture, the software, and the hardware that is used in Skolelinux
- both the backup- and restore-part must be extensively tested
- all communication between the backup server and the file server/LTSP-server(s) must be encrypted
- must be possible to place the backup server on another network than the file server and LTSP-server(s)

2.3.1 Availability and reliability

Backup shall start after 12PM and be completed before 5am (except for initial backup). The system shall be available for restore between 6am and 11PM.

The backup service shall in case of any failure send a failure report to the SA.

The schools/institutions who want to use the backup system have to decide how much money they will invest in hardware, but this backup project must document the available options and how they impact the reliability and functionality of the system.

2.3.2 Security

The backup service must be secured in a way that denies intruders (over the network). No other computers than the file server and the LTSP-server(s) shall be accepted by the backup server. The schools/institutions have to decide how much money they will invest in physically securing the backup server (burglary, physical destruction etc.). The documentation shall point out possible options to increase the backup servers (physically) security.

2.3.3 Capacity

The network bandwidth shall be at least 100Mbit/s. The speed on initial backup shall be at least 2GB/h.

The schools/institutions who are going to use the backup system have to decide how much capacity (hard drive, CPU, and memory) they can afford. The documentation shall contain guidelines to planning how much capacity that is needed for the service to work in a satisfactorily way.

2.3.4 Extensibility

It shall be possible to extend the backup servers capacity with more hard disk, CPU, and memory.

2.3.5 User-friendliness

There shall not be made any demands on the SA's IT-knowledge. Every manual operation that is offered by the backup service shall be briefly described in the documentation, and the documentation should be easy to understand.

2.4 Requirements to the systems functional properties

It is **required** that the backup service have the following properties:

- take daily backup of specified directories/files to a hard drive on an external backup server (on a Skolelinux-installation these directories/files would be predefined, but changes shall be possible)

- the daily backup transfer rate (not initial backups) shall exceed 1GB/h
- keep ownership and privileges to files and directories
- perform restore of directories/files (user decides placing)
- restore of 1MB data shall at a maximum take 2 minutes (excluding the SA's preparations and keyboard typing)
- the backup software on the file server and LTSP-server(s) shall be included on the Skolelinux CD and the only input at configuration is the IP-address to the backup server and a password
- the backup server is installed on a separate computer and shall only require input from the SA about which computers (IP-addresses) it shall receive backup from, which computer it shall use as syslog-server (if any), and the email-address to the SA

It is **desirable** that the backup service have the following properties:

- archives are stored encrypted on the backup server (at least 128bits key length and a cipher equal to or better than DES)
- possibility to fetch data from the backup server (over the network) for an offline backup
- the end-user, pupils, and teachers shall be able to restore their personal files (the SA must be able to enable or disable this service)
- the backup system is managed through the Webmin interface, and the backup system's Webmin-module should have the following properties:
  - define all files and directories to be backed up at each server
  - define off-site or on-site machines where backup should be stored on daily, weekly and monthly basis
  - check the quality of backup-files (monitor file sizes, numbers, successful/failed logins, disk full, disk failure etc.)
  - define mail and SMS recipients in case of trouble and errors
  - maintain logs
  - provide browser facilities to restore files from backup
  - restore data
  - syslog monitoring

2.5 Sketch of the technical solution

Figure 2.1 shows how the Skolelinux-network is planned with the backup server included. The plan is to take daily backup of files/directories at the file server and the LTSP-server(s).

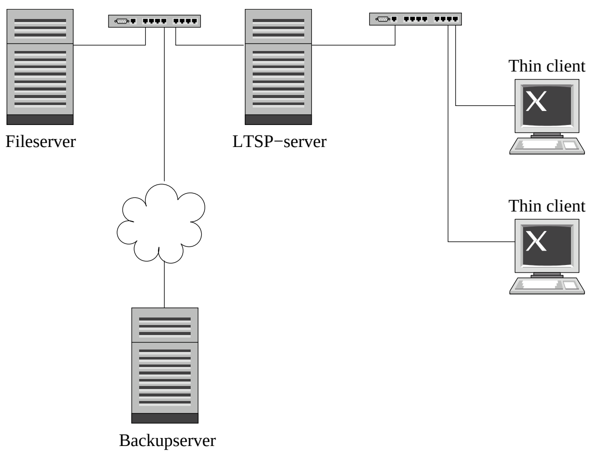


Figure 2.1: Sketch of the Skolelinux network

Daily backups are to be stored for a month, weekly backups for four months and monthly backups for one year. This is illustrated in figure 2.2.

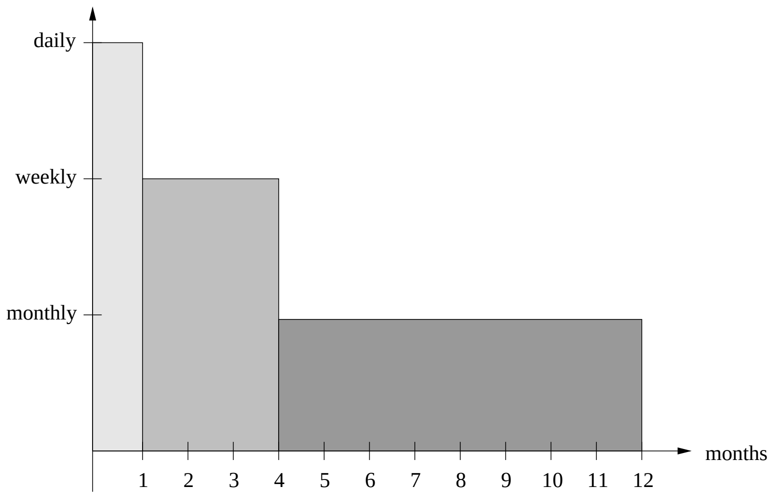


Figure 2.2: Storage of daily, weekly and monthly backups

2.6 Documentation

All code that is written shall be well documented.

These following documents shall be included (in Norwegian):

- User's manual
  (contains information about how to do restore of files/directories)
- System administrator's manual
  (contains information about how to install and configure the backup service, alternative actions to make the backup server more reliable, alternative actions to increase the backup servers physical security, information about how to plan the capacity of the service and how to upgrade the backup server with hard drive, CPU and memory)

Chapter 3

Preliminary studies

In this chapter Skolelinux and their ambitions and requirements will be described. The chapter will also contain a study of different backup systems, the reasons for not choosing most of them, and the results and comments from tests that have been carried out.

3.1 Skolelinux

Skolelinux wants their distribution to be as simple to install and maintain as possible. When it comes to a backup solution it is acceptable with an initial amount of work to configure and install the backup system, but the system administrators in the schools are often teachers who do not have time to many proactive and routine tasks. The backup system must of course inform the system administrators when there is work to be done that are critical to the systems main tasks.

3.1.1 Ambitions

Under the development of Skolelinux, it is a goal to use as much software and solutions that are already developed or under development as possible. The main reason for that is when a software project is active, Skolelinux do not have to maintain that part themselves and one will achieve a greater progression and can concentrate on the main goal, which is developing an easy to install and a next to maintenance free Linux distribution for schools. One example is that Skolelinux is built on the Debian GNU/Linux distribution and uses a lot of the software that is already prepared for Debian.

This will also be important when the backup system in Skolelinux shall be put together. As little time as possible should be used to develop a piece of software that already has been developed. Time should be used to choose the right software and to put this together and make it as easy as possible to install and maintain for the schools system administrators. Writing documentation is also a part of the project.

3.1.2 Requirements

If a backup system is to be shipped with the Skolelinux distribution, some requirements must be met (see chapter 2). These requirements also applies to external software used in the backup system.

- The backup system and the external programs used in the backup system must have a license similar to GPL or BSD. This because Skolelinux shall be a free alternative to the schools and because Skolelinux is build on Debian GNU/Linux where all programs included must follow The Debian Free Software Guidelines[Sch].
- The backup system must support the architecture, the software and the hardware used in Skolelinux.
- It must be possible to place the backup server on another network than the file server and LTSP server(s).
- The communication between the backup server and the machines that shall be backed up, must be encrypted.
- It must be possible to specify which files and directories that shall be backed up.
- The uid/gid ownership and privileges must be stored as well.
- The backup system has to store the backup to a hard-disk. The reason for this is that tape is expensive, unreliable and leads to routine work for the system administrator.
- The backup system must handle snapshots in such a way that it is possible to restore files from different points of time.
- The software used in the backup system should be packaged for Debian. Preferably it should be packaged for the Debian stable release, but it better that it only exist in Debian testing or unstable than it is not packaged for Debian at all.

3.2 Backup software

afbackup

Author	: Albert Flugel
Last stable release	: March 1st, 2003
License	: GPL
Web site	: http://sourceforge.net/projects/afbackup/

afbackup is a client-server backup system that offers several workstations a centralized backup service. The backup can be started by a cronjob on each workstation, or remotely from a central administration host. Any streaming device can be used for storing the data, but afbackup is designed for using tapes.

afbackup also provides authentication of clients, access restrictions for the streamer device, a lot of functionality for handling tapes, client-side per-file compression for reliability, client and server buffering for maximal throughput and a Tk-based graphical configuration tool.

afio

Author	: Koen Holtman
Last stable release	: October 10th, 2001
License	: GPL
Web site	: http://freshmeat.net/projects/afio/

Afio is a utility that makes cpio-format archives. It deals with input data corruption and supports multi-volume archives during interactive operation. Afio is best used as an "archive engine" in a backup script.

Amanda - Advanced Maryland Automatic Network Disk Archiver

Author	: James da Silva
Last stable release	: June 27th, 2003
License	: BSD
Web site	: http://www.amanda.org/

Amanda is backup utility developed at the University of Maryland. It's main goal is to back up multiple hosts to a single backup drive. According to John R. Jackson and Alexandre Oliva, which are members of the Amanda Core Development Team, Amanda is as advanced as a free backup utility gets and that Amanda has an installed base of at least 1500 sites. Amanda uses native dump and/or GNU tar and can back up a large number of workstations running multiple versions of Unix.

The Amanda approach is to use a "holding disk" on the tape server machine, do several dumps in parallel into files in the holding disk, and have an independent process that moves the data from the holding disk to a tape device.

Amanda provides its own network protocol on top of TCP and UDP. Each client backup program writes data to standard out, which Amanda collects and transmits to the tape server host. This allows Amanda to use compression and/or decryption on the data transferred over the network. [Pre99, pg. 146-150]

Arkeia

Author	:	Arkeia Corporation
Last stable release	:	July 28th, 2003
License	:	Proprietary
Web site	:	http://www.arkeia.com/

Arkeia is a commercial backup system developed by the former Knox Software, now Arkeia Corporation. It is mainly developed for use with a tape device or bigger robotic tape library. According to their web site[ark] Arkeia's server software is available for many different operating systems including AIX, HP-UX, IRIX, Linux, Solaris, Tru64 and UnixWare. They has client software for even more operating systems, e.g. BSD, Digital Unix, Mac OS X, Novell and Windows 95/98/NT4/ME/2000/XP/2003.

According to Marcel Gagné, Arkeia is a sophisticated and powerful backup solution with some initial strangeness, a *cool* graphical interface with some cryptic error-messages and lack of troubleshooting information. In addition to the regular product with a regular license Arkeia offers a product they call Arkeia Light, which is a free version of the same product but with support for only one Linux server, one tape drive and two clients in either a personal or commercial environment. [Gag03]

Backup Exec

Author	:	Veritas Software
Last stable release	:	January 22th, 2003
License	:	Proprietary
Web site	:	http://www.veritas.com/us/products/backupexec/

Backup Exec is a high-performance, easy-to-use, flexible network backup-and-restore solution designed for departmental and workgroup environments. Backup Exec provides certified solutions for Microsoft Windows and Novell NetWare environments. Veritas web pages do not provide information about any solutions that fits in a clean Linux environment. [bacb]

BackupPC

Author	:	Craig Barratt
Last stable release	:	June 14th, 2003
License	:	GPL
Web site	:	http://backuppc.sourceforge.net/

BackupPC is a system for backing up Linux and Windows PCs and laptops to a server's disk. According to their web site BackupPC is highly configurable and easy to install and maintain. BackupPC is written in Perl and extracts backup data via SMB using Samba, tar over SSH/rsh/NFS, or rsync.

One of the most important features for BackupPC is that identical files from multiple backups from the same or different PCs are stored only once resulting in substantial savings in disk storage and disk I/O. Other important features is compression support, a powerful web user interface, mobile environment support (laptops using dynamical IP's), periodical email's to PC-users that haven't had their PC backed up for a while, and detailed documentation.

According to BackupPC's web site, BackupPC is tested on Linux, Freenix, and Solaris hosts, and Linux, Win95, Win98, Win2000 and WinXP clients.

Bacula - The Network Backup Solution

Author	: Kern Sibbald
Last stable release	: August 1st, 2003
License	: GPL
Web site	: http://www.bacula.org/

Bacula is a set of computer programs that permits backup management, recovery and verification of computer data across a network of computers of different kinds. In technical terms, it is a network client/server based backup program. Bacula is designed to use removable media for storage, but it also support hard drives.

According to their web site [baca] Bacula is relatively easy to use and efficient, while offering many advanced storage management features that make it easy to find and recover lost or damaged files. Bacula has been built and tested on: RedHat, Solaris and FreeBSD.

BRU Backup and Restore Utility

Author	: The Tolis Group, Inc.
Last stable release	: -
License	: Proprietary
Web site	: http://www.tolisgroup.com/

BRU is a series of backup software developed by the Tolis Group. According to their web site [tol], BRU-Pro, the Tolis Groups highest level application, supports simultaneously backups of multiple clients to different tape drives, network traffic encryption, network data compression, end-user restore of personal files and additional more or less interesting features.

BRU has server software available for Linux and client software available for Linux, x86-based Windows, Mac OS X, and a lot of Unix systems.

Burt - The backup and recovery tool

Author	: Eric Melski and Dean Jansa
Last stable release	: May 12th, 1999
License	: Freely distributable own license
Web site	: http://www.cs.wisc.edu/~jmelski/burt/

Burt is a freely distributed parallel network backup system written at the University of Wisconsin, Madison. It is designed to backup large heterogeneous networks. Burt uses Tcl and standard backup programs like dump and tar. It is able to back up many different data sources, including UNIX and Windows workstations, AFS based storage, and others. Burt supports parallel backups to ensure high backup speeds and checksums to ensure data integrity. The principal contribution of Burt is that it provides a powerful I/O engine within the context of a flexible scripting language; this combination enables graceful solutions to many problems associated with backup of large installations. [Mel99]

Dirvish

Author	: J.W. Schultz
Last stable release	: July 15th, 2003
License	: GNU
Web site	: http://www.pegasys.ws/dirvish/

Dirvish is a utility that maintains multiple backups on online storage. The utility is built up on rsync, and each backup is available as a sort of snapshot directory, where common files are shared between the different backup generations. For changed files only those parts that actually change are transfered over the network.

Even though the backups are made incrementally, each image can be used independently for restores or to make off-site copies or archives. The removal of an image will have no effect on other images.

FauBackup

Author	: Volmar Sieh and Martin Waitz
Last stable release	: September 2nd, 2003
License	: GPL
Web site	: http://faubackup.sourceforge.net/

FauBackup is a backup system which uses a file system on a hard drive as backup media. The way FauBackup does incremental backup, is to hard-link the files which are not changed since last backup. All backups can be accessed using standard file system tools like ls, find, cp et al. FauBackup support usage of SSH to store the backup on a remote system or fetch the files to backup from a remote client.

It is possible to configure FauBackup to do a specified number of daily, weekly, monthly and yearly backups, i.e. FauBackup cleans up/deletes the old backups that is not supposed to exist.

Hdup

Author	: Miek Gieben
Last stable release	: August 9th, 2003
License	: GPL
Web site	: http://www.miek.nl/projects/hdup/hdup.shtml

Hdup is a tar-based backup utility which main features are encryption, remote storage and incremental dumps. It has been tested on Linux (packaged for Debian and Gentoo), Solaris and FreeBSD, but does also work on other versions of Unix.

Hdup uses three backup "formats"; monthly, weekly and daily. The monthly backup is a full dump of the file system, the weekly backup is an incremental dump of the filesystem related to the latest monthly backup and the daily backup is a incremental dump of the filesystem reacted to the latest weekly backup.

Legato NetWorker

Author	: Legato Software
Last stable release	: -
License	: Proprietary
Web site	: http://portal1.legato.com/products/networker/

With over 100.000 server licenses worldwide Legato NetWorker does centralizing backup and recovery operations across UNIX, Windows, Linux, OpenVMS, Macintosh and NetWare platforms. Advanced indexing and media management, cluster support, high speed parallelism, cross-platform tape interoperability, comprehensive NDMP support, backup to disk, tape cloning, archive, server-less backup, and dynamic drive sharing are among key components that Legato Software announce on their website[leg].

Mondo Rescue

Author	: Hugo Rabson
Last stable release	: August 1st, 2003
License	: GPL
Web site	: http://www.microwerks.net/ hugo/

Mondo Rescue does backup to tape, CD-R, CD-RW, NFS or a hard disk partition. Its main function is to create a set of CD's that can be used to do a complete restore of the system. Mondo Rescue supports the most used filesystem on the Linux platform and some others too.

In addition to full rescue of a system, Mondo Rescue also offers additional functionality, like accidently deleted files.

MultiCD

Author	: Daniel Born
Last stable release	: April 4th, 2003
License	: GPL
Web site	: http://danborn.net/multiCD/

MultiCD can be configured to backup your whole or parts of your Linux system to one or more CD's. MultiCD does no compression on the files, just make a CD image of them, and if you want, burns it out to one or more CD's.

rdiff-backup

Author	: Ben Escoto
Last stable release	: August 8th, 2003
License	: GPL
Web site	: http://rdiff-backup.stanford.edu/

rdiff-backup is a Python program that backs up one directory to another. The target directory ends up as a copy of the source directory, and reverse diffs are stored together with the backup to ensure that you can recover files lost some time ago.

rdiff-backup also handle subdirectories, sym-links, special files, permissions, uid/gid ownership and modification times. SSH or rsync can be used to back up to a remote computer, and only the differences are transmitted.

rlbackup - Remote Linked Backup

Author	: John C. Bowman
Last stable release	: July 29th, 2003
License	: GPL
Web site	: http://www.math.ualberta.ca/imaging/rlbackup/

Remote Linked Backup does as the name describes, takes backup to a remote location and uses hard linking between the same files in different backup directories. Every backup becomes a snapshot, and the snapshots are independent of each others. rlbackup uses rsync to do the backup task.

Storebackup

Author	: Heinz-Josef Claes
Last stable release	: August 30th, 2003
License	: GPL
Web site	: http://sourceforge.net/projects/storebackup/

Storebackup is a backup utility that stores files on other disks. It includes several optimizations that reduce the disk space needed and improve performance, and unifies the advantages of traditional full and incremental backups. It includes tools for analyzing backup data and restoring.

The only option on backing up to a remote file system is to use NFS. Once archived, files are accessible by mounting file system (locally, or via Samba or NFS). Storebackup also includes a tool for System Administrators to restore (sub)trees.

3.2.1 Preliminary sorting

In this section the backup systems that not do satisfy the requirement listed in chapter 3.1.2 will be disqualified.

The first requirement is that the software had to be distributed with a license that makes it possible to distribute it together with the Skolelinux CD.

The following software fail the requirement of a satisfying license:

- Arkeia
- Backup Exec
- BRU Backup and Restore Utility
- Burt
  Burt has a sort of free license, and it have been discussed on the debian-legal mailing list, and considered as non-free in a thread[Did99] started by Paolo Didonè.
- Legato NetWorker

The backup systems that failed due to the license, will not be considered when the other requirements are considered. This because there may be some problems with testing some of them without a license.

The second requirement is that the backup system must support the architecture, the software and the hardware used in Skolelinux. To achieve this requirement and the third requirement, which is that it must be possible to place the backup server on another network than the file server and LTSP server(s), the backup software used in the backup system must allow a scenario where the file server only uses the backup server as a remote hard drive. I.e. the client-software installed on the file server (possibly also on the LTSP server(s)) has to take the initiative to each backup session.

The following software fail these requirements:

- **Amanda**
  Amanda fails at this point because the Amanda backup server is the component that connects to the client when a backup session is initialized. When the backup server must have the possibility to be placed on another network than the rest of the Skolelinux network, the request from the backup server has to go into a network which is NATed (see the Skolelinux architecture document [Rei02]). This is not impossible to solve, but nothing that will work out of the box when Skolelinux does not include a gateway or rules for such gateway.

- **BackupPC**
  BackupPC is designed to take backup of a lot of workstations, including laptops that are not continuous connected to the network. Like Amanda, it is the server part of BackupPC that request the clients when backup is to be taken, and this does not fit well in a NATed network.

- **Bacula**
  The Bacula Director service is one of the services running on the backup server and is the service that request backups from the Bacula file services on the clients. Like Amanda, this does not fit well in a NATed network.

- **Dirvish**
  When using Dirvish the backup server is the host that initializes the backup session with the clients. Like Amanda, this does not fit well in a NATed network.

- **Hdup**
  Hdup can be used perfectly to backup to a remote location, but when it comes to the restore part, you will have to start the restore process from the remote machine, which will not fit well in a NATed network.

- **Storebackup**
  The way Storebackup stores backup to another host over the network, is using NFS and when the backup server must have the possibility to be placed on another network than the rest of the Skolelinux network, NFS is a bad idea.

The fourth requirement is that the communication between the backup server and the machines that shall be backed up, must be encrypted. This requirement does not exclude any of the backup systems, because it is possible to transmit almost any stream of data over an SSH connection.

The fifth requirement is also an argument that does not exclude any of the backup systems reviewed in this project. For all of them, it is possible to choose the files you want to back up.

The sixth requirement is that uid/gid ownership and privileges must be stored. None of the remaining backup software has problems with achieving this.

The seventh requirement is that the software has to store the backup to a hard disk. The following software fail these requirement:

- **afbackup**
  afbackup is primary designed to back up to a streaming device. afbackup's documentation says nothing about how to use a hard disk or a file on a hard disk as a streaming device, therefore afbackup fails at this requirement.

The eighth requirement is that the backup system must handle snapshots in such way that it is possible to restore files from different points in time. The following software fail these requirement:

- **afio**
  afio is a script that creates cpio-formated archives, and afio does not deal with snapshots. afio is, as the Freshmeat project page [afi] says, "afio is best used as an 'archive engine' in a backup script".
- **Mondo Rescue**
  Mondo Rescue is a backup system designed to make it possible to recover from scratch if necessary. In one way, you can say that Mondo Rescue can keep different snapshots of your backup on different CD's, but this is considered a much too time consuming operation, so it is disqualified.
- **MultiCD**
  MultiCD was created to take a full backup of a system to one or more CD's. It is also possible to choose which files/directories to backup, but in the same way as Mondo Rescue, MultiCD is disqualified because of the too time consuming way of restoring files from different points of time.

3.2.2 Test: the remaining software

These backup systems have passed all the requirements:

- FauBackup
- rdiff-backup
- rlbackup

In the rest of the test chapter, each of the backup systems will be tested against the requirements and some other functionality to see if one of them point it self out as crystal clear candidate. The requirements checked in the previous section will not be tested if not one of the backup systems is much better than the other. There are logs that describes the procedure carried out for each software system as appendixes to this report. The FauBackup log is in appendix A, rdiff-backup log is in appendix B and the rlbackup log is in appendix C.

The computers used to test these backup systems are one Pentium Celeron 450MHz and one Pentium III 1000 MHz both with Debian Stable installed. The software was not installed using any precompiled packages, but the tar.gz-files offered on the backup softwares web sites (see the logs for details).

Installation

The first thing one notices when doing a software-test is how easy/hard the software is to install. FauBackup installed smoothly and without any problems. The installation of rdiff-backup also went very smooth. rlbackup was a bit more problematic. To get the software compiled, the -ansi-option for g++ in in the Makefile had to be removed (see appendix C for details).

File ownership and privileges

In a FauBackup- and rlbackup repository the files and directories keeps its uid/gid ownership and privileges, while rdiff-backup stores this both in separate files and with the files.

All the backup systems handle soft links and hard links. FauBackup also handles files with holes. rdiff-backup and rlbackup does not handle files with holes (see logs for details).

Snapshot storage

FauBackup first copies the whole directory from the client to the backup server, and then compares the newest backup to the last backup and then hard linking of the identical files. The negative part here is that all files are copied at each backup, which leads into a lot of network usage. FauBackup gives you the opportunity to choose how many daily, weekly, monthly and yearly backups to keep on the backup server.

rdiff-backup has the last snapshot stored as the primary mirror, and uses reverse diff-files to store the differences between newer and older files. rdiff-backup does not have any functionality to choose the numbers of daily, weekly, monthly or yearly backups to keep, but gives the user the possibility to delete every backup that is older than backup at a specified time.

rlbackup has one directory for each snapshot and hard-links equal files between snapshot directories. rlbackup uses rsync to compare files in the repository with the fresh files. rsync calculates deltas for the whole backup set before it starts transferring information to the backup server. According to Jonas Smedegaard [Sme03] this can make both the backup server and the client to crack if the data set is to large.

With rlbackup you can specify how sequentially backups you want to retain. This option is explained like this in the rlbackup configuration file:

> The number (≥ 2) of sequential backups to retain. After this number, retain only every second backup, then every fourth backup, and so on, for each successive power of 2 (except for the necessary intermediate files).

Hard disk usage

In this test all backup systems were set to backup a home directory every night. To get more changes to the files, a copy of Skolelinux CVS-repository was updated in the home directory before each backup. The home directory has been growing from about 600MB to about 950MB. After these seven weeks FauBackup did use 3.2GB, rdiff-backup 1.3GB and rlbackup 2.3G.

Velocity

During each backup session run in this test, the time used was logged to see the differences between the backup systems. The average time elapsed during FauBackup's sessions was about six minutes, rdiff-backup about four minutes and rlbackup about three minutes.

Restoring

While none of the FauBackup man-pages have any information on how to restore at all, most likely this is supposed to be done manually with a tool like scp, FTP or NFS. rdiff-backup and rlbackup do not have an option that allows the end-users to restore their own files (you have to have system administrator privileges), the only restoring can be done by SA. With all systems you have the possibility to give the users read-access to the files in a secure way via the snfs filesystem or you can make an additional piece of software that runs as root and prepares a zip- or tar-file with the files the user requests.

Debian prepared (packaged)

FauBackup is packaged for Debian, version 0.5-pre1 in the stable archives, 0.5.1 in testing and 0.5.2 in unstable (0.5.2 is the latest release). One advantage for FauBackup is that the developer, Martin Waitz, also is the Debian Developer responsible for packaging it. There have been some bug fixing between the 0.5-pre1 and 0.5.2, so if FauBackup is the software chosen, we want to include backporting the 0.5.2 version to Debian stable as a part of the project.

rdiff-backup is packaged for Debian, but in the stable archives only version 0.6.0 is present, so if rdiff-backup is chosen as the backup software to use in this project, the project have to include backporting the newest release of the software to fit into Debian's stable release. This should not be a big problem, as the newest release of rdiff-backup easily installed on a Debian stable computer.

rlbackup is not packaged for Debian at all, so the project has to include packaging it in case of using it in this project. As there was some problems with compiling and installing the newest release of rlbackup on a Debian stable computer and the fact that rlbackup has not been packaged for Debian at all, there may be more difficult to package rlbackup than FauBackup or rdiff-backup for Debian stable.

Propagation

Just to get a little feeling about the size of the systems, this is the results of searches on the names (FauBackup, rdiff-backup and rlbackup) on AllTheWeb and Google (the searches was performed October 7th):

AllTheWeb		
FauBackup	-	1,476 results
VBackup	-	868 results
rdiff-backup	-	3,554 results
rlbackup	-	11 results

Google		
FauBackup	-	1,880 results
VBackup	-	1,010 results
rdiff-backup	-	7,400 results
rlbackup	-	53 results

These results **just** shows us that probably rdiff-backup is more in use and discussed than FauBackup, and rlbackup is not very discussed on public Internet forum. (We will have in mind that FauBackup was renamed from VBackup to FauBackup early in 2002.)

Documentation

When using a software package, it is important that the documentation following the package is good. FauBackup's documentation consists of four man-pages which is quite informative but misses examples. rdiff-backup has a good documentation page on the web site, including e.g. examples of usage, FAQ, man-page and more detailed information about rdiff-backup. The only documentation rlbackup has, is the installation routines which includes examples on how to take backup and how to recover.

Activity and support

When going to choose a Open Source software package to use in a project like this, it is positive that the software development team is active in developing and supporting their product. If they are, you will likely get a more positive experience when reporting a bug or needing help.

FauBackup has mainly one developer, Martin Waitz (working at Dept. of Computer Science 3, Friedrich-Alexander University Erlangen). The last six months two new releases have been announced (from v0.5.1 March 12th to v.0.5.2 September 2nd). The average number of postings on the mailing list is 3 per month. The author, Martin Waitz, is one of the most active persons on the mailing list and almost always responds quickly to questions. The FauBackup web site is simple, but informative.

rdiff-backup also has mainly one developer, Ben Escoto (graduate at Stanford University). The last six months seven new stable releases have been announced (from v0.11.4 Mars 15th to v0.12.4 September 13th [Esc03a]). The average number of postings on the mailing list is 70 per month, and the number is increasing from about 50 in April to about 120 in September. The author, Ben Escoto, is one of the most active persons on the mailing list and almost always responds quickly to questions. The rdiff-backup website is well organized, informative, and frequently updated.

rlbackup also has mainly one developer, John C. Bowman (professor at Dept. of Mathematical Sciences, University of Alberta). The last six months two new releases has been announced (v2.05 May 21 to v2.06 July 29th [Bow03]). rlbackup does not have a public mailing list. The rlbackup website is quite informative and was last updated when the v2.06 was released.

3.2.3 Conclusion

The Debian packaging is a very important issue for Skolelinux, as they do not want to have the package responsibility for more packages than necessary.

Mixing this point with the installation phase, FauBackup and rdiff-backup scores almost equal and rlbackup is some steps behind.

With FauBackup, you can decide how many daily, weekly, monthly and yearly backups to keep, rlbackup gives you the opportunity to specify how sequentially backups you want to retain (as specified above), and rdiff-backup does only allow you to delete every backups that are older than a specified date. This gave FauBackup full score, rlbackup medium score and rdiff-backup zero score.

But the fact that rlbackup uses rsync, and that rsync, as mentioned above, can make both the backup server and the client to crash, gives rlbackup negative score.

The disk usage made rdiff-backup to a winner, before slbackup on a second, and FauBackup last. There was big difference between the three.

When time usage is considered, FauBackup was the slowest, with rdiff-backup on a second, and slbackup as the fastest. As Faubackup transfers all the data at each backup session, this was not a surprising result. slbackup and rdiff-backup scores highest on this one.

When it comes to propagation, activity and support, due to the very active mailing list, the responsive author, good documentation and frequent releases rdiff-backup is the backup software that scores absolutely best. FauBackup and rlbackup scores almost equal, but it is negative for rlbackup that it does not have a public mailing-list.

All things considered, rdiff-backup is winner backup system in this project. The main reasons for this is Debian packaging, disk usage, network bandwidth usage, documentation, activity, and support.

Chapter 4

Design

This chapter contains the design specification of the system. The first part is a description of the architecture of the system and the last part contains more detailed descriptions of some parts of the system. This chapter is written for those who are going to implement the system and those who want to understand how the system is designed after the implementation is finished.

4.1 Architecture

First an overview of the system is presented, before the system is split in four different times of view; configuration, backup, restore and maintenance. Two types of diagrams are used in this chapter, a sort of flowchart diagram and a sort of sequence diagram (not following the standard, but the ideas are taken from UML sequence diagrams). The flowchart diagram shows the three different computers in the project as rectangular boxes containing the storage medias and the processes that are playing vital parts in this project. The sequence diagrams show the interaction between the SA, the processes and the storage medias in the different views of time as described above.

4.1.1 Overview

The system contains one file server, one backup server and one or more LTSP-server(s). Both the backup server and the LTSP server can be included as a part of the file server, but in the rest of this chapter they will be treated as separate computers. Figure 4.1 is presenting the different computers in the system with their relevant storage systems and processes running.

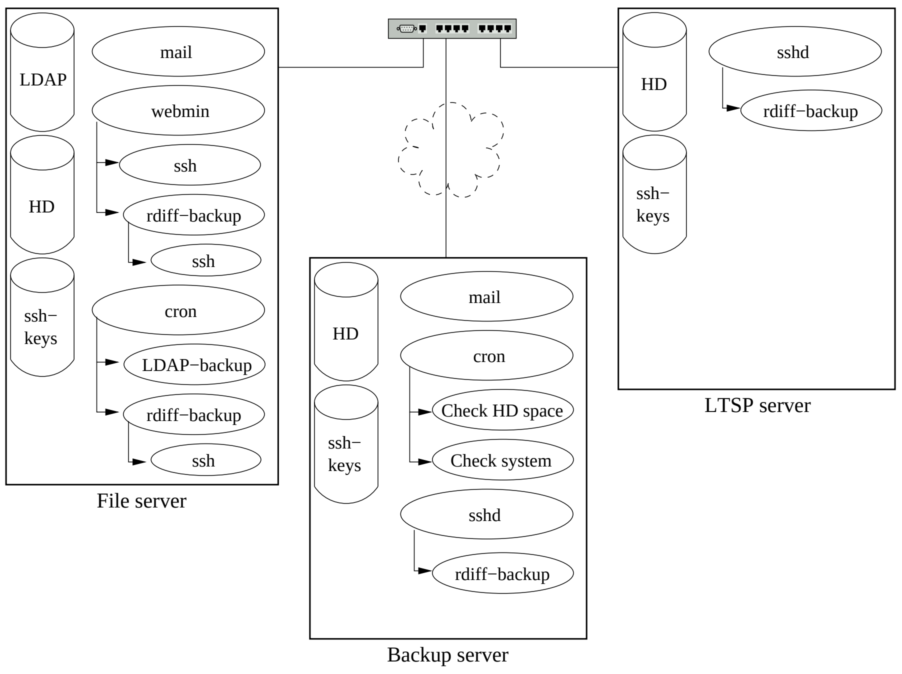


Figure 4.1: Overview of the system with relevant processes and storage systems

File server

The file server has mail functionality for use by the other processes that runs on the server. It also handles the configuration of the backup system, the configuration files, and the Webmin interface between the SA and the backup system. There are cron jobs that handles both backup of the LDAP database, backup of its own files, and backup of the LTSP servers files.

Backup server

The backup server also has mail functionality if some of the other processes that runs on the computer needs to report something to the SA. There are some cron jobs running to check if the hard disk has enough free space and checking other parts of the system. A SSH daemon runs so the file server is able to connect, and the backup system also starts a rdiff-backup process that runs in server-mode when the backup session is running.

LTSP server(s)

On the LTSP server the only process that is relevant to the backup system is the SSH daemon that serves the file server with SSH connections. The backup process on the file server starts a rdiff-backup process in server-mode on the LTSP server(s) when the backup session is running.

4.1.2 Configuration

This phase is where the System Administrator, SA, is configuring the system. This includes specifying what to back up, when to backup, the SA's mail address, the maximum age of backups to keep, and handling SSH-keys.

In figure 4.2 the processes that are relevant to the configuration phase are filled. Figure 4.3 shows the interaction between the processes.

The following list describes the set of arrows in figure 4.3:

1. The interface between the SA and the system is a Webmin-module. The Webmin module is described in further detail in chapter 4.3.

2. The Webmin module stores its configuration in cron jobs and other files on the hard disk.

3. Webmin also have functionality to handle the SSH keys used by the backup process to connect to the backup server and LTSP server(s).

4. SSH is used by Webmin to send the public part of the SSH key to the backup server and the LTSP server(s).

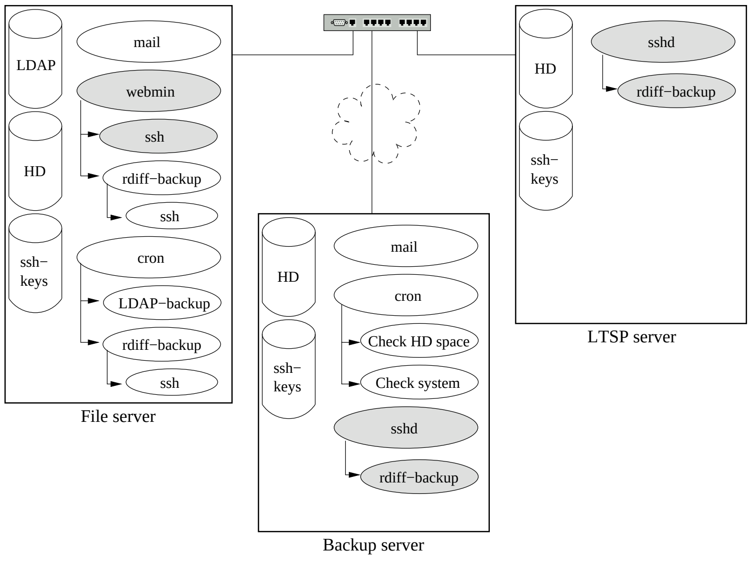


Figure 4.2: Overview of the system with the processes that are relevant to the configuration phase filled

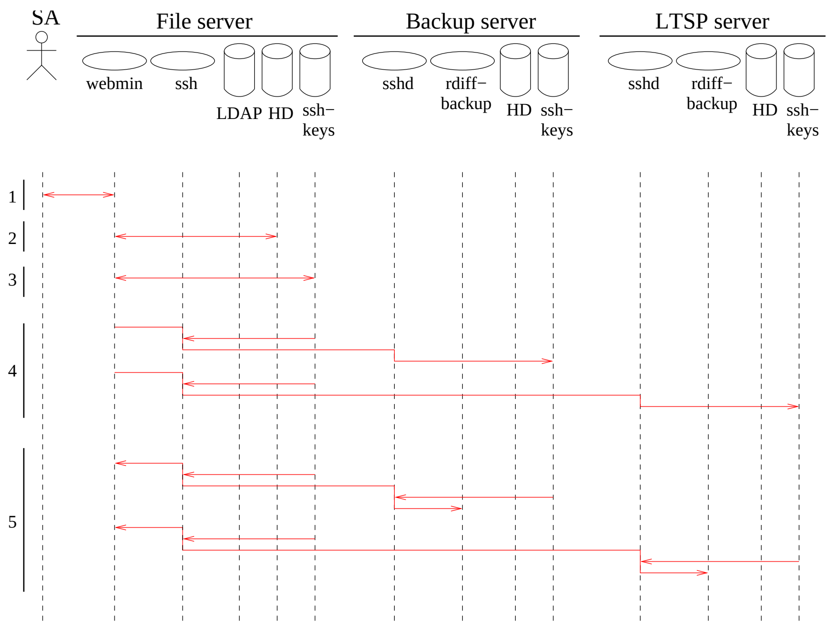


Figure 4.3: Interaction between the active processes in the configuration phase

5. SSH is also used by Webmin to test if the connection between the backup server and the LTSP server(s) is working properly and to verify that rdiff-backup is installed and working on the remote computers and that it has the same version on all computers.

4.1.3 Backup

This phase is when the backup is running and store files from the file server and LTSP server(s) on the backup server. Figure 4.4 shows which processes that are active in the backup process and figure 4.5 illustrates the interactions between the processes.

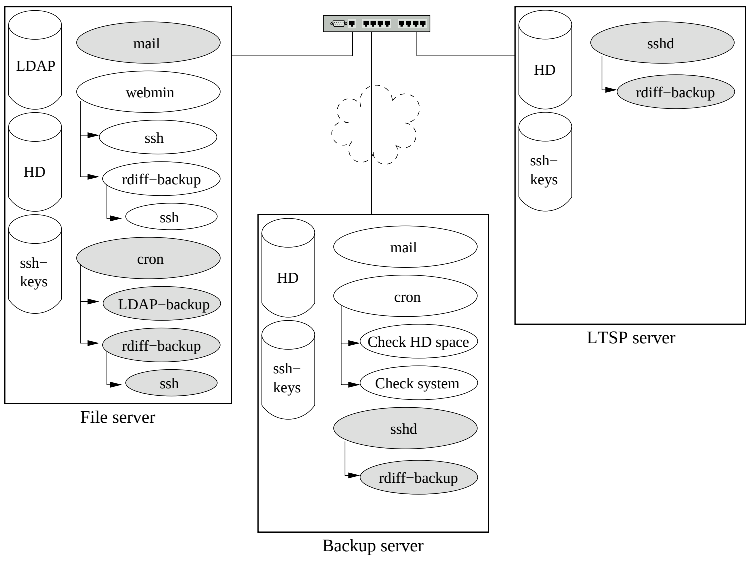


Figure 4.4: Overview of the system with the processes that are relevant to the backup phase filled

The following list describes the set of arrows in figure 4.5:

1. A cron job that stores the LDAP database in a file on the hard disk. If a failure appears, the cron job will send an email to the SA.

2. A cron job that takes backup of specified files and directories on the file server. rdiff-backup uses SSH to send data over the network to the backup server. Because of the SSH keys that are generated in the installation / configuration phase, no passwords are needed.

3. Like 2, but it is the LTSP server(s) and not the file server that is backed up.

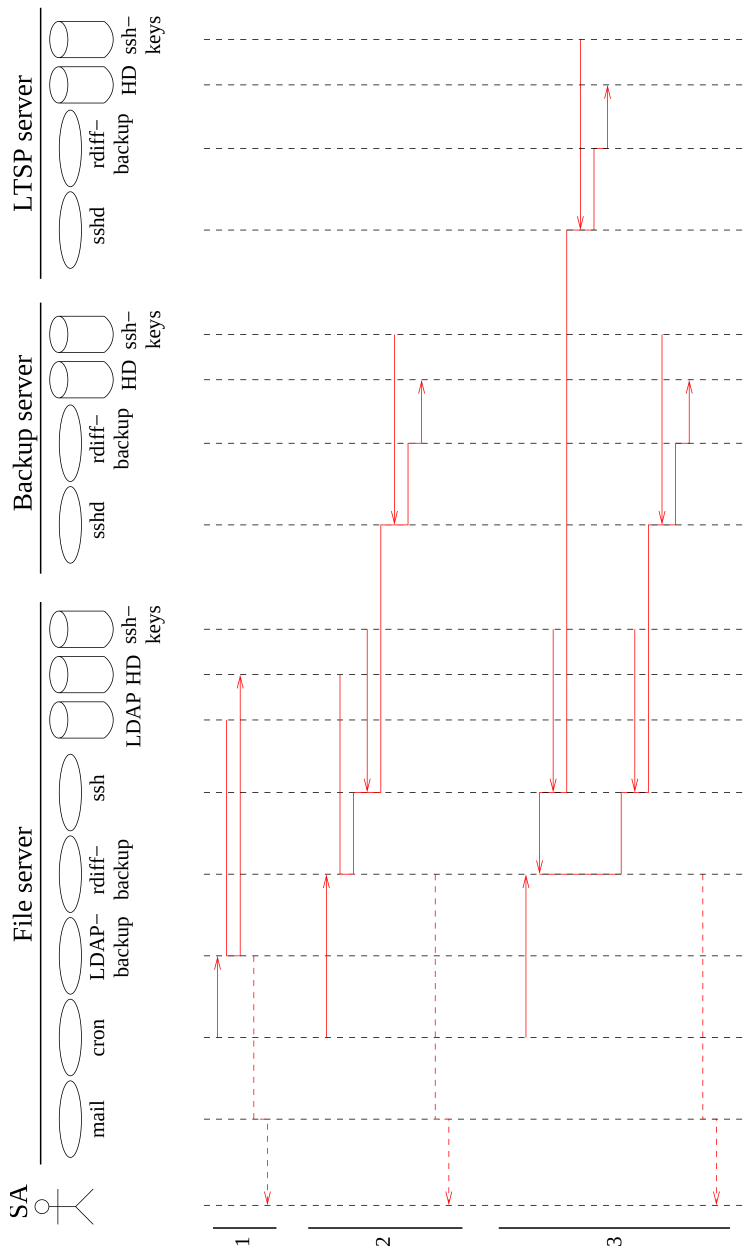


Figure 4.5: Interaction between the active processes in the backup phase

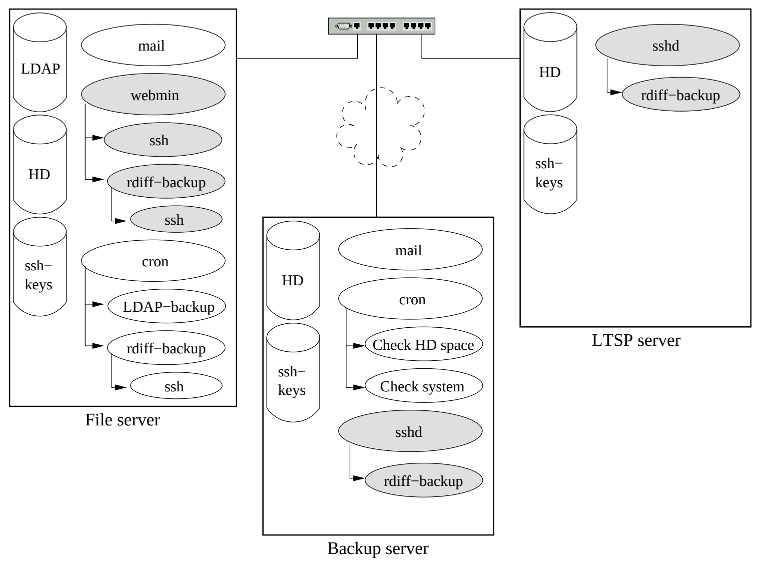


Figure 4.6: Overview of the system with the processes that are relevant to the restore phase filled

4.1.4 Restore

This is the phase where the SA restores files from backup. Figure 4.6 shows which processes that are active in the restore phase and figure 4.7 illustrates the interaction between the processes.

The following list describes the set of arrows in figure 4.7:

1. The restore is initiated via the Webmin module, and a rdiff-backup process is started to restore files from the backup server to the file server. Here, SSH is also used to send data over the network, and the SSH keys prevent any passwords handling.
2. The same as 1, but the files are restored to the LTSP server(s) instead of the file server.

4.1.5 Maintenance

In this phase the backup server is checking that everything is working properly. If anything goes wrong, the system will warn the SA through an email describing the problem(s).

Figure 4.8 shows which processes that are active in the maintenance phase and figure 4.9 illustrates the interaction between the processes. The following list describes the set of arrows in figure 4.9:

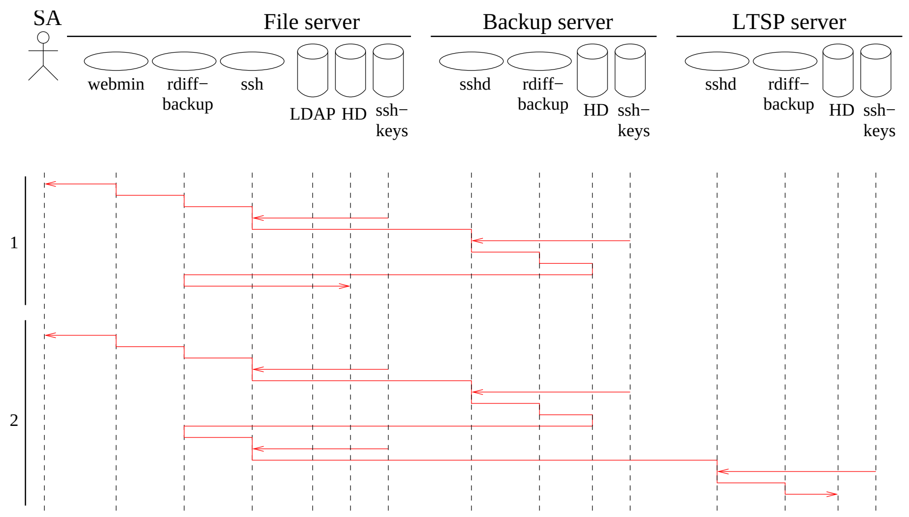


Figure 4.7: Interaction between the active processes in the restore phase

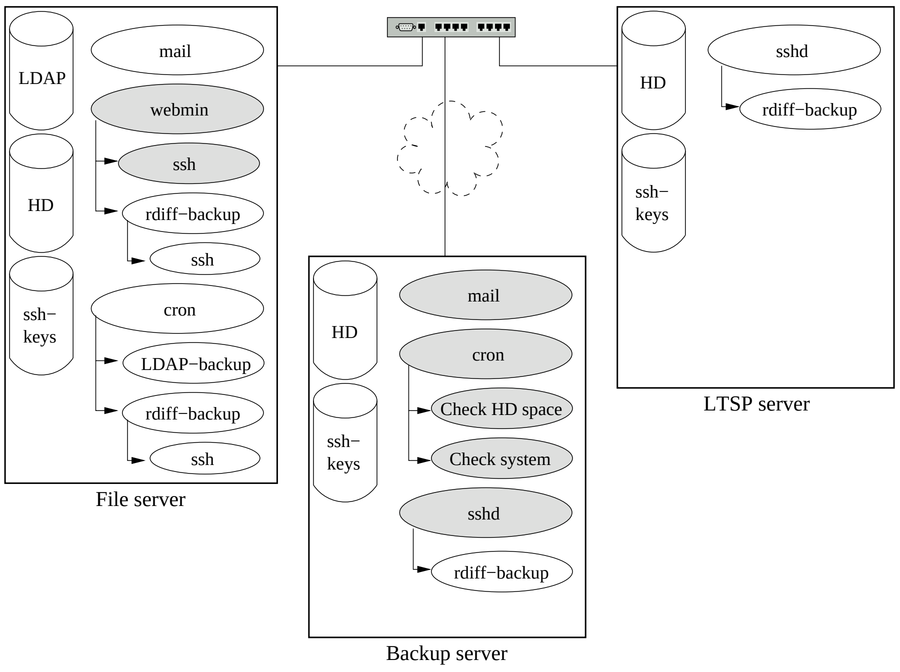


Figure 4.8: Overview of the system with the processes that are relevant to the maintenance phase filled

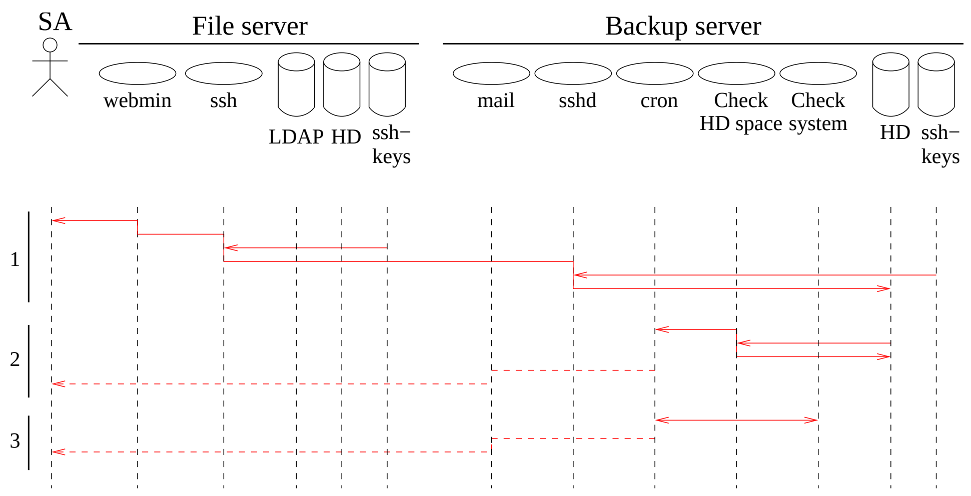


Figure 4.9: Interaction between the active processes in the maintenance phase

1. It is possible to manually delete old backups from the Webmin interface.

2. One important thing to check on a backup server is if the disk is getting filled up, so the free space on the hard drive is crossing a specified limit, a mail will be sent to the SA with a warning.

3. Other system preferences can also be monitored and send warnings to the SA by mail if anything is wrong.

4.2 Software packages

This section describe the software structure in this project. There will be written several pieces of software, which will be packaged in different software packages:

- **slbackup**
  This package contains the configuration file, the script that starts the backup session, and the cron job for running rdiff-backup once a night.

- **webmin-slbackup**
  This package contains the Webmin module that the SA uses to configure slbackup, restore files/directories, do some manual maintenance, and configure the SSH keys.

- **slapd-backup**
  This package contains the configuration that sets up a replicating LDAP server and a cron job that dumps this database to a file once a day.

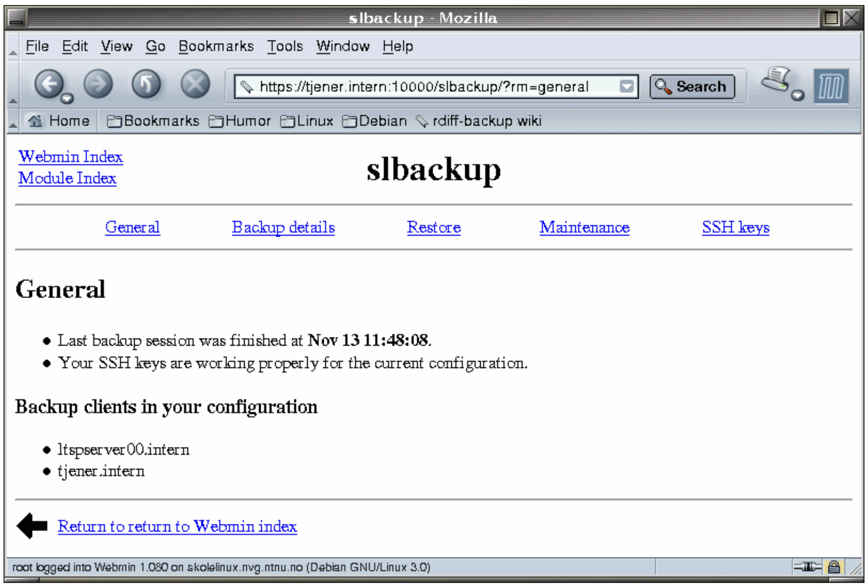
4.3 Webmin

The configuration interface is to be written as a Webmin module. The Webmin module for the backup system is called *webmin-slbackup* and has five main sections:

- **General**
  This section contains an introduction to the Webmin module and some status information about the backup system.
- **Backup details**
  This section is where the user specify all details about the backup configuration.
- **Restore**
  This is the section where the user can restore files and directories. There are two different option; full restore and restore one file or directory.
- **Maintenance**
  In this section the user can specify to delete backups that are older than a specified date.
- **SSH keys**
  This section is handling the SSH keys on the computers in the configuration.

4.3.1 General

The General page contains an introduction to the Webmin module and some statistics about the backup system. In figure 4.10, we can see what the General page look like.


slbackup - Mozilla
File Edit View Go Bookmarks Tools Window Help
https://tjener.intern:10000/slbackup/?rm=general
Search
Home Bookmarks Humor Linux Debian rdiff-backup wiki
Webmin Index
Module Index

slbackup

General Backup details Restore Maintenance SSH keys

General

- Last backup session was finished at **Nov 13 11:48:08**.
- Your SSH keys are working properly for the current configuration.

Backup clients in your configuration

- ltspserver00.intern
- tjener.intern

Return to return to Webmin index

root logged into Webmin 1.080 on skolelinux.nvg.ntnu.no (Debian GNU/Linux 3.0)

Figure 4.10: Snapshot of the "General" page

4.3.2 Backup details

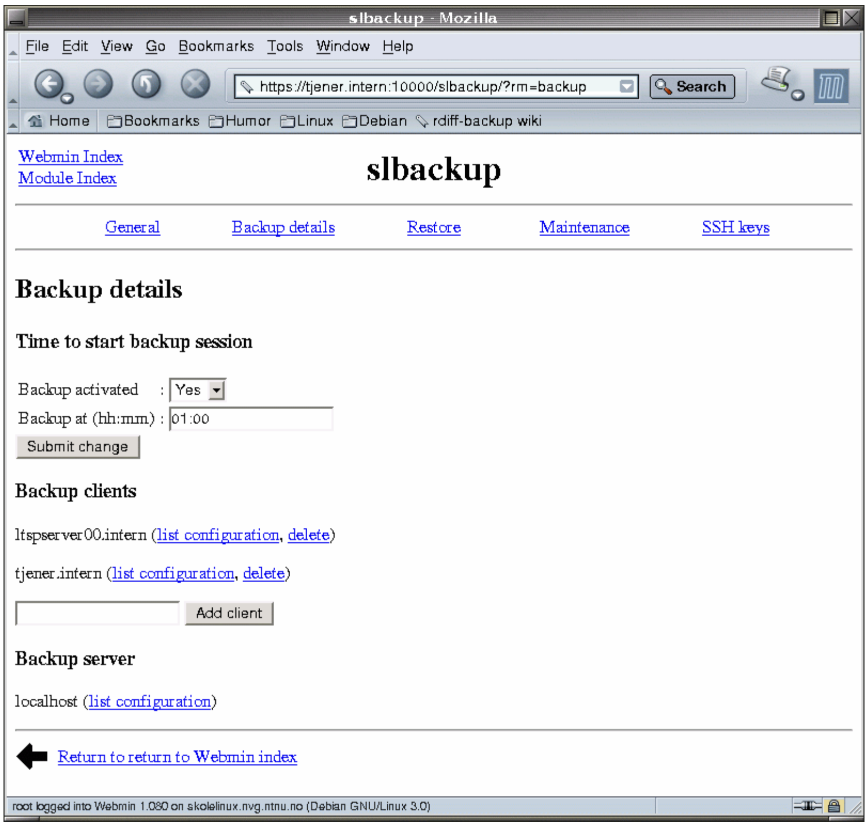
The Backup details page contains the configuration of the backup process. The main window of this section is shown in figure 4.11. On the Backup details page the following actions are supported:

- specify when the backup session is started (assumed to be once a day)
- configure clients
- add clients
- delete clients
- configure backup server

When a user change the time of day that the backup session shall start, he will be sent back to the "Backup details" page and be informed on the top of the page that the time has been changed.

When a user wants to configure a client he only has to click the configure link, and a configuration page will be shown as in figure 4.12.

When a user adds a client, he will be sent to a page where he can configure the client where the files/directories that is to be backed up on the client can be configured. This page is shown in figure 4.13.


slbackup - Mozilla
File Edit View Go Bookmarks Tools Window Help
https://tjener.intern:10000/slbackup/?rm=backup
Search
Home Bookmarks Humor Linux Debian rdiff-backup wiki
Webmin Index
Module Index
General Backup details Restore Maintenance SSH keys

slbackup

Backup details

Time to start backup session

Backup activated : Yes
Backup at (hh:mm) : 01:00
Submit change

Backup clients

ltspserver00.intern (list configuration, delete)

tjener.intern (list configuration, delete)

Add client

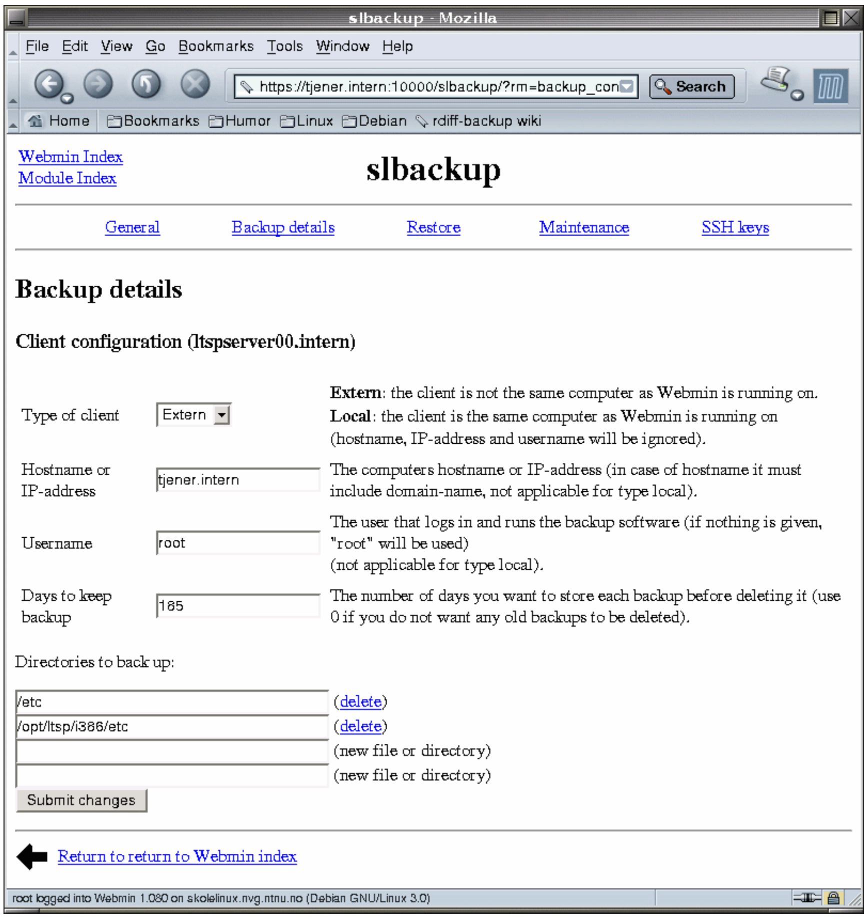
Backup server

localhost (list configuration)

Return to return to Webmin index

root logged into Webmin 1.080 on skolelinux.nvg.ntnu.no (Debian GNU/Linux 3.0)

Figure 4.11: Snapshot of the “Backup details” page


slbackup - Mozilla
File Edit View Go Bookmarks Tools Window Help
https://tjener.intern:10000/slbackup/?rm=backup_con
Search
Home Bookmarks Humor Linux Debian rdiff-backup wiki

Webmin Index
Module Index

slbackup

General | Backup details | Restore | Maintenance | SSH keys

Backup details

Client configuration (ltspserver00.intern)

Type of client	Extern	**Extern**: the client is not the same computer as Webmin is running on. **Local**: the client is the same computer as Webmin is running on (hostname, IP-address and username will be ignored).
Hostname or IP-address	tjener.intern	The computers hostname or IP-address (in case of hostname it must include domain-name, not applicable for type local).
Username	root	The user that logs in and runs the backup software (if nothing is given, "root" will be used) (not applicable for type local).
Days to keep backup	185	The number of days you want to store each backup before deleting it (use 0 if you do not want any old backups to be deleted).

Directories to back up:

/etc	(delete)
/opt/ltsp/i386/etc	(delete)
	(new file or directory)
	(new file or directory)

Submit changes

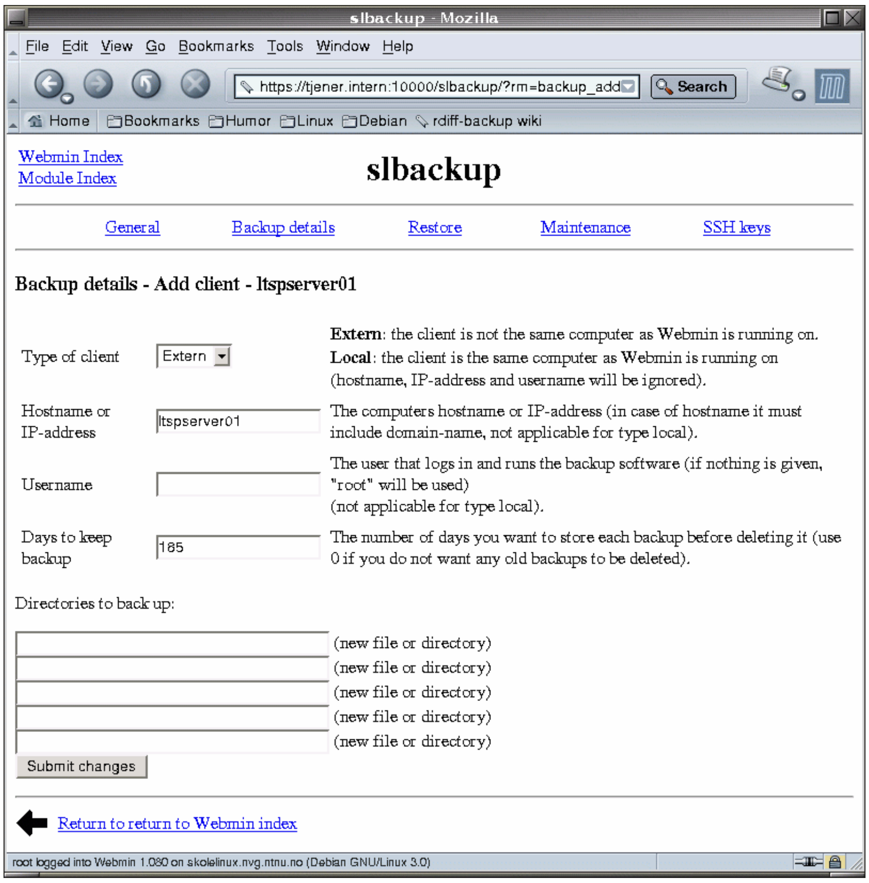
Return to return to Webmin index

root logged into Webmin 1.080 on skolelinux.nvg.ntnu.no (Debian GNU/Linux 3.0)

Figure 4.12: Snapshot of the "Configure client" page

When a user deletes a client, he will be asked to confirm the deletion as shown in figure 4.14. When the user press the "Delete"-button he will be sent back to the "Backup details" page (figure 4.11) and see a message that verifies that the deletion is successfully completed.

If a user chooses to configure the backup server, he will be sent to the configuration page shown in figure 4.15. When the user press the "Submit changes" he will be sent back to the same page with verify messages on the changes made.


slbackup - Mozilla
File Edit View Go Bookmarks Tools Window Help
https://tjener.intern:10000/slbackup/?rm=backup_add
Search
Home Bookmarks Humor Linux Debian rdiff-backup wiki
Webmin Index
Module Index
General
Backup details
Restore
Maintenance
SSH keys
Submit changes
Return to return to Webmin index
root logged into Webmin 1.080 on skolelinux.nvg.ntnu.no (Debian GNU/Linux 3.0)

slbackup

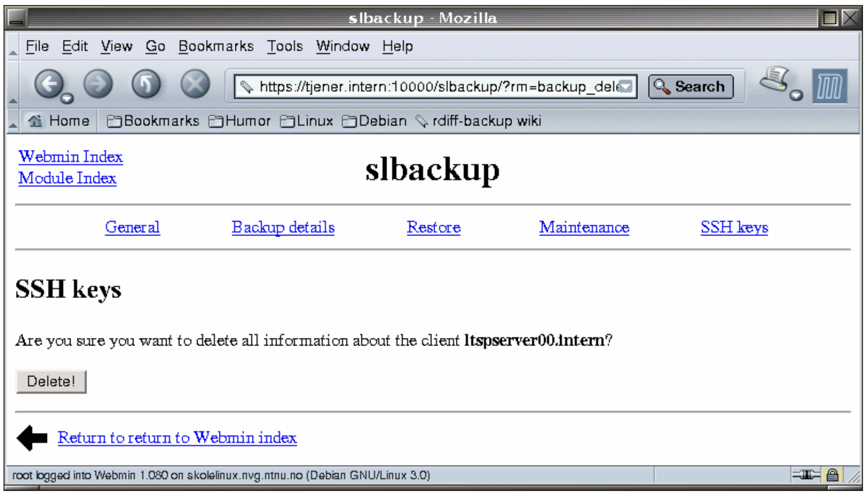
Backup details - Add client - ltspserver01

Type of client	Extern	**Extern**: the client is not the same computer as Webmin is running on. **Local**: the client is the same computer as Webmin is running on (hostname, IP-address and username will be ignored).
Hostname or IP-address	ltspserver01	The computers hostname or IP-address (in case of hostname it must include domain-name, not applicable for type local).
Username		The user that logs in and runs the backup software (if nothing is given, "root" will be used) (not applicable for type local).
Days to keep backup	185	The number of days you want to store each backup before deleting it (use 0 if you do not want any old backups to be deleted).

Directories to back up:

(new file or directory)
(new file or directory)
(new file or directory)
(new file or directory)
(new file or directory)

Figure 4.13: Snapshot of the “Add client” page

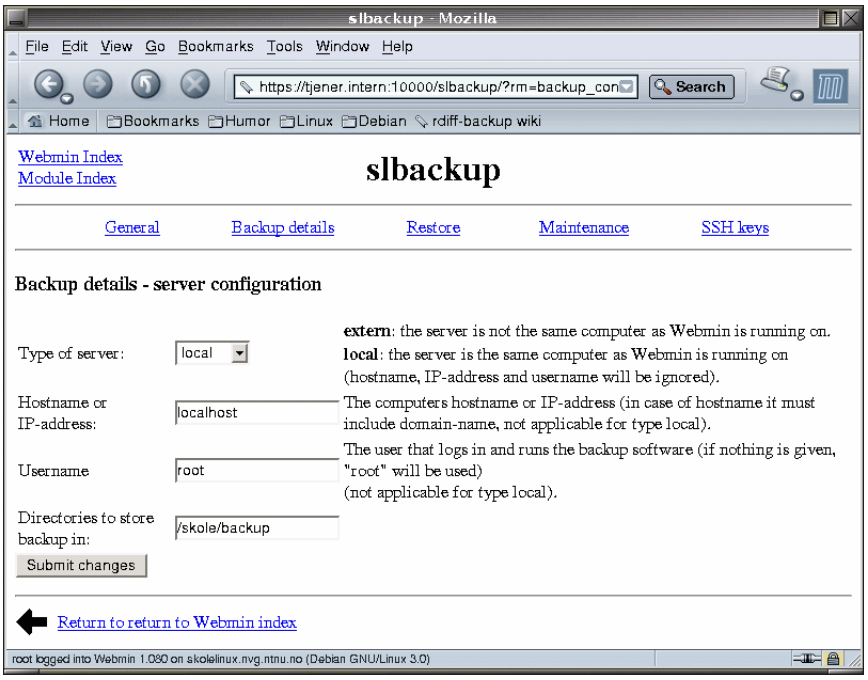

slbackup - Mozilla
File Edit View Go Bookmarks Tools Window Help
https://tjener.intern:10000/slbackup/?rm=backup_dele
Search
Home Bookmarks Humor Linux Debian rdiff-backup wiki
Webmin Index
Module Index
General
Backup details
Restore
Maintenance
SSH keys
Delete!
Return to return to Webmin index
root logged into Webmin 1.080 on skolelinux.nvg.ntnu.no (Debian GNU/Linux 3.0)

slbackup

SSH keys

Are you sure you want to delete all information about the client **ltspserver00.intern**?

Figure 4.14: Snapshot of the “Delete client” page


slbackup - Mozilla
File Edit View Go Bookmarks Tools Window Help
https://tjener.intern:10000/slbackup/?rm=backup_con
Search
Home Bookmarks Humor Linux Debian rdiff-backup wiki
Webmin Index
Module Index
General
Backup details
Restore
Maintenance
SSH keys
Submit changes
Return to return to Webmin index

slbackup

Backup details - server configuration

Type of server:	local	**extern**: the server is not the same computer as Webmin is running on. **local**: the server is the same computer as Webmin is running on (hostname, IP-address and username will be ignored).
Hostname or IP-address:	localhost	The computers hostname or IP-address (in case of hostname it must include domain-name, not applicable for type local).
Username	root	The user that logs in and runs the backup software (if nothing is given, "root" will be used) (not applicable for type local).
Directories to store backup in:	/skole/backup	

root logged into Webmin 1.080 on skolelinux.nvg.ntnu.no (Debian GNU/Linux 3.0)

Figure 4.15: Snapshot of the "Server configuration" page

4.3.3 Restore

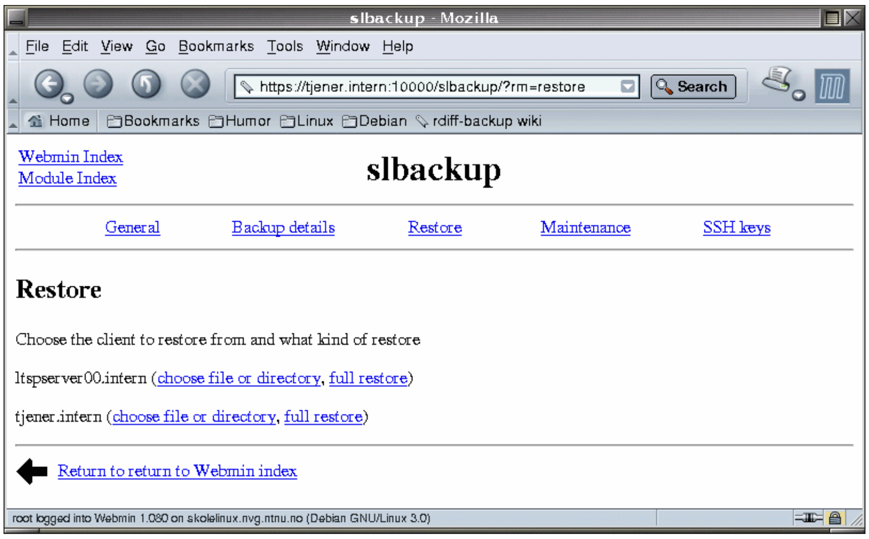
The restore section is, as the name says, where the user can restore files from the backup server to the system. On the Restore main page, the user has two options per computer in the backup configuration as seen in figure 4.16:

- **choose files/directories**
- **full restore**

If the user chooses to restore just one file or directory, the page shown in figure 4.17 comes up. On this page the user must type in the file/directory he wants to restore. When the user presses the "Choose file/directory" the page shown in figure 4.18 is shown.

Also if the user chooses to perform a full restore the page shown in figure 4.18 comes up. On this page the user can choose which snapshot he wants to restore from (displayed as dates) and which directory the restore shall be restored to.

When the user now presses the "Execute restore" button, a page like the one in figure 4.19 will be shown. On this page the user is presented with the details from the execution process, with a log from the backup system at the bottom.


slbackup - Mozilla
File Edit View Go Bookmarks Tools Window Help
Search
Home Bookmarks Humor Linux Debian rdiff-backup wiki

https://tjener.intern:10000/slbackup/?rm=restore

Webmin Index
Module Index

slbackup

General Backup details Restore Maintenance SSH keys

Restore

Choose the client to restore from and what kind of restore

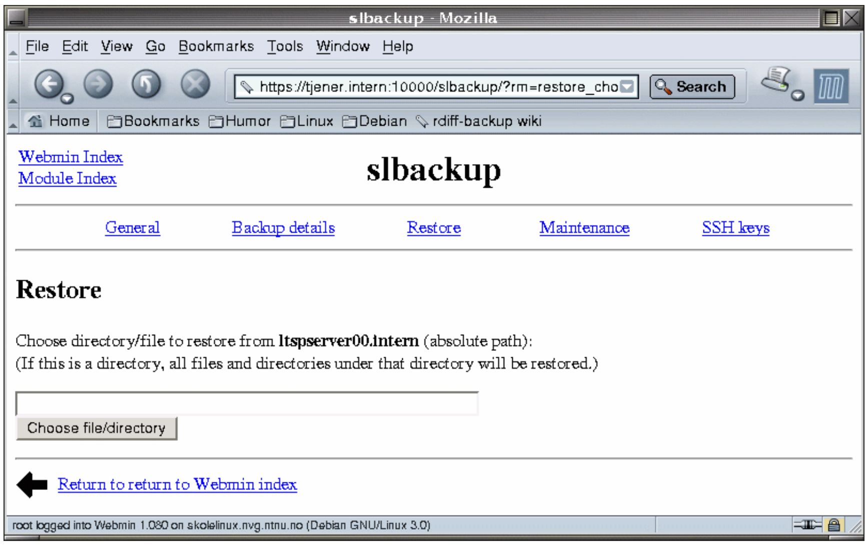
ltspserver00.intern (choose file or directory, full restore)

tjener.intern (choose file or directory, full restore)

Return to return to Webmin index

root logged into Webmin 1.080 on skolelinux.nvg.ntnu.no (Debian GNU/Linux 3.0)

Figure 4.16: Snapshot of the “Restore” main page


slbackup - Mozilla
File Edit View Go Bookmarks Tools Window Help
Search
Home Bookmarks Humor Linux Debian rdiff-backup wiki

https://tjener.intern:10000/slbackup/?rm=restore_cho

Webmin Index
Module Index

slbackup

General Backup details Restore Maintenance SSH keys

Restore

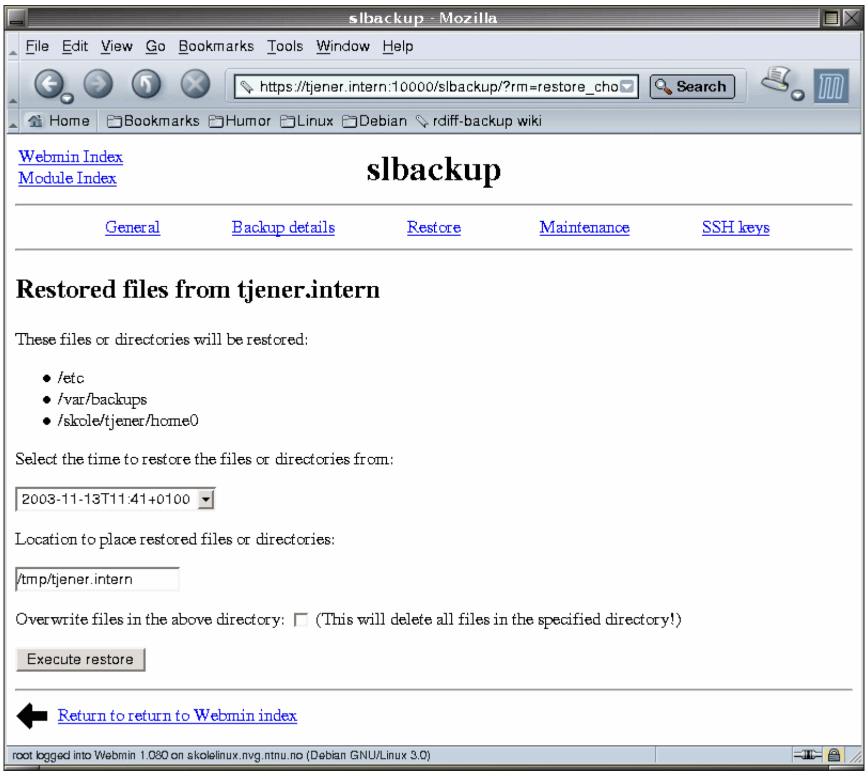
Choose directory/file to restore from **ltspserver00.intern** (absolute path):
(If this is a directory, all files and directories under that directory will be restored.)

Choose file/directory

Return to return to Webmin index

root logged into Webmin 1.080 on skolelinux.nvg.ntnu.no (Debian GNU/Linux 3.0)

Figure 4.17: Snapshot of the “Choose files/directories” page


slbackup - Mozilla
File Edit View Go Bookmarks Tools Window Help
https://tjener.intern:10000/slbackup/?rm=restore_cho
Search
Home Bookmarks Humor Linux Debian rdiff-backup wiki

Webmin Index
Module Index

slbackup

General Backup details Restore Maintenance SSH keys

Restored files from tjener.intern

These files or directories will be restored:

- /etc
- /var/backups
- /skole/tjener/home0

Select the time to restore the files or directories from:

2003-11-13T11:41+0100

Location to place restored files or directories:

/tmp/tjener.intern

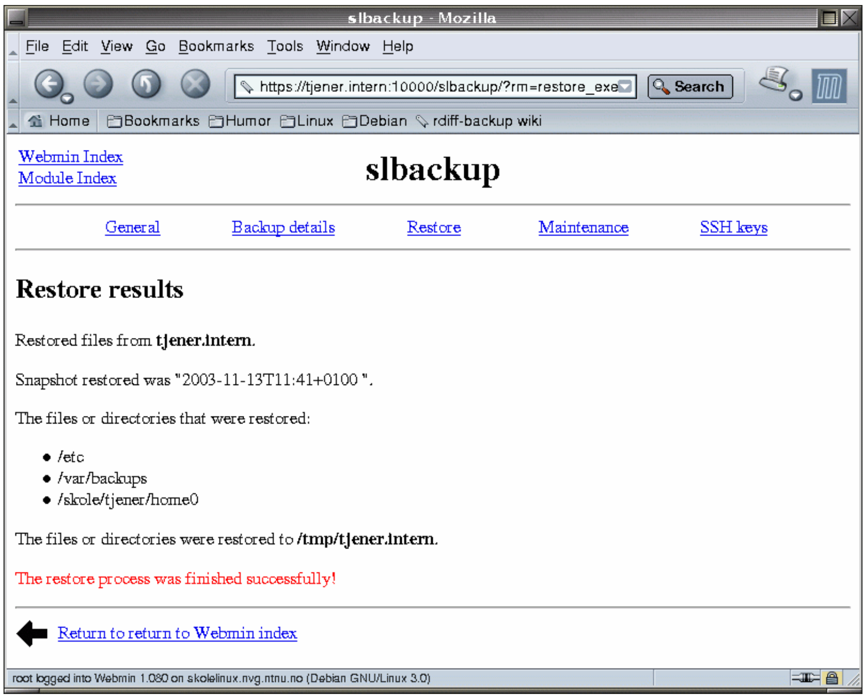
Overwrite files in the above directory: ☐ (This will delete all files in the specified directory!)

Execute restore

Return to return to Webmin index

root logged into Webmin 1.080 on skolelinux.nvg.ntnu.no (Debian GNU/Linux 3.0)

Figure 4.18: Snapshot of the “Restore details” page


slbackup - Mozilla
File Edit View Go Bookmarks Tools Window Help
https://tjener.intern:10000/slbackup/?rm=restore_exe
Search
Home Bookmarks Humor Linux Debian rdiff-backup wiki
Webmin Index
Module Index
General
Backup details
Restore
Maintenance
SSH keys

slbackup

Restore results

Restored files from **tjener.intern**.

Snapshot restored was "2003-11-13T11:41+0100 ".

The files or directories that were restored:

- /etc
- /var/backups
- /skole/tjener/home0

The files or directories were restored to **/tmp/tjener.intern**.

The restore process was finished successfully!

Return to return to Webmin index

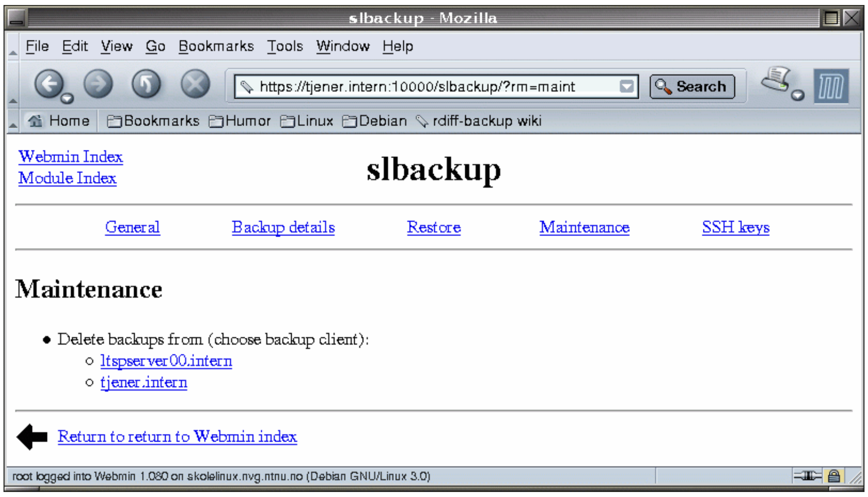
root logged into Webmin 1.080 on skolelinux.nvg.ntnu.no (Debian GNU/Linux 3.0)

Figure 4.19: Snapshot of the “Restore results” page

4.3.4 Maintenance

In this section the user can delete old snapshots from the backup server. The reason to do this is to free space on the backup servers hard disk. The main window for the “Maintenance” is shown in figure 4.20.

When the user chooses which client he wants to delete snapshots from, the page shown in figure 4.21. Here the user must choose which snapshots to delete (every snapshot older that the one that is chosen will be deleted). When the user presses the “Delete” button, a confirm page comes up as shown in figure 4.22. When the user confirms that he wants to delete, the page in figure 4.20 shows up with a verification that the snapshot(s) has been deleted.


slbackup - Mozilla
File Edit View Go Bookmarks Tools Window Help
https://tjener.intern:10000/slbackup/?rm=maint
Search
Home Bookmarks Humor Linux Debian rdiff-backup wiki

Webmin Index
Module Index

slbackup

General Backup details Restore Maintenance SSH keys

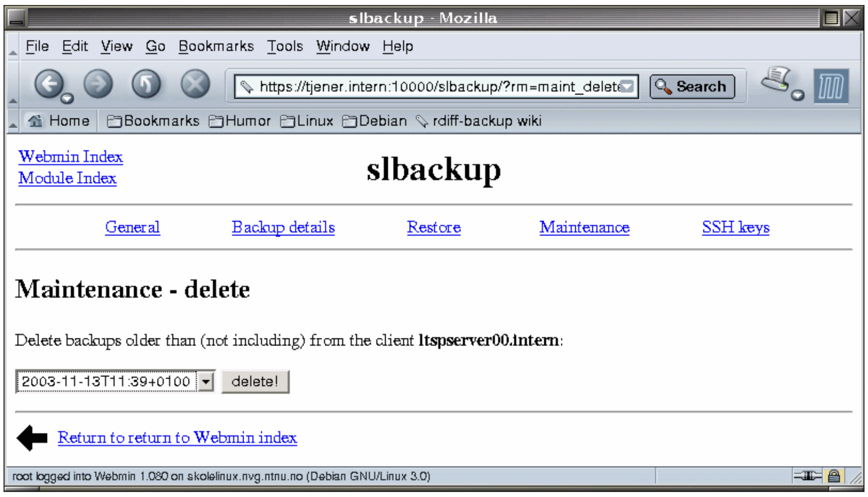
Maintenance

- Delete backups from (choose backup client):
  - ltspserver00.intern
  - tjener.intern

Return to return to Webmin index

root logged into Webmin 1.080 on skolelinux.nvg.ntnu.no (Debian GNU/Linux 3.0)

Figure 4.20: Snapshot of the "Maintenance" page


slbackup - Mozilla
File Edit View Go Bookmarks Tools Window Help
https://tjener.intern:10000/slbackup/?rm=maint_delete
Search
Home Bookmarks Humor Linux Debian rdiff-backup wiki

Webmin Index
Module Index

slbackup

General Backup details Restore Maintenance SSH keys

Maintenance - delete

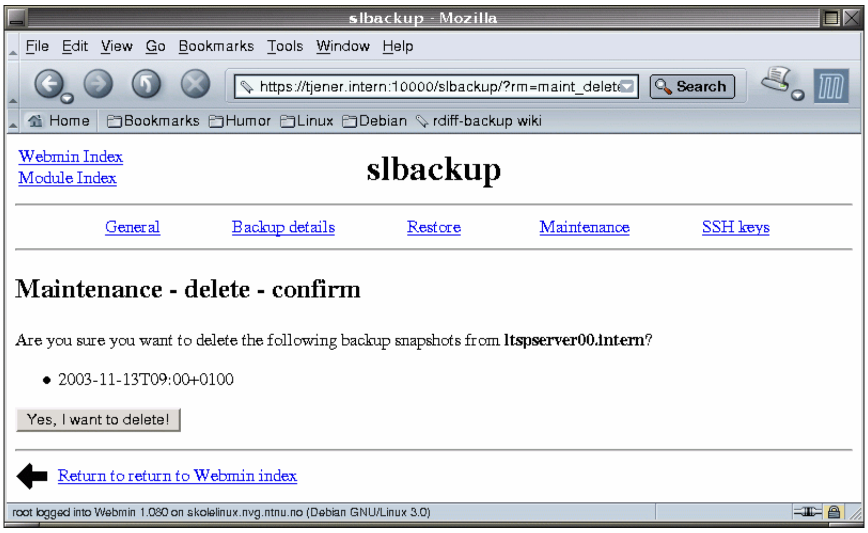
Delete backups older than (not including) from the client **ltspserver00.intern**:

2003-11-13T11:39+0100 delete!

Return to return to Webmin index

root logged into Webmin 1.080 on skolelinux.nvg.ntnu.no (Debian GNU/Linux 3.0)

Figure 4.21: Snapshot of the "Maintenance - delete" page


slbackup - Mozilla
File Edit View Go Bookmarks Tools Window Help
https://tjener.intern:10000/slbackup/?rm=maint_delete
Search
Home Bookmarks Humor Linux Debian rdiff-backup wiki
Webmin Index
Module Index
General Backup details Restore Maintenance SSH keys
Yes, I want to delete!
Return to return to Webmin index

slbackup

Maintenance - delete - confirm

Are you sure you want to delete the following backup snapshots from **ltspserver00.intern**?

- 2003-11-13T09:00+0100

root logged into Webmin 1.080 on skolelinux.nvg.ntnu.no (Debian GNU/Linux 3.0)

Figure 4.22: Snapshot of the "Maintenance - delete - confirm" page

4.3.5 SSH keys

In this section the user can manage the SSH keys on the computers in the current configuration. For the system to be satisfied with the SSH keys, there must exist a private-/public key pair on the Skolelinux tjener, and the public part of that keys must exist on the backup server and every extern client. The main page is shown in figure 4.23.

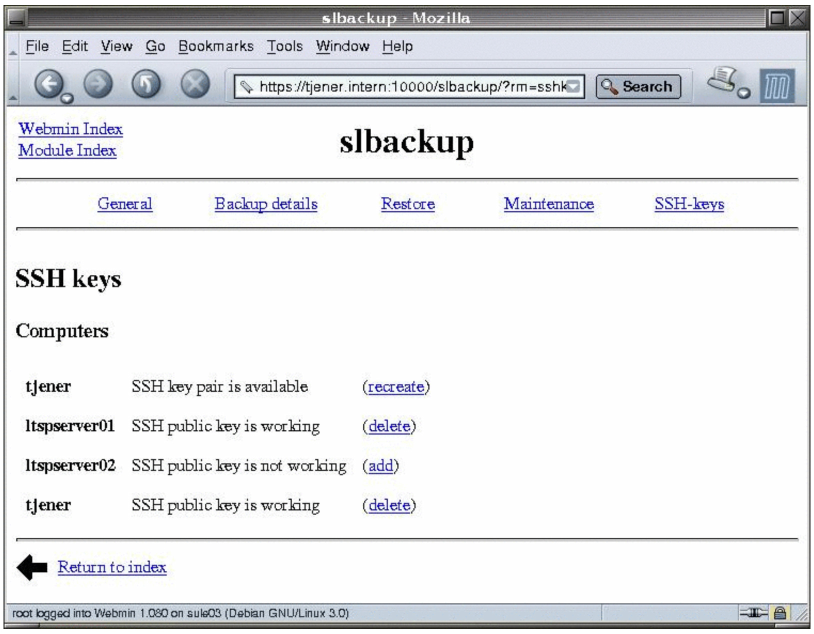
If the user either adds, deletes, or recreates the keys for a computer, a confirmation web page as shown in figure 4.24 will be displayed. If the requested action requires a password, a simple web form will ask for this.

4.4 Backup server

As explained earlier, the backup server is assumed to be an external computer, but shall also be designed in a way that it can be a part of the Skolelinux server. In this section we assume that it is an external computer if nothing else is written.

On the backup server the following software packages shall be installed in addition to a minimal installation of the system:

- IP-tables
- Nagios client


slbackup - Mozilla
File Edit View Go Bookmarks Tools Window Help
https://tjener.intern:10000/slbackup/?rm=sshk
Search
Webmin Index
Module Index
General
Backup details
Restore
Maintenance
SSH-keys
Return to index
root logged into Webmin 1.080 on sule03 (Debian GNU/Linux 3.0)

slbackup

SSH keys

Computers

tjener	SSH key pair is available	(recreate)
ltspserver01	SSH public key is working	(delete)
ltspserver02	SSH public key is not working	(add)
tjener	SSH public key is working	(delete)

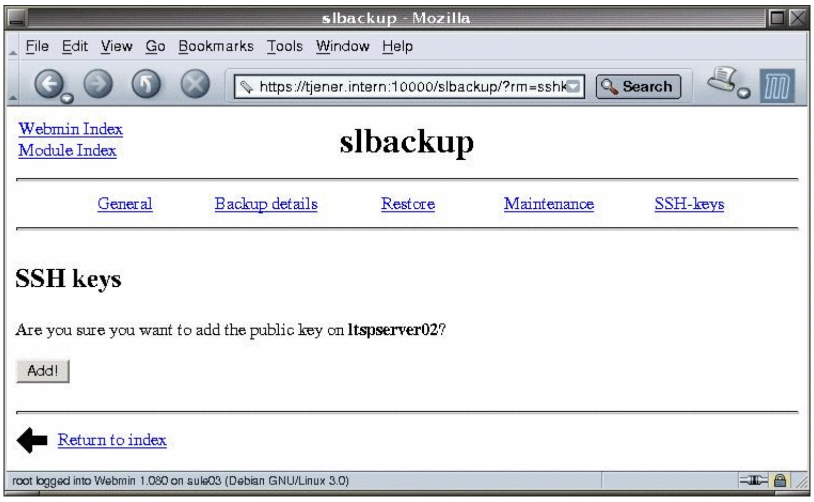
Figure 4.23: Snapshot of the "SSH keys" page

- rdiff-backup
- SSH

The operating system installed on the backup server shall be an adjusted version of one of the Skolelinux profiles. All unnecessary software packages shall be uninstalled, and the above mentioned software shall be installed and properly configured. The hard disk on the backup server will be divided into two logical parts. One will store the operating system and the other will be used to store backup data. The part storing the backup data will be created as a LVM-partition and preferably mounted on */skole/backup*. The advantage of using LVM is that the partition easily can be enlarged. It is also possible to add an additional hard disk and add its space to the LVM-partition.

The IP-tables package, which is firewall software, shall be set up in a way where no other computers than the Skolelinux server will be permitted to log in, send/receive files and monitor the backup server. This is to prevent attacks from unwanted computers.

Nagios is installed to monitor the backup server. Parameters that shall be monitored are backup errors, backup consistency, disk space, and that the SSH server is running. A Nagios plugin for slbackup is needed to monitor backup errors and backup consistency.



slbackup

SSH keys

Are you sure you want to add the public key on **ltspserver02**?

root logged into Webmin 1.080 on sule03 (Debian GNU/Linux 3.0)

Figure 4.24: Snapshot of the "SSH keys - confirm" page

rdiff-backup has to be installed if the backup service shall work. When rdiff-backup uses an external computer for storing data, a rdiff-backup process will be started on the external computer in server-mode.

SSH must be installed because rdiff-backup uses SSH to transfer data over the network. SSH will also provide security when key pairs are used for authentication.

4.5 LDAP backup

Backing up the LDAP database is important to Skolelinux, as all user data is stored in it. To make a good backup copy of the database, you will have to stop the LDAP database engine (*slapd*), dump the database into a ldif formatted file and start the LDAP database engine.

Due to the problems described by Rune Nordbøe Skillingstad in [Ski03], the best solution would be to start an additional *slapd* process, from now called *slurpd* that replicates the main LDAP database. To do backups, we now have to stop *slurpd*, dump the *slurpd* LDAP database, and start the *slurpd*-process. While this is done, all other processes can still use the main LDAP database, there will not be unnecessary out-of-service time, and we will omit the problems with other programs crashing due to the stopped *slapd*-process.

Figure 4.25 shows this process in more detail. In phase one, the system is in normal state with processes using information in the LDAP database thru the *slapd*-process and the *slurpd*-process replicating the *slapd*-process. In phase two, the LDAP-backup job has stopped the *slurpd*-process and runs

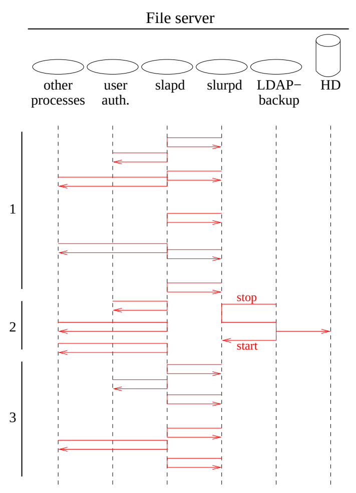


Figure 4.25: Interaction between the two *slapd*-processes and other parts of the system

slapcat to dump data info a ldif-formatted file on the hard drive, this at the same time as normal processes use information from the LDAP database thru the *slapd*-process. In phase three, the system is back in normal state, like phase one.

4.6 Security

This backup system assumes that the backup server is a separate computer which can be placed on another location than the other Skolelinux servers. If the backup server is a separate computer, there should be restricted physical access to the backup server. No physical security like encrypted file systems have been considered due to shortage of time in the project.

When it comes to software security, the backup server will be an additional Skolelinux server for the system administrator to keep secure. How to do this is covered in the “IKT-driftshåndbok for Skolelinux” [VB03].

Chapter 5

Implementation

This implementation chapter will first have a section about one design change Skolelinux requested after the design was finished. The rest of the chapter will describe the software written, which packages that is released and the changes made to the Skolelinux CD to implement the backup solution.

As the backup system does not depend upon the package *slapd-backup*, this package will not be implemented due to shortage of time.

5.1 Design change

The Skolelinux distribution is built with a specific architecture in mind. This architecture is specified in [Rei02]. The architecture contains mainly of two types of servers, file server and LTSP server. The LTSP server(s) is the terminal server for a number of thin clients and the file server provides the LTSP server(s) with services like DNS, email, file sharing, printer server, web proxy, web server, and user authorization. Skolelinux is designed to make it possible to extract these services and place them on external computers if needed (often due to capacity problems). Skolelinux wants to develop a common way to extract these services, but this is still on the planning stage.

The backup service is similar to the other services supported by the Skolelinux distribution. Skolelinux wants the backup service to be extracted in the same way as the other services when the common procedure for extracting services is developed. It would be irrelevant and counterproductive to build an own distribution around the backup service, when Skolelinux nevertheless will do this in another way when the common procedure is developed. This design change was decided by Skolelinux about three weeks before the deadline of this project.

Skolelinux are aware of the following disadvantages when not placing the backup service on an external computer, but still wants to wait with the extraction of the backup service:

- **Computer failure**
  If the backup data is placed on the same computer as the files/directories that shall be backed up, the backup will have no value if the computer fails (eg. disk crash).

- **Environmental disaster**
  Disaster like fire, flooding, and theft will strike both the backup and the live system if the data is stored on the same computer, even in the same room or building.

Chapter 4 describes the design Skolelinux actually wants for a backup solution. But because of the yet unimplemented common solution to extract services, they want the backup server implemented on the Skolelinux file server. But, this implementation shall make it easy to extract the backup service to an external computer, but as default it shall be located on the Skolelinux file server.

5.2 Programming language

The web interface for the backup system will be implemented as a Webmin module, because most of the configuration interfaces in Skolelinux already are Webmin modules. When Webmin, and most of the Webmin modules are written in Perl, it will be natural that the Webmin module for the backup system also is written in Perl.

In addition to the cron job on the Skolelinux file server that will contain the information about when the backup session is started, a configuration file that contains information about the backup server and the client(s) is needed. There is a very good Perl module, *Config::General*, that handles configuration files very well. It is natural to also write the rest of the software in Perl, so that the backup project does not become a soup of mixed programming languages. Another possibility is to write everything in Perl, is that it is possible to share some code between different pieces of software.

5.3 Backup software

The first phase of the implementation is to rebuild a new release of *rdiff-backup* to a Debian stable package. This also result in rebuilding the *librsync* package to Debian stable. There are two possible techniques to use when rebuilding a package for the stable release of Debian.

1. Take the fresh sources, and build a Debian package from the bottom up. Here it is important to find all the dependencies, or the package will fail to run on systems missing these dependencies.

2. If the version of the package from the unstable package archive is new enough, download this source package, check that the package dependencies are met in Debian stable, fulfil the build dependencies, and try to rebuild the package.

After trying both methods, Petter Reinholdtsen recommended technique two, because the packages already have been tested by lots of people and it will be easier when Skolelinux and Debian releases a new stable release (the packages would already be there).

5.3.1 librsync

The version of *librsync* used is 0.9.6. To rebuild librsync, build dependencies to *autoconf* and *automake1.7* was removed as they were not used in the current version of the package. After removing these build dependencies the package successfully build in Debian stable.

5.3.2 rdiff-backup

The version of *rdiff-backup* used is 0.12.5. The only dependency problem when rebuilding *rdiff-backup* was *python2.3*, and when *rdiff-backup* really only depends on *python2.2*, the package was just changed to use and depend on *python2.2* instead of *python2.3*. After this change, *rdiff-backup* also rebuild successfully in Debian stable.

5.4 Package: slbackup

The package *slbackup* contains a standard configuration file, a Perl module which contains configuration file functionality, a cron job that starts the backup session once a day, and the backup session script to be run by cron every day to perform backups as the configuration file describes.

The configuration file was created so the Perl module *Config::General* could read from and write to it. The configuration file is placed in */etc/slbackup/slbackup.conf* and for a standard Skolelinux file server configuration it will be equivalent with the one in appendix E.1. The reason for choosing a *Config::General* format for the configuration file, is that it will be an easy and common way for all software that is going to read from and/or write to the file.

The Perl module is located in */usr/share/perl5/SLBackup.pm* and provides functionality for reading and writing data to and from the configuration file. The configuration handling uses the Perl module *Config::General* and has a subroutine that reads the configuration file and returns a hash with the configuration, and a subroutine that writes the hash back to the configuration file. A copy of the modules is in appendix E.2.

By default, the cron job will run the backup script as the *root*-user each night at 1AM. The cron job will be located in */etc/cron.d/slbackup*. A copy of the file is in appendix E.3.

The script that executes the *rdiff-backup* process reads the configuration file and parses it. First the script deletes all backups older than the number of days specified in the configuration file (default is set to six months). After the deletion the script builds one command string for each host to back up. While building this string, the script is testing that the configuration file provides enough information about the server and clients if some of these are external computers. These tests are valid hostname/IP-address, username, working SSH-connection, and right version of *rdiff-backup*. This string will be executed, and status and error messages from the script and extended output from *rdiff-backup* is logged to */var/log/slbackup/slbackup.log*. The format of each line in this log file is "<date/time> - <logmessage>". A copy of the script is in appendix E.4.

The Debian package, which provides the installation routine of this package, also set up logrotate to rotate the */var/log/slbackup/slbackup.log* monthly.

The complete source code for this package is located on the CD following this report (cvs/alioth/slbackup/).

5.5 Package: webmin-slbackup

The package *webmin-slbackup* is a graphical front-end to the configuration file (*/etc/slbackup/slbackup.conf*), restore functionality, maintenance functionality, and the configuration of SSH keys. The Webmin module is described in detail in chapter 4.3.

To handle a Webmin module with this size, the Perl modules *CGI::Application* [Erl] and *HTML::Template* have been used. *CGI::Application* provides a framework for building reusable Web-applications and *HTML::Template* provides a way of separating the HTML-code from the Perl code. HTML-templates been made for each Web page, and all text strings printed is fetched from an own file with strings.

The most important files in this package is the Perl module *WebminSL-Backup.pm*, a lot of HTML templates, and the files containing all the text strings used in the Webmin module, one for each language.

WebminSLBackup.pm contains one subroutine for each web page displayed to the user and some common subroutines used to create several pages. *WebminSLBackup.pm* also uses the subroutines provided by the Perl module *SLBackup.pm* from the *slbackup* package.

To provide the HTML templates with the strings from the language files, a subroutine written by Andreas Schuldei in connection with the Skolelinux package *webmin-ldap-skolelinux*, is used.

The information about the latest backup sessions is fetched from the log-file */var/log/slbackup/slbackup.log*.

The data provided in connection with restore pages, is provided by *rdiff-backup* and its options specified by the man page[Esc03b].

To test if the SSH connections work without providing passwords, a SSH session is started with the option "*-o PasswordAuthentication=no*" and checks that "*echo -n 1*" actually returns *1* and not an error message.

To handle the SSH connection that requires password interaction, the Perl module *Expect.pm*[Gie] was used.

The complete source code for this package is located on the CD following this report (cvs/alioth/webmin-slbackup/).

5.5.1 libexpect-perl

The version of *libexpect-perl* located in Debian stable (version 1.11) did not work properly, so the package was rebuild with sources from Debian unstable (version 1.15). It was not necessary to do any changes to the package to make it build in Debian stable.

5.6 Implementation on the Skolelinux CD

The implementation of the backup software on the Skolelinux CD is divided into three parts.

1. Set up a LVM partition and mount this to */skole/backup*. The rules for the size of this partition will specify the minimum and maximum size for the partition.

   The files containing these rules are called "*Main-Server.table*" and "*Main-Server+Thin-Client-Server.table*" for the Skolelinux file server and combined file server and LTSP server respectively. These files are located in the Skolelinux CVS tree (src/base-config-skolelinux/autopartkit/). A complete copy of the Skolelinux CVS tree is on the CD following this report (cvs/skolelinux/).

2. Edit the list of packages that shall be included on the CD and add *slbackup* and *webmin-slbackup*. The CD-builder script handles dependencies.

   The package lists are called "*task-skolelinux-server.txt*" and "*task-skolelinux-ltsp.txt*" for the Skolelinux file server and LTSP server respectively. These files are located in the Skolelinux CVS tree (src/task-skolelinux/lists/). A complete copy of the Skolelinux CVS tree is on the CD following this report (cvs/skolelinux/).

3. Modify the *slbackup* Debian package to use Debconf in the configuration phase. Debconf is a configuration management system for Debian packages. When making a package using Debconf for configuration, a lot of templates are used. Each template reflects a variable that will be used for instance in a configuration file. All Debconf templates has a default value that will be used if the user does not want to be asked questions during the installation.

   The Debconf template-file and the other installation files, is located in *slbackup*'s CVS tree on Alioth (slbackup/debian/). A complete copy of *slbackup*'s CVS tree on Alioth is on the CD following this report (cvs/alioth/).

   The great advantage of using Debconf in this case, is that a general scenario (not Skolelinux) the configuration scripts asks the user or uses the default-values which is valid for a Debian installation, and in the Skolelinux installation all the templates have predefined answers (specified in the Skolelinux installation). So for instance, the locations to back up in a Skolelinux installation is defined to be:

   - */etc*
   - */skole/tjener/home0*
   - */var/backups*

   for a Skolelinux file server, and

   - */etc*
   - */skole/tjener/home0*
   - */opt/ltsp/i386/etc*
   - */var/backups*

   for a combined Skolelinux file server and LTSP server.

When these three parts work properly, the backup solution should be installed and work "out of the box".

Chapter 6

Installation

The backup system have been installed and tested on four systems, one Debian stable environment, three small Skolelinux environments, and some fresh Skolelinux installations just to test the installation procedure after implementing the solution on the Skolelinux CD.

6.1 Debian stable environment

In this environment, *rdiff-backup* has been tested for nearly two months and *slbackup* has been tested for two weeks. The files that have been backed up is one home directory including an updated version of the Skolelinux CVS to insure that files are heavily changed. The only errors that have occurred during the test are *rdiff-backup*-specific:

```
SpecialFileError <filename> Socket error: AF_UNIX path too long
```

This error does not influence of the rest of the backup, and a restore does not fail. In this case the file that caused the `SpecialFileError` was a file that was recreated if missing, and therefore not an important file to lose.

Beside this the *slbackup* software has not run into any errors. When using the Webmin interface, a lot of bugs has been discovered and fixed.

In this environment an external tester did a total review of the entire Webmin interface, and had a lot of comments. Most of them were implemented. During this evaluation session more bugs in the Webmin interface were discovered, and fixed.

6.2 Skolelinux environment

The first Skolelinux environment that *slbackup* were tested, was at NVG (a club for students and employees at NTNU). *slbackup* has been used to back up configuration files on this installation. Except for some bugs discovered in the Webmin interface, that have been fixed, no problems have surfaced.

The second Skolelinux installation where *slbackup* has been tested, was a test installation used at the Skolelinux gathering in Ulsrud comprehensive school in Oslo 14th to 16th of November. On this installation another external test person reviewed the Webmin interface, and found some bugs that were fixed.

The third Skolelinux installation that tested *slbackup* was the test network of the system administrator at Holmlia comprehensive school. This test was carried out one week before the project deadline, but resulted in some bugs fixed, and positive comments about the product.

All the external testers were satisfied with the Webmin interface, but had some ideas that are added to a wish-list for implementation in future releases of the product.

6.3 Installation tests

About 5–10 test installations were done to test that the installation procedure is working properly. One of the problems was to find free space on the Skolelinux CD (Skolelinux shall be install able from only one CD). This problem solved it self, when the CD size was increased from 650MB to 700MB.

Chapter 7

Conclusion

The project is finished and Skolelinux have a working backup solution. In this chapter we will look at the positive results, the present solution with the requirement specification in mind, discuss what could have been done different during the project, sum up what can be done with the product in the future, and discuss the working method used in a Skolelinux project.

7.1 What has been achieved?

This projects main goal is reached, namely to put together a working backup solution for the Skolelinux distribution that works "out of the box" without any needs for the SA to do any initial configuration. The project has succeeded when all things are considered. A short summary of the achievements in the project is presented in the list below.

- A requirement specification is put together and adopted by Skolelinux.
- Backup software was found and evaluated, and finally *rdiff-backup* was chosen (see chapter 3 for details).
- A design for the system and how to use it was made up.
- The design was implemented; necessary software packages was rebuild for Skolelinux, and two software packages was written (*slbackup* and *webmin-slbackup*).
- The system was installed and tested.
- The system was included on the Skolelinux CD.

The project has ended up with a working backup solution that will be used in the Skolelinux distribution as from the next pre-release of the CD (number 43).

Things I have learnt from the project and things I interpret as positive during the project are listed below.

- The communication channels used in Skolelinux (IRC and email-lists) have been frequently used as requested by Skolelinux, and I have good experience with asking questions and discussing challenges I have met.
- At the time this project finishes, the backup solution will solve problems like when someone deletes files/directories, and when someone changes files, but wants the original back. When Skolelinux develops a common way of extracting services, the backup solution will automatically cover problems like disk crash, fire, flooding, and so on.
- I have submitted a bug report on the backup software the product in this project is based on, *rdiff-backup*, and within a week, the author of *rdiff-backup* fixed the bug and released a new version with the fix included.

7.2 Achieving the requirement specification

The product that has been developed have most of the requested features that are requested in the requirement specification. The features that are missing or partly missing are listed below. The sentences printed in bold are obtained from the requirement specification (see chapter 2).

- **The backup service shall in case of any failure send a failure report to the SA**
  No mail is sent to the SA and no information is presented by the Webmin module if the backup fails. Skolelinux are going to use Nagios to monitor their services, and a Nagios module for *slbackup* should have been made. More information about this is in chapter 7.4.
- **The documentation shall contain guidelines to planning how much capacity that is needed for the service to work properly.**
  As the implementation does not include a separate backup server the documentation does not contain any information about scaling one, but the documentation contain guidelines to planning how much disk capacity one would need.
- **Take daily backup of specified directories/files to a hard drive on an external backup server (...)**

Skolelinux decided three weeks before the project deadline that they did not want to have the backups stored on an external backup server, but on the Skolelinux main server in a default installation. This is because, as explained in chapter 5.1, Skolelinux wants a common way to extract services, but this is not implemented yet.

The implemented backup solution supports storing the backups on an external computer, but in the Skolelinux installation the default is to back up the Skolelinux main server to an own partition on the same computer.

- **The backup server is installed on a separate computer and shall only require input from the SA about which computers (IP-addresses) it shall receive backup from, which computer it shall use as syslog-server (if any), and the email-address to the SA.**
  This item was omitted as common way to extract services (including backup server) from the Skolelinux main server has been implemented yet.

Most of the missing features are missing because of the decision to exclude a separate backup server and the rest was not prioritized.

The product has some desired features in the requirement specification that are missing in the developed product:

- **Archives are stored encrypted on the backup server (...)**
  This has not been implemented. Partly because there has not been developed a backup server, and partly because it would have been to complex to implement this on the Skolelinux main server (that per default stores the backup).

- **The end user, pupils, and teachers shall be able to restore their personal files (...)**
  The restore functionality provided by the Webmin interface is minimal, and does not include support for user restore. This has been suggested as a student project.

- **Webmin interface: check the quality of the backup files (...)**
  There has not been any implemented functionality in the Webmin interface for this desired feature.

- **Webmin interface: define mail and SMS recipients in case of trouble and errors**
  The monitor software in Skolelinux shall be Nagios, but the monitor solution is not finished yet, so a temporary solution is to log everything into */var/log/slbackup/slbackup.log*.

- **Webmin interface: provide browser facilities to restore files from backup**
  A minimal but working restore functionality was preferred over a restore functionality with a lot of facilities, but to provide a new restore interface with the browser facility among others is suggested as a new student project.

- **Webmin interface: syslog monitoring**
  Logs are written to a separate file (*/var/log/slbackup/slbackup.log*) and the Webmin interface uses it to find out when the last backup session was finished. A more detailed statistic page in the Webmin interface is suggested as an item in the "Further work" chapter.

The requirement specification required one Users Manual and one System Administrators manual. No restore interface for the end users of the system is created, and the manuals have been merged into one Users Manual that is meant for the System Administrators. The Users Manual is in appendix D.

7.3 What could have been done different?

If the project was to be repeated, the issues that would have been modified are:

- **More than one project participant**
  There are a lot of positive effects by being more than one person doing a project.
  - In the design phase, two people will give you two different views of the system, and maybe give a better design.
  - If a problem occur, two brains think better than one.
  - It is more motivating being more than one person working alone, and if one of the members have a bad day, the other(s) are there to motivate and push in the right direction.
  - In the most cases, more work would be done.

- **More discussion with the customer**
  One experienced that more detailed discussion with the customer when doing the design, could have saved some work hours. The best example in this project, is that three weeks before the project deadline, Skolelinux decided to not implement a backup server, but use the Skolelinux main server instead. Of course, this reduce saved the project member for some time, but I think if this part of the design had been discussed with the customer at an earlier moment of time, it could have been saving even more time.

- **Longer period of time for testing and bug fixing**
  This project had a product for testing ready about a month before the project deadline. A month is not enough to test a backup system extensively. A longer period for testing and bug fixing would have been a target to reach for in another project.

- **Better planning of actions that relied on other parties**
  Early in the project period I talked to SA's on some schools and agreed with them to test the backup solution when it was ready. The problem was that when there was less than a month to the project deadline, it was hard to make a test fit into the SA's schedules. The lesson to learn from this, is that if some actions relies on other parties than the project group, it is important to make agreements including a date or week number for the planned happening (in this case the test installations).

The last missing feature with the backup solution, is a drawback with using *rdiff-backup* as the backup software. *rdiff-backup* has no functionality that makes it possible to store daily backups for a specified amount of time, weekly backups for another specified amount of time, and monthly backups for yet another specified amount of time. This conflicts with figure 2.2 in chapter 2, which gives an estimate of how long to keep the backups. *slbackup* is therefore, after discussions with Skolelinux, storing daily backups for six months.

7.4 Further work

Each of the software packages *slbackup* and *webmin-slbackup* contains a TODO file including features to add and trivial bugs that users have requested during this project. The most important feature are

- add installation procedure for other systems than Debian
- check that there exists any backups before trying to delete some (will prevent the error messages in the log-file)
- make a command line client which provides restore functionality, idea: https://init.linpro.no/pipermail/skolelinux.no/devel/2003-August/000577.html

for *slbackup*, and

- add one more step before executing restore (with checking of files, and reporting to the user what actually will happen)

- check if a SSH-public key is needed when showing "Configure client" page, and print a warning and a "link" to the SSH keys-page
- new page with some statistics from the backups taken
- add browsing functionality in the restore section

for *webmin-slbackup*.

A goal for these software packages is to push them into the Debian unstable package archive. The benefits of doing this, is that in the future the packages hopefully will be a part of Debian's stable distribution, and also Skolelinux. The reason Skolelinux wants their packages to be a part of Debian, is that they (Skolelinux) does not have to handle security updates of the packages.

In addition to the two software packages, Skolelinux should make a common way to extract services from the Skolelinux main server to separate computers. This is important to the backup system, because the value of backups will increase a lot when extracting the backups to a separate computers (discussed in detail in chapter 5.1).

One missing feature is monitoring the backup server. The main reason for the shortcoming on this area, is that no backup server was designed. In any case monitoring is an important feature of a backup system, and the three main conditions to monitor is backup failures, backup consistency and disk space. As Skolelinux is going to use Nagios as the main monitoring system, an own Nagios plugin that at least monitored the three conditions mentioned above should have been developed for *slbackup*.

The features provided by the products that are developed in this projects, have been reported as missing features of *rdiff-backup* on the rdiff-backup-users email-list. Some of the functionality in this project could have been merged into *rdiff-backup*.

When using and designing a system based on *rdiff-backup*, one notices that one important feature is missing. The way *rdiff-backup* handle incremental backups today is that it maintains a copy of all files like a fresh snapshot and stores the reverse differences in separate files called reverse diff-files. The problem is that *rdiff-backup* does not have a function to merge these reverse diff-files, and the backup repository stores all daily differences for as long as you want to keep backups (default is six months for a standard Skolelinux installation). It could have been useful to be able to merge these reverse diff-files so one would not need to store daily, but weekly when the backups were older than a specified amount of time (default could be one month). This will need an additional software module in either *slbackup* or *slbackup*.

Skolelinux Backup only handles UNIX/Linux computers in the network, and most schools have some computers with Windows installed on them. One approach to cover these too, would have been if *slbackup* searched for SMB shares available on the network and presented these in the web interface, so that the SA would have the possibility to choose which of them he wanted to back up. *slbackup* could have mounted these SMB shares and backed them up each night as a part of the daily backup session.

For most schools backup to hard disks are the best solution. Some schools or other organizations using *slbackup* may want to use tape as the backup media. A *slbackup* module that handled backup to tape is a good idea. The system administrators would have the possibility to choose the backup media to use in the web interface. Another approach is to use backup-to-disk for daily backups and backup-to-tape for archive backups.

Another idea for the backup media, is to use distributed storage. That would be to use free space on the computers in the network to store the backups. This does not give you total control over the files, but you could use private-/public-key to encrypt/decrypt the data. You could never been 100% sure that you could restore a file, but statistically you are almost sure that you could.

7.5 Releasing slbackup as an Open Source project

The software packages *slbackup* and *webmin-slbackup*, has during the project been released as their own Open Source project and is now located on Debian's SourceForge clone Alioth. The reason for doing this is that these packages hopefully will get contributers from others than the Skolelinux community which can test and help developing the product.

I will continue to maintain the Debian packages of the software made in this project and do what I can to push them into Debian's archive. As the next release of Skolelinux (after version 1.0) probably will be based on the next stable release of Debian, it will be time saving for Skolelinux if the Debian packages *slbackup* and *webmin-slbackup* make it into the next stable release of Debian.

7.6 Working method

As mentioned earlier in this report, Skolelinux practices a special form of management model called a do-o-cracy and the communication between the Skolelinux developers takes place almost only on the Internet (IRC and email-lists). To accomplish a successful project within a Skolelinux community, it is important to make the most of the capable participators in the project. To do this, one must use the communication channels actively from the start of the project. An advantage one has as a participator in a Skolelinux project is the freedom the do-o-cracy gives you, but it is important to discuss important details with the other participators in the project. It is also important to notice that the most active people participating to The Skolelinux Project, often are the people you can trust ones advice and decisions. So the conclusion is that a project for Skolelinux can be very instructive if the project member(s) make the most of the capabilities that the other participators in the Skolelinux community.

Bibliography

[afi] Freshmeat project-page for afio.
http://freshmeat.net/projects/afio/ (accessed: 2003/09/24).

[ark] Arkeias website.
http://www.arkeia.com/platforms.html (accessed: 2003/09/07).

[baca] Bacula website.
http://www.bacula.org/ (accessed: 2003/09/15).

[bacb] Veritas website.
http://www.veritas.com (accessed: 2003/11/25).

[Bow03] John C. Bowman. Changelog for rlbackup, 2003.
http://www.math.ualberta.ca/imaging/rlbackup/ChangeLog
(accessed: 2003/09/27).

[Did99] Paolo Didonè. Burt license thread on the debian-legal mailing list, 1999.
http://lists.debian.org/debian-legal/1999/debian-legal-199906/msg00212.html (accessed: 2003/10/07).

[Erl] Jesse Erlbaum. Perl module: Cgi::application.
http://search.cpan.org/dist/CGI-Application/
(accessed: 2003/11/05).

[Esc03a] Ben Escoto. Changelog for rdiff-backup, 2003.
http://rdiff-backup.stanford.edu/CHANGELOG
(accessed: 2003/09/07).

[Esc03b] Ben Escoto. Manpage for rdiff-backup, 2003.
http://rdiff-backup.stanford.edu/rdiff-backup.1.html
(accessed: 2003/11/09).

[Gag03] Marcel Gagné. Arkeia corporation's arkeia, version 5.0.16. *UnixReview.com*, March 2003.
http://www.unixreview.com/documents/s=7822/sam0303web/
(accessed: October 7th, 2003).

[Gie] Roland Giersig. Perl module: Expect.
http://search.cpan.org/ rgiersig/Expect-1.15/
(accessed: 2003/11/05).

[leg] Legato software website.
http://www.legato.com (accessed: 2003/11/25).

[Mel99] Eric Melski, editor. *Burt: The Backup and Recovery Tool.* USENIX, 1999.

[Pre99] W. Curtis Preston, editor. *Unix Backup & Recovery.* O'Reilly, Sebastopol, Canada, first edition, 1999.

[Ray01] Eric S. Raymond. *The Cathedral and the Bazaar.* O'Reilly, Sebastopol, Canada, 2001.

[Rei02] Petter Reinholdtsen. Skolelinux - architecture, 2002.
http://developer.skolelinux.no/arkitektur/arkitektur.html.en
(accessed: 2003/09/23).

[Sch] Ian Jackson & Christian Schwarz. The debian free software guidelines.
http://www.debian.org/doc/debian-policy/ch-archive.html#s-dfsg
(accessed: 2003/09/12).

[Ski03] Rune Nordbøe Skillingstad. slapd backup, 2003.
http://developer.skolelinux.no/r̃unesk/slapd-backup.html
(accessed: 2003/10/16).

[Sme03] Jonas Smedegaard. Programmer til delta-baseret spejling, 2003.
https://init.linpro.no/pipermail/skolelinux.no/linuxiskolen/2003-August/036296.html (accessed: 2003/09/23).

[tol] The tolis group website.
http://www.tolisgroup.com/ (accessed: 2003/09/09).

[VB03] Tor Harald Nordnes og Truls Teigen Vibeke Braaten, Christian Juell. Ikt-driftshåndbok for skolelinux, 2003.
http://developer.skolelinux.no/dokumentasjon/IKT-bok.html
(accessed: 2003/10/23).

[Yrv01] Knut Yrvin. Gjørokratiet, 2001.
http://developer.skolelinux.no/info/prosjektet/innlegg/gjoerokrati.txt
(accessed: 2003/09/26).

Glossary

AFS Andrew File System

BSD Berkeley Software Distribution

build dependency to be able to build a package, some other packages has to be installed

calculate delta a calculation that gives the difference between two files

cpio a program that manages archives of files

CVS Concurrent Versions System

debian-legal mailing list mailing list where discussions about legality issues such as copyrights, patents etc.

dump utility that examines files on a file system and determines which files needed to be backed up

FAQ Frequently asked questions

files with holes if you have a file that has a lot of sectors containing only NUL characters, it does not assign those sectors to actual sectors on the disk

FTP File Transfer Protocol

GPL Gnu General Public License

ldif LDAP Data Interchange Format, used to represent LDAP entries in a simple text format

IRC Internet Relay Chat

NFS Network File System

Northern Sami the language of the nomadic Lapp people in northern Scandinavia and the Kola Peninsula

Python a programming/scripting language

rdiff tool to make diffs for binary files

rebuild one can rebuild a Debian package from the unstable archives on a Debian stable installation to make the package installable on Debian stable (if not all package dependencies are met, you have to fix them to make the package build)

scp secure copy

SMB Server Message Block, a lightweight protocol designed to allow the sharing of files and printers in a small network

snfs Secure NFS and NIS via SSH Tunnel

Sourceforge is the world's largest Open Source software development website

SSH Secure Shell

tar utility for packaging a set of files as a single archive

Tcl scripting language

Webmin a web-based interface for system administration for Unix which is the main configuration utility in Skolelinux

Appendix A

Testlog: FauBackup

A.1 Installation and configuration

FauBackup was downloaded from SourceForge (https://sourceforge.net/project/showfiles.php?group_id=73776), unpacked in */usr/src* and compiled and installed with *make; make install*. There was no problems during the installation phase.

To prevent the ssh-connection asking for password, there were created a key pair without password protection on the client computer and the public-part of the key was added to the list of authorized keys on the root-account on the backup server (*sandbox.idi.ntnu.no*).

A.2 Testing hard links, sym links and files with holes

Hard links were tested with making a hard link between *python_stat.zip* and *python_stat-hardlink.zip*:

```
werner@sule03:~$ ls -i pyse\_stat*
  32708 pyse\_stat-hardlink.zip    32708 pyse\_stat.zip
```

On the backup server, these files also have the same inode-number:

Sym links were tested with a sym link, already in the home directory, to */local/docs*:

```
werner@sule03:~$ ls -l docs
lrwxrwxrwx    1 werner   1000          12 Aug 20 13:29 docs -> /local/docs/
```

On the backup server, this file looks like:

```
werner@sule03:~$ ls -l docs
lrwxrwxrwx    1 werner   1000           12 Aug 20 13:29 docs -> /local/docs/
```

To test if the backup system handles files with holes, a C++ program that makes files with holes was used to make a 1GB file. The home directory still had the size of approximately 600MB, and on the backup server the size did not change (see the line including faubackup):

```
sandbox:/backup# du --max-depth=1 -h .
16k     ./lost+found
1.6G    ./rdiff-backup
8.0k    ./.ssh
619M    ./faubackup
1.6G    ./sule03.idi.ntnu.no
3.7G    .
```

Appendix B

Testlog: rdiff-backup

B.1 Installation and configuration

rdiff-backups requires librsync version 0.9.6 or later. librsync was downloaded from Sourceforge (https://sourceforge.net/project/showfiles.php?group_id=56125) and unpacked in */usr/src* with *tar xvzf /local/download/librsync-0.9.6*. Configuration was executed with *./configure*, building, and testing with *make all check* and installed it with *make install*. rdiff-backup also needed python2.2, so it was installed using *apt-get install python2.2 python2.2-dev*.

rdiff-backup was downloaded from the website of rdiff-backup (http://rdiff-backup.stanford.edu/rdiff-backup-0.12.4.tar.gz), and unpacked in */usr/src* with *tar xvzf /local/download/rdiff-backup-0.12.4.tar.gz*. rdiff-backup was build and installed with *python2.2 setup.py install*. Every step in installing both librsync and rdiff-backup completed smoothly.

To prevent the ssh-connection asking for password, there were created a keypair without password protection on the client computer and the public-part of the key was added to the list of authorized keys on the root-account on the backup server (*sandbox.idi.ntnu.no*).

B.2 Testing hard links, sym links and files with holes

Hard links were tested with making a hard link between *python_stat.zip* and *python_stat-hardlink.zip*:

```
werner@sule03:~$ ls -i pyse\_stat*
  32708 pyse\_stat-hardlink.zip    32708 pyse\_stat.zip
```

On the backup server, these files also have the same inode-number:

```
sandbox:/backup/rdiff-backup/sule03-home-werner# ls -i pyse\_stat*
 180617 pyse\_stat-hardlink.zip   180617 pyse\_stat.zip
```

Sym links were tested with a sym link, already in the home directory, to */local/docs*:

```
werner@sule03:~$ ls -l docs
lrwxrwxrwx    1 werner   1000          12 Aug 20 13:29 docs -> /local/docs/
```

On the backup server, this file looks like:

```
sandbox:/backup/rdiff-backup/sule03-home-werner# ls -l docs
lrwxrwxrwx    1 root     root          12 Oct  4 03:01 docs -> /local/docs/
```

To test if the backup system handles files with holes, a C++ program that makes files with holes was used to make a 1GB file. The home directory still had the size of approximately 600MB, and on the backup server the size was 1GB larger than it should be (see the line including rdiff-backup):

```
sandbox:/backup# du --max-depth=1 -h .
16k     ./lost+found
1.6G    ./rdiff-backup
8.0k    ./.ssh
619M    ./faubackup
1.6G    ./sule03.idi.ntnu.no
3.7G    .
```

rdiff-backup does not handle files with holes.

Appendix C

Testlog: rlbackup

C.1 Installation and configuration

rlbackup requires rsync version 2.5.6 or later. rsync was downloaded from *http://dp.samba.org/ftp/rsync/rsync-2.5.6.tar.gz*, unpacked in */usr/src* and compiled and installed with *./configure; make; make install.*

rlbackup was downloaded from (http://www.math.ualberta.ca/imaging/rlbackup/rlbackup-2.06.tar.gz), unpacked in */usr/src* and made a symbolic link to the rsync source directory called *rsync-src*. The client installation was as easy as *make install-client* and edit the file */usr/local/etc/rlbackup.conf*. On the server, rlbackup is supposed to run in a chrooted shell to increase the security. So when trying to compile and install rlbackup with *make install-server*, this problem occurs:

```
werner:/usr/src/rlbackup-2.06# make install-server
g++ -ansi -D_FILE_OFFSET_BITS=64 -D_LARGEFILE_SOURCE -O3 -Wall -DBACKUP=\"/backup\"
  -DBINDIR=\"/bin\" -DMOUNT=\"/bin/mount\" -DUMOUNT=\"/bin/umount\"
  -DMOUNTOPT=\""-t ext2 -o ro,loop"\" -DBINFS=\"/usr/local/bin/bin.ext2\" -DNICE=19
  -o rsyncsh rsyncsh.cc
rsyncsh.cc: In function ‘char ** args(const char *)’:
rsyncsh.cc:70: implicit declaration of function ‘int strdup(...)’
rsyncsh.cc:70: initialization to ‘char *’ from ‘int’ lacks a cast
rsyncsh.cc:73: implicit declaration of function ‘int index(...)’
rsyncsh.cc:73: assignment to ‘char *’ from ‘int’ lacks a cast
rsyncsh.cc:77: assignment to ‘char *’ from ‘int’ lacks a cast
rsyncsh.cc: In function ‘int System(const char *, const char * = 0)’:
rsyncsh.cc:90: implicit declaration of function ‘int chroot(...)’
rsyncsh.cc: In function ‘char * remotehost()’:
rsyncsh.cc:111: initialization to ‘char *’ from ‘int’ lacks a cast
rsyncsh.cc:116: implicit declaration of function ‘int inet_aton(...)’
rsyncsh.cc: In function ‘int main(int, char **)’:
rsyncsh.cc:142: initialization to ‘char *’ from ‘int’ lacks a cast
make: *** [rsyncsh] Error 1
```

A quick solution was to remove the *-ansi*-option to g++ in rlbackup's *Makefile*. Then it worked doing a *make install-server*. Now, after the *INSTALL*-file's instructions, the command *mkdir -p /backup/sule03.idi.ntnu.no/bin* had to be run.

C.2 Testing hard links, sym links and files with holes

Hard links were tested with making a hard link between *python_stat.zip* and *python_stat-hardlink.zip*:

```
werner@sule03:~$ ls -i pyse\_stat*
  32708 pyse\_stat-hardlink.zip    32708 pyse\_stat.zip
```

On the backup server, these files also have the same inode-number:

```
sandbox:/backup/sule03.idi.ntnu.no/1/werner# ls -i pyse\_stat*
 542026 pyse\_stat-hardlink.zip   542026 pyse\_stat.zip
```

Sym links were tested with a sym link, already in the home directory, to */local/docs*:

```
werner@sule03:~$ ls -l docs
lrwxrwxrwx    1 werner   1000           12 Aug 20 13:29 docs -> /local/docs/
```

On the backup server, this file looks like:

```
sandbox:/backup/sule03.idi.ntnu.no/1/werner# ls -l docs
lrwxrwxrwx    1 werner   1000           12 Oct  4 03:54 docs -> /local/docs/
```

To test if the backup system handles files with holes, a C++ program that makes files with holes was used to make a 1GB file. The home directory still had the size of approximately 600MB, and on the backup server the size was 1GB larger than it should be (see the line including rlbackup):

```
sandbox:/backup# du --max-depth=1 -h .
16k     ./lost+found
1.6G    ./rdiff-backup
8.0k    ./.ssh
619M    ./faubackup
1.6G    ./sule03.idi.ntnu.no
3.7G    .
```

rlbackup does not handle files with holes.

Appendix D

Users manual

Ved aktiv bruk av IT-systemer, er det vanlig at harddisker og andre deler i systemet feiler. Det er også vanlig at brukere er uheldige og sletter filer. Grunnen til å ha et backupsystem er at man i disse tilfellene har en mulighet til å komme tilbake til der man var før uhellet var ute.

Kostnadene ved det å ha et backupsystem, kan egentlig deles i to; utstyr og arbeid. Utstyret du trenger er stort sett en stor harddisk som har plass til alt du skal ta backup av i tillegg til de historiske endringene du ønsker å lagre. Det vil si hvis du ønsker å ta vare på backup i et halvt år, må du, hvis du bruker backupsystemet i Skolelinux, lagre alle endringene som har skjedd dette halve året. Det finnes backupsystemer som tar vare på daglig backup i en uke, ukentlige backup i en måned, månedlige backup i et år osv, men det backupsystemet som er brukt i Skolelinux tar vare på alle endringer fra dag til dag i den tiden du ønsker dette.

Antall brukere	**GB**		
	3 mnd.	**6 mnd.**	**12 mnd.**
10	0,8GB	1,4GB	2,6GB
100	7,8GB	14GB	26GB
500	40GB	70GB	130GB

Table D.1: Forslag til anbefalt diskplass for backup basert på 2 måneders statistikk fra skolene i Time kommune, Runni ungdomsskole og Ulsrud VGS. NB! dette er **kun** et forslag.

Arbeidskostnaden med backup er hovedsakelig forbundet med gjenskapning av tapte data. To typiske eksempler på dette er at en bruker har ved et uhell slettet filer/kataloger på hjemmeområdet sitt og ønsker disse tilbake og at en maskin eller harddisk har blitt ødelagt og man ønsker tilbake viktige data på denne.

D.1 Skolelinux Backup

Skolelinux kommer med et ferdig konfigurert og igangsatt backupsystem, “Skolelinux Backup”. For å forklare hvordan tjenesten “Skolelinux Backup” er bygd opp, er det tre roller i backupsystemet som må være definert:

tjener Maskinen som “Skolelinux Backup” er installert på, hvor konfigurasjonsfilen ligger og eventuelt Webmin-modulen er installert.

backupklient En maskin som er definert som klient i konfigurasjonen til “Skolelinux Backup”. Maskinen(e) som har denne rollen, blir tatt backup av.

backuptjener En maskin som er definert som backuptjener i konfigurasjonen til “Skolelinux Backup”. Det er på denne maskinen backup lagres.

En viktig ting å huske på, er at en maskin kan ha flere av disse rollene, men kun en maskin kan inneha rollene tjener og backuptjener (kan være på samme eller forskjellig maskin) og flere maskiner kan ha rollen backupklient.

Hver natt starter backuptjenesten på maskinen som har tjener-rollen. Her blir backupklientene behandlet i rekkefølge, hvor de filene/katalogene som er oppgitt for den backupklienten i konfigurasjonsfilen blir tatt backup av. Backupen blir plassert på backuptjeneren.

Følgende aksjoner i forbindelse med backupsystemet er nødvendige:

1. Konfigurere, beskrevet i kapittel D.3.2
2. Gjenskape data, beskrevet i kapittel D.3.3
3. Konsistenssjekke backupsystemet, beskrevet i kapittel D.4

D.2 Installasjon

Installasjonen av backupsystemet på tjenermaskinen gjøres når du installerer Skolelinux. Hvis du ønsker å ta backup av flere enn tjenermaskinen, eller ønsker å lagre backup på en annen maskin enn tjener, krever dette at du installerer noe programvare.

D.2.1 Ny klient

Hvis du ønsker å ta backup av flere enn tjenermaskinen, f.eks. en LTSP-tjener, må du på denne maskinen installere følgende programpakker (gjøres f.eks. med *apt-get install*):

- *rdiff-backup*
- *ssh*

For at backupsystemet faktisk skal ta backup av denne klienten, må den også legges til i konfigurasjonen. Se kapittel D.3 for en detaljert beskrivelse av dette.

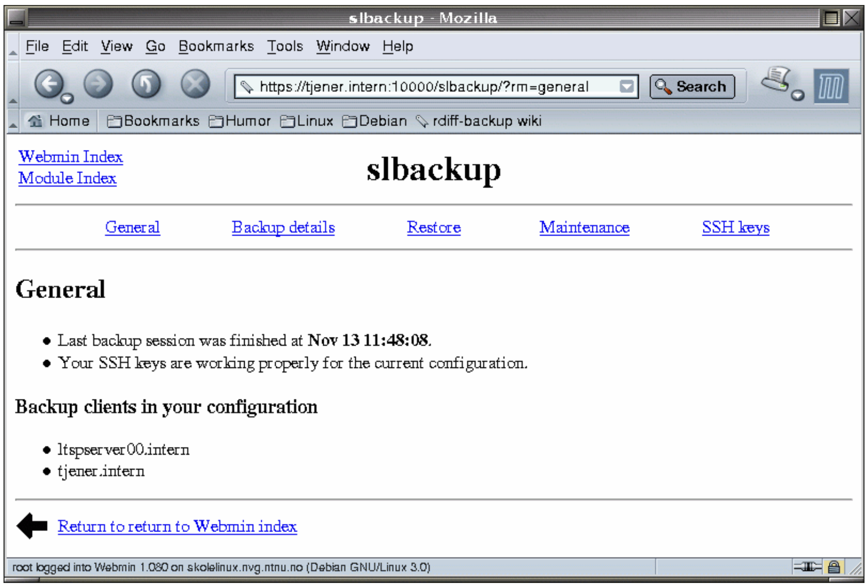
D.3 Konfigurasjon

Konfigurering av backupsystemet kan gjøres på to måter; i et vevgrensesnitt som er endel av Webmin eller direkte editering av konfigurasjonsfiler. I denne manualen tar vi for oss vevgrensesnittet.

Vevgrensesnittet består av fem hoveddeler som er kort beskrevet i listen under:

- **General**
  Denne delen er en slags startside, og inneholder litt generell informasjon om backupsystemet.
- **Backup details**
  Det er i denne delen brukeren har mulighet til å konfigurere backupsystemet. Backupsystemet kan bli koplet inn og ut, starttidspunktet for backupsesjonen (som kjøres en gang per døgn) kan spesifiseres, klientmaskiner kan legges til, slettes og konfigureres og backupserveren kan konfigureres.
- **Restore**
  I denne delen har brukeren mulighet til å gjenopprette filer fra backup.
- **Maintenance**
  Denne delen gjør det mulig for brukeren å vedlikeholde backupen. Den eneste funksjonen som er tilgjengelig her, er å slette gammel backup fra backupserveren for å spare plass.
- **SSH keys**
  For at backup skal kunne tas av eksterne klientmaskiner hver natt uten at noen skal være der å taste inn et passord, benytter backupsystemet SSH nøkler. Denne delen av systemet konfigurerer nøklene for de eksterne klientmaskinene.

I de neste delkapitlene vil hver av delene i listen over være detaljert beskrevet.


slbackup - Mozilla
File Edit View Go Bookmarks Tools Window Help
https://tjener.intern:10000/slbackup/?rm=general
Search
Home Bookmarks Humor Linux Debian rdiff-backup wiki

Webmin Index
Module Index

slbackup

General Backup details Restore Maintenance SSH keys

General

- Last backup session was finished at **Nov 13 11:48:08**.
- Your SSH keys are working properly for the current configuration.

Backup clients in your configuration

- ltspserver00.intern
- tjener.intern

Return to return to Webmin index

root logged into Webmin 1.080 on skolelinux.nvg.ntnu.no (Debian GNU/Linux 3.0)

Figure D.1: Skjermbilde av “General” siden i vevgrensesnittet

D.3.1 General

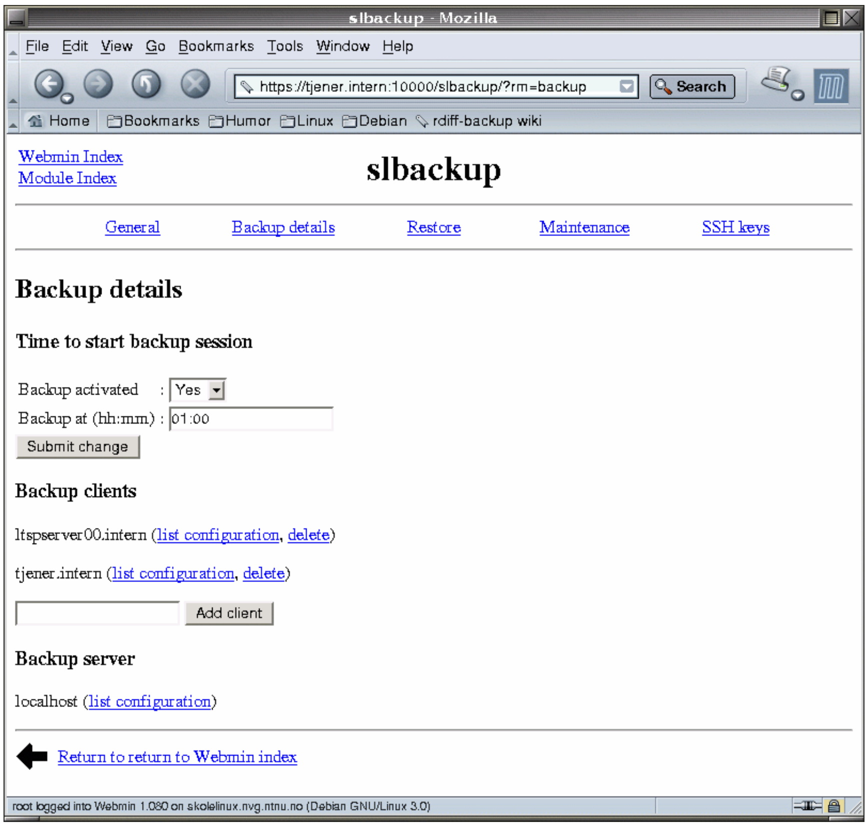
Denne delen er kun en slags startside, og inneholder ikke noe funksjonalitet, kun informasjon om backupsystemet. I figur D.1, ser vi et eksempel på hvordan en slik side kan se ut.

D.3.2 Backup details

På denne siden har du tilgang til å se på og forandre all konfigurasjon av både backupklienter og backupserver. Figur D.2 viser et eksempel på hvordan en slik side kan se ut.

På denne siden kan du spesifisere følgende:

- om du vil at det skal bli tatt backup hver dag og eventuelt når på døgnet du vil at dette skal skje (kl. 01:00 er Skolelinux standardvalg, og også et smart valg)
- se på og forandre konfigurasjonen til en enkelt backupklient
- legge til backupklienter
- slette backupklienter
- se på og forandre konfigurasjonen til backupserveren


slbackup - Mozilla
File Edit View Go Bookmarks Tools Window Help
https://tjener.intern:10000/slbackup/?rm=backup
Search
Home Bookmarks Humor Linux Debian rdiff-backup wiki
Webmin Index
Module Index
General
Backup details
Restore
Maintenance
SSH keys
Submit change
Add client
Return to return to Webmin index

slbackup

Backup details

Time to start backup session

Backup activated : Yes
Backup at (hh:mm) : 01:00

Backup clients

ltspserver00.intern (list configuration, delete)

tjener.intern (list configuration, delete)

Backup server

localhost (list configuration)

root logged into Webmin 1.080 on skolelinux.nvg.ntnu.no (Debian GNU/Linux 3.0)

Figure D.2: Skjermbilde av “Backup details” siden i vevgrensesnittet

Hvis du forandrer et av valgene som omhandler backuptidspunkt og trykker “Submit”-knappen vil du bli sent tilbake til den samme siden med en melding om at tidspunktet ble forandret.

Hvis du vil se på og eventuelt forandre konfigurasjonen til en backupklient, trykker du på “list configuration” og du vil få opp en egen side som vist i figur D.3. Hvert valg på siden er beskrevet under:

Type of client Det finnes to typer backupklienter, ekstern og lokal. Med ekstern menes at klienten ikke er den maskinen Webmin kjøres på. I Skolelinux vil lokal være *tjener.intern*, og ekstern være alle de andre maskinene som f.eks. *ltspserver00* hvis du har en egen LTSP-tjener.

Hostname or IP-address Dette er nettverksadressen til klienten (brukes bare hvis klienten er av type ekstern). Du kan enten bruke maskinnavnet (f.eks. *ltspserver00.intern*) eller IP-adressen (som for *ltspserver00* oftest er *10.0.2.10*).

Username Hvis klienten er av typen ekstern, må backupsystemet logge inn på klienten med en bruker når backup skal taes. Denne brukeren vil i de fleste tilfeller være *root*, men kan i spesielle tilfeller være noe annet. Dette feltet blir ikke brukt hvis klienten er av type lokal.

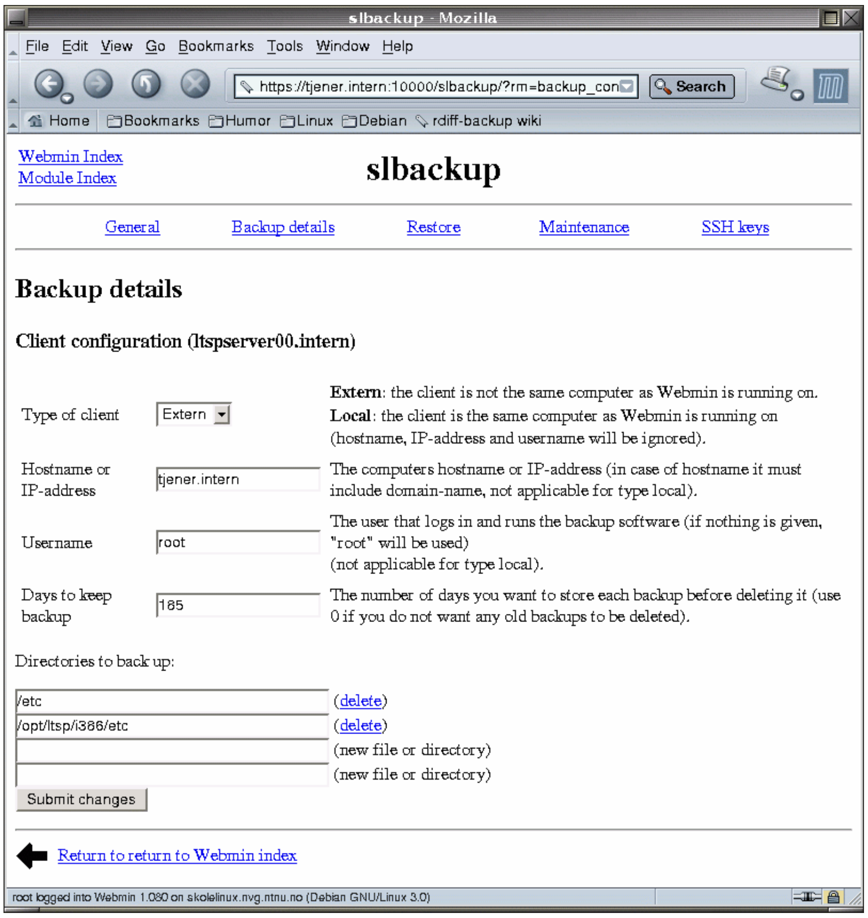
Days to keep backup Du kan velge hvor lenge du vil ta vare på gammel backup. Skolelinux anbefaler et halvt år (ca. 185 dager), men dette avhenger blant annet av hvor mye harddiskplass du har tilgjengelig. Hvis du ikke ønsker at backupsystemet skal slette gammel backup automatisk, velger du 0 i dette feltet (du kan også slette gammel backup manuelt på “Maintenance”-siden).

En viktig ting å bemerke seg i dette punktet, er at jo flere dager du velger å ta vare på backupen, jo mer diskplass trenger du (se tabell D.1).

Directories to back up Denne listen viser hvilke filer eller kataloger du ønsker at backupsystemet skal ta backup av på klientmaskinen. I konfigurasjonsgrensesnittet har du mulighet til å legge til to ekstra filer eller kataloger (hvis du ønsker å legge til flere enn to, må du legge til to og to av gangen).

Bak de katalogene som allerede er definert for klienten, er det en lenke som heter “delete”. Denne bruker du hvis du ønsker at den tilhørende katalogen ikke lengre skal taes backup av.

Submit changes Når du har gjort noen forandringer i grensesnittet beskrevet over, og ønsker å lagre dette trykker du på denne knappen. Du vil da bli sent tilbake til den samme siden, men med de nye verdiene fylt inn og en melding på toppen av siden som bekrefter forandringen(e) du gjorde.


slbackup - Mozilla
File Edit View Go Bookmarks Tools Window Help
https://tjener.intern:10000/slbackup/?rm=backup_con
Search
Home Bookmarks Humor Linux Debian rdiff-backup wiki
Webmin Index
Module Index

slbackup

General | Backup details | Restore | Maintenance | SSH keys

Backup details

Client configuration (ltspserver00.intern)

Type of client	Extern	**Extern**: the client is not the same computer as Webmin is running on. **Local**: the client is the same computer as Webmin is running on (hostname, IP-address and username will be ignored).
Hostname or IP-address	tjener.intern	The computers hostname or IP-address (in case of hostname it must include domain-name, not applicable for type local).
Username	root	The user that logs in and runs the backup software (if nothing is given, "root" will be used) (not applicable for type local).
Days to keep backup	185	The number of days you want to store each backup before deleting it (use 0 if you do not want any old backups to be deleted).

Directories to back up:

/etc	(delete)
/opt/ltsp/i386/etc	(delete)
	(new file or directory)
	(new file or directory)

Submit changes

Return to return to Webmin index

root logged into Webmin 1.080 on skolelinux.nvg.ntnu.no (Debian GNU/Linux 3.0)

Figure D.3: Skjermbilde av "Configure client" siden i vevgrensesnittet

Hvis du vil slette en backupklient, trykker du på “delete” på samme linjen som klienten du vil slette, og du vil få opp en ny side som spør deg om du er sikker på dette. Her må du trykke “Delete!”, og du vil få opp siden vist i figur D.2 på nytt.

Hvis du vil legge til en backupklient, skriver du inn navnet på klienten i tekstfeltet til venstre for knappen “Add client” (dette navnet er noe du velger, og det er lurt å velge et navn som gjør at du kjenner igjen klienten) og trykker “Add client”. Da vil du få opp en side som likner på den som vises i figur D.4.

Valgene du har på denne siden er veldig like de på siden for konfigurasjon av klienten. Den eneste forskjellen er at det ikke er fylt inn noen filer eller kataloger som skal taes backup av. For de forskjellige Skolelinux maskinene har vi følgende forslag:

Tjener:

- */etc*
- */skole/tjener/home0*
- */var/backups*

LTSP-tjener:

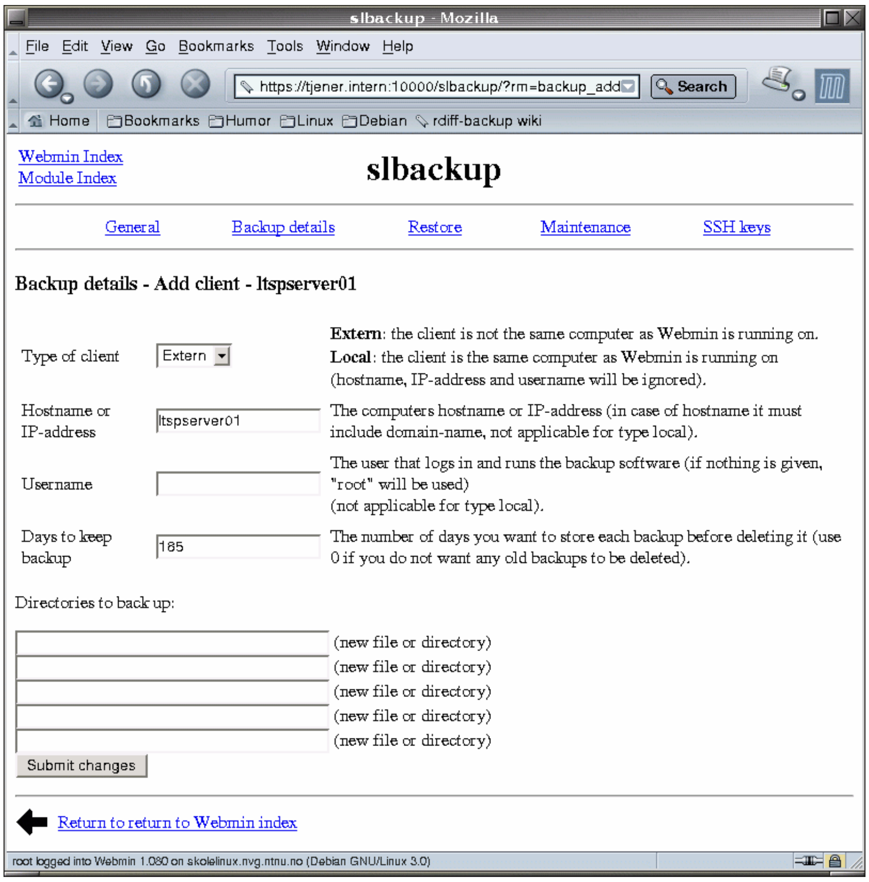
- */etc*
- */opt/ltsp/i386/etc*

Kombinert Tjener og LTSP-tjener:

- */etc*
- */opt/ltsp/i386/etc*
- */skole/tjener/home0*
- */var/backups*

Når du trykker “Submit changes” vil du få opp konfigurasjonssiden beskrevet over, nå med de verdiene du valgte samt en beskjed på toppen av siden om at er lagt til og eventuelt en advarsel hvis noen av verdiene du skrev inn ikke var gyldige.

Hvis du ønsker å se på eller forandre konfigurasjonen til backuptjeneren, trykker du på “list configuration” under overskriften “Backup server” nederst på siden. Da vil du få opp en egen side, som vist i figur D.5.



Backup details - Add client - ltspserver01

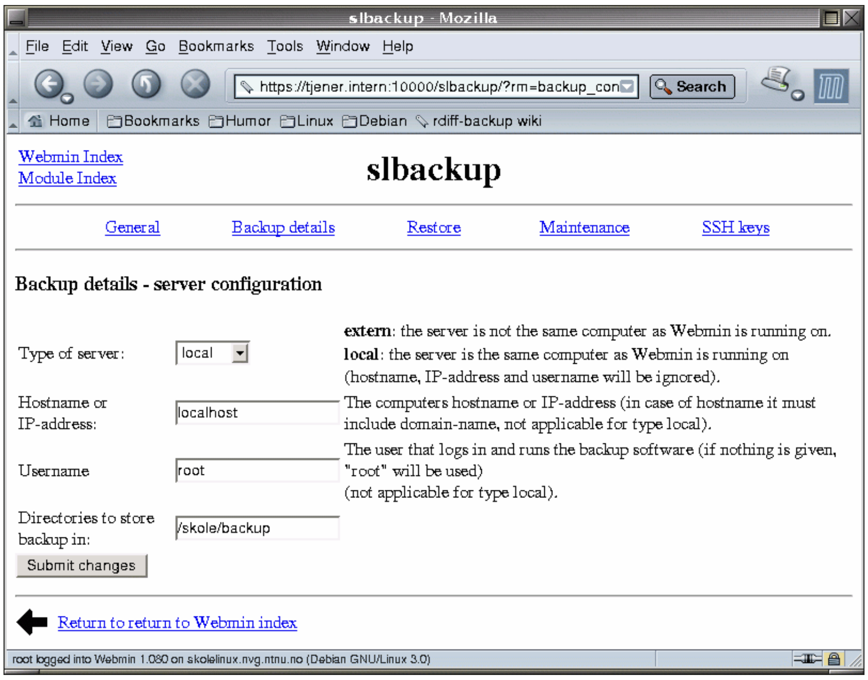
Type of client	Extern	**Extern**: the client is not the same computer as Webmin is running on. **Local**: the client is the same computer as Webmin is running on (hostname, IP-address and username will be ignored).
Hostname or IP-address	ltspserver01	The computers hostname or IP-address (in case of hostname it must include domain-name, not applicable for type local).
Username		The user that logs in and runs the backup software (if nothing is given, "root" will be used) (not applicable for type local).
Days to keep backup	185	The number of days you want to store each backup before deleting it (use 0 if you do not want any old backups to be deleted).

Directories to back up:

(new file or directory)
(new file or directory)
(new file or directory)
(new file or directory)
(new file or directory)

root logged into Webmin 1.080 on skolelinux.nvg.ntnu.no (Debian GNU/Linux 3.0)

Figure D.4: Skjermbilde av “Add client” siden i vevgrensesnittet


slbackup - Mozilla
File Edit View Go Bookmarks Tools Window Help
https://tjener.intern:10000/slbackup/?rm=backup_con
Search
Home Bookmarks Humor Linux Debian rdiff-backup wiki
Webmin Index
Module Index
General
Backup details
Restore
Maintenance
SSH keys
Submit changes
Return to return to Webmin index

slbackup

Backup details - server configuration

Type of server:	local	**extern**: the server is not the same computer as Webmin is running on. **local**: the server is the same computer as Webmin is running on (hostname, IP-address and username will be ignored).
Hostname or IP-address:	localhost	The computers hostname or IP-address (in case of hostname it must include domain-name, not applicable for type local).
Username	root	The user that logs in and runs the backup software (if nothing is given, "root" will be used) (not applicable for type local).
Directories to store backup in:	/skole/backup	

root logged into Webmin 1.080 on skolelinux.nvg.ntnu.no (Debian GNU/Linux 3.0)

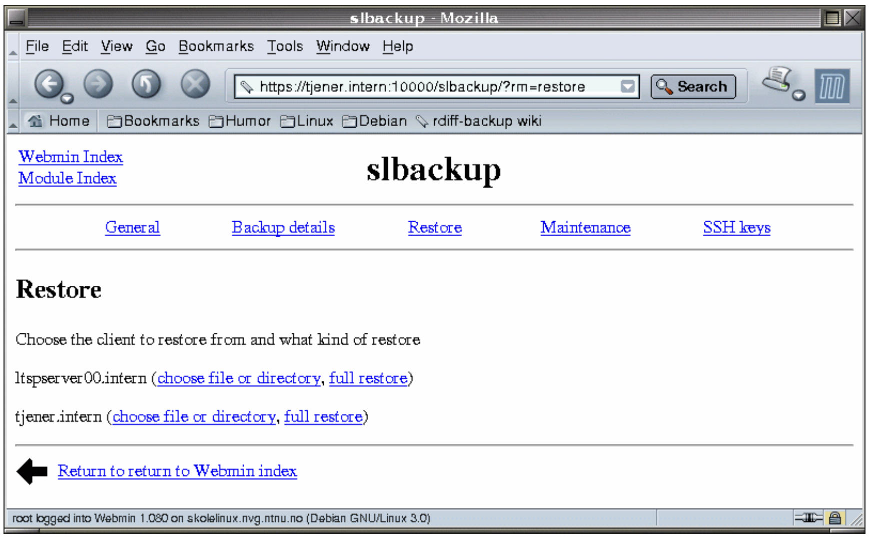
Figure D.5: Skjermbilde av "Server configuration" siden i vevgrensesnittet

Type of server Det finnes to typer backuptjenere, ekstern og lokal. Med ekstern menes at tjeneren ikke er den maskinen Webmin kjøres på. I Skolelinux vil lokal bety at du lagrer backup på *tjener.intern*, og ekstern være å lagre backup på en annen maskin, noe som vil være å anbefale da dette vil gjøre at du kan dra nytte av flere sider ved backup.

Hostname or IP-address Dette er nettverksadressen til backupserveren (brukes bare hvis den er av type ekstern). Du kan enten bruke maskinnavnet (f.eks. *backup.intern* hvis den er satt opp i DNS) eller IP-adressen (som for *backup.intern* kan være *10.0.2.5*).

Username Hvis backupserveren er av typen ekstern, må backupsystemet logge inn på den med en bruker når backup skal taes. Denne brukeren vil i de fleste tilfeller være *root*, men avhengig av oppsettet på backupserveren kan dette i noen tilfeller være noe annet. Dette feltet blir ikke brukt hvis backupserveren er av type lokal.

Directory to store backup in Her skal katalogen hvor backup skal lagres spesifiseres. I en standard Skolelinux-installasjon vil dette være */skole/backup* på tjeneren.


slbackup - Mozilla
File Edit View Go Bookmarks Tools Window Help
https://tjener.intern:10000/slbackup/?rm=restore
Search
Home Bookmarks Humor Linux Debian rdiff-backup wiki
Webmin Index
Module Index
General Backup details Restore Maintenance SSH keys

slbackup

Restore

Choose the client to restore from and what kind of restore

ltspserver00.intern (choose file or directory, full restore)

tjener.intern (choose file or directory, full restore)

Return to return to Webmin index

root logged into Webmin 1.080 on skolelinux.nvg.ntnu.no (Debian GNU/Linux 3.0)

Figure D.6: Skjermbilde av "Restore" siden i vevgrensesnittet

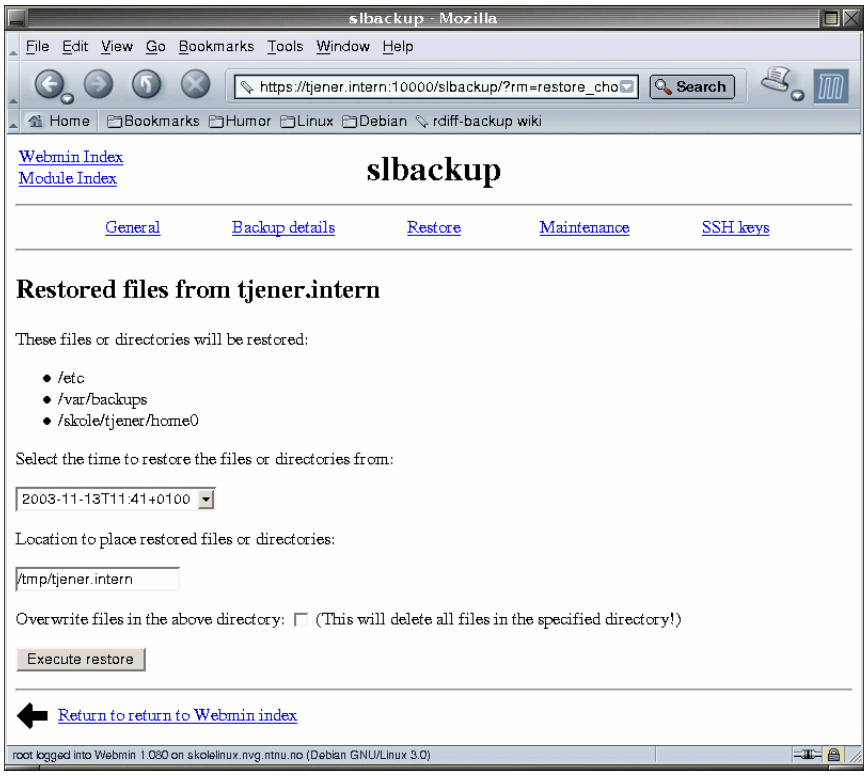
D.3.3 Restore

Denne siden gir deg et enkelt, grensesnitt til å gjenskape filer fra backup. På denne siden, som er vist i figur D.6, har du to valg. Enten kan du gjenskape en enkeltfil eller katalog, eller du kan gjenskape alle filer som har blitt tatt backup av fra en klient.

Hvis du ønsker å gjenskape kun en fil eller en katalog, trykker du på "choose file or directory" tilhørende den klienten du ønsker å gjenskape fra. Nå vil du få opp en side hvor du må skrive inn full bane til den filen eller katalogen du vil gjenskape (eksempel: */skole/tjener/home0/olan/rapport.sxw*) og trykke "Choose file/directory". Nå vil en side, som likner på den i figur D.7 komme opp. Hvis du hadde valgt "Full restore" ville du ha kommet direkte til denne siden.

Du vil få presentert en liste over hvilke filer eller kataloger som vil bli gjenskapt. Det er også en liste over hvilke datoer backupsystemet har tilgjengelige backup fra. Den nyeste backup vil være den som er forhåndsvalgt. Du må også skrive inn hvor du vil at den/de gjenskapte filen(e) eller katalog(ene) skal lagres. Standard er å gjenskape filene til */tmp/<maskinnavn>/*, men her kan du bytte ut hvis du vil ha filene et annet sted.

NB! Hvis den filbanen du oppgir her finnes fra før, vil backupsystemet feile, og du får opp en side som forteller deg at filen(e) eller katalog(ene) finnes fra før. Hvis du er sikker på at du vil overskrive denne/disse, må du gå tilbake til forrige side (med "Tilbake"-knappen i nettleseren din) og krysse av for "Overwrite files in the above directory".


slbackup - Mozilla
File Edit View Go Bookmarks Tools Window Help
https://tjener.intern:10000/slbackup/?rm=restore_cho
Search
Home Bookmarks Humor Linux Debian rdiff-backup wiki
Webmin Index
Module Index
General
Backup details
Restore
Maintenance
SSH keys
Execute restore
Return to return to Webmin index

slbackup

Restored files from tjener.intern

These files or directories will be restored:

- /etc
- /var/backups
- /skole/tjener/home0

Select the time to restore the files or directories from:

2003-11-13T11:41+0100

Location to place restored files or directories:

/tmp/tjener.intern

Overwrite files in the above directory: ☐ (This will delete all files in the specified directory!)

root logged into Webmin 1.080 on skolelinux.nvg.ntnu.no (Debian GNU/Linux 3.0)

Figure D.7: Skjermbilde av “Restore details” siden i vevgrensesnittet

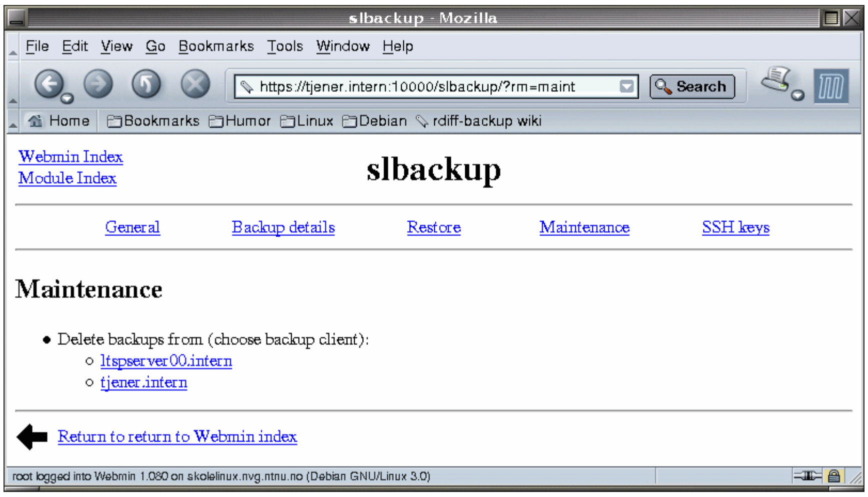
NB2! Hvis du krysser av for dette, og du skal gjenskape en katalog, vil alt i katalogen bli slettet, og gjenopprettet på nytt (dvs. du vil miste filer som har blitt laget/forandret siden sist backup), så bruk denne muligheten med omhu!

NB3! Når du trykker på “Execute restore”-knappen, er det viktig at du ikke trykker Escape-knappen på tastaturet eller Stopp-knappen i nettleseren, for da vil du avbryte gjenskapingen, noe som høyst sannsynlig vil føre til at det du gjenskaper vil bli ødelagt.

D.3.4 Maintenance

På denne siden kan du vedlikeholde backup som er tatt. Vedlikeholde betyr her å slette gammel backup. Bakgrunnen for å ville gjøre dette, er f.eks. manglende harddisk-plass. Siden du får opp ser omtrent ut som den i figur D.8.

Når du har bestemt hvilken klient du vil slette backup for, klikker du på klientens navn, og du vil få opp en side som ser ut som figur D.9. På denne siden må du velge den datoen som skal være den siste du beholder. Dvs. at


slbackup - Mozilla
File Edit View Go Bookmarks Tools Window Help
https://tjener.intern:10000/slbackup/?rm=maint
Search
Home Bookmarks Humor Linux Debian rdiff-backup wiki

Webmin Index
Module Index

slbackup

General Backup details Restore Maintenance SSH keys

Maintenance

- Delete backups from (choose backup client):
  - ltspserver00.intern
  - tjener.intern

Return to return to Webmin index

root logged into Webmin 1.080 on skolelinux.nvg.ntnu.no (Debian GNU/Linux 3.0)

Figure D.8: Skjermbilde av "Maintenance" siden i vevgrensesnittet

alle backups som er eldre enn valgte dato vil bli slettet. Når du så trykker på "delete"-knappen, vil du få opp en side med en liste over alle backup'ene som vil bli slettet, og du må bekrefte at du faktisk vil slette disse.

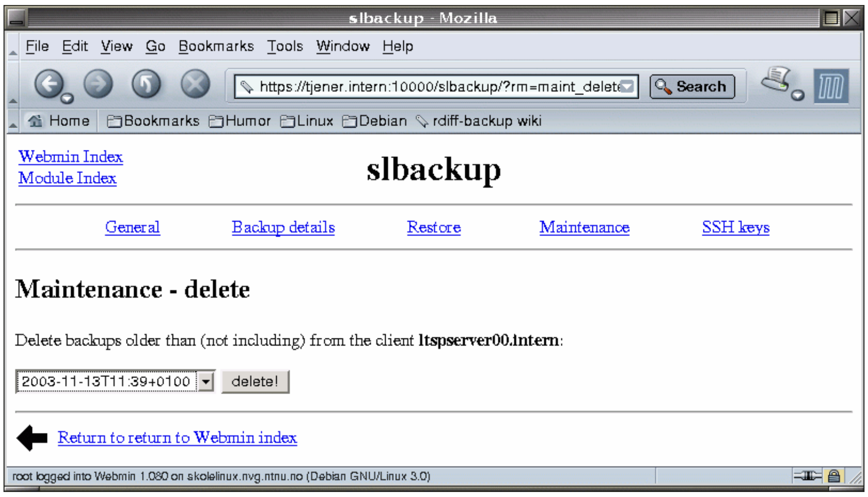
Når du bekrefter dette, vil backupsystemet begynne jobben med å slette disse, noe som kan ta litt tid. **NB!** Det er nå viktig at du ikke trykker Escape-tasten på tastaturet eller Stopp-knappen i nettleseren din, for da vil du avbryte slette-jobben, noe som kan ødelegge deler av backupen din.

D.3.5 SSH keys

På denne siden sjekker du om oppsettet ditt fungerer med tanke på SSH-nøkler. Dette er siden du vil besøke hvis du på "General"-siden får beskjed om at SSH-nøklene dine ikke fungerer tilfredsstillende for oppsettet ditt. Skolelinux Backup bruker SSH-nøkler for å unngå lagring av passord. En kort forklaring på SSH-nøkler kommer under. Denne siden automatiserer det arbeidet det er å få SSH-nøklene på plass for at den passordløse forbindelsen skal fungere når backupsystemet kjører igang midt på natta (SSH-nøkler er en sikker måte å gjøre dette på, selv om det kan høres litt farlig ut med passord-løs innlogging). Siden kan likne på det som er vist i figur D.10

Funksjonaliteten som blir presentert på denne siden er (for å forstå detaljene, se den enkle forklaringen under lista over funksjonene):

Add Dette kopierer den offentlige delen av nøkkelparet over på den eksterne maskinen. For å gjøre dette får du opp en side hvor du blir spurt om å oppgi passord. Dette passordet tilhører brukeren du har oppgitt i klientkonfigurasjonen på klientmaskinen.


slbackup - Mozilla
File Edit View Go Bookmarks Tools Window Help
https://tjener.intern:10000/slbackup/?rm=maint_delete
Search
Home Bookmarks Humor Linux Debian rdiff-backup wiki
Webmin Index
Module Index
General Backup details Restore Maintenance SSH keys
delete!
Return to return to Webmin index

slbackup

Maintenance - delete

Delete backups older than (not including) from the client **ltspserver00.intern**:

2003-11-13T11:39+0100

root logged into Webmin 1.080 on skolelinux.nvg.ntnu.no (Debian GNU/Linux 3.0)

Figure D.9: Skjermbilde av “Maintenance - delete” siden i vevgrensesnittet

Create Dette lager et nøkkel-par (privat og offentlig del) på den lokale maskinen.

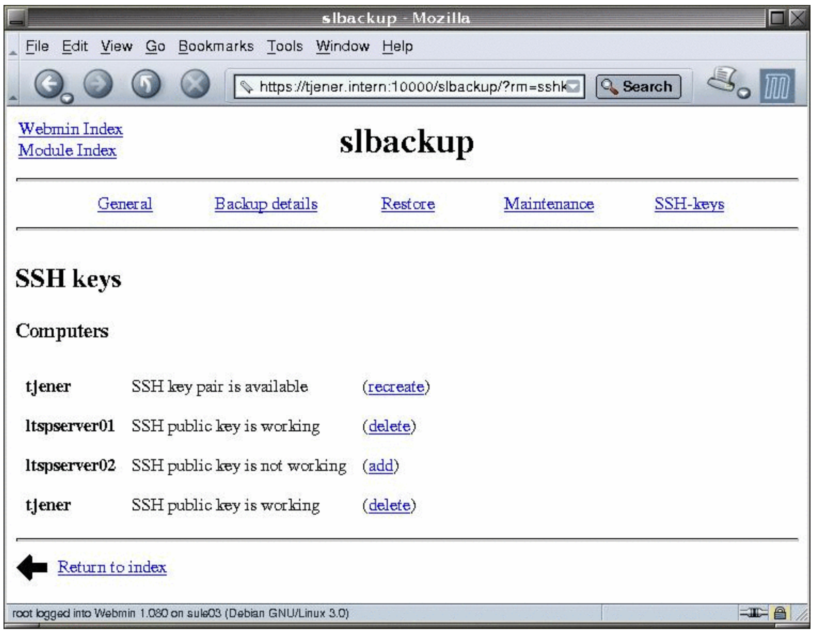
Hva er SSH-nøkler og hva brukes de til?
Kort fortalt kan man bruke SSH-nøkler for å logge seg inn på en maskin over nettverket uten å bruke passord. Det første som må gjøres, er å lage et nøkkel-par, som består av en privat og en offentlig nøkkel, på hjemmeområdet sitt (*.ssh/id_dsa* og *.ssh/id_dsa.pub* er hendholsvis den private og offentlige nøkkelen). Det neste som må gjøres er å legge den offentlige nøkkelen (innholdet i fila) til i fila *-.ssh/authorized_keys* på hjemmeområdet til brukeren du skal logge inn som på den eksterne maskinen.

For en mer detaljert beskrivelse av SSH (eget kapittel om nøkler), se på dette dokumentet: http://www.mindrot.org/d̃jm/auug2002/ssh-tutorial.pdf

D.4 Konsistenssjekk

Det er viktig og konsistenssjekke backupen med jevne mellomrom. Grunnen til dette er at det er bedre å oppdage at backupen eventuelt er ødelagt før du virkelig trenger den.

En enkel måte å konsistenssjekke backupen på, er å gjøre en test-gjenskapning av noen data. Et eksempel er at du kan gjenskape alle data til for eksempel en bruker til */tmp/brukernavn* og så sjekke at dette går smertefritt og at filene ligger der og går an å åpne. Hvordan dette gjøres er beskrevet i kapittel D.3.3.



slbackup

SSH keys

Computers

tjener	SSH key pair is available	(recreate)
ltspserver01	SSH public key is working	(delete)
ltspserver02	SSH public key is not working	(add)
tjener	SSH public key is working	(delete)

root logged into Webmin 1.080 on sule03 (Debian GNU/Linux 3.0)

Figure D.10: Skjermbilde av "SSH keys" siden i vevgrensesnittet

D.5 Referanser

Skolelinux Backup:
http://slbackup.alioth.debian.org/

Studentprosjektets:
http://developer.skolelinux.no/info/studentgrupper/2003-backup/

rdiff-backup:
http://rdiff-backup.stanford.edu/

Skolelinux:
http://www.skolelinux.no/

Appendix E

Source code: slbackup

E.1 /etc/slbackup/slbackup.conf

```
<client>
    <localhost>
        address     localhost
        location    /etc
        location    /home
        location    /var/backups
        type        local
        user        root
        keep        185
    </localhost>
#    <externhost>
#        address     extern.domain
#        location    /etc
#        location    /var/backups
#        type        extern
#        user        root
#        keep        0
#    </externhost>
</client>
server_address backupserver.domain
server_destdir  /backup
server_type     local
server_user     root
```

E.2 /usr/share/perl5/SLBackup.pm

```
#!/usr/bin/perl
#
# Library for use with slbackup (Skolelinux Backup)
#
```

```
# Content:
# - deal with configuration files
# - deal with log files
#
# $Id: SLBackup.pm,v 1.2 2003/11/27 17:13:47 werner Exp $
#
# Most of the code in this module is copied from the
# LRRD project (http://www.linpro.no/project/lrrd/)
#
# Thanks to Linpro AS for well written perl code!
#

package SLBackup;

use Exporter;
@ISA = ('Exporter');
@EXPORT = ('slbackup_overwrite',
           'slbackup_readconfig',
           'slbackup_writeconfig',
           'slbackup_config' );

use strict;
use Config::General;

my $config = undef;
my $configfile = '/etc/slbackup/slbackup.conf';
my $DEBUG = 0;


sub slbackup_readconfig {
    my ($conf, $missingok) = @_;
    $conf ||= $configfile;

    if (! -r $conf and ! $missingok) {
        print "slbackup_readconfig: cannot open '$conf'\n";
        return undef;
    }

    my $conffile = new Config::General($conf);
    my $config = { $conffile->getall };
    return ($config);
}


sub slbackup_writeconfig {
    my ($datafilename, $data) = @_;

    my $datafile = new Config::General();
    $datafile->save_file($datafilename, $data);
}


1;

__END__
```

E.3 /etc/cron.d/slbackup

```
# cron job for Skolelinux Backup (every night at 01:00)
#0 1 * * * root if [ -x /usr/share/slbackup/slbackup-cron -a -f
  /etc/slbackup/slbackup.conf ]; then /usr/share/slbackup/slbackup-cron
  ; fi
```

E.4 /usr/share/slbackup/client/slbackup-cron

```
#!/usr/bin/perl
#
# Script to be run by cron each night to backup locations specified in
# /etc/slbackup/slbackup.conf
#
# $Id: slbackup-cron.pl,v 1.2 2003/11/27 17:13:47 werner Exp $
#

use strict;
use Config::General;
use POSIX qw(strftime);
use SLBackup;

my $logfile = "/var/log/slbackup/slbackup.log";


# open logfile
open (LOG, ">>$logfile") or die ("Unable to open $logfile\n");
logger ("Starting slbackup:");

# fetch configuration
my $conffile = '/etc/slbackup/slbackup.conf';
my $config = &slbackup_readconfig($conffile);

# run rdiff-backup for each client in configuration
for my $key (keys %{$config->{client}}) {
    my $client = $config->{client}->{$key};
    my $execstr = "";
    my $execstr_serverpart = "";
    my $execstr_clientpart = "";

    # check if server is not of type "local" ->
#  add server-part of the exeecstr in a
if (exists ($config->{server_type}) and
    $config->{server_type} ne "local") {

    # check if server_address is present in configuration
    #FIXME - check that the address is valid
    if (!exists ($config->{server_address})) {
        logger ("Address for server is not present in configuration " .
                " file ... please fix!");
        logger("Failed backing up clients.");
        last;
    }

    # check if server_user is present in configuration
    if (!exists ($config->{server_user})) {
        logger ("Username for server is not present in configuration " .
                " file ... please fix!");
        logger("Failed backing up clients.");
        last;
    }

    # test if ssh-connection to server works ok
    my $sshteststr = "ssh -o PasswordAuthentication=no " .
        "$config->{server_user}" . "@" .
            "$config->{server_address} 'echo -n 1'";
    if (`$sshteststr` ne "1") {
        logger ("ssh-connection to server $key failed...");
        logger ("Failed backing up clients.");
        last;
    }

    # test that rdiff-backup has the same version as here
    $sshteststr = "ssh $config->{server_user}" . "@" .
            "$config->{server_address} 'rdiff-backup -V'";
    if (`$sshteststr` ne `rdiff-backup -V`) {
        logger ("rdiff-backup does not have the same version on " .
                "this computer and the backup server... please fix!");
        logger ("Failed backing up clients.");
        last;
    }

    # the server-part of the configuration shall be ok, so
    # build server-part of execstr and continue
    $execstr_serverpart =
        "$config->{server_user}\@$config->{server_address}::";
}

# check if destination directory on backup server is represented in
# configuration file -> return, else add it :)
```

```
    if (!exists ($config->{server_destdir})) {
        logger ("Destination␣directory␣on␣the␣server␣is␣not␣specified␣" .
                "in␣the␣configuration...␣please␣fix!");
        logger ("Failed␣backing␣up␣clients.");
        last;
    }
    $execstr_serverpart .= "$config->{server_destdir}/$key";


    # start with the client-handling
    logger ("Starting␣backup␣of␣client␣$key");

    # check if client not is of type "local" ->
    # - check if necessary configuration options are present
    # - check if ssh-connection is ok
    # - check if rdiff-backup is the same version as here
    if (exists ($config->{client}->{$key}->{type}) and
        $config->{client}->{$key}->{type} ne "local") {

        # check that address is provided
        #FIXME - check that the address is valid
        if (!exists ($config->{client}->{$key}->{address})) {
            logger ("Address␣for␣client␣$key␣is␣not␣present␣in␣" .
                    "configuration...␣please␣fix!");
            logger ("Backup␣of␣client␣$key␣failed.");
            next;
        }

        # check that username is provided
        if (!exists ($config->{client}->{$key}->{user})) {
            logger ("Username␣for␣client␣$key␣is␣not␣present␣in␣" .
                    "configuration...␣please␣fix!");
            logger ("Backup␣of␣client␣$key␣failed");
            next;
        }

        # test that ssh connection to the client works ok
        my $sshteststr = "ssh␣-o␣PasswordAuthentication=no␣" .
            "$config->{client}->{$key}->{user}" . "@" .
                "$config->{client}->{$key}->{address}␣'echo␣-n␣1'";
        if (`$sshteststr` ne "1") {
            logger ("ssh-connection␣to␣$key␣failed...");
            logger ("Failed␣backing␣up␣client␣$key.");
            next;
        }

        # test that rdiff-backup on the client is the same version as here
        $sshteststr = "ssh␣$config->{client}->{$key}->{user}" . "@" .
            "$config->{client}->{$key}->{address}␣'rdiff-backup␣-V'";
        if (`$sshteststr` ne `rdiff-backup -V`) {
            logger ("rdiff-backup does not have the same version on this " .
                    "computer and the client $key... please fix!");
            logger ("Failed backing up client $key.");
            next;
        }

        # client configuration shall be ok, so we continue:
        # add client address in the client-part of execstr
        $execstr_clientpart .=
            "$config->{client}->{$key}->{user}\@" .
            "$config->{client}->{$key}->{address}::";
    }

    # add the common part of the client execstring
    # (specify '/' as the location)
    $execstr_clientpart .= "/";

    # build execute string
    my $execstr = "rdiff-backup --print-statistics ";

    # include clients locations if exists
    if (!exists ($config->{client}->{$key}->{location})) {
        logger ("Locations for client $key is not present in " .
                "configuration... please fix!");
        logger ("No files from client $key will be backed up.");
        next;
    } elsif (ref ($config->{client}->{$key}->{location}) eq "ARRAY") {
        # there are more than one location => location is an array
        for my $loc (@{$config->{client}->{$key}->{location}}) {
            $execstr .= "--include $loc ";
        }
    } else {
        # there is only one location => location is a string
        my $loc = $config->{client}->{$key}->{location};
        $execstr .= "--include $loc ";
    }

    # exclude everything else
    $execstr .= "--exclude '/*' ";

    # include client-part and server-part
    $execstr .= "$execstr_clientpart $execstr_serverpart";

    # before backing up, remove old backups
    my $client_keep;
    if (($client_keep = $config->{client}->{$key}->{keep}) and
        ($client_keep gt 0)) {
        my $removestr = "rdiff-backup --force --remove-older-than ";
        $removestr .= "$client_keep" . "D␣";
        my $server_type = $config->{server_type};
        my $server_destdir = $config->{server_destdir};
        my $server_address = $config->{server_address};
        my $server_user = $config->{server_user};

        if ($server_type eq "extern") {
            $removestr .= "$server_user" . "@" . "$server_address" . "::";
        }
        if (grep (/\/$/, $server_destdir)) {
            $removestr .= "$server_destdir";
        } else {
            $removestr .= "$server_destdir" . "/";
        }
        $removestr .= "$key";

        # remove backups older than $client_keep
        #FIXME - check if there are backups there...
        logger ("Trying␣to␣remove␣backups␣older␣than␣$client_keep␣days:");
        my $output .= `$removestr 2>&1`;
        logger ("$output");

        # 0 mean success -> invert it
        my $retval = ! $?;

        # log
        if ($retval) {
            logger ("Removing␣backups␣older␣than␣$client_keep␣days␣sccesseded!");
        } else {
            logger ("Failed␣removing␣backups␣older␣than␣$client_keep.");
        }
    }

    # run rdiff-backup for client $key
    my $output .= `$execstr 2>&1`;
    logger ("\n$output");

    # 0 mean success -> invert it
    my $retval = ! $?;

    # log
    if ($retval) {
        logger ("Successfully␣finished␣backing␣up␣client␣$key");
    } else {
        logger ("Failed␣backing␣up␣client␣$key");
    }
}

logger ("Finished␣slbackup.");
```

```
close (LOG);


sub logger {
     my ($comment) = @_;
     my $now = strftime "%b %d %H:%M:%S", localtime;
     printflush LOG ("$now - $comment\n");
}



1;
```

Appendix F

Source code: webmin-slbackup

F.1 WebminSLBackup.pm

```
#
# WebminSLBackup.pm -- Webmin module for the Skolelinux Backup service
# By Morten Werner Olsen <werner@skolelinux.no>
#
# Copyright (c) 2003, Skolelinux.
# See COPYING in the source distribution for license details.
#
# $Id: WebminSLBackup.pm,v 1.2 2003/11/27 17:13:47 werner Exp $
#

# This is a subclass of CGI::Application
package WebminSLBackup;
use base 'CGI::Application';
use Expect;
use Net::Ping;
use POSIX qw(strftime);
use SLBackup;

$Expect::Exp_Internal = 0;
$Expect::Log_Stdout = 0;

use strict;

my $conffile = "/etc/slbackup/slbackup.conf";
my $logfile = "/var/log/slbackup/webmin-slbackup.log";

# unfortunately webmin's web-lib.pl uses a lot of global functions so because
# we use strict we have to predeclare them here.
use vars qw(%config %gconfig $module_name $module_config_directory $tb $cb
  $scriptname $remote_user $base_remote_user $current_theme $root_directory
  $module_root_directory %module_info %text);
```

```
# This function is the first one called when the script is run. Ideally this
# and cgiapp_postrun would be in a class which contained all the infrastructure
# that webmin modules using CGI::Application share in common.
# The inheritance tree would look like this:
#
#                                                   ,---> SLBackup.pm
# CGI::Application --> CGI::Application::Webmin -+---> Foo.pm
#                    (or some similiar name)  `---> Bar.pm
#                                                     (etc.)
sub cgiapp_init
{
  my ($self) = @_;

  # Import the webmin helper functions.
  do '/usr/share/webmin/web-lib.pl';

  # initalize webmin configuration.
  &init_config();

  # Convert all the webmin global variables into class parameters to make
  # it more OOP.
  map { $self->param($_, $config{$_}) } keys %config;
  map { $self->param($_, $config{$_}) } keys %gconfig;
  map { $self->param($_, $config{$_}) } keys %module_info;
  $self->param('module_name', $module_name);
  $self->param('module_config_directory', $module_config_directory);
  $self->param('tb', $tb);
  $self->param('cb', $cb);
  $self->param('scriptname', $scriptname);
  $self->param('remote_user', $remote_user);
  $self->param('base_remote_user', $base_remote_user);
  $self->param('current_theme', $current_theme);
  $self->param('root_directory', $root_directory);
  $self->param('module_root_directory', $module_root_directory);
}


#
# This function is called after each run mode (See CGI::Application docs.)
# Here we wrap the output of each template with the webmin header and footer
#
sub cgiapp_postrun
{
  # $output is a reference to the results of a run mode.
  my ($self, $output) = @_;

  # We want to send out the input unbuffered.
  local $| = 1;

  # We don't want CGI::Application to send the http headers.  The function
  # below will do that.
  $self->header_type('none');

  # print the webmin header.
  header($self->param('module_name'), '');

  print $$output;

  # print the webmin footer
  #footer();
  &footer("/", $text{'index_return'});

  # If we don't erase the contents of $output, the body of the page will be
  # displayed twice
  $$output = '';
}

#
# Here we set up the SLBackup class. Think of it as the constructor.
#
sub setup
{
  my ($self) = @_;

  # The initial run mode or if no run mode is specified.
  $self->start_mode('general');

  # Our run modes and the functions they map to.
  $self->run_modes(
    'general' => 'general',
    'backup' => 'backup',
    'backup_configure_client' => 'backup_configure_client',
    'backup_add_client' => 'backup_add_client',
    'backup_delete_client' => 'backup_delete_client',
    'backup_configure_server' => 'backup_configure_server',
    'restore' => 'restore',
    'restore_choose' => 'restore_choose',
    'restore_choose_snapshot' => 'restore_choose_snapshot',
    'restore_execute' => 'restore_execute',
    'maint' => 'maint',
    'maint_delete_older' => 'maint_delete_older',
    'maint_delete_confirm' => 'maint_delete_confirm',
    'sshkeys' => 'sshkeys',
    'sshkeys_add' => 'sshkeys_add',
    'sshkeys_create' => 'sshkeys_create',
    'sshkeys_delete' => 'sshkeys_delete',
    # AUTOLOAD is called if something other than the above run modes is asked
    # for. In this case we just fall back to general.
    AUTOLOAD => 'general',
  );

}

#
# This run mode ...
#
sub general
{
  my ($self) = @_;
  my $last_session;
  my $session_finished = 0;
  my $sshkeys_ok = 0;

  # fetch configuration data
  my $config = slbackup_readconfig ($conffile);

  # find out details about the last session
  if (open (LOGFILE, "/var/log/slbackup/slbackup.log")) {
      my @finished_sessions = grep (/Finished slbackup/, <LOGFILE>);
      close (LOGFILE);

      for my $session (@finished_sessions) {
          ($last_session) = split (/ - /, $session);
      }
      $session_finished = 1;
  }

  ## does the ssh keys work properly?
  my $sshkeys_localhost = 0;
  my $sshkeys_server = 0;
  my $sshkeys_client = 0;

  # localhost test
  my @sshprivstat = stat ("/root/.ssh/id_dsa");
  my @sshpubstat = stat ("/root/.ssh/id_dsa.pub");
  if (($sshprivstat[7] gt 0) and ($sshpubstat[7] gt 0)) {
      $sshkeys_localhost = 1;
  }

  my ($server_address, $server_destdir, $server_type, $server_user) =
       list_server_config();
  # server test
  if ($sshkeys_localhost) {
      my $execstr = "ssh -o PasswordAuthentication=no $server_user" . "@" .
          "$server_address 'echo 1'";
```

```
        if (($server_type eq "local") or (`$execstr` eq "1")) {
            $sshkeys_server = 1;
        }
    }

    # clients test
    my @clients = list_clients();

    if ($sshkeys_localhost and $sshkeys_server) {
        for my $key (keys %{$config->{client}}) {
            if ($config->{client}->{$key}->{type} eq "extern") {
                my $client_address = $config->{client}->{$key}->{address};
                my $client_user = $config->{client}->{$key}->{user};
                my $execstr = "ssh␣-o␣PasswordAuthentication=no␣$client_user" .
                    "@" . "$client_address␣'echo␣-n␣1'";
                if (`$execstr` eq "1") {
                    $sshkeys_client = 1;
                } else {
                    $sshkeys_client = 0;
                    last;
                }
            } else {
                $sshkeys_client = 1;
            }
        }
    }

    if ($sshkeys_client) {
        $sshkeys_ok = 1;
    }

    # find the clients in configuration
    @clients = list_clients_html();

    # See the HTML::Template man page to understand all this.
    my $template = $self->load_tmpl('general', die_on_bad_params => 0);

    # fill template with text-strings
    fill_in_template($template);

    # fill template with "local" variables
    $template->param(last_session => $last_session,
                     session_finished => $session_finished,
                     clients => \@clients, sshkeys_ok => $sshkeys_ok,);

    return $template->output;
}

#
```

```
# This run mode ...
#
sub backup
{
  my ($self) = @_;
  my $backup_enable = 1;
  my $backuptime_changed;
  my $client;
  my $client_deleted = 0;
  my $newtime = "";
  my $output = "";

  # Fetching query from webform
  my $q = $self->query();

  # fetch details from cron job
  open (CRONFILE, "/etc/cron.d/slbackup");
  my @cronline = grep (/slbackup-cron/, <CRONFILE>);
  close (CRONFILE);

  if (grep (/^\#/, $cronline[0])) {
      $cronline[0] =~ s/\#//g;
      $backup_enable = 0;
  }
  my ($min, $hour) = split(/ /, $cronline[0], 3);


  # Check if a new time was submitted
  if ($q->param("backuptime-submit")) {
      my $enable = $q->param("backup_enable");
      $newtime = $q->param("newtime");
      # make new cron-line
      ($hour, $min) = split(/:/, $newtime);
      my $newcron = "#␣cron␣job␣for␣Skolelinux␣Backup␣";
      $newcron .= "(every␣night␣at␣$hour:$min)\n";
      if ($enable =~ /[Nn]o/) {
          $newcron .= "#";
          $backup_enable = 0;
      } else {
          $backup_enable = 1;
      }
      $newcron .= "$min␣$hour␣*␣*␣*␣root␣";
      $newcron .= "if␣[␣-x␣/usr/share/slbackup/slbackup-cron␣" .
          "-a␣-r␣/etc/slbackup/slbackup.conf␣];␣then␣" .
          "/usr/share/slbackup/slbackup-cron␣;␣fi\n";

      # save this in cron job
      open(CRONFILE, ">/etc/cron.d/slbackup");
      print(CRONFILE "$newcron");
```

```
        close(CRONFILE);

        # set bool newtime_changed
        $backuptime_changed = 1;
    } else {
        if (length($min) le 1) {
            $min = "0$min";
        }
        if (length($hour) le 1) {
            $hour = "0$hour";
        }
        $newtime = "$hour:$min";

        # set bool newtime_changed
        $backuptime_changed = 0;
    }

    # Check if a client has been deleted
    if ($q->param("action") eq "delete_client") {
        # which client is to be deleted
        $client = $q->param("client");

        # fetch configuration
        my $config = slbackup_readconfig ($conffile);

        # delete client from config
        delete ($config->{client}->{$client});

        # write configuration
        slbackup_writeconfig($conffile, $config);

        # set bool client_deleted
        $client_deleted = 1;
    }

    # find the clients and servername in configuration
    my $config = slbackup_readconfig ($conffile);
    my $backupserver = $config->{server_address};
    my @clients = list_clients_html();

    # See the HTML::Template man page to understand all this.
    my $template = $self->load_tmpl('backup', die_on_bad_params => 0);

    # fill template with text-strings
    fill_in_template($template);

    $template->param(newtime => $newtime,
                     backuptime_changed => $backuptime_changed,
                     backupserver => $backupserver,
                    backup_enable => $backup_enable,
                    client => $client,
                    clients => \@clients,
                    client_deleted => $client_deleted,
                    configure => $text{"configure"},
                    delete => $text{"delete"},);
  return $output . $template->output;
}

#
# This run mode ...
#
sub backup_add_client
{
  my ($self) = @_;

  # Fetching query from webform
  my $q = $self->query();

  # find which client to add
  my $client = $q->param("clientname");

  # See the HTML::Template man page to understand all this.
  my $template = $self->load_tmpl('backup_add_client', \
                                  die_on_bad_params => 0);

  # fill template with text-strings
  fill_in_template($template);

  $template->param(client => $client,);
  return $template->output;
}

#
# This run mode ...
#
sub backup_configure_client
{
  my ($self) = @_;
  my $client;
  my $client_added = 0;
  my $client_added_success = 0;
  my $client_added_error = "";
  my $client_confd = 0;
  my $client_changed = 0;
  my $client_hostname;
  my $client_unreach = 0;
  my $client_unreach_err = "";
  my $client_type_extbool;
my $client_user;
my $client_keep;
my $delete_dir;
my $delete_dir_bool = 0;
my $delete_dir_deleted = 0;
my $output = "";

# Fetching query from webform
my $q = $self->query();

# fetch old configuration
my $config = slbackup_readconfig($conffile);

# is the user coming from backup_configure_client?
if ( $client = $q->param("client_confd")) {
    # fetch old configuration
    my @oldfiles = list_files ( $client );
    my $oldfiles_len = scalar (@oldfiles);
    my @newfiles;
    my $files_changed = 0;
    my $type_changed = 0;
    my $hostname_changed = 0;
    my $user_changed = 0;
    my $keep_changed = 0;

    ## update configuration
    # check if client type is updated
    if ($q->param("type") ne $config->{client}->{$client}->{type}) {
        $config->{client}->{$client}->{type} = $q->param("type");
        $type_changed = 1;
    }

    # check if client hostname is updated
    if ($q->param("hostname") ne $config->{client}->{$client}->{address}) {
        $config->{client}->{$client}->{address} = $q->param("hostname");
        $hostname_changed = 1;
    }

    # check if client username is updated
    if ($q->param("username") ne $config->{client}->{$client}->{user}) {
        $config->{client}->{$client}->{user} = $q->param("username");
        $user_changed = 1;
    }

    # check if days to keep is updated
    if ($q->param("keep") ne $config->{client}->{$client}->{keep}) {
        if ($q->param("keep") =~ /\d?/) {
            $config->{client}->{$client}->{keep} = $q->param("keep");
            $keep_changed = 1;
            }
        }

        # check for changed file
        for my $file ( @oldfiles ) {
            my $newfile = $q->param($file);
            push (@newfiles, $newfile);
            if ( $file ne $newfile) {
                $files_changed = 1;
            }
        }

        # check if new file(s) are added
        my $newfile0 = $q->param("newfile0");
        my $newfile1 = $q->param("newfile1");
        logger("newfile0: .$newfile0., newfile1: .$newfile1.");
        if ($newfile0 ne "") {
            push (@newfiles, $newfile0);
            $files_changed = 1;
        }
        if ($newfile1 ne "") {
            push (@newfiles, $newfile1);
            $files_changed = 1;
        }

        # save the new configuration if change has
        if ($type_changed or $hostname_changed or $user_changed or
            $files_changed or $keep_changed) {
            delete ($config->{client}->{$client}->{location});
            if (scalar (@newfiles) eq 1) {
                $config->{client}->{$client}->{location} = $newfiles[0];
            } else {
                @{$config->{client}->{$client}->{location}} = @newfiles;
            }
            slbackup_writeconfig ($conffile , $config);

            # set bool client_changed
            $client_changed = 1;
        }

        # set bool client_confd
        $client_confd = 1;
    } else {
        # find which client to configure
        $client = $q->param("client");
    }

    # check if some directories is to be deleted and delete it
    if ($delete_dir = $q->param("delete_dir")) {
        # delete file from configuration:
        # build a new list of files to backup
        my @newfiles;

        # check how many locations in config file
        if (ref ($config->{client}->{$client}->{location}) eq "") {
            # check that location actually is something
            if ($config->{client}->{$client}->{location} ne "") {
                if ($config->{client}->{$client}->{location} ne $delete_dir) {
                    push (@newfiles, $config->{client}->{$client}->{location});
                } else {
                    $delete_dir_deleted = 1;
                }
            }
        } elsif (ref ($config->{client}->{$client}->{location}) eq "ARRAY") {
            for my $file (@{$config->{client}->{$client}->{location}}) {
                if ( $file ne $delete_dir) {
                    push( @newfiles, $file );
                } else {
                    $delete_dir_deleted = 1;
                }
            }
        }

        # rewrite configuration
        # check if @newfiles is empty
        if (scalar (@newfiles) eq 0) {
            delete ($config->{client}->{$client}->{location});
        } else {
            @{$config->{client}->{$client}->{location}} = @newfiles;
        }
        slbackup_writeconfig ($conffile, $config);

        # set bool delete_dir_bool
        $delete_dir_bool = 1;
    }

    # is the client added?
    if ($q->param("addclient")) {
        # set bool client_added
        $client_added = 1;

        # fetch data from web form
        my $client = $q->param("client");
        my $client_type = $q->param("type");
        my $client_hostname = $q->param("hostname");
        my $client_username = $q->param("username");
        my $client_keep = $q->param("keep");
        my @files;
    for (my $i = 0; $i lt 5; $i++) {
        my $file = $q->param("newfile" . $i);
        if ( $file ne "") {
            push (@files, $file );
        }
    }

    # check that user has given us enough information
    if ( $client eq "") {
        # client name has to ...
        $client_added_error = $text{"added_error_name"};
    } elsif (( $client_type ne "local") and ($client_type ne "extern")) {
        # something really strange has happened
        # (not post from original form or something)
        $client_added_error = $text{"added_error_type"};
    } elsif (( $client_type eq "extern") and ($client_hostname eq "")) {
        # client hostname cannot be nothing
        $client_added_error = $text{"added_error_type_name"};
    # check that client does not already exist
    } elsif (! $config->{client}->{$client}) {
        # client does not exist -> adding
        # building hash
        my %newclient;
        $newclient{"type"} = $client_type;
        if ( $client_hostname ne "") {
            $newclient{"address"} = $client_hostname;
        }
        if ( $client_username eq "") {
            $newclient{"user"} = "root";
        } else {
            $newclient{"user"} = $client_username;
        }
        if ( $client_keep =~ /\d?/) {
            $newclient{"keep"} = $client_keep;
        } else {
            #FIXME test if this really is a number, not just set it to 185
            $newclient{"keep"} = 185;
        }

        if (scalar (@files) eq 1) {
            $newclient{"location"} = $files[0];
        } elsif (scalar (@files) gt 1) {
            $newclient{"location"} = \@files;
        }

        # store configuration
        $config->{client}->{$client} = \%newclient;
        slbackup_writeconfig($conffile, $config);

```

```
            # set client_added_success-bool
            $client_added_success = 1;
        } else {
            # client existed (?)
            $client_added_error = $text{"added_error_client_existed"};
        }
    } else {
        # set bool client_added
        $client_added = 0;
    }

    # fetch data about client from configuration file
    # check the hostname-validity if client is extern
    if ($config->{client}->{$client}->{address}) {
        $client_hostname = $config->{client}->{$client}->{address};
    } else {
        $client_hostname = "";
    }
    if ($config->{client}->{$client}->{type} eq "extern") {
        $client_type_extbool = 1;

        # check that the hostname/ip-address is valid
        $client_unreach_err = hostname_validate($client_hostname);

        if ( $client_unreach_err) {
            $client_unreach = 1;
        }
    } else {
        $client_type_extbool = 0;
    }

    $client_user = $config->{client}->{$client}->{user};
    $client_keep = $config->{client}->{$client}->{keep};
    my @files = list_files_html ($client);

    # See the HTML::Template man page to understand all this.
    my $template = $self->load_tmpl('backup_configure_client', \
                                    die_on_bad_params => 0);

    # fill template with text-strings
    fill_in_template($template);

    $template->param(client => $client, client_added => $client_added,
                     client_confd => $client_confd,
                     client_changed => $client_changed,
                     client_added => $client_added,
                     client_added_error => $client_added_error,
                     client_added_success => $client_added_success,
                     client_hostname => $client_hostname,
                    client_unreach => $client_unreach,
                    client_unreach_err => $client_unreach_err,
                    client_type_extbool => $client_type_extbool,
                    client_user => $client_user,
                    client_keep => $client_keep,
                    delete_dir => $delete_dir,
                    delete_dir_bool => $delete_dir_bool,
                    delete_dir_deleted => $delete_dir_deleted,
                    files  => \@files,);

  return $output . $template->output;
}


#
# This run mode ...
#
sub backup_delete_client
{
  my ($self) = @_;
  my $client;

  # Fetching query from webform
  my $q = $self->query();

  # find which client to delete public key on
  $client = $q->param("client");

  # See the HTML::Template man page to understand all this.
  my $template = $self->load_tmpl('backup_delete_client', \
                                  die_on_bad_params => 0);

  # fill template with text-strings
  fill_in_template($template);

  $template->param(client => $client,);
  return $template->output;
}

#
# This run mode ...
#
sub backup_configure_server
{
  my ($self) = @_;
  my $backupdir;
  my $serveraddr;
  my $servertype;
  my $serveruser;
my $serverconf_changed = 0;
my $localhost = 1;
my $output = "";

# Fetching query from webform
my $q = $self->query();

# fetch config from file
my $config = slbackup_readconfig ($conffile);

# Check if some information has changed
if ($q->param("serverconf")) {
    # get data from form
    $backupdir = $q->param("backupdir");
    $config->{server_destdir} = $backupdir;

    $servertype = $q->param("servertype");
    $config->{server_type} = $servertype;

    $serveruser = $q->param("serveruser");
    if ($serveruser eq "") {
        $serveruser = "root";
    }
    $config->{server_user} = $serveruser;

    $serveraddr = $q->param("serveraddr");
    $config->{server_address} = $serveraddr;

    # write to configuration file
    slbackup_writeconfig ($conffile, $config);

    $serverconf_changed = 1;
} else { #fetch from config-file
    $backupdir = $config->{server_destdir};
    $servertype = $config->{server_type};
    $serveruser = $config->{server_user};
    $serveraddr = $config->{server_address};
}

# set value of servertype (info to html-template)
if ($servertype eq "local") {
    $localhost = 1;
} else {
    $localhost = 0;
}

# See the HTML::Template man page to understand all this.
my $template = $self->load_tmpl('backup_configure_server', \
                                die_on_bad_params => 0);
```

```
  # fill template with text−strings
  fill_in_template($template);

  $template−>param(backupdir => $backupdir,
                   localhost => $localhost,
                   serveraddr => $serveraddr,
                   serveruser => $serveruser,
                   serverconf_changed => $serverconf_changed,);
  return $output . $template−>output;
}

#
# This run mode ...
#
sub restore
{
  my ($self) = @_;

  # fetch names on all computers that is possible to restore from
  my @clients = list_clients_html();

  # See the HTML::Template man page to understand all this.
  my $template = $self−>load_tmpl('restore', die_on_bad_params => 0);

  # fill template with text−strings
  fill_in_template($template);

  $template−>param(clients => \@clients, error => 0);
  return $template−>output;
}


#
# This run mode ...
#
sub restore_choose
{
  my ($self) = @_;

  # Fetching query from webform
  my $q = $self−>query();

  # find which client to restore to
  my $client = $q−>param("client");

  # See the HTML::Template man page to understand all this.
  my $template = $self−>load_tmpl('restore_choose', \
                                          die_on_bad_params => 0);
  # fill template with text−strings
  fill_in_template($template);

  $template−>param(client => $client, error => 0, error_msg => "",);
  return $template−>output;
}

#
# This run mode ...
#
sub restore_choose_snapshot
{
  my ($self) = @_;
  my $restoredir;
  my @restoredirs;

  # Fetching query from webform
  my $q = $self−>query();

  # find which client to restore to and what type of restore ( full or not)
  my $client = $q−>param("client");
  my $type = $q−>param("type");

  if ($type eq "full") {
      my @files = list_files($client);
      for (my $i = 0; $i < scalar (@files); $i++) {
          push (@restoredirs, { id => "dir$i", location => $files[$i]});
      }
  } else {
      # not full restore −> directory has been passed with a web−form
      $restoredir = $q−>param("dir0");
      @restoredirs = ( { id => 'dir0', location => $q−>param("dir0") } );
      $type = "not_full";
  }

  # find the available snapshots from the client
  my @snapshots = list_snapshots ($client, $type, $restoredir);

  # if there are no snapshots and type not equals full, give warning
  if (scalar (@snapshots) eq 0 and $q−>param("type") ne "full") {
      my $error_msg = $text{"error_no_snapshots_avail"};

      # See the HTML::Template man page to understand all this.
      my $template = $self−>load_tmpl('restore_choose', \
                                              die_on_bad_params => 0);

      # fill template with text−strings
      fill_in_template($template);
```

```
    $template->param(client => $client, error => 1,
                     error_msg => $error_msg);
    return $template->output;
  } elsif (scalar (@snapshots) eq 0 and $q->param("type") eq "full") {
    # if there are no snapshots and type equals full
    my $error_msg = $text{"error_no_snapshots_avail_client"};

    # fetch names on all computers that is possible to restore from
    my @clients = list_clients_html();

    # See the HTML::Template man page to understand all this.
    my $template = $self->load_tmpl('restore', die_on_bad_params => 0);

    # fill template with text-strings
    fill_in_template($template);

    $template->param(clients => \@clients, error => 1,
                     error_msg => $error_msg,);
    return $template->output;
  } else {
    # See the HTML::Template man page to understand all this.
    my $template = $self->load_tmpl('restore_choose_snapshot', \
                                    die_on_bad_params => 0);

    # fill template with text-strings
    fill_in_template($template);

    $template->param(backuptype => $type, client => $client,
                     restoredirs => \@restoredirs,
                     snapshots => \@snapshots,);
    return $template->output;
  }
}

#
# This run mode ...
#
sub restore_execute
{
  my ($self) = @_;

  # Fetching query from webform
  my $q = $self->query();

  # fetch config from configuration file
  my $config = slbackup_readconfig ($conffile);

  # find which client to restore to
```

```
my $client = $q->param("client");

# find what kind of backup (full / not full)
my $backuptype = $q->param("backuptype");

# find which snapshot to restore from
my $temp = $q->param("snapshot");
my ($timestamp, $filetype) = split (/;/, $q->param("snapshot"));
my $snapshot = ssepoch_to_iso ($timestamp);

# find which directory to restore to
my $targetdir = $q->param("targetdir");

# find which directories/ files to restore
my $i = 0;
my $tempentry;
my @restoredirs;
while ($tempentry = $q->param("dir" . $i)) {
    push @restoredirs, { id => "dir" . $i, location => $tempentry };
    $i++;
}

# build the execute string
my $execstr = "rdiff-backup --print-statistics --restore-as-of $timestamp ";

my $overwrite = $q->param("overwrite");
if ($overwrite eq "on") {
    $execstr .= "--force ";
}

if ($config->{server_type} eq "extern") {
    my $user = $config->{server_user};
    if ($user eq "") {
        $user = "root";
    }
    my $address = $config->{server_address};
    $execstr .= "$user" . "@" . "$address" . "::"
}
my $destdir = $config->{server_destdir};
$execstr .= "$destdir" . "/" . "$client";
my $restoredir = $restoredirs[0]->{location};

# restore file to the computer webmin runs on (until later)...
#if ($config->{client}->{$client}->{type} eq "extern") {
#   my $user = $config->{client}->{$client}->{user};
#   if ($user eq "") {
#     $user = "root";
#   }
#   my $address = $config->{client}->{$client}->{address};
```

```
  #    $execstr .= "$user" . "@" . "$address" . "::";
  #}
  if ($backuptype eq "full") {
      $execstr .= "/␣$targetdir";
  } else {
      $execstr .= "$restoredir␣$targetdir";
  }

  # execute restore
  my $output;
  my $target;
  my $retval;
  if ( $targetdir eq "/") {
      # we wont let any user do this...
      $output = $text{"restore_not_allow"};
      $retval = 0;
  } else {
      $output = `$execstr 2>&1`;
      $retval = $?;
      if ( $retval ne 0) {
          $retval = 0;
      } else {
          $retval = 1;
      }
  }

  # See the HTML::Template man page to understand all this.
  my $template = $self->load_tmpl('restore_execute', \
                                  die_on_bad_params => 0);

  # fill template with text-strings
   fill_in_template($template);

  $template->param(client => $client, restoredirs => \@restoredirs,
                   restorelog => $output, snapshot => $snapshot,
                   success => $retval, targetdir => $targetdir,);
  return $template->output;
}

#
# This run mode ...
#
sub maint
{
  my ($self) = @_;
  my $client;
  my $delete_confirm;
  my $execstr;
  my $output;
my $retval;

# Fetching query from webform
my $q = $self->query();

# check if this is the return from deleted snapshots
if ($q->param("client")) {
    # set the bool delete_confirm
    $delete_confirm = 1;

    # fetch data about the client and snapshot to delete older than
    $client = $q->param("client");
    my $snapshot = $q->param("snapshot");

    # fetch data about backup server
    my $config = slbackup_readconfig ($conffile);
    my ($server_address, $server_destdir, $server_type, $server_user) =
        list_server_config();

    # delete data
    $execstr = "rdiff-backup␣--force␣--remove-older-than␣$snapshot␣";
    if ($server_type eq "extern") {
        $execstr .= "$server_user" . "@" . "$server_address" . "::";
    }
    if (grep (/\/$/, $server_destdir)) {
        $execstr .= "$server_destdir";
    } else {
        $execstr .= "$server_destdir" . "/";
    }
    $execstr .= "$client";

    # execute restore
    $output = `$execstr 2>&1`;
    $retval = $?;
    if ($retval ne 0) {
        $retval = 0;
    } else {
        $retval = 1;
    }

    $delete_confirm = 1;
} else {
    # set the bool delete_confirm to 0 and the string client to ""
    $delete_confirm = 0;
    $client = "";
}

# find all available computers
my @clients = list_clients();
```

```
  # See the HTML::Template man page to understand all this.
  my $template = $self->load_tmpl('maint', die_on_bad_params => 0);

  # fill template with text-strings
  fill_in_template($template);

  $template->param(client => $client, clients => \@clients,
                   delete_confirm => $delete_confirm,
                   delete_log => $output, success => $retval,);
  return $template->output;
}

#
# This run mode ...
#
sub maint_delete_older
{
  my ($self) = @_;
  my $deleting_none = 0;

  # Fetching query from webform
  my $q = $self->query();

  # find which client to delete snapshots for
  my $client = $q->param("client");

  # find all available snapshots
  my @snapshots_all = list_snapshots($client, "full", "");
  my @snapshots;
  for (my $i = (scalar (@snapshots_all) - 2); $i ge 0; $i--) {
      push (@snapshots, $snapshots_all[$i]);
  }

  # check number of snapshots available
  # (only one, none is going to be deleted)
  if (scalar (@snapshots) le 0) {
      $deleting_none = 1;
  }

  # See the HTML::Template man page to understand all this.
  my $template = $self->load_tmpl('maint_delete_older', \
                                  die_on_bad_params => 0);

  # fill template with text-strings
  fill_in_template($template);

  $template->param(client => $client, deleting_none => $deleting_none,
                   snapshots => \@snapshots,);
  return $template->output;
}

#
# This run mode ...
#
sub maint_delete_confirm
{
  my ($self) = @_;

  # Fetching query from webform
  my $q = $self->query();

  # find which client to delete snapshots for
  my $client = $q->param("client");

  # find the latest snapshot to keep
  my ($snapshot, $stype) = split (/;/, $q->param("snapshot"));

  # find which snapshots will be deleted
  my @snapshots = list_snapshots($client, "full", "");
  my @snapshots_deleted;
  for my $snap (@snapshots) {
      #FIXME - something wrong here...
      ($snap, my $temp) = split (/;/, $snap->{id});
      if ($snap lt $snapshot) {
          my $time = ssepoch_to_iso ($snap);
          push (@snapshots_deleted, {date => $time});
      }
  }

  # See the HTML::Template man page to understand all this.
  my $template = $self->load_tmpl('maint_delete_confirm', \
                                   die_on_bad_params => 0);

  # fill template with text-strings
   fill _in_template($template);

  $template->param(client => $client, snapshot => $snapshot,
                   snapshots_deleted => \@snapshots_deleted,);
  return $template->output;
}

#
# This run mode ...
#
sub sshkeys
{
  my ($self) = @_;
my $client;
my $client_publickey_added = 0;
my $client_publickey_deleted = 0;
my $client_keys_recreated = 0;
my $error_msg;
my $retval;

# Fetching query from webform
my $q = $self->query();

## find information about the localhost
my $config = slbackup_readconfig ($conffile);

# check if some action was done and returned to this page
# we would not want to delete sshkeys until slbackup is using it's own
# SSH private/public-keys
#if ($q->param("action") eq "delete") {
#    # delete clients public-key
#    $client  = $q->param("client");
#    $client _publickey_deleted = 1;
#    my $type = $q->param("type");
#    my $address;
#    my $username;
#    if ($type eq "server") {
#        $address = $config->{server_address};
#        $username = $config->{server_user};
#    } else {
#        $address = $config->{client}->{$client}->{address};
#        $username = $config->{client}->{$client}->{user};
#    }
#    $error_msg = ssh_session ("delsshpubkey", $address, $username, "");

if ($q->param("action") eq "add") {
    # add public key to the client
    $client  = $q->param("client");
    $client _publickey_added = 1;
    my $password = $q->param("password");
    my $address;
    my $username;
    if ($q->param("type") eq "backupserver") {
        $address = $config->{server_address};
        $username = $config->{server_user};
    } else {
        $address = $config->{client}->{$client}->{address};
        $username = $config->{client}->{$client}->{user};
    }
    $error_msg = ssh_session ("addsshpubkey", $address, $username,
                              $password);
} elsif ($q->param("action") eq "create") {
        # create private/public key pair on localhost
        my $cmd = 'ssh-keygen -t dsa -N "" -f ~/.ssh/id_dsa';
        my $output = `$cmd 2>&1`;
        $retval = $?;
        if ($retval ne 0) {
            $retval = 0;
        } else {
            $retval = 1;
        }

        if (! $retval) {
            $error_msg = "<br>\n" . $output;
        }
    }

    # hostname
    my $localhost = `hostname`;
    for my $tmpclient (keys %{$config->{client}}) {
        if ($config->{client}->{$tmpclient}->{type} eq "local") {
            $localhost = "$tmpclient";
        }
    }

    # status of the host
    my @sshprivstat = stat ("/root/.ssh/id_dsa");
    my @sshpubstat = stat ("/root/.ssh/id_dsa.pub");
    my $localhost_status;
    my $localhost_choises;
    # only checking that the size is greater then zero
    if (($sshprivstat[7] gt 0) and ($sshpubstat[7] gt 0)) {
        $localhost_status = $text{"sshkeys_pair_avail"};
        # we don't want to be able to recreate ssh key pair after all
        # it can break other things depending on the ssh key pair located
        # in /root/.ssh/id_dsa[.pub]
        $localhost_choises = "&nbsp;";
    } else {
        $localhost_status = $text{"sshkeys_pair_unavail"};
        $localhost_choises = "&nbsp;";
        $localhost_choises =
            "(<a href=\"?rm=sshkeys_create\">" .
            $text{create} . "</a>)";
    }

    ## find information about the backup server
    my ($server_address, $server_destdir, $server_type, $server_user) =
        list_server_config();

    my $server_status;
    my $server_choises = "&nbsp";
```

```
  # check if server _is_ localhost
  if ($server_type eq "local") {
      $server_status = $text{"sshkeys_not_needed"};
  } elsif ($localhost_status eq $text{"sshkeys_pair_unavail"}) {
      $server_status = $text{"sshkeys_pub_unavail"} . " $localhost";
      $server_choises = "&nbsp;";
  } else {
      # if not, test ssh connection with rdiff-backup -V
      my $execstr = "ssh $server_user" . "@" .
          "$server_address rdiff-backup -V";
      if (`$execstr` eq `rdiff-backup -V`) {
          $server_status = $text{"sshkeys_pub_work"};
          $server_choises = "&nbsp;";
# we do not want to delete keys until slbackup uses it's own SSH keys
#         $server_choises =
#             "(<a href=\"?rm=sshkeys_delete&type=server\">" .
#             $text{"delete"} . "</a>)";
      } else {
          $server_status = $text{"sshkeys_pub_notwork"};
          $server_choises = "&nbsp;";
          $server_choises =
              "(<a href=\"?rm=sshkeys_add&type=server\">" .
              $text{"add"} . "</a>)";
      }
  }

  ## find the computers available in the configuration
  # hostname, status, choises
  my @clients = list_clients();
  my @clients_var;

  for my $bclient (keys %{$config->{client}}) {
      my $client_status;
      my $client_choises;
      if ($config->{client}->{$bclient}->{type} eq "extern") {
          # if keypair is not available on localhost, none of the clients
          # public keys could work
          if ($localhost_status eq $text{"sshkeys_pair_unavail"}) {
              # set same status and choises on all clients
              $client_status = $text{"sshkeys_pub_unavail"} . " $localhost";
              $client_choises = "&nbsp;";
          } else {
              my $client_address = $config->{client}->{$bclient}->{address};
              my $client_user = $config->{client}->{$bclient}->{user};
              my $execstr = "ssh $client_user" . "@" .
                  "$client_address rdiff-backup -V";
              if (`$execstr` eq `rdiff-backup -V`) {
                  $client_status = $text{"sshkeys_pub_work"};
# we do not want to delete keys until slbackup uses it's own keys
#                       $client_choises =
#                           "(<a␣href=\"?rm=sshkeys_delete&client=$bclient\">" .
#                               $text{"delete"} . "</a>)";
                    } else {
                        $client_status = $text{"sshkeys_pub_notwork"};
                        $client_choises =
                            "(<a␣href=\"?rm=sshkeys_add&client=$bclient\">" .
                                $text{"add"} . "</a>)";
                    }
                }
                push (@clients_var, { client => $bclient,
                                      client_status => $client_status,
                                      client_choises => $client_choises });
        }
  }

  # See the HTML::Template man page to understand all this.
  my $template = $self->load_tmpl('sshkeys', die_on_bad_params => 0);

  # fill template with text-strings
  fill_in_template($template);

  # check if something returned an error
  my $error = 0;
  if ($error_msg ne "") {
      $error = 1;
  }

  $template->param(localhost => $localhost,
                   localhost_status => $localhost_status,
                   localhost_choises => $localhost_choises,
                   backupserver => $server_address,
                   backupserver_status => $server_status,
                   backupserver_choises => $server_choises,
                   clients => \@clients_var,
                   client => $client,
                   client_publickey_added => $client_publickey_added,
                   client_publickey_deleted => $client_publickey_deleted,
                   client_keys_recreated => $client_keys_recreated,
                   error => $error, error_msg => $error_msg,);
  return $template->output;
}


#
# This run mode ...
#
sub sshkeys_add
{
  my ($self) = @_;

  # Fetching query from webform
  my $q = $self->query();


  # check if the computer to add a pubkey to is the server
  # and set address and user-variables
  my $type = $q->param("type");
  my $client;
  my $username;
  my $address;
  my $config = slbackup_readconfig();
  if ($type eq "server") {
      $client  = $config->{server_address};
      $username = $config->{server_user};
      $address = $client;
      $type = "backupserver";
  } else {
      $client  = $q->param("client");
      $username = $config->{client}->{$client}->{user};
      $address = $config->{client}->{$client}->{address};
      $type = "client";
  }

  # See the HTML::Template man page to understand all this.
  my $template = $self->load_tmpl('sshkeys_add', \
                                  die_on_bad_params => 0);

  # fill  template with text-strings
   fill _in_template($template);

  $template->param(client => $client, username => $username,
                  address => $address, type => $type,);
  return $template->output;
}


#
# This run mode ...
#
sub sshkeys_create
{
  my ($self) = @_;

  # See the HTML::Template man page to understand all this.
  my $template = $self->load_tmpl('sshkeys_create', \
                                  die_on_bad_params => 0);
  # fill template with text-strings
  fill_in_template($template);

  return $template->output;
}


#
# This run mode ...
#
sub sshkeys_delete
{
  my ($self) = @_;

  # Fetching query from webform
  my $q = $self->query();

  # fetch configuration
  my $config = slbackup_readconfig ($conffile);

  # find which client to delete public key on
  my $client = $q->param("client");
  my $type = $q->param("type");

  # find server address if type eq server
  if ($type eq "server") {
      $client = $config->{server_address};
  }

  # See the HTML::Template man page to understand all this.
  my $template = $self->load_tmpl('sshkeys_delete', \
                                  die_on_bad_params => 0);

  # fill template with text-strings
  fill_in_template($template);

  $template->param(client => $client, type => $type,);
  return $template->output;
}

#
# This run mode displays the information for one alternative group.
#
#sub list
#{
# my ($self) = @_;
# my $alternative = $self->query->param('name');
#
# my $file = $self->parse_alternative(\$alternative);
# my $template = $self->load_tmpl('list', die_on_bad_params => 0);
# $template->param(name => $file->{name}, links => $file->{links},
#   scriptname => $self->param('scriptname'), tb => $self->param('tb'),
#   cb => $self->param('cb'),);
#
# return $template->output;
#}


# function to fill in a template
# (Thanks to Andreas Schuldei - webmin-ldap-skolelinux)
sub fill_in_template {
    my ($template) = @_;

    my @parameter_names = $template->param();

    for my $name (@parameter_names) {

        if( $text{$name} ){
            $template->param ($name => $text{$name} );
        }
    }
    $template->param( JavaScriptStatus => generate_status() )
        if($template->query(name => "JavaScriptStatus"));
    $template->param( cb              => $cb              )
        if($template->query(name => "cb" ));
    $template->param( tb              => $tb              )
        if($template->query(name => "tb" ));
    $template->param( CgiName         => "entermanyusers.cgi")
        if($template->query(name => "CgiName" ));
}


# function to fill in a template
# (Thanks to Andreas Schuldei - webmin-ldap-skolelinux)
sub generate_status {
    my $status = "\"";
    $status .= $ENV{'ANONYMOUS_USER'} ? "Anonymous user" :$remote_user;
    $status .= $ENV{'SSL_USER'} ? " (SSL certified)" :
        $ENV{'LOCAL_USER'} ? " (Local user)" : "";
    $status .= " logged into " .$text{'programname'};
    $status .= " " . get_webmin_version();
    $status .= " on " . get_system_hostname();
    $status .= "(" . $ENV{'HTTP_USER_AGENT'};
    $status .= ")\"";

    return $status;
}
```

```
# function that lists the clients in the configuration
# and applies some language data used in html−templates
# (do not use data from this functions to other things than html−templates)
sub list_clients_html {
    # fetch configuration
    my $config = slbackup_readconfig($conffile);

    my @clients;
    for my $client (keys %{$config−>{client}}) {
        push (@clients, {client => $client,
                         configure => $text{"configure"},
                         delete => $text{"delete"},
                         choose_filesdir => $text{"choose_filesdir"},
                         restore_full => $text{"restore_full"}});
    }
    return @clients;
}


# function that lists the clients in the configuration
sub list_clients {
    # fetch configuration
    my $config = slbackup_readconfig($conffile);

    my @clients;
    for my $client (keys %{$config−>{client}}) {
        push (@clients, {client => $client});
    }
    return @clients;
}


# function that lists all files / directories to back up on a clients
sub list_files {
    my ($client) = @_;
    # fetch configuration
    my $config = slbackup_readconfig($conffile);

    # check if location is scalar or array, and put value(s) in @files
    my @files;
    if (ref ($config−>{client}−>{$client}−>{location}) eq "") {
        # check that location actually is something
        if ($config−>{client}−>{$client}−>{location} ne "") {
            push (@files, $config−>{client}−>{$client}−>{location});
        }
    } elsif (ref ($config−>{client}−>{$client}−>{location}) eq "ARRAY") {
        foreach my $file (@{$config−>{client}−>{$client}−>{location}}) {
            push (@files, $file );
        }
    }
    return @files;
}


# function that lists all files / directories to back up on a clients
# and applies some language data used in html-templates
# (do not use data from this functions to other things than html-templates)
sub list_files_html {
    my ($client) = @_;;
    # fetch configuration
    my $config = slbackup_readconfig($conffile);

    # check if location is scalar or array, and put value(s) in @files
    my @files;
    if (ref ($config->{client}->{$client}->{location}) eq "") {
        # check that location actually is something
        if ($config->{client}->{$client}->{location} ne "") {
            push (@files, { file => $config->{client}->{$client}->{location},
                            delete => $text{"delete"},
                            remove => $text{"remove"},
                            client => $client});
        }
    } elsif (ref ($config->{client}->{$client}->{location}) eq "ARRAY") {
        foreach my $file (@{$config->{client}->{$client}->{location}}) {
            push (@files, { file => $file,
                            delete => $text{"delete"},
                            remove => $text{"remove"},
                            client => $client});
        }
    }
    return @files;
}


#
sub list_snapshots {
    my ($client, $backup_type, $restoredir) = @_;

    # fetch client configuration
    my $config = slbackup_readconfig ($conffile);
    my $client_user = $config->{client}->{$client}->{user};
    my $server_address = $config->{server_address};
    my $server_destdir = $config->{server_destdir};
    my $server_type = $config->{server_type};
    my $server_user = $config->{server_user};

    # build execute string
    my $execstr = "rdiff-backup --parsable-output --list-increments ";

    if ($server_type eq "extern") {
        $execstr .= "$server_user" . "@" . "$server_address" . "::";
    }

    if ($backup_type eq "full" ) {
        $execstr .= "$server_destdir" . "/" . "$client";
    } else {
        $execstr .= "$server_destdir" . "/" . "$client" . "$restoredir";
    }

    my $output .= `$execstr 2>&1 | grep -v missing`;
    my $retval = $?;
    if ($retval ne 0) {
        $retval = 0;
    } else {
        $retval = 1;
    }

    my @snapshots;

    if ($retval eq 1 and ! ($output =~ /error/i)) {
        my @lines = split (/\n/, $output);
        my $increments = scalar (@lines);

        for (my $i = $increments - 1; $i >= 0; $i--) {
            $lines[$i] =~ /^(\d*)\s(\w*)$/;
            my ($time_se, $type) = ($1, $2);
            my $id = "$time_se;$type";
            my $time = ssepoch_to_iso($time_se);
            push (@snapshots, { id => $id, time => $time });
        }

    } else { # error handling
        #FIXME, please :)
    }

    return @snapshots;
}


# return configuration data about server
sub list_server_config {
    # fetch config from configuration files
    my $config = slbackup_readconfig ($conffile);

    # get server configuration
    my $address = $config->{server_address};
    my $destdir = $config->{server_destdir};
    my $type = $config->{server_type};
    my $user = $config->{server_user};

    return ($address, $destdir, $type, $user);
}


# reformat "seconds␣since␣epoch␣to␣"DDMMYYYY HH:MM"-format
sub ssepoch_to_iso {
    my ($timestamp) = @_;

    return `date -d "1970-01-01␣$timestamp␣sec␣UTC" --iso-8601=minutes`;
}


#
sub ssh_session {
    my ($action, $address, $username, $password) = @_;
    my $cmd;

    # check action
    if ($action eq "addsshpubkey" ) {
        # fetch ssh public key from ~/.ssh/id_dsa.pub
        open (DSAPUB, "/root/.ssh/id_dsa.pub") ||
            return "Could␣not␣open␣public␣key␣on␣localhost";
        my $sshpubkey = readline (\*DSAPUB);
        close (DSAPUB);

        # provide command to the SSH session
        $cmd = "echo␣-n␣'$sshpubkey'␣»␣~$username/.ssh/authorized_keys";
    } elsif ($action eq "delsshpubkey") {
        # fetch ssh public key from ~/.ssh/id_dsa.pub
        open (DSAPUB, "/root/.ssh/id_dsa.pub") ||
            return "Could␣not␣open␣public␣key␣on␣localhost";
        my $sshpubkey = readline (\*DSAPUB);
        close (DSAPUB);

        # delete ssh public key from ~/.ssh/authorized_keys
        $cmd = "grep␣-v␣'$sshpubkey'␣~$username/.ssh/authorized_keys␣>␣" .
            "~$username/.ssh/authorized_keys";
    }

    # Check that host is up'n'running
    #FIXME

    # connect to host and execute command
    (my $connect = Expect->spawn("ssh␣-o␣StrictHostKeyChecking=no␣" .
                                 "$username\@$address")) ||
        return "Problems with SSH connection";

    if ($action ne "delsshpubkey") {
        if ($connect->expect(10, "assword: ")) {
            print $connect "$password\r\n";
        } else {
            return "Host did not ask for password";
        }
    }

    # recognise prompt...
    #FIXME - need a better solution here... (regexp and which fetches
    # more possible prompts)
    if ($connect->expect(10, '# ') || $connect->expect(10, '$ ')) {
        print $connect "$cmd\r";
    } else {
        return "No prompt given by host";
    }

    if ($connect->expect(10, '# ') || $connect->expect(10, '$ ')) {
        print $connect "exit\r";
    } else {
        return "Prompt is not returned after executing command";
    }

    if (! ($connect->expect(10, "closed"))) {
        return "Connection was not properly closed";
    }

    $connect->hard_close();

    return "";
}


sub hostname_validate {
    my ($hostip, $trash) = @_;
    my $hostip_valid = 0;

    my $hosttest = '^([a-z]([-a-z\d]*[a-z\d])?[\.]?)+?$';

    my $iptest = '^(([01]?[0-9][0-9]?|2[0-4][0-9]|25[0-5])\.){3}([01]?[0-9][0-9]?|2[0-4][0-9]|25[0-5])$';

    # test if the hostname or ip-address is in a valid format
    if (! ($hostip =~ /$hosttest/i or $hostip =~ /$iptest/)) {
        return "<b>" . $text{"error_str"} . ":</b> " .
            $text{"hostname_not_valid"};
    }
    # test if the hostname or ip-address is reachable
    my $p = Net::Ping->new("icmp", 5);
    if ($p->ping($hostip)) {
        logger("hostname: $hostip, ping successfull\n");
        $p->close();
        return undef;
    } else {
        $p->close();
        logger("hostname: .$hostip., ping NOT successfull\n");
        return "<b>" . $text{'warn_str'} . ":</b> " .
            $text{'host_not_reachable'};
    }
}

sub logger {
    open (LOG, ">>$logfile");
    my ($comment) = @_;
    my $now = strftime "%b %d %H:%M:%S", localtime;
    printflush LOG ("$now - $comment\n");
    close (LOG);
}


1;

__END__
```