StoreBackup 3.5

http://storebackup.org

April 20, 2014

1 Super Quick Start

StoreBackup is a very space efficient disk-to-disk backup suite for GNU/Linux and other unixoid systems. Additional details and help are provided in later sections of this document.
In these brief quick start steps we make certain simplifying assumptions. If you are OK with that, then proceed as follows:

1. Download the source from http://download.savannah.gnu.org/releases/storebackup/
2. Unpack it (using `tar -jxvf)` into `/opt` (it will make the folder `/opt/storeBackup`.)[1]
3. Create symbolic links. In a terminal, run these 2 commands (the 2nd line ends with: space,dot):

```
# cd /usr/local/bin
# ln -s /opt/storeBackup/bin/* .
```

4. Run your first backup with this command (substituting your actual username in the command):[2]

```
storeBackup.pl --sourceDir /home/your_username --backupDir /tmp/my_master_backup
```

   This may take a while. Open a second shell and see what happens in the backup directory. You have now backed up your home directory to `/tmp/my_master_backup`.

For more details, please continue reading; especially see installation, section 2 and storeBackup.pl, section 6.2. If the above steps gave you any challenges, don't worry. This document will cover everything from storeBackup installation to NFS server settings in much more detail.

See storeBackup's Top Features on the next page

[1]You need root permissions to install storeBackup at `/opt/storeBackup` and to follow the next steps. You can also unpack and run storeBackup from a place where you do not have root permissions. If you start storeBackup without root permissions, it will run with the permissions you have at that moment.

[2]If you install storeBackup from the Debian or Ubuntu repository via the packet manager, all programs will come without the "`.pl`" at the end, so instead of `storeBackup.pl` you have to call `storeBackup`.

1.1 storeBackup's Top Features

- Restore easily – even without storeBackup! The most important aspect of a backup tool is easy restoring from a transparent (native) storage format.
- Copies / compresses files to another disk and generates backups with time stamps while saving space by recognizing files with identical *contents* (even if renamed or copied) and hardlinking them (so each backup is totally complete, independent and autonomous)
- Detects identical files in different, independent backups (eg. of different computers)
- Splits big image files (from e.g., TrueCrypt, mbox, Xen, KVM, VMware, etc.) or complete devices into small pieces and saves only differences to existing backups, thereby saving space and time
- Sophisticated possibilities for including and excluding files and directories.
- Support of sparse files.
- Supports isolated incremental backups (e.g., when travelling with a laptop) and later integration in master backup.
- Supports time shifted replication of backups to additional other disks / locations, even for complex schemes.
- Supports checking of backups via md5 sums to eg. recognize bit rot of the hard disk.
- Enables checking of (old) files in the source directory with md5 sums in the backup to eg. recognize bit rot of the hard disk.
- Fast backups even over slow or high latency network connections.
- Contains an entire suite of backup-related tools.

1.2 Why should you back up your files?

Simple answer – for two reasons.

1. To restore the last state after a hardware or software crash.
2. To recover old versions of a file or folder if it was deleted / destroyed unnoticed (e.g., by a software bug) or by mistake.

The most important backup is the one you did not make!

New releases are announced at http://freshmeat.net/projects/storebackup. Please subscribe to get recent information.

If you have any hints, comments or questions, send an email to hjclaes at web.de

StoreBackup is licensed under the terms of the GPL-v3 or any later version.

Heinz-Josef Claes with support of contributors, April 20, 2014

Contents

2 Installation

You should read The Idea Behind It (Abstract), see section 5.1, as well as Supported Platforms and Tips, see section 6.1, to see if storeBackup fits to your needs.

Installation is straightforward:
Download the archive from http://download.savannah.gnu.org/releases/storebackup/ and go to a directory where you want to unpack it.
If you are not sure where to unpack it, allow me to suggest `/opt`. (You need root permissions to write at `/opt`.) If you chose `/opt`, then in the example below *path* is equal to `/opt`.

```
$ cd path
$ tar jxvf pathToArchive/storeBackup-3.3.tar.bz2
```

This will create a directory "storeBackup" where you will find four sub-directories: `bin`, `lib`, `man`, and `doc`. If you do not want to type the whole path every time to start storeBackup.pl (or any of the programs in `bin`), there are two easy choices.

One choice is to set your `$PATH` variable:

```
$ cd storeBackup/bin
$ export PATH=‘pwd‘:$PATH
```

(The quotes around the `pwd` must be back quotes, ascii code 96; some pdf readers will render them as them as normal quotes in this document!)
Also set `$PATH` in your .bashrc or whatever shell you are using.

The second choice is to make symbolic links from a place where `$PATH` is already set. For instance, if your `$PATH` also points to `/usr/local/bin` (and you have write permissions), you can do:

```
# cd /usr/local/bin
# ln -s path/storeBackup/bin/* .
```

Don't use hard links for that. StoreBackup will not find it's libraries if you do so.

If you want to have access to the man pages via the man command, you should set `MANPATH`:

```
$ cd storeBackup/man
$ export MANPATH=‘pwd‘:$MANPATH
```

Naturally, you have to change the path after `cd` depending on your location in the filesystem.
Also, you should set `MANPATH` in your .bashrc or whatever shell you are using.

Please have a look into the file `README.1ST` which is located in the `doc` folder.

3 Getting Started

Let's make your first backup.
Imagine you want to backup your home directory to `/tmp/my_master_backup`. (If your home directory is too big to do this, choose a small directory inside your home directory.)
Go into your home directory and type:

```
$ mkdir /tmp/my_master_backup
$ cd
$ storeBackup.pl --sourceDir . --backupDir /tmp/my_master_backup
```

Alternatively, you can call storeBackup.pl with an additional option:

```
$ storeBackup.pl --sourceDir . --backupDir /tmp/my_backup_destination --checkCompr
```

The Option `--checkCompr` will tell storeBackup.pl to check the contents of each file for compressibility. Without this option, storeBackup.pl will use static lists of file types to make a decision – which is much less precise (in chapter 7.4.1, "How to define if a file should be compressed" you will get a more detailed view on this).
If storeBackup.pl is not in your path, you will get an error message from the shell and need to set `$PATH` or type the full path to storeBackup.pl.[3]
This can take some time depending on how much data is in your home directory, as storeBackup.pl will compress your files. It will use all cores of your system for this. Because of these compressions, the first backup is very slow.
If you want to reduce the load from storeBackup (especially in the first run), you might use option `--compress 'nice bzip2'` and maybe call `storeBackup.pl` with `ionice -t -c 3 storeBackup.pl`.
After the backup is finished, create a new file, copy a file and rename a file and or directory and start a second run:

```
$ cd
$ storeBackup.pl --sourceDir . --backupDir /tmp/my_master_backup
```

or

[3]see installation, section 2 if you do not know what this means

```
$ storeBackup.pl --sourceDir . --backupDir /tmp/my_backup_destination --checkCompr
```

You will notice that the backup is much faster now.

Hint: You can use the short form of the options, too:

```
$ storeBackup.pl --sourceDir . --backupDir /tmp/my_backup_destination --checkCompr
```

This is identical with:

```
$ storeBackup.pl -s . -b /tmp/my_backup_destination -C
```

Go to `/tmp/my_master_backup`. We call this location your "master backup" for consistency with the rest of the documentation. There you will find a directory named `default`. This is called a series because this directory will hold a series of backups for your computer. You can change the default series name from "default" to the name of your computer. This is easily accomplished with the storeBackup configuration file (explained later).
Inside of the `default` directory you will see two sub directories whose names reflect the date and time of the two backups you just completed. Go into these directories (use two shells, one for each) and look at the files with the command:
`$ ls -li`
Option "`i`" tells ls to show the inode number, which you can see in the very left column. You will notice that files with the same content (especially the ones you copied, renamed, moved and the ones in renamed directories) refer to the same inode – so the file exists only once on your disk thanks to storeBackup's efficient technology.

If you used storeBackup in Versions prior to 2.0 and simply made a backup with

```
storeBackup.pl -s sourceDir -t targetDir    # !!! old syntax !!!
```

and now want to continue making backups with version 2.0 or above, use

```
storeBackup.pl -s sourceDir --backupDir targetDir -S .
```

Where the parameters of `sourceDir` and `targetDir` are the same in both versions.

4 What's new?

For a list of all changes see chapter 12, "ChangeLog". This chapter will just list a summary of new features and bug fixes.

4.1 Version 3.5

This version corrects some bugs (see ChangeLog for details). The following new features are added (beside minor ones):

Including / excluding files There are several methods to include or exclude files. You can read how they interact or complement in the new section 7.2 "Selecting Directories / Files to Backup".

Option `stayInFileSystem` By using this option only file systems specified by options `sourceDir` and the symlinks of `followLinks` are saved.

Wildcards for linking series and for replication To configure `otherBackupSeries`, you can use wildcards now. More on that topic can be found at the beginning of section 7.8.7 "Using Wildcards for Replication".

Tagging of (un)finished backups Prior to this version (3.5), unfinished backups where tagged with the existence of file `.md5CheckSums.notFinished` during the creation of the backup. This file was deleted after the backup and a filesystem sync was finished.
During a run of a backup (as well as a never finished one) it is not used for external linking from any storeBackup program running on other backups.
This behavior of `storeBackup.pl` now is changed to the following steps:

1. Create the new backup directory and run the backup part of `storeBackup.pl`.
2. Run a filesystem sync.
3. Create the tag file `.storeBackupLinks/backup.Finished`.
4. Perform deletion of old backups (and maybe unfinished backups if option `deleteNotFinishedDirs` is set).

If you upgrade to this version and run your backups (etc.), you do not have to follow any special instructions. *But if you want to go back to a previous version (like 3.4.3) after running backups with version 3.5 (or higher), you have to make sure, that all unfinished backups created by version 3.5 (or later) have been deleted!*

Option `--force` for setting up isolated mode When setting up isolated mode, you now can force the usage of the last existing backup, even if it is not completed with `storeBackupUpdateBackup.pl`.

Log file options for storeBackupMount.pl Added the log file options known from `storeBackup.pl` to `storeBackupMount.pl`. See section 6.11, `storeBackupMount.pl` for details.

Changed option name in multiTail.pl The option `noOldFiles` was changed to `noOfOldFiles` ("number of old files") to make it consistent among all programs.

Rasberry Pi You can find now some explanations about Raspberry Pi in the FAQ 8.

4.2 Version 3.4.3

This version corrects some bugs. See the ChangeLog for details about this. The following new features are added (beside minor ones):

multitail Name of `multitail.pl` is now changed to `multiTail.pl` to avoid conflicts with the open source program `multitail`.[4] You have to change calls of `multitail.pl` to `multiTail.pl` or maybe `multiTail`.

multiTail.pl now supports colored output via the options `--color` and `--grep`. New option `--print` shows the options set.

Deduplication sometimes missed identical blocks when using "blocked files" (corrected).

linkToDirs.pl Added option `--printDepth` and time frame to `--progressReport`.

Ubuntu (maybe other distributions) does not set `$PWD` when starting a program with `sudo`. Now, `storeBackup.pl` does not depend any more on `$PWD`.

4.3 Version 3.4.2

This version corrects some bugs. See ChangeLog for details about this. The following new features are added (beside minor ones):

Sparse files When restoring blocked files from the backup with `storeBackupRecover.pl` you now can select the creation of a sparse file with the option `--createSparseFiles` (or the shortcut `-s`). See chapter 6.4, "storeBackupRecover.pl". Additionally, `linkToDirs.pl` also has the same options to support sparse files, see chapter 6.16, "linkToDirs.pl" for further information.

Time scheduled progress report The option `--progressReport` (`-P`) of `storeBackup.pl` now allows additionally a time frame (eg. 30 seconds) after which a log messages is printed.

4.4 Version 3.4.1

This version corrects some bugs. See ChangeLog for details about this. The following new features are added (beside minor ones):

saveRAM bug The rule functions `MARK_DIR` and `MARK_DIR_REC` did not work together with the option `saveRAM`.

Documentation Added chapter 8, "Internals" which describes how storeBackup components fit together. Knowing this makes it easier to react in unplanned situations like interrupted backups or replications.

[4]Debian and Debian dependent distributions remove the `.pl` from the names of the storeBackup programs.

4.5 Version 3.4

This version corrects some bugs. See ChangeLog for details about this. The following new features are added (beside minor ones):

Storing of system-incompatible file systems A backup eg. on NTFS or via sshfs cannot store “special files”. Now, `storeBackup.pl` can use options `archiveTypes` and `specialTypeArchiver` to save these files in individual cpio or tar archives.

Special treatment of directories You can use file system flags set outside of storeBackup to exclude defined directories (optionally with its subdirectories) from backup or to compress files. See example 5 in chapter 7.4, “Defining Rules”.

Simple recreation of isolated mode A configuration file generated for isolated mode can now be used again for setting up the identical isolated mode again.

Readability The pdf version of this document (which is included in each download) now uses microtype for better readability, see
http://mirrors.ibiblio.org/CTAN/macros/latex/contrib/microtype/microtype.pdf.

4.6 Version 3.3.1

This is mainly a bug fix release. See ChangeLog for details about this. Nevertheless, some new features are added (beside minor ones):

Mounting `storeBackupMount.pl` is rewritten, so you have to adjust the options. You can now call other programs than `storeBackup.pl`, too. See 6.11, `storeBackupMount.pl` for details.

Easy compression rule `storeBackup.pl` now supports an additional new option `--checkCompr` (`-C`) on the command line. When using this option, the contents of each file bigger than 1k is checked if compression should be useful. See storeBackup.pl, section 6.2 for details.

4.7 Version 3.3

Compression: Prior to this version, you could define a list of file suffixes (extensions) that would never be compressed. You could also set a minimal size of files so that files below that size would not be compressed. If you do not change anything about this in your configuration file, the behavior stays the same. However, now you can also define a list of file suffixes *always to compress* in addition to the prior options which let you define a list of file suffixes which should *not be compressed*, plus a minimal file size. For files not fitting in these categories `storeBackup.pl` will make an estimation if compression might reduce the file size. Naturally, you can also define your own rules with this functionality. See section 7.4.1, “How to define if a file should be compressed” for detailed information.

Isolated mode: If you are traveling, for example, with a laptop and have no connection to your backup, you now can store delta backups (relative to your “big” local backup) on a small media device (e.g., a memory stick) and *integrate* these backups later into your central one. See section 7.7, “isolated mode / offline backups” for detailed information.

Replication of backups: Allows you to set up the replication of your backup to other disks / locations. This can be used to make continuous copies of your backup. See section 7.8, “replication of backups” for detailed information.

linkToDirs.pl: Allows you to copy / hard link backups to other ones. It is like running `cp -a` but hard linking all identical files to selectable directories. It can also be used to support the replication of backups in special situations. If you have small backups you want to copy onto another disk, this might be the right tool for you also. See section 6.16, `linkToDirs.pl` for detailed information.

storeBackupCheckSource.pl: This tool is intended to find files in the source that might have changed over the time without the user’s interaction or knowledge, for example by bit rot. See section 6.13, `storeBackupCheckSource.pl` for detailed information.

5 The Idea Behind It

5.1 Abstract

StoreBackup is a disk-to-disk backup tool for GNU/Linux. It also runs on other Unix-like machines. You can directly browse through the backed-up files (locally or via NFS, Samba, SSH[5] or almost any other network file system). This gives the users the ability to restore files easily and quickly. The user only has to copy (and optionally uncompress) the files to restore them. There is also a tool for easy restoring (sub-) trees for the administrator. Every single backup from a specific time can be deleted without affecting the other existing backups.
StoreBackup recognizes files by their content rather than just by their name or location. It detects if files have been copied, renamed or moved. If the file is identical, but differs by name or location, storeBackup has an efficient way (hardlinks) to include that file in the current backup without copying it again. When a user reorganizes his photo or music collection, most backup software must transfer all those files over the network and store them again in the backup location, wasting time and space. StoreBackup will simply hardlink to the identical content that is already stored in the backup location, saving a lot of time and space.
StoreBackup can split big image files (e.g., from virtual machines) in little pieces and needs only the space for changes in these splits. Restoring these parts to the full image is also easily possible with simple tools: `cat` or possibly `bzcat` (or whatever you used for compression). Naturally, storeBackup delivers a tool to restore everything easier. You can also split devices or partitions (like `/dev/sdb1`) in the same way.
StoreBackup offers itself to the general user who does not necessarily own a tape backup but a second hard drive or another computer. It also offers itself to the users in the professional environment for extremely fast and comfortable access to their backups, also to save on the costs of tapes as well as administrative expenses.
StoreBackup is a command line tool. You can start it via cron automatically. Normally, you would not want to have a graphical surface on a server and most important: If your machine crashed, you probably do not have a running gui.
Storage on hard drives, memory sticks or similar devices offers itself as an alternative or additional resource to data backup on tapes. StoreBackup performs well, saves storage capacity, and increases administrative flexibility:

- Directories, including their tree structure, may be copied to another location (e.g. `/home` $\longrightarrow$ `/var/bkup/2003.12.13_02.04.26`). Permissions to the files remain, enabling users to access the backup directly. The most important aspect of a backup tool is easy and safe restoring.
- The content of each file being backed up is compared with the existing backup to make sure there is only one backup for each file. That means files with the same content exist physically only once in the backup.
- Identical files are hard linked and appear in the backup in the same locations as in the original.
- You can exclude files from the backup by excluding whole directories or by specifying rules depending on regular expressions, file size, groups, users and other criteria.
- Backed up files will be compressed, unless they are marked 'exclude'. Compression may also be excluded entirely or may be the result of a file based analysis of storeBackup itself.
- Image files or mass storage devices, where only parts change from backup to backup can be evaluated for differences. In the backup, you will only need the space for the changed blocks (which can be compressed).
- Backup series, generated independently (e.g. from different machines) may refer through hard links to shared files. Full or partial backups may be executed with this method, always under the condition that files with the same content may exist only once in the backup.
- The final result of running storeBackup is always a full backup. This can be automatically deleted with easy or high sophisticated deletion rules.
- StoreBackup supports a lot of other options. They are described in this document.

[5]see FAQ4 for details about making a backup via SSH

5.2 Another Backup Tool? – The roots of storeBackup

Possibly, there are thousands of backup programs. So, why another one? The reason arose from my activities as a consultant. The entire week I was moving around and I had no way to secure my data during the week at home. All I had was a 250MB ZIP drive connected to the parallel port of my laptop. The backup on the ZIP drive did not give me a lot of storage space and I had to live with a low bandwidth (about 200KB/s) and high latency. In contradiction to that I wanted fast, simple access to my data - I did not like the usual options of full, differential and incremental backups (e.g. with tar or dump): On one hand, it is usually too cumbersome to retrieve one of the versions, on the other hand it is not possible to delete an old backup at will, as this has to be planned carefully at the generation of the backup.
It was my goal to be able to backup quickly during my work and find my files quickly and without hassle. So, at the end of 1999 the first version of storeBackup was created. It was, however, not suitable for large environments. It was not performing well enough, did not scale sufficiently and was not able do deal with nasty file names (e.g. '\n' in a name).
Based on that experience with the first version, I wrote a new one which was published a little bit less than a year later under the GPL. In the meantime the number of users had grown - from home user applications, securing of (mail) directories at ISPs, small and medium sized companies or hospitals as well as universities and for general archiving.

5.3 What would be an ideal Backup Tool?

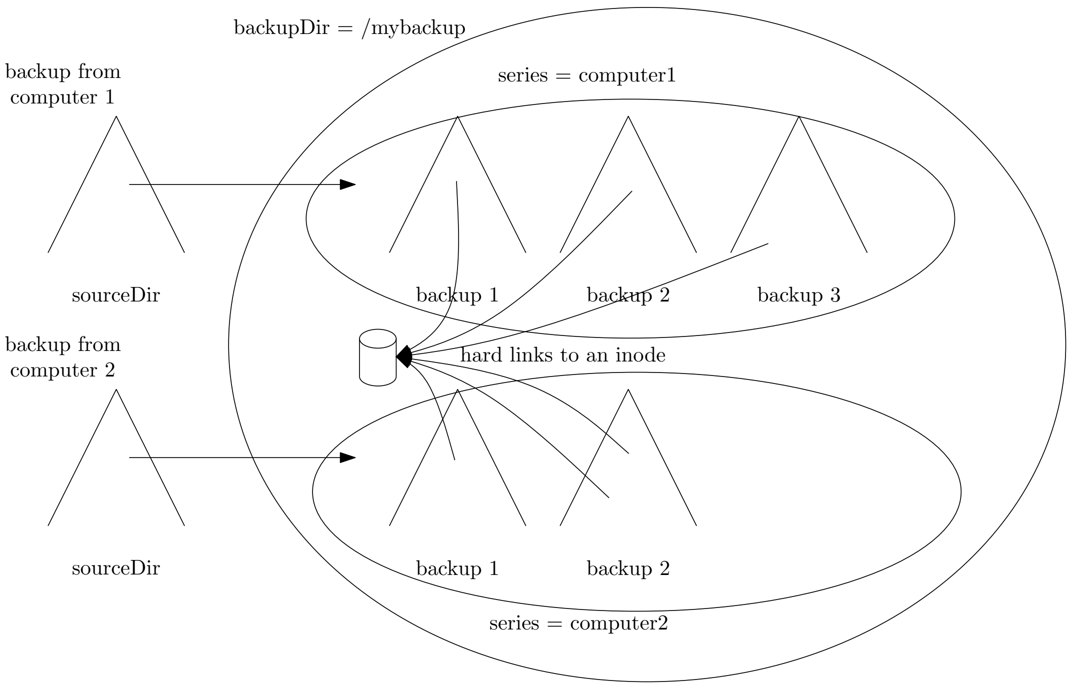
The most important aspect of a backup is that you are not only able to restore but to do this easily.
The following reflects backups of files, not databases.
The ideal backup tool would create every day a complete copy of the entire data system (including the applicable access rights) on another data system with minimal effort for the administrator and maximal comfort for the user. The computer and hard disk systems to make this possible should be in a distant, secure building, of course. With the help of a file system browser the user could search and access the data and copy data directly back. The backup would be usable directly and restoring possible without problems or special learnings. Dealing with backups would become something *normal* - since the route over the administration would in general be unnecessary.
The process described here has a “small” disadvantage: it needs a lot of hard drive space and it is quite slow because each time the total amount of data needs to be copied.

5.4 How storeBackup works

StoreBackup tries to accomplish the “ideal backup” and to solve the two problems: storage space and performance.
You can run it on a machine to store your backups to a local drive (or temporarily connected drive) or to a network mount (e.g., NFS, sshfs, cifs). This is the most efficient way. But you can also read the data from a machine which offers its data e.g., via cifs or NFS.

5.4.1 Illustration

storeBackup is a disk-to-disk backup tool for GNU/Linux based on hard links. Have a look at the picture:



Imagine you have two computers you want to backup onto one file server via NFS[6]. (Even if you only want to backup on an external disk (e.g., usb or e-sata), you should also read the following to understand the basic parameters.)
The default values of all storeBackup options are designed in a way that you can use storeBackup with only two options: --sourceDir and --backupDir. (The shortforms of these two options are -s and -b.) Everything else works with acceptable default values. The way you can define soureDir is very flexible[7]
First, you have to define a "backupDir" directory, where all your backups will reside. This is called the master backup repository. Let's say, this is `/my_master_backup`. To separate your different backups from multiple computers, you will make a separate directory for each; in the illustration, it is `computer1` and `computer2`.
In storeBackup terminology, `computer1` is a series and `computer2` is a series. The term "series" is used because e.g. `computer1` will contain a set of sequential backups (i.e., a series) that belong to computer1. These are shown as backup 1, backup 2 and backup 3 in the illustration.
You will identify these directories each as a "series" in storeBackup's configuration file. The full path to these directories is `/my_master_backup/computer1` and `/my_master_backup/computer2`. In each "series" directory you will generate a *series of backups* for the sourceDir on the computer associated with that series. You will see that it is all organized very logically and the organization is natural and easy to follow.
storeBackup will find all files that already exist in your backup repository ("my_master_backup") with the same content (either within a single backup or across multiple backups, or even across multiple backups for multiple computers!).
storeBackup will hard link all these files to only one inode (see section 7.11). Changing names or paths of the files does not present a problem, because storeBackup can tell if files are the same based on the content. In the picture above, you see that in backup 3 of computer 1 the location of the file has moved (perhaps also renamed), or in backup 2 of computer 2 the file has been copied. StoreBackup will recognize this and always link to the same inode.
storeBackup supports multiple series of backups (e.g., daily or weekly backups) from different sources (e.g., different servers or workstations). As mentioned, the default name for a series is `default`. However, if you plan to back up multiple computers, each series should be configured with a name that describes the computer (e.g., the source) using the --series option (short form -S) on the command line or simply "series" in the config file. See storeBackup.pl, section 6.2.
The next few sections will go into further details about the details of how storeBackup works.

[6]See section 7.10 how to configure NFS properly – you must have write access to the directory where you want to save your backup. If you are root, you should have root permissions on this mounted directory.
[7]If you want to backup more than one top-level directory tree at a time, you should have a look at option `followLinks` in section 6.2.

5.4.2 Reducing Disk Space

Saving files as a whole

The first measure to decrease the necessary hard drive storage space would be the compression of data – if that makes sense. storeBackup allows the use of any compression algorithm as an external program. The default is `bzip2`.
Looking at the stored data closely, it is apparent that from backup to backup relatively few files change – which is the reason for incremental backups. We also find that many files with the same content may be found in a backup because users copy files or a version administration program (like cvs) is active. In addition, files or directory structures are re-named by users, in incremental backups they are again (unnecessarily) secured. The solution to this is to check the backup for files with the same content (possibly compressed) and to refer to those. Within storeBackup, a hard link is used for referencing. With this trick of adding hard links, which were already created in existing backup files, each file is present in each backup although it exists physically on the hard drive only once. Copying and renaming of files or directories takes only the storage space of the hard links – nearly nothing.
Most likely not only one computer needs to be secured but a number of them. They often have a high proportion of identical files, especially with directories like `/etc`, `/usr` or `/home`. Obviously, there should be only one copy of identical files stored on the backup drive. To mount all directories from the backup server and to backup all computers in one sweep would be the most simple solution. This way duplicate files get detected and hard linked. However, this procedure has the disadvantage that all machines to be secured have to be available for the backup time. That procedure can in many cases not be feasible, for example, if notebooks shall be backed up using storeBackup. Specifically with notebooks we can find a high overlap rate of files since users create local copies. In such cases or if servers are backed up independently from one another, and the available hard drive space shall be utilized optimally through hard links, storeBackup is able to hard link files in independent backups (meaning: independent from each other, possibly from different machines).

Splitting files into parts: blocked files

The method of compressing and hard linking files works pretty well for "normal" files like office, configuration, program code and all other type of small files.
It more or less fails for big image files where only parts are changed. Such a file with e.g., 3 GB has only a few megabytes of changes, but the method described above would copy or compress the whole 3 GB into the backup, which is neither space nor time efficient. To solve this problem, storeBackup can handle such files in a special way.
In the configuration file you can specify which one should be handled as "blocked files". For these blocked files, a directory instead of a plain file is created in the backup. (The name of the directory is identical to the original file name.) The affected file from the source is not stored as a whole in the backup – instead it is stored as (small) numbered blocks in the created directory. These blocks can be compressed.
In the next backup (after something has changed in the original file,) storeBackup checks which of these blocks have changed and only copies / compresses that blocks. For the now missing *unchanged* blocks a hard link is generated to the fitting blocks in the old backup(s). This md5 sum based comparison is also done with other blocked files, so if you duplicate a VM for different use, storeBackup will find the identical blocks. It will also find identical blocks within one blocked file. This may happen when unused areas in an image are blanked or massively when saving sparse files.
As a result the needed space is reduced dramatically (compared with copying / compressing the whole file) and it is still possible to restore the contents of the original file without a running storeBackup which is the philosophy of storebackup (restoring is the most important part of a backup) and might be useful in e.g., 10 years. (Who knows what's happening then!?)

Deleting Backups

For the deletion of files storeBackup offers a set of options. It is a great advantage for deletion when each backup is a full backup, as those may be deleted indiscriminately. Unlike traditional backups, there is no need to consider if an incremental backup is depending on previous ones.
The options permit the deletion or saving of backups on specific workdays, first or last existing backup of the week/month or year. It can be assured that a set of a minimum number of backups remains. This is especially useful if backups are not generated on a regular basis. It is possible to keep the last backups of

a laptop until the end of a four week vacation even though the period to keep it is set to three weeks. Furthermore it is possible to define the maximal number of backups. There are more options to resolve the existence of conflicts between contradictory rules (by using common sense).

5.4.3 Performance

The procedure described above assumes that an existing backup is being checked for identical files prior to a new backup of a file. This applies to files in the previous backup as well as to the newly created one. Of course it does not make much sense to directly compare every file to be backed up with the previous backup. So, the md5 sums of the previous backup are being compared with the md5 sum of the file to be backed up with the utilization of the hash table.
Computing the md5 sum is fast, but in case of a large amount of data it is still not fast enough. For this reason storeBackup checks initially if the file was not altered since the last backup (path + file name, ctime, mtime and size are the same). If that is the case, the md5 sum of the last backup is being adopted and the hard link set. If the initial check shows a difference, the md5 sum is being computed and a check takes place to see if another file with the same md5 sum exists. (The comparison with a number of backup series uses an expanded but similarly efficient process). For this approach only a few md5 sums need to be calculated for a backup. If you want to tune storeBackup, especially if you save via NFS, there are two things you can do:

- tune NFS (see section 7.10)
- use the lateLinks option of storeBackup, and possibly delete your old backups independent from the backup process.
  Using storeBackup with lateLinks is like using an asynchronous client / server application or to be more precisely like using multiple batches on (normally) multiple machines:
  - Checking the source directory to know what has changed and to be compressed and save the changed data to the backup directory on the backup server. The missing directories and hard links in the backup are stored in a protocol file.
  - Take this information and restore a “normal” fully linked backup.
  - Delete old backups depending on the rules for the deletion.

The follwing performance measurements only show the direct backup time (without calling storeBackupUpdateBackup.pl[8]). They have been done with a beta version of storeBackup 2.0.
Some background information to the following numbers: The backup was run on an Athlon X2, 2.3 GHz, 4 GB RAM. The NFS server was an Athlon XP, 1.84 GHz, 1.5 GB RAM. The network was running with 100 MBit/s, storeBackup was used with standard parameters. The units of the measurements are in hours:minutes:seconds or minutes:seconds. The size of sourceDir was 12GB, the size of the backup done with storeBackup was 9.2 GB. The backups were done with 4769 directories and 38499 files. StoreBackup.pl linked 5038 files internally which means these were duplicates. The source for the data were my files and the “Desktop” from my Windows XP Laptop, i.e. “real” data.
The first table shows the time for copying the data to the nfs server with standard programs. The nfs server is mounted with option `async`[9], which is a performance optimization and not the standard configuration.

command	duration	size of backup
cp -a	28:46	12 GB
tar jcf	01:58:20	9.4 GB
tar cf	21:06	12 GB

All is like it was to expect: `tar` with compression is much slower than the other ones; and `cp` is slower than `tar`, because it has to create lots of files. There is one astonishing number: The size of the backup file of `tar jcf` ist 9.4 GB, while the resulting size of the backup with `storeBackup.pl` is only 9.2 GB. We see the reason for this in the internal linked 5038 files – the duplicates are stored only once with storeBackup.

We do not see the effect of comparing the contents in this benchmark again, but it makes a lot of differences

[8]This is only necessary if you use storeBackup.pl with option lateLinks. The necessary time for running storeBackupUpdateBackup.pl can be seen in the next section 5.4.4.
[9]see configuring nfs, section 7.10

in performance and especially used disk space. If the time stamp of a file is changed, then traditional backup software will store this file in an incremental backup – storeBackup will only create a hard link.

Now let's run `storeBackup.pl` on the same contents. The nfs server is still mounted with option `async`. There are no changes in the source directory between the first to the second or third backup.

storeBackup	1.19, Standard		2.0, Standard		2.0, lateLinks		mount with `async`
1. backup	49:51	100%	49:20	99%	31:14	63%	
2. backup	02:45	100%	02:25	88%	00:42	25%	file system read cache empty
3. backup	01:51	100%	01:54	100%	00:26	23%	file system read cache filled

We can see the following:

- The first run of `storeBackup.pl` is faster than `tar jcf` (tar with compression.) It is easy to understand why: storeBackup.pl uses both cores of the machine, while the compression with tar uses only one. But if you look a little bit deeper to the number, you see that `storeBackup.pl` needs less than half the time (42%) of tar with compression. It naturally additionally calculates *all* md5 sums and has to perform the overhead of creating thousands of files (look at the difference between `cp` and tar cf above). The effect of reducing the time for copying more than 50% comes from two effects: `storeBackup.pl` does not compress all files (depending on their suffix, e.g., `.bz2` files are not compressed again) and it recognizes files with the same content and sets just a hard link (also the reason for 9.2 instead of 9.4 GB).

- The second backup was done with a new mount of the source directory, so the read cache for it was not filled. You can see some improvement between version 1.19 and 2.0 because of better parallelization of reading the data in storeBackup itself.
  You see no difference in the third run between version 1.19 and 2.0, because reading the source directory entries is now in the file system cache, which means that the blocking factor is now the speed of the nfs server – and that's the same in both runs.

- With option lateLinks, you can see an improvement by a factor of 4. The time you see depends massively on the time needed for reading the source directory (plus reading the information from the previous backup, which is always the same).

Now let's do the same with an nfs mount without "tricks" like configuring `async`:

command	duration	size of backup
cp -a	37:51	12 GB
tar jcf	02:02:01	9.4 GB
tar cf	25:05	12 GB

storeBackup	1.19, Standard		2.0, Standard		2.0, lateLinks		mount with `sync`
1. backup	53:35	100%	49:20	100%	38:53	63%	
2. backup	05:36	100%	05:24	96%	00:43	13%	file system read cache empty
3. backup	05:10	100%	04:54	95%	00:27	9%	file system read cache filled

We can see the following:

- Everything is more or less slower, because of higher latency due to the synchronous communication with the nfs server. If only one file is written (like with `tar),` the difference to the backups with `async` is smaller, if many files are written, it's bigger.

- We see that the difference between `sync` and `async` using lateLinks is very small. The reason is simple: Only a few files are written over nfs, so the latency only has a small impact on the overall time for the backup. This results in the fact that the backup with lateLinks and a very fast source directory (cache) is now *10 times* faster.

- Because the latency is not important for making a backup, I mounted this file server over a vpn[10] over the Internet. This means very high latency and a bandwidth of about 20KByte/s from the nfs server and 50KByte/s to the nfs server (seen on a network monitoring tool). With same same boundary conditions as before (mounted with `async`, source directory file system in cache, no changes) I got a speed up with lateLinks (compared with non-lateLinks backup) by a *factor of 70*.
  So if your changed or new files are not too big compared with the available bandwidth, you can also

[10]The vpn software was openvpn, the connection was tunneled trough several firewalls.

use storeBackup (with lateLinks) for making a backup over a vpn on high latency lines.[11] Naturally you should not choose option lateCompress in such a case. Another advantage with lateLinks in such cases is, that parallelization works much better, because reading unchanged data in the source directory nearly needs no action on the NFS mount.

Conclusion: If you mount with nfs, you can make it really fast using option lateLinks. See section 7.6 for how to configure it.

Using "blocked files" also improves performance a lot because only a small percentage of an image file has to be copied or compressed. See the description about using blocked files (see section 7.5) for the influence of this option to performance and space needed.

5.4.4 Example of a Run

Here you can see the statistical output of a big backup I ran on my laptop and saved to an NFS server. (I'm running this backup including OS once or twice a week and a smaller one every day, similar to the description of example 3, section 9.4.) I had to backup more than 500,000 entries:

```
STATISTIC 2008.09.08 23:40:17  3961  [sec] |      user|    system
STATISTIC 2008.09.08 23:40:17  3961 -------+----------+----------
STATISTIC 2008.09.08 23:40:17  3961 process|    386.30|    166.27
STATISTIC 2008.09.08 23:40:17  3961 childs |    209.02|    116.96
STATISTIC 2008.09.08 23:40:17  3961 -------+----------+----------
STATISTIC 2008.09.08 23:40:17  3961 sum    |    595.32|    283.23 => 878.55 (14m39s)
STATISTIC 2008.09.08 23:40:17  3961                directories = 43498
STATISTIC 2008.09.08 23:40:17  3961                      files = 482516
STATISTIC 2008.09.08 23:40:17  3961             symbolic links = 12024
STATISTIC 2008.09.08 23:40:17  3961                 late links = 462267
STATISTIC 2008.09.08 23:40:17  3961                named pipes = 3
STATISTIC 2008.09.08 23:40:17  3961                    sockets = 48
STATISTIC 2008.09.08 23:40:17  3961              block devices = 0
STATISTIC 2008.09.08 23:40:17  3961          character devices = 0
STATISTIC 2008.09.08 23:40:17  3961      new internal linked files = 178
STATISTIC 2008.09.08 23:40:17  3961           old linked files = 462089
STATISTIC 2008.09.08 23:40:17  3961            unchanged files = 0
STATISTIC 2008.09.08 23:40:17  3961               copied files = 2896
STATISTIC 2008.09.08 23:40:17  3961           compressed files = 5204
STATISTIC 2008.09.08 23:40:17  3961    excluded files because rule = 78
STATISTIC 2008.09.08 23:40:17  3961    included files because rule = 0
STATISTIC 2008.09.08 23:40:17  3961         max size of copy queue = 22
STATISTIC 2008.09.08 23:40:17  3961  max size of compression queue = 361
STATISTIC 2008.09.08 23:40:17  3961           calculaed md5 sums = 50606
STATISTIC 2008.09.08 23:40:17  3961                forks total = 9176
STATISTIC 2008.09.08 23:40:17  3961                  forks md5 = 3957
STATISTIC 2008.09.08 23:40:17  3961                 forks copy = 12
STATISTIC 2008.09.08 23:40:17  3961                forks bzip2 = 5204
STATISTIC 2008.09.08 23:40:17  3961              sum of source =  10G (10965625851)
STATISTIC 2008.09.08 23:40:17  3961          sum of target all = 10.0G (10731903808)
STATISTIC 2008.09.08 23:40:17  3961          sum of target all = 97.87%
STATISTIC 2008.09.08 23:40:17  3961          sum of target new = 109M (114598007)
STATISTIC 2008.09.08 23:40:17  3961          sum of target new = 1.05%
STATISTIC 2008.09.08 23:40:17  3961         sum of md5ed files = 744M (779727492)
STATISTIC 2008.09.08 23:40:17  3961         sum of md5ed files = 7.11%
STATISTIC 2008.09.08 23:40:17  3961     sum internal linked (copy) =  32k (32472)
STATISTIC 2008.09.08 23:40:17  3961    sum internal linked (compr) = 6.2M (6543998)
STATISTIC 2008.09.08 23:40:17  3961          sum old linked (copy) = 3.3G (3515951642)
STATISTIC 2008.09.08 23:40:17  3961         sum old linked (compr) = 6.6G (7094777689)
STATISTIC 2008.09.08 23:40:17  3961           sum unchanged (copy) = 0.0  (0)
STATISTIC 2008.09.08 23:40:17  3961          sum unchanged (compr) = 0.0  (0)
STATISTIC 2008.09.08 23:40:17  3961                 sum new (copy) =  11M (11090534)
STATISTIC 2008.09.08 23:40:17  3961                sum new (compr) =  99M (103507473)
STATISTIC 2008.09.08 23:40:17  3961     sum new (compr), orig size = 321M (336637589)
STATISTIC 2008.09.08 23:40:17  3961                 sum new / orig = 32.96%
STATISTIC 2008.09.08 23:40:17  3961       size of md5CheckSum file =  16M (16271962)
STATISTIC 2008.09.08 23:40:17  3961     size of temporary db files = 0.0  (0)
STATISTIC 2008.09.08 23:40:17  3961            precommand duration = 1s
```

[11]You also can exclude too big files with the option exceptRule of storeBackup.pl from the backup and save them later when you have access to a better line.

```
STATISTIC 2008.09.08 23:40:17  3961           deleted old backups = 0
STATISTIC 2008.09.08 23:40:17  3961           deleted directories = 0
STATISTIC 2008.09.08 23:40:17  3961                 deleted files = 0
STATISTIC 2008.09.08 23:40:17  3961          (only) removed links = 0
STATISTIC 2008.09.08 23:40:17  3961 freed space in old directories = 0.0  (0)
STATISTIC 2008.09.08 23:40:17  3961       add. used space in files = 125M (130869969)
STATISTIC 2008.09.08 23:40:17  3961               backup duration = 27m3s
STATISTIC 2008.09.08 23:40:17  3961 over all files/sec (real time) = 297.30
STATISTIC 2008.09.08 23:40:17  3961  over all files/sec (CPU time) = 549.22
STATISTIC 2008.09.08 23:40:17  3961                     CPU usage = 54.13%
```

It took about 27 minutes to run the backup.
But look at the number of calculated md5 sums: 50,606. This is the number of files, a “normal” backup (which does not examine the contents) would have saved because a time stamp has changed or they have moved (I didn’t move files around, the changes were mainly from OS updates.). StoreBackup calculates the md5 sums and recognises that only 8,100 files (copied + compressd files) have changed.
So only 16% of the files which normally whould have been saved had to be stored. Over the time, this makes a big difference in the space you need for your backups. And naturally, the files in the backup are compressed (if reasonable).

Because the backup ran with the option lateLinks, I later had to run (via cron) storeBackupUpdate-Backup.pl to set all the links etc.:

```
INFO      2008.09.09 02:17:52 13323 updating </disk1/store-backup/fschjc-gentoo-all/2008.09.08_23.13.14>
INFO      2008.09.09 02:17:52 13323 phase 1: mkdir, symlink and compressing files
STATISTIC 2008.09.09 02:18:18 13323 created 43498 directories
STATISTIC 2008.09.09 02:18:18 13323 created 12024 symbolic links
STATISTIC 2008.09.09 02:18:18 13323 compressed 0 files
STATISTIC 2008.09.09 02:18:18 13323 used 0.0  instead of 0.0  (0 <- 0)
INFO      2008.09.09 02:18:18 13323 phase 2: setting hard links
STATISTIC 2008.09.09 02:27:55 13323 linked 462267 files
INFO      2008.09.09 02:27:55 13323 phase 3: setting file permissions
STATISTIC 2008.09.09 02:31:05 13323 set permissions for 482442 files
INFO      2008.09.09 02:31:05 13323 phase 4: setting directory permissions
STATISTIC 2008.09.09 02:31:47 13323 set permissions for 43498 directories
```

It took about 14 minutes to “complete” the backup for 500,000 entries.

6 Components / Programs to use

storeBackup.pl	performs the backup, is able to generate a configuration file for itself
storeBackupUpdateBackup.pl	If you choose the option 'lateLinks' in storeBackup.pl, it will not directly perform all the necessary hard links and is therefore much faster, especially when storing via nfs. This program will check all your dependencies and generate the hard links. As a result, your backup will have the same structure as calling storeBackup.pl without 'lateLinks'. This program is also used to perform the replication of backups.
storeBackupRecover.pl	Recovers files or (sub) trees from the backup uncompresses, restores all permissions and re-creates hard links like they were in the source.
storeBackupVersions.pl	Analyse the versions of backed up files.
storeBackupSearch.pl	Search with various criteria (rules) in the backup(s)
st...upSetupIsolatedMode.pl	Copies meta data to external media for backup eg. when traveling
st...upMergeIsolatedBackup.pl	Merges backups on local media to central master backup
storeBackupCheckBackup.pl	Checks integrity of all files in the backup by recalculating the md5 sum of all files and comparing them with stored ones
storeBackupCheckSource.pl	Checks Integrity of all unchanged files in the source directory by recalculating the md5 sum of all files and comparing them with stored ones
storeBackupls.pl	Lists backed up directories (versions) with additional information (week day, age of backup)
storeBackupDel.pl	Delete old backups using the same rules as in storeBackup.pl. This can be used to delete backups asynchronously. It can read the configuration file of storeBackup.pl
storeBackupMount.pl	You can use this program if you want to make a backup via nfs. It pings the server, mounts file system(s), calls storeBackup.pl and umounts the file system(s). It writes a log file and has a detailed error handling.
storeBackup_du.pl	Evaluates the disk usage in one or more backup directories.
storeBackupConvertBackup.pl	Convert (very) old backups to new format. Only use this if storeBackup.pl tells you to do.
linkToDirs	copy / link directories to other ones
llt	Shows atime, ctime and mtime of files.
multiTail.pl	Allows you to show (multiple) log files. You can also write multiple log files to one. It is more robust than 'tail -f'.

`st...up` is an abbreviation of `storeBackup`.

You can get a description of the options by calling the programs above with option '`-h`'.

6.1 Supported Platforms and Tips

The storeBackup tools have been reported to run on GNU/Linux, FreeBSD, Solaris and AIX. They should be able to run on all Unix platforms. Perl was used as the programming language, so you need a working perl implementation for starting one of the programs described above.
StoreBackup is developed and tested on GNU/Linux. For all programs you will get a short help message if you call it with option `-h`.

StoreBackup stores its data on a local filesystem – or something that looks like a local filesystem. You can store to any filesystem (or virtual filesystem) that supports hard link and the type of data you want to save (e.g., symbolic links or special files like named pipes if you want to save them). The following *examples* show some of the possibilites. (If you write to remote filesystems, you can speed up things by using option lateLinks, see section 7.6.)

ext4 is currently (2012) the fastest filesystem for Linux. It is well supported by the kernel and will be available for the foreseeable future.

ext2 You can use this filesystem, but there are several reasons not to do so: file system checks may last "forever" and it doesn't support hashes for filenames, which means access to the many small files generated from "blocked files" is slow.

reiserfs is the actually most space efficient filesystem for Linux because of tail packing. Space in filesystems is organized in blocks. The block size is typically 4kB, so as an average you will not be able to use around 2kB for each file. If you have a lot of files (esp. when using blocked files with compression and therefore undefined blocked file length) you will lose a high percentage of your space. With tail packing, these not filled last blocks of the files are packed together by the filesystem. Reiserfs is slower than ext4. It is well supported by the kernel and will be available for the foreseeable future.

vfat This fossil filesystem doesn't support hard links or differentiation of files written in uppercase and lowercase letters (try to store a file with filename `fileA` and one with `filea` into the same directory). You cannot store your backups with storeBackup on such a filesystem. Naturally, you can save data *from* such a filesystem using storeBackup.

ntfs First of all, you can store your backups on an nfts filesystem. But ownership and permissions will not be available in the backup. Especially if you use ntfs on an external disk or memory stick this might not be an issue. Read the "important note" at item "CIFS" below in this list!

NFS The Network File System allows you to store your backups transparently over the network (see configuring NFS, section 7.10). Naturally, you can also read your data via NFS if you do not want to run storeBackup natively on the system to save (e.g., for very old Unix system where you do not have a running perl 5).

CIFS It is possible to store your data on a CIFS (Samba) share. Beside being a little bit slower than NFS it does not support a multi user mount. So all your data will be stored with ownership of *one user* only. If your environment is a multi user environment where each user should have direct access to his backup data only, this type of storage is not sufficient for you. If each user is allowed to see all data in the backup or if an administrator does the restore, it's no problem to use e.g., a samba server (which is often the only available storage on small NAS boxes) to store your backups. Naturally, you also read data from a CIFS share, but you have to consider that CIFS only can be mounted on a user basis. It is not a transparent network file system like NFS.
Important note: If you restore your data with `storeBackupRecover.pl` you will get correct permissions and ownerships back. `StoreBackupRecover.pl` doesn't care in any way about the permissions of the files in the backup. The meta information (including hard links in the source) is taken from the meta data files `storeBackup.pl` stores. **BUT** if you use `storeBackup.pl` with the option `lateLinks` and if you can run `storeBackupUpdateBackup.pl` *locally* on your Samba file server, you will get all permissions in the backup directory like in the source directory.

sshfs A short description how to configure sshfs is placed in FAQ 4. Read the comments about CIFS in the item above for a description of possible restrictions.

6.2 storeBackup.pl

This is the basic program to make a backup. Beside a lot of options, there are two modes you can use:

1. Make a direct backup and do all the necessary copying, compressing, linking, permission settings etc. If you are not familiar with storeBackup, you should start with this mode.

2. Only do the absolutely necessary (deltas) and leave the rest to storeBackupUpdateBackup.pl which you have to run later. This is a kind of client / server mode.

```
NAME
    storeBackup.pl - fancy compressing managing checksumming hard-linking cp
    -ua
DESCRIPTION
    This program copies trees to another location. Every file copied is
    potentially compressed (see --exceptSuffix). The backups after the first
    backup will compare the files with an md5 checksum with the last stored
    version. If they are equal, it will only make an hard link to it. It
    will also check mtime, ctime and size to recognize idential files in
    older backups very fast. It can also backup big image files fast and
    efficiently on a per block basis (data deduplication).
    You can overwrite options in the configuration file on the command line.
SYNOPSIS
            storeBackup.pl --help
    or
            storeBackup.pl -g configFile
    or
            storeBackup.pl [-f configFile] [-s sourceDir]
                  [-b backupDirectory] [-S series] [--checkCompr] [--print]
                  [-T tmpdir] [-L lockFile] [--unlockBeforeDel]
                  [--exceptDirs dir1] [--contExceptDirsErr]
                  [--includeDirs dir1]
                  [--exceptRule rule] [--includeRule rule]
                  [--exceptTypes types]
                  [--specialTypeArchiver archiver [--archiveTypes types]]
                  [--cpIsGnu] [--linkSymlinks]
                  [--precommand job] [--postcommand job]
                  [--followLinks depth] [--highLatency]
                  [--ignorePerms] [--lateLinks [--lateCompress]]
                  [--checkBlocksSuffix suffix] [--checkBlocksMinSize size]
                  [--checkBlocksBS] [--checkBlocksCompr check|yes|no]
                  [--checkBlocksParallel] [--queueBlock]
                  [--checkBlocksRule0 rule [--checkBlocksBS0 size]
                   [--checkBlocksCompr0 key] [--checkBlocksRead0 filter]
                   [--checkBlocksParallel0]]
                  [--checkBlocksRule1 rule [--checkBlocksBS1 size]
                   [--checkBlocksCompr1 key] [--checkBlocksRead1 filter]
                   [--checkBlocksParallel1]]
                  [--checkBlocksRule2 rule [--checkBlocksBS2 size]
                   [--checkBlocksCompr2 kdey] [--checkBlocksRead2 filter]
                   [--checkBlocksParallel2]]
                  [--checkBlocksRule3 rule [--checkBlocksBS3 size]
                   [--checkBlocksCompr3 key] [--checkBlocksRead3 filter]
                   [--checkBlocksParallel3]]
                  [--checkBlocksRule4 rule [--checkBlocksBS4 size]
                   [--checkBlocksCompr4 key] [--checkBlocksRead4 filter]
                   [--checkBlocksParallel4]]
                  [--checkDevices0 list [--checkDevicesDir0]
                   [--checkDevicesBS0] [checkDevicesCompr0 key]
                   [--checkDevicesParallel0]]
                  [--checkDevices1 list [--checkDevicesDir1]
                   [--checkDevicesBS1] [checkDevicesCompr1 key]
                   [--checkDevicesParallel1]]
                  [--checkDevices2 list [--checkDevicesDir2]
                   [--checkDevicesBS2] [checkDevicesCompr2 key]
                   [--checkDevicesParallel2]]
                  [--checkDevices3 list [--checkDevicesDir3]
                   [--checkDevicesBS3] [checkDevicesCompr3 key]
                   [--checkDevicesParallel3]]
                  [--checkDevices4 list [--checkDevicesDir4]
                   [--checkDevicesBS4] [checkDevicesCompr4 key]
                   [--checkDevicesParallel1]]
                  [--saveRAM] [-c compress] [-u uncompress] [-p postfix]
                  [--noCompress number] [--queueCompress number]
                  [--noCopy number] [--queueCopy number]
                  [--withUserGroupStat] [--userGroupStatFile filename]
                  [--exceptSuffix suffixes] [--addExceptSuffix suffixes]
                  [--compressSuffix] [--minCompressSize size] [--comprRule]
                  [--doNotCompressMD5File] [--chmodMD5File] [-v]
```

```
[-d level][--progressReport number[,timeframe]] [--printDepth]
[--ignoreReadError]
[--suppressWarning key] [--linkToRecent name]
[--doNotDelete] [--deleteNotFinishedDirs]
[--resetAtime] [--keepAll timePeriod] [--keepWeekday entry]
[[--keepFirstOfYear] [--keepLastOfYear]
 [--keepFirstOfMonth] [--keepLastOfMonth]
 [--firstDayOfWeek day] [--keepFirstOfWeek]
 [--keepLastOfWeek] [--keepDuplicate] [--keepMinNumber]
 [--keepMaxNumber]
  | [--keepRelative] ]
[-l logFile
 [--plusLogStdout] [--suppressTime] [-m maxFilelen]
 [[-n noOfOldFiles] | [--saveLogs]]
 [--compressWith compressprog]]
[--logInBackupDir [--compressLogInBackupDir]
 [--logInBackupDirFileName logFile]]
[otherBackupSeries ...]
```

You have to set at least two options: `--sourceDir` *and* `--backupDir`. *It doesn't matter if you set them on the command line, in the configuration file or mixed.*

Options which can be used only on command line: There is always a long option (like `--file`) and sometimes also a shortcut (`-f`).

`--help` Generate a long help message with a short description of all options.

`--generate / -g` Generate a template for a configuration file. After generation, you can edit it with the editor of your choice. It is recommended to use the configuration file if you want to configure more than a simple backup.

`--print` Print the options used (from command line *and* from the configuration file) and stop after printing the options. In case of difficult quoting (especially on the command line) this gives you the chance to see what's really used in the program.

`--file / -f` Name of the configuration file you want to use when calling storeBackup.pl for a backup run.

`--checkCompr / -C` If you just want to run a simple backup and do not want to rely on the presets for "exceptSuffix" (which maybe not fit to your needs), use this option. It overwrites all settings for "exceptSuffix", "addExceptSuffix", "minCompressSize" and "comprRule".
If you use this option, `storeBackup.pl` will check if compression makes sense for all files bigger than 1k.

6.2.1 storeBackup.pl Options

The following options can be used on the command line and in the configuration file (see section 7.1). There is a long option for the command line (like `--sourceDir`), sometimes also a shortcut for the command line (like `-s`) and the name of the term used in the configuration file (like `sourceDir`). For options which can be used on command line only, see (previous) chapter 6.2.

`--sourceDir / -s / sourceDir` The path to the directory you want to backup. You can only backup *one* directory with storeBackup.pl. If you want to backup more than one directory, you can use `--includeDirs`, `--exceptDirs` or better the recommended option `--followLinks` (see below).

`--backupDir / -b / backupDir` The repository, where *all* your master backups are stored. In this document, this is often referred to as the master backup repository. You may have additional copies of your master backups in other locations (created via storeBackup's replication feature) but you normally run this program on the master backup repository. If you have one series of backups (e.g., from one computer), this parameter value will normally be the directory where your backups are. In this case, set the following option (`series`) to "`.`". Example:
`backupDir = /backup`

```
series = .
```
Then you will see your backups directly in `/backup`:
```
$ ls -l /backup
drwxr-xr-x 14 root root 528 Aug 24 21:33 2008.08.22_02.18.43
drwxr-xr-x 14 root root 528 Aug 24 21:33 2008.08.23_02.01.11
drwxr-xr-x 14 root root 528 Aug 24 21:33 2008.08.24_02.03.51
drwxr-xr-x 14 root root 528 Aug 24 21:33 2008.08.24_13.04.55
```
If you have different series of backups in your repository, you normally will create sub directories for each different backup series (perhaps from different computers) and configure `series` to these directory names. Let's assume you have three different computers to backup, "bob", "joe" and "bill". Then you can create three different directories:
```
$ ls -l /backup
drwxr-xr-x 2 root root 40 Aug 25 17:02 bill
drwxr-xr-x 2 root root 40 Aug 25 17:02 bob
drwxr-xr-x 2 root root 40 Aug 25 17:02 joe
```
Below these directories, you will find the individual backups for "bill", "bob" and "joe". E.g. for "bill" you will set:
```
backupDir = /backup
series = bill
```
Then you will see your backup in `/backup/bill`:
```
$ ls -l /backup
drwxr-xr-x 11 root root 432 Aug 24 21:33 2008.08.20_02.18.25
drwxr-xr-x 11 root root 432 Aug 24 21:33 2008.08.21_02.11.53
drwxr-xr-x 11 root root 432 Aug 24 21:33 2008.08.22_02.36.18
drwxr-xr-x 11 root root 432 Aug 24 21:33 2008.08.23_02.17.18
drwxr-xr-x 11 root root 432 Aug 24 21:33 2008.08.24_02.15.45
drwxr-xr-x 11 root root 432 Aug 24 21:33 2008.08.24_13.17.21
```

`--series / -S / series` see option `backupDir` above.
The default value for `series` is "default". To rename an existing series do the following:

- Run storeBackupUpdateBackup.pl so no unresolved lateLinks (see option `lateLinks` below) exists.
- Rename the directory below the directory specified with option `backupDir` to whatever name you want.
- Configure this option (`series`) to the name of the directory you have chosen in the step before.

`--tmpdir / -T /tmpdir` Directory for temporary files, the default value is picked from the environment variable `$TMPDIR`. If it does not exist, `/tmp` is set as the default value.

`--lockFile / -L / lockFile` storeBackup.pl uses a lock file to avoid it running multiple times. The default name of the lock file is `/tmp/storeBackup.lock`.
This type of lock files does not work across multiple servers and is not designed to separate `storeBackup.pl` and `storeBackupUpdateBackup.pl` or any other storeBackup process in a separate PID[12] space. If you want to use lock files across multiple servers you should build your own solution (maybe on top of these lock files) or use an "enterprise job scheduler".

`--unlockBeforeDel / unlockBeforeDel` Remove the lock file before deleting old backups. Default is to delete the lock file after removing old backups. This "shortens" the time for a backup from some perspective.

`--exceptDirs / -e / exceptDirs` You can specify a list of directories to be excluded from the backup. It must be a *relative path* from the point specified with option `sourceDir`. You can also use wildcards. To give an example, if all your users reside below *`sourceDir`*`/home` and you want to avoid to backup the directory `tmp` in each home directory, you can say:
```
exceptDirs = home/*/tmp
```
For interpreting the wildcards, storeBackup.pl uses a shell. So if the resulting list of directories is

[12]process ID

too long (about 4K), then this will not work any more. Then you should use option `exceptRule` (see below).
If you want to specify a list of directories in the configuration file simply write:
`exceptDirs = home/*/tmp 'otherdir/temp'`
On the command line, simply repeat the option:
`-e 'home/*/tmp' -e 'otherdir/tmp'`
Here, quoting `home/*/tmp` is important to avoid the expansion of the term by the shell.
(In section 7.2“Selecting Directories / Files to Backup” you will get an overview about different options to include / exclude files or directories.)

`--contExceptDirsErr / contExceptDirsErr` storeBackup.pl will continue to backup even if one or more directories specified with `exceptDirs` does not exist. Default is to print and error message and stop.

`--includeDirs / -i / includeDirs` If this option is set, only files in directories specified here are backed up. StoreBackup.pl will only include files which are *not in* the exceptDirs and *in* the includeDirs. This option can be used in the way as described for exceptDirs.
(In section 7.2“Selecting Directories / Files to Backup” you will get an overview about different options to include / exclude files or directories.)

`--exceptRule / exceptRule` If this rule matches, the affected file is excluded from the backup. The rules are executed on regular files. You can read more about rules in section 7.4.
(In section 7.2“Selecting Directories / Files to Backup” you will get an overview about different options to include / exclude files or directories.)

`--includeRule / includeRule` If a definition for this option exists then only files which match this rule are backup up. StoreBackup.pl will back up files which are *not* excluded by the backup and *match* the includeRule. You can read more about rules in section 7.4.

`--writeExcludeLog / writeExcludeLog` This option tells storeBackup.pl to write a file with the names of files which have been excluded because of rules. The file will be stored in the top level of the actual backup with the name `.storeBackup.notSaved.bz2`. It is compressed with bzip2.

`--exceptTypes / exceptTypes` Do not save the files of the specified type. StoreBackup.pl knows:
`S` — file is a socket
`b` — file is a block special file
`c` — file is a character special file
`f` — file is a plain file
`p` — file is a named pipe
`l` — file is a symbolic link
`Sbc` can only be stored if you have gnu-cp in your path and activate the “gnucp” option (see below).
If you specify
`exceptTypes = Sbc`
then files of these types will not be stored in the backup and no warning will be generated. This rule is evaluated before “exceptRule” and “includeRule”. If you want to exclude some file types in general, use this option (it's faster and easier to use).

`--archiveTypes archiveTypes` Save the specified type of files in an achive instead of saving them directly in the file system. This is useful, if you want to backup those file types but your target file or transport mechanism (e.g. sshfs or non gnu-cp) system does not support those types of files. You can select the following types of files:
`S` — file is a socket
`b` — file is a block special file
`c` — file is a character special file
`p` — file is a named pipe

`--specialTypeArchiver / specialTypeArchiver` Possible values are `cpio` and `tar`. Default option is `cpio`. See option `archiveTypes` described above.
Note: `tar` is not able to archive sockets; `cpio` is not part of the actual posix standard any more.

`--cpIsGnu / cpIsGnu` If you choose this option, you will be able to backup (and restore) files of type `Sbc` (see above). For restoring with storeBackupRecover.pl, you also need gnu-cp. If you are using a linux system, your cp will be gnu cp.

`--linkSymlinks / linkSymlinks` If you store your backups on a file system which supports hard links to symbolic links, you should activate this option. GNU/Linux does support this feature. Default is not to hard link symbolic links.

`--precommand / precommand` You can define *one* command (or script) to be executed before storeBackup.pl starts the backup. It will only start after the lock file is checked. If the return value of this command / script is != 0, then storeBackup.pl will stop immediately. The output of this command to stdin is printed as a warning to the storeBackup.pl log file, the output to stderr is printed as an error.
The cli parameter to this option is parsed like a line in the configuration file and normally has to be quoted. This means, you can use parameters, e.g.:
`precommand = /backup/pre.sh param1 param2`
is the same as:
`--precommand '/backup/pre.sh param1 param2'`

`--postcommand / postcommand` This command is executed after finishing the backup, but before starting the deletion of old backups. StoreBackup.pl reports, if the exit status is != 0.
The cli parameter to this option is parsed like a line in the configuration file, see option "precommand".

`--followLinks / followLinks` If you want to backup more than one directory, you should use this option. For instance, if you want to backup `/boot`, `/etc` and `/home/tom`, then you should do (as root) something similar to:

```
# mkdir /backup
# cd /backup
# ln -s /boot boot
# ln -s /etc etc
# ln -s /home/tom home_tom
# ln -s . backup
# storeBackup.pl -g stbu.conf
```

Then you should configure your backup by editing file `stbu.conf`. Configure (among others):

```
# sourceDir = /backup
# followLinks = 1
```

This will tell storeBackup.pl to take the fist level of symbolic links below `/backup` like directories. With "`ln -s . backup`" you will get a sub directory inside of your backup which exactly reflects `/backup`.
"followLinks" configures storeBackup.pl to treat n levels of directories or symbolic links as directories. Simply by adding or deleting a symbolic link to your backup directory, you can add or remove any directory in your file system to `/backup` from your backup.

`--stayInFileSystem / stayInFileSystem` Only files and directories in file systems which are specified by option `sourceDir` of via the symbolic links of option `followLinks` are saved.

`--highLatency / highLatency` Use this option if you are running storeBackup on line with a very high latency, like a vpn over the internet. This option will use more parallelisation at the cost of more cpu needed. If you use this option, then it would be a good idea to use `--lateLinks` and probably `--lateCompress`.
Do not use this option for regular backups to a another local disk or to nfs mounts on the local network.

`--ignorePerms / ignorePerms` With this option, files in the backup will not necessarily have the same permissions and owners as the original ones. This speeds up the backup. Recovery with storeBackupRecover.pl will restore the permissions and owners correctly. There are several possibilities to improve performance, see section 5.4.3.

`--lateLinks / lateLinks` This option will reduce your *direct* backup time at the cost of a second process you have to run later. For a local backup onto another disk, you will see an improvement of 30–50%. If you write a backup over NFS, you will see an improvement by a *factor* of 5 to 10. This value can vary depending on how many new files you have to backup and how fast your network is. Saving over a vpn over the Internet I measured an improvement with lateLinks by a factor of 70.
If you want to use "lateLinks" you have to read section 7.6.

`--lateCompress / lateCompress` This option can only be used if "lateLinks" is set. Compression of files ≥ "minCompressSize" will be done later when starting storeBackupUpdateBackup.pl. See also section 7.6.

`--checkBlocksSuffix` The configuration is similar to `exceptSuffix`, a list of suffixes which are checked for a match, e.g., \\`.vdmk` for VMware images. They simply mean that the last part of the file name must be similar to what you define here.
The next options described here are only used if `checkBlocksSuffix` is set.
See blocked files (section 7.5) for more information about the options with "`block`" in their name.

`--checkBlocksMinSize` Only files with this minimum size will the treated as blocked files. You can use the same shortcuts as described in defining rules, see section 7.4, e.g., 50M means 50 megabytes. The default value is 100M.

`--checkBlocksBS` Defines the block size in which the files which matches have to be split by storeBackup.pl. The format is equal to `checkBlocksMinSize`. The default value is 1M. The minimal value is 10k.

`--checkBlocksParallel` Read the files specified here parallel to the files *not* specified in checkBlocksSuffix. This normally only makes sense if the files specified here are small or if the are on a separate device. Default is `no`, which means not to parallelize.

`--checkBlocksCompr` Defines if the blocks are compressed. Possible values are `yes`, `no` or `check`; the default value is `no`.
This option only affects files selected with `checkBlocksSuffix`. If you set this option to `check`, every block is checked for compression (or not), see How to define if a file should be compressed (section 7.4.1).

`--checkBlocksRule`i The ith rule specifying files to treat as blocked files in the backup. You can define 5 rules, beginning from `checkBlocksRule0` to `checkBlocksRule4`.
See blocked files (section 7.5) for more information about the options with "`block`" in their name.

`--checkBlocksBS`i The corresponding block size for the blocks in the backup. The default value is 1 megabyte (1M). The minimal value is 10k.

`--checkBlocksCompr`i Defines if the blocks are compressed. Possible values are `yes`, `no` or `check`; the default value is `no`.
This option only affects files selected with `checkBlocksSuffix`. If you set this option to `check`, every block is checked for compression (or not). See How to define if a file should be compressed (section 7.4.1).

`--checkBlocksRead`i Defines a filter for reading the specified file in `sourceDir`, e.g., `gunzip` or `gzip -d`. This option is useful if you have to save an already compressed image file. (Using the "blocked file" feature of storeBackup with already compressed files compressed as a whole does not make sense.)

`--checkBlocksParallel`i Read the files specified here in parallel to the files *not* specified in checkBlocksRule$i$ or checkDevices$i$. This normally only makes sense if the files specified here are small or if the are on a separate device.
Default is `no`, which means not to parallelize.
You have to know, that files and devices specified in checkBlocksRule$i$ or checkDevices$i$ are *never* parallelized.

`--checkDevices`i List of devices (e.g., `/dev/sdd2 /dev/sde1`) to backup.

`--checkDevicesDir`*i* Directory where the devices are stored in the backup (*relative* path). The image file will also be restored in that directory if you restore the backup with storeBackupRecover.pl (if you use default options). In this directory, storeBackup will create a subdirectory which name is generated from the parameters of `checkDevices`, e.g., `/dev/sdc` will result in `dev_sdc`.

`--checkDevicesBS`*i* Defines the block size in which the devices specified have to be split by storeBackup.pl. The format is equal to `checkBlocksMinSize`. The default value is 1M. The minimal value is 10k.

`--checkDevicesCompr`*i* Defines if the blocks are compressed. Possible values are `yes`, `no` or `check`; the default value is `no`.
If you set this option to `check`, every block is checked for compression. See How to define if a file should be compressed (section 7.4.1).

`--checkDevicesParallel`*i* Read the devices specified here parallel to the files *not* specified in checkBlocksRule*i* or checkDevices*i*. This normally only makes sense if the files specified here are small or if they are on a separate device.
Default is `no`, which means not to parallelize.
Please note that files and devices specified in checkBlocksRule*i* or checkDevices*i* are *never* parallelized.

`--saveRAM / saveRAM` Use this option if storeBackup.pl runs on a system with very low memory. You will then see some dbm files in "tmpDir". This will slow down storeBackup.pl a little bit, so do this only if you run into problems without it. On modern computers, it should only be necessary to use this option if you backup millions of files.

`--compress / -c / compress` The command storeBackup.pl uses for compression. Default is `bzip2`.
The cli parameter to this option is parsed like a line in the configuration file and normally has to be quoted on the command line. This means you can use parameters, e.g.:
```
compress = gzip -9
```
which is similar to:
```
--compress 'gzip -9'
```

`--uncompress / -u / uncompress` The command storeBackup.pl uses for uncompressing the files in the backup with storeBackupRecover.pl. Default is "`bzip2 -d`". It must fit to the parameter of the option "compress".
The cli parameter to this option is parsed like a line in the configuration file and normally has to be quoted on the command line. This means, you can use parameters, e.g.:
```
uncompress = gzip -d
```
which is similar to:
```
--uncompress 'gzip -d'
```

`--postfix / -p /postfix` The postfix storeBackup.pl will use for compressed files. This should fit to the option compress. Default is `.bz2`.

`--noCompress / noCompress` Maximal number of parallel compression operations. With GNU/Linux, the default value is chosen automatically as the number of cores plus 1.

`--queueCompress / queueCompress` Maximal length of a queue to store files before they are compressed. Default value is 1000.

`--noCopy / noCopy` Maximal number of parallel copy operations. The default value is 1.

`--queueCopy / queueCopy` Maximal length of a queue to store files before they are copied. The default value is 1000.

`--withUserGroupStat / withUserGroupStat` Write statistics about used space in sourceDir by user and groups in the log file.

`--userGroupStatFile / userGroupStatFile` Write statistics about used space in sourceDir by user and groups in this file. This file will be overwritten each time.

`--exceptSuffix / exceptSuffix` Do not compress files with these suffixes. On the command line, you can repeat this option multiple times. The default value is:

```
exceptSuffix = '\.zip' '\.bz2' '\.gz' '\.tgz' '\.jpg' '\.gif' '\.tiff'
    '\.tif' '\.mpeg' '\.mpg' '\.mp3' '\.ogg' '\.gpg' '\.png'
```

You should use a backslash (\) to mask the dot. If you do not do so, the dot is interpreted as any character.
If you do not want to compress any file, you can use:
`exceptSuffix = .*`

`--addExceptSuffix / addExceptSuffix` If you only want to *add* suffixes to the above, use this option. On the command line, you can repeat this option multiple times. See the examples above (option exeptSuffix) how to use it in the configuration file.

`--compressSuffix / compressSuffix` List of suffixes of files to be compressed (in `exceptSuffix` format). If you enter this value, then a rule will be generated depending on `exceptSuffix`, `addExceptSuffix`, `minCompressSize` and a special rule-function to theck if a compression of the files not affected by the suffix based criteria will be done or not. Easy examples and detailed explanations are presented in How to define if a file should be compressed (section 7.4.1).

`--minCompressSize / minCompressSize` Files with a size smaller than this value will not be compressed. The default value is 1024.
If you change this value from one backup to the next (e.g. if you make the fist backup with the default value and the second with 512), then this change affects only files which have a new content. Files with a content which exists already in the backup will be linked to the ones in the old backup. (So, in the example a (new) file with 600 bytes will not be compressed in the second backup if there was already a file with the same content in the first backup.)

`--comprRule / comprRule` You can use this rule as an alternative to options `exceptSuffix`, `minCompressSize`, `addExceptSuffix` and `compressSuffix`. If this rule is set, the just mentioned options are ignored (this means that no rule is generated from these options). See defining rules (section 7.4) to understand how to configure rules. You can e.g., define a rule that the data of serveral uses will not be compressed for easier restore by the users themself.

`--doNotCompressMD5File / doNotCompressMD5File` StoreBackup.pl stores information about each file in the backup in the top level directory of each backup in a file called `.md5CheckSums`. It normally is compressed with bzip2. Using this option avoids this compression. Use this if your computer is very slow and has only one core. It will speed up things a little bit.

`--chmodMD5File` Everybody who wants to use storeBackupRecover.pl needs to be able to read the file `.md5CheckSums` (see option above). Default permission on this file is 0600, which means only the one who generated the backup has access to it. With this option you can give access to other people. If you do so, this can be a kind of a security hole: for all files `.md5CheckSums` stores md5sum, times, uid, gid, mode (and some other information).
Direct access to files in the backup is independent of this option.

`--verbose / -v / verbose` Generate verbose messages.

`--debug / -d / debug` Generates debug messages:
0 — no debug messages (default)
1 — some debug messages
2 — many debug messages
This option is especially helpful in combination with options exceptRule and includeRule.

`--resetAtime / resetAtime` Restores the access time in the backups (same as in source), but changes ctime (creation time). Normally, you will not use this option.

`--doNotDelete / doNotDelete` Do all checks what backups should be deleted, but do not delete anything. This option is useful in combination with storeBackupDel.pl which can read the configuration file of storeBackup.pl. StoreBackupDel.pl can delete old backups later asynchronously.
For understanding the rules what file should be deleted, see the "keep∗" options below.

`--deleteNotFinishedDirs / deleteNotFinishedDirs` Delete backups which have not been finished and are therefore not complete. `StoreBackup.pl` or `storeBackupDel.pl` will only delete unfinished backups if option `doNotDelete` is set to `yes` or if this option is used on the command line. The default value for this option is `no`.

`--keepAll / keepAll` Keep all backups of a series for the specified amount of time. This is like a default value for *all* days in option keepWeekday (see below). Deletion of old backups is done after the actual backup is finished or with storeBackupDel.pl. The time range has to be specified in format "dhms", e.g. "10d2h" means 10 days and 2 hours. To do so is useful if you want to specify 10 days, because if you define this exactly, then checking a few minutes or seconds before or later can result in 9 days. StoreBackups internal calculation is in seconds.
The default value is "30d".

`--keepWeekday / keepWeekday` This option overwrites the settings of option keepAll for special days of the week. `Mon,Wed:40d5m Sat:60d10m` means:

- keep backups from Monday and Wednesday 40 days + 5 minutes
- keep backups from Saturday 60 days + 10 minutes
- keep backups from the rest of the weekdays like specified via option keepAll.

You can also use the "archive flag" which means not to delete the affected directories because option "keepMaxNumber" hits. `Mon,Wed:a40d5m Sat:60d10m` means:

- keep backups from Monday and Wednesday 40 days + 5 minutes + archive
  If you have more than "keepMaxNumber" backups, then Monday and Wednesday backups falling in this category will not be deleted.
- keep backups from Saturday 60 days + 10 minutes
  If you have more than "keepMaxNumber" backups, then Saturday Backups falling in this category will be deleted.
- keep backups from the rest of the weekdays like specified via option keepAll. If you have more than "keepMaxNumber" backups, then they will be deleted if they are falling in this category.

On the command line, the parameter to this option is parsed like a line in the configuration file and therefore normally has to be quoted on the command line.

`--keepFirstOfYear / keepFirstOfYear` Do not delete the first existing backup of a year. The format is a time period (see option keepAll) with a possible "archive flag".

`--keepLastOfYear / keepLastOfYear` Do not delete the last existing backup of a year. The format is a time period (see option keepAll) with a possible "archive flag".

`--keepFirstOfMonth / keepFirstOfMonth` Do not delete the first existing backup of a month. The format is a time period (see option keepAll) with a possible "archive flag".

`--keepLastOfMonth / keepLastOfMonth` Do not delete the last existing backup of a month. The format is a time period (see option keepAll) with a possible "archive flag".

`--firstDayOfWeek / firstDayOfWeek` Sets the first day of the week for the calculations depending on options keepFirstOfWeek and keepLastOfWeek. Default value is "Sun".

`--keepFirstOfWeek / keepFirstOfWeek` Do not delete the first existing backup of a week. The format is a time period (see option keepAll) with a possible "archive flag".

`--keepLastOfWeek / keepLastOfWeek` Do not delete the last existing backup of a week. The format is a time period (see option keepAll) with a possible "archive flag".

`--keepDuplicate / keepDuplicate` Keep multiple backups of one day up to the specified value. If it's older than specified here, delete all except the oldest backup of that day. Usage of the "archive" flag is not possible. The format is like described for option keepAll. The default value is "7d".

`--keepMinNumber / keepMinNumber` Keep that minimum of backups. Multiple backups of one day are counted as one backup. The default value is "10".

`--keepMaxNumber / keepMaxNumber` Try to keep only that maximum number of backups. If you have more backups than specified here, the following sequence of deletion will happen:

- Delete all duplicates of a day, beginning with the oldest ones, except the last of every day.
- If this is not enough, delete the rest of the backups beginning with oldest, but *never* a backup with the "archive flag" or the last backup. See option keepWeekday for explanations of the "archive flag".

`--keepRelative / -R / keepRelative` This is a alternative deletion scheme. If this option is set, all other keep∗ options are ignored. On the command line, the parameter to this option is parsed like a line in the configuration file and normally has to be quoted.
This backup deletion scheme allows you to specify the relative age of the backups you would like to have rather then the period over which a backup should be kept.
Imagine that you always want to have the following backups available:

- 1 backup from yesterday
- 1 backup from last week
- 1 backup from last month
- 1 backup from 3 months ago

Note that this is most likely *not* what you really want to have, because it simply means that you have to do daily backups and have to keep every backup for exactly 3 months. Otherwise you wouldn't always have a backup that is of *exactly* the requested age.
What you really want to have is therefore probably something like this:

- 1 backup of age 1 hour to 24 hours / 1 day
- 1 backup of age 1 day to 7 days
- 1 backup of age 14 days to 31 days
- 1 backup of age 80 days to 100 days

This is now a very common backup strategy, but you would have difficulty to achieve this with the usual keepFirstOf∗ options, especially if you do not do backups with perfect regularity. However, you can implement it very easily using keepRelative. All you need to write is:
`keepRelative = 1h 1d 7d 14d 31d 80d 100d`
i.e. you list all the intervals for which you want to have backups. storeBackup will delete backups in such a way that you come as close as possible (if you do not do backups often enough, there is of course nothing that storeBackup can do) to your requested backup scheme.
Note that this may mean that storeBackup keeps more backups that you think it has to, i.e. it may keep two backups in the same period. In this case storeBackup "looks into the future" and determines that both backups will *later* be necessary in order to have a backups for all periods. This is also the reason why in the above example you have somehow implicitly specified the period 7 days to 14 days, although you didn't really want to have a backup in this period – in order to have backups in the next period (14 days to 31 days) you always need to have a backup in the period 7 days to 14 days as well. Therefore the syntax doesn't allow you to exclude some periods.
Finally you should be aware that storeBackup shifts all the intervals if it cannot find a recent enough backup: if your first intervals is from 10 days to 20 days, but your most recent backup is actually 25 days old, all subsequent periods will be extended by 5 days. This ensures that if you haven't made any backups over a large period, this period is not taken into account for your backup scheme. To give an example why this is useful: If you wanted to have backups 1, 3, 7 and 10 days old and then went on vacation for 14 days, it is pretty unlikely that you want all your backups deleted when you come back, hence storeBackup ignores these 14 days and keeps the backups appropriately longer.

`--progressReport / -P / progressReport` Print a progress report after the specified number of files. If you want to get a message at least after a specific time frame, you may add that time frame separated by a comma, eg:
`-P 1000,1m10s` on the command line or
`progressReport = 1000,1m10s` in the configuration file.
There must be no white space in the parameter to that option. The syntax of the time frame is the same as with the keep∗ options.

`--printDepth / -D / printDepth` Print depth of actually read directories during the backup.

`--ignoreReadError / ignoreReadError` Setting this option lets storeBackup.pl ignore read errors in the source directory – not readable directories do not cause storeBackup.pl to stop processing. This option was implemented for reading shares from a windows server which sometimes generated such faults. Normally, you should not use this option.

`--suppressWarning / suppressWarning` Suppresses (unwanted) warnings that would normally be written to the log (and/or standard output). This is an advanced option. *For normal use of storeBackup, you can ignore this option.* In some situations, an advanced user may not want to see certain warnings. This option allows the user to turn those warnings off. This feature is only available for certain non-critical warnings, e.g. missing excluded directories, files changed during backup, or creation of the 'default' series. [13]

- Using suppressWarning with the `excDir` key suppresses the warning that an excluded directory does not exist.
- Using the `fileChange` key suppresses any warning when storeBackup notices that a file has changed since the current backup began.
- Using the `crSeries` key suppresses the warning that storeBackup had to create a directory for the 'default' series.
- Using the `hashCollision` key suppresses the warning that storeBackup found a possible md5 hash collision.
- Using the `fileNameWIthLineFeed` key suppresses the warning if a filename contains a line feed.
- Using the `use_DB_File` key suppresses the warning if in your perl installation the CPAN module `DB_File` is not installed. If possible, you should install it – it helps improve performance and reduces memory usage.
- Using the `use_MLDBM` key suppresses the warning which is generated if you may use the rule functions `MARK_DIR` or `MARK_DIR_REC` in combination with option `saveRAM` *and* the perl CPAN module `MLDBM` is not installed on you computer.
- Using the `use_IOCompressBzip2` key suppresses the warning that you should install the perl CPAN module `IO::Compress::Bzip2` for better performance. Ignore / suppress this message if you do not use `bzip2` as compression method.
- Using the `noBackupForPeriod` key suppresses the warning that there are no backups for certain periods when using the option `keepRelative`.

`--linkToRecent` After a successful backup, a symbolic link to the most recent backup of this series (that's the backup just done) is created with the name specified by this option. If an older symbolic link exists, it will be deleted. If you change the name of this symbolic link in the configuration, the old link will *not* be removed – you have to delete it manually.

`--logFile / -l / logFile` Name of the log file. Default is stdout.

[13]The logic behind the suppressWarning option is that *r*epeated non-critical warnings can cause the user to ignore most warnings in general. Here is an example of how you could benefit from this option. Say you have defined a list of directories to exclude from the backup such as temporary directories. Sometimes you limit your list of included directories also. If you limit the included directories in such a way that the excluded directories are not part of the backup, storeBackup would normally generate a warning for every such "missing" excluded directory. However, you may choose to leave the excluded directories defined in the configuration file because when you expand your included directory list you do not want to risk forgetting to again define the excluded directories. But you also do not want the warnings because too many non-critical warnings might prevent you from seeing an important one. In that situation, you can use this option. It means that when altering your included directories list, you only have to make one change (includeDirs) rather than two changes (includeDirs and exceptDirs).
However, there is a situation where using this option to suppress warnings of missing excluded directories could have a negative consequence. Say you have an excluded temporary directory named `testing` that you do not want to back up. Now you rename `testing` to `app1_testing` but you still do not want it to be backed up. If you do not update your storeBackup config file, and if `app1_testing` is under an included directory, it will now be backed up. However, if you have not suppressed this class of warning, storeBackup will alert you that `testing` (the previously excluded directory name) cannot be found. That will probably remind you of your change and let you update your configuration. So use this option with caution. If you are not sure whether you should use it, you probably should not.

`--plusLogStdout / plusLogStdout` If the option logFile is set you can configure here storeBackup.pl to also print the log messages to stdout.

`--suppressTime / suppressTime` Suppress the output of the actual time in the log file.

`--maxFilelen / -m / maxFilelen` Maximal size of a log file. After reaching this size, the log file will be rotated (see option noOfOldFiles) or compressed (see option saveLogs).

`--noOfOldFiles / -n / noOfOldFiles` Number of old rotated log files, default is 5. With default values, it will look like this:

```
$ ls -l /tmp/storebackup.log*
-rw------- 1 hjc  root  328815 30. Aug 12:12 /tmp/storebackup.log
-rw------- 1 root root 1000087 27. Aug 21:18 /tmp/storebackup.log.1
-rw------- 1 root root 1000038 20. Aug 19:02 /tmp/storebackup.log.2
-rw------- 1 root root 1000094 11. Aug 18:51 /tmp/storebackup.log.3
-rw------- 1 root root 1000147 11. Aug 18:49 /tmp/storebackup.log.4
-rw------- 1 root root 1000030 11. Aug 18:49 /tmp/storebackup.log.5
```

Older log files than *.5 have been deleted automatically.

`--saveLogs / saveLogs` Save the log files with a time and date stamp instead of deleting them after rotating. (Setting this option deactivates option noOfOldFiles.)

`--compressWith / compressWith` Specifies the program to compress the saved log files (e.g., with `gzip -9`). Default value is `bzip2`.
On the command line, the parameter to this option is parsed like a line in the configuration file and normally has to be quoted.

`--logInBackupDir / logInBackupDir` Write the log file (also) in the backup directory (default name is `.storeBackup.log`, also see option logInBackupDirFileName below). This log file as the case may be does not contain all error messages like the one specified with option logFile. (The backup directory must exist before any message can be written into this log file.)
This is useful for having a (historical) log file, while the "global" log file (from option logFile) is useful for monitoring.

`--compressLogInBackupDir / compressLogInBackupDir` The log file in the backup directory will be compressed if you specify this option.

`--logInBackupDirFileName / logInBackupDirFileName` File name of the log file to be stored in the backup directory. Default is `.storeBackup.log`.

`...otherBackupSeries...  / otherBackupSeries` On the command line, this is not an option; it is a list parameter. So you have to write on the command line e.g.:

```
# storeBackup.pl <all_options> 0:server2 0-2:server3
```

In the configuration file this is similar to:
`otherBackupSeries = 0:server2 0-2:server3`
Here you can specify a list of other backup directories to consider for hard linking. The path to their backup directories must be a relative path from backupDir!
Format (examples):

```
otherSeries/2002.08.29_08.25.28 -> consider exactly this otherSeries
```

or

```
0:otherSeries -> last (youngest) in <backupDir>/otherSeries
1:otherSeries -> first before last in <backupDir>/otherSeries
n:otherSeries -> n'th before last in <backupDir>/otherSeries
3-5:otherSeries -> 3rd, 4th and 5th in <backupDir>/otherSeries
all:otherSeries -> all in <backupDir>/otherBackupSeries
```

If you do not specify `otherBackupSeries` then the youngest backup from all series in the top level directory you specified with the option `backupDir` is considered automatically.

To configure `otherBackupSeries`, you can use wildcards. More on that topic can be found at the beginning of section 7.8.7 "Using Wildcards for Replication"

6.3 storeBackupUpdateBackup.pl

You need to run this program if you use the option lateLinks in storeBackup.pl. See section 5.4.3 about performance why you perhaps want to use this option, and section 7.6 how to use it.
This program includes the necessary copying if you are using the replication of backups, see chapter 7.8.
StoreBackupUpdateBackup.pl does the job of generating hard links, directories, symbolic links, compression of files and setting permissions storeBackup.pl does not do with option lateLinks. Before it starts doing this, it will check the consistency of your references resulting from the use of lateLinks in your backup repository, e.g. it detects if one backup is missing.

To correct inconsistencies, use storeBackupUpdateBackup.pl in interactive mode (option `--interactive`).

To generate configuration files for replication of backups, use the options `--genBackupBaseTreeConf` and `--genDeltaCacheConf`.

```
    storeBackupUpdateBackup.pl - updates / finalizes backups created by
    storeBackup.pl with option --lateLink, --lateCompress
SYNOPSIS
            storeBackupUpdateBackup.pl -b backupDirectory [--autorepair]
                  [--print] [--verbose] [--debug] [--lockFile] [--noCompress]
                  [--progressReport number] [--checkOnly] [--copyBackupOnly]
                  [--dontCopyBackup] [-A archiveDurationDeltaCache]
                  [--dontDelInDeltaCache] [-T tmpdir]
                  [--logFile
                   [--suppressTime] [-m maxFilelen]
                   [[-n noOfOldFiles] | [--saveLogs]]
                   [--compressWith compressprog]]
            storeBackupUpdateBackup.pl --interactive --backupDir topLevlDir
                  [--autorepair] [--print]
            storeBackupUpdateBackup.pl --genBackupBaseTreeConf directory
            storeBackupUpdateBackup.pl --genDeltaCacheConf directory
```

The only option you have to specify is backupDir, the rest of the options are optional. This program only accepts parameters on the command line. It is not possible to use a configuration file.

`--interactive / -i` Interactive mode for repairing/deleting currupted backups created with the option lateLinks.

`--backupDir` The repository, where all your backups are stored. This program can be used on the master backup or a backup copy (e.g., a copy created via replication). The meaning of this parameter is similar to the option backupDir of storeBackup, see section 6.2.

`--autorepair / -a` If storeBackupUpdateBackup.pl detects inconsistencies which do not harm your repository, it will repair the reference files automatically without asking you. It will only write an INFO message in the log file and tells you what it repaired.
For instance, if you deleted your *last* backup with lateLinks using `rm` (which you should not do before successfully running this program!), then the internal referencing structure of your backups is inconsistent. StoreBackupUpdateBackup.pl (and also storeBackup.pl) will recognize that a backup which referenced to another one is missing. But correcting the reference structure does not lead to a loss of data, so this is an example when it can be repaired without user intervention. (For more information see section 7.9 about special files.)

`--print` Prints the options used and stops after printing the options. In case of difficult quoting (especially on the command line) this gives you the chance to see what's really used in the program.

`--verbose / -v` Generate verbose messages.

`--debug / -d` Generate detailed information about linking, compressing, etc.

`--lockFile / -L` Specify a lock file. If the lock file exists and a process with the id stored in it is running, then the program will immediately stop to avoid running it multiple times (which would be a very bad idea). The default is `/tmp/storeBackupUpdateBackup.lock`. You should also not run storeBackupUpdateBackup.pl parallel to storeBackup.pl.
This type of lock files does not work across multiple servers and is not designed to separate `storeBackup.pl` and `storeBackupUpdateBackup.pl` or any other storeBackup process in a separate PID[14] space. If you want to use lock files across multiple servers you should build your own solution (maybe on top of these lock files) or use an "enterprise job scheduler".

`--noCompress` Maximal number of parallel compression operations. The default value is chosen automatically (number of cores plus 1).

`--checkOnly / -c` Do not perform any action, only check the consistency. Use this in combination with option –debug to get detailed information.

`--copyBackupOnly` If you use replication, only replicate to/from delta cache, do not do any hard linking, compressing, or changing of file permissions

`--dontCopyBackup` Do not do any replication tasks.
NOTE: If used on a master backup this option will disrupt the data flow for replication!

`--archiveDurationDeltaCache / -A` Duration after which already in backupCopy copied and linked backups will be deleted. This affects all series in deltacaches processedBackups directory. The duration has to be specified in format 'dhms', e.g. `10d4h` means 10 days and 4 hours, default is `99d` (the format is similar to option `keepAll` in `storeBackupDel.pl`)

`--dontDelInDeltaCache` Do not delete any backup in deltaCache

`--tmpdir / -T /tmpdir` Directory for temporary files, the default value is picked from the environment variable `$TMPDIR`. If it does not exist, `/tmp` is set as the default value.

`--progressReport` Print a progress report:
after each *number* files when compressing
after each *number * 1000* files when linking
after each *number * 10000* files when performing chmod

`--logFile / -l` Name of the log file. Default is stdout.

`--suppressTime` Suppress the output of the actual time in the log file.

`--maxFilelen / -m` Maximal size of a log file. After reaching this size, the log file will be rotated (see option noOfOldFiles) or compressed (see option saveLogs).

`--noOfOldFiles / -n` Number of old rotated log files, default is 5. With default values, it will look like this:

```
 $ ls -l /tmp/storebackup.log*
-rw------- 1 hjc  root  328815 30. Aug 12:12 /tmp/storebackup.log
-rw------- 1 root root 1000087 27. Aug 21:18 /tmp/storebackup.log.1
-rw------- 1 root root 1000038 20. Aug 19:02 /tmp/storebackup.log.2
-rw------- 1 root root 1000094 11. Aug 18:51 /tmp/storebackup.log.3
-rw------- 1 root root 1000147 11. Aug 18:49 /tmp/storebackup.log.4
-rw------- 1 root root 1000030 11. Aug 18:49 /tmp/storebackup.log.5
```

Older log files than *.5 have been deleted automatically.

[14] process ID

`--saveLogs` Save the log files with a time and date stamp instead of deleting them after rotating. (Setting this option overwrites the default value of option noOfOldFiles.)

`--compressWith` Specifies the program to compress the saved log files (e.g., with `gzip -9`). Default value is `bzip2`.
On the command line, the parameter to this option is parsed like a line in the configuration file and normally has to be quoted.

`--genBackupBaseTreeConf` If you want to replicate your backups, you can use this option to generate one of the required configuration file as a template. See replication of backups, section 7.8 for details.

`--genDeltaCacheConf` If you want to replicate your backups, you can use this option to generate one of the required configuration file as a template. See replication of backups, section 7.8 for details.

6.4 storeBackupRecover.pl

Restores the backup tree or parts of the backup tree.

```
storeBackupRecover.pl -r restore [-b root] -t targetDir [--flat]
        [-o] [--tmpdir] [--noHardLinks] [-p number] [-v] [-n]
        [--cpIsGnu] [--noGnuCp] [-s]
```

To restore one file or a small number of files, the easiest way is to use cp or a file system browser. This tool is intended to restore (and if necessary uncompress) files. It recreates the backed up data in the same way it was in the original source directory: permissions are set (even if option ignorePerms was set in storeBackup.pl; this option affects only the permissions in the backup tree) and also existing hard links which were in the source tree are reconstructed.

You have to use at least two options: restoreTree and targetDir. StoreBackupRecover.pl only supports command line arguments.

`--restoreTree / -r` Backup tree or part of a backup tree to restore. The easiest way to restore something is to go into the backup directory where the tree you want to restore is located. I now assume its name is `mydir`. Then type:
`# storeBackupRecover.pl -r mydir -t /tmp/myRestorePlace`
where `/tmp/myRestorePlace` is the place where you want that directory and all of its content to be restored (see option targetDir).

`--backupRoot / -b` Normally, there should be no need to use this option! When you restore a directory, storeBackupRecover.pl does this by searching for `.md5CheckSum.info` which is in the root directory of each backup. If it finds more than one of these files it generates an ERROR message. This normally will happen if you make a storeBackup backup of a storeBackup backup and want to restore.
If you get an error message like "found info file a second time . . . ", you need to specify the root of this backup (where you recover with the option restoreTree).

`--targetDir / -t` The directory where you want the recovered files to be stored. Unless you use option flat, storeBackup always restores the complete backup path to the tree you specified with option restoreTree.

`--flat` The directory structure is not restored. All files are stored directly in "targetDir". This is only useful if you recover a small number of files.

`--overwrite / -o` Overwrite existing files (normally not a good idea). It is better to restore in a separate directory and move files around later.

`--tmpdir / -T /tmpdir` Directory for temporary files, default is picked from environment variable `$TMPDIR`. If it does not exist, `/tmp` is set as the default value.

`--noHardLinks` Do not reconstruct hard links in the restore tree, always copy files.

`--noRestoreParallel / -p` Maximal number or parallel started processes to uncompress the files in the backup. Default is 12.
Reduce this number if you are restoring blocked files and the system has insufficient RAM.

`--verbose / -v` Print verbose messages.

`--noRestored / -n` At the end of restoring, print the number of restored dirs, hard links, symbolic links, files, etc.

`--noGnuCp` If you configured storeBackup.pl to use gnucp (option cpIsGnu), so it can backup special files like character devices, then storeBackupRecover.pl reads this information in the backup. But if the computer where you restore the backup has no gnucp installed, you can configure storeBackupRecover.pl not to use cp.
If you made your backup without gnucp, storeBackcupRecover.pl will not use its functionality. There would be no need to do so, because no special files could be backed up.
If you are using GNU/Linux based systems only, it is the best to forget this option.

`--createSparseFiles / -s` Creates sparse files from files backuped as blocked files if full blocks are filled with zeros.

6.5 storeBackupVersions.pl

storeBackupVersions.pl locates different versions of a file saved with storeBackup.pl. This is the right program if you want to see how many different versions of a specific file exist and where a file with a specific content is located anywhere in a backup series.
If you want to make a more "high sophisticated" search depending on file names, sizes, dates or other stuff, have a look at storeBackupSearch, see section 6.6.

```
storeBackupVersions.pl -f file [-b root]  [-v]
 [-l [-a | [-s] [-u] [-g] [-M] [-c] [-m]]]
```

This program only accepts options on the command line. The only option you have to set is `--file`.

`--file / -f` The name (and path) of the file in the backup. Write the file name exactly as it is written in the backup.

`--backupRoot / -b` Normally there should be no need to use this option! When you restore a directory, storeBackupRecover.pl does this by searching for `.md5CheckSum.info` which is in the root directory of each backup. If it finds more than one of these files it generates an ERROR message. This normally will happen if you make a storeBackup backup of a storeBackup backup and want to restore.
If you get an error message like "found info file a second time . . . ", you need to specify the root of this backup (where you recover with option restoreTree).

`--verbose / -v` Print verbose messages.

`--locateSame / -l` Locate files with the same contents in the backup.

`--showAll / -A` same as `-s -u -g -M -c -m`

`--size / -s` show the size (human readable) of the source file

`--uid / -u` show the uid of the source file

`--gid / -g` show the gid of the source file

`--mode / -M` show the permissions of the source file

`--ctime / -c` show the creation time of the source file

`--mtime / -m` show the modify time of the source file

`--atime / -a` show the access time of the source file

6.6 storeBackupSearch.pl

storeBackupSearch.pl allows you to search in specific backups in a backup series or in all backups under backupDir. You can define any rule built from combinations of file name, size, mode (permissions), owner (uid, gid), creation time, modify time and file type – naturally the ones of the original source directory. See section 7.4 how to define rules. It will help you if you have at least some very basic knowledge about scripting or programming.

```
storeBackupSearch.pl -g configFile

storeBackupSearch.pl [-f configFile] [-b backupDirectory]
      [-s rule]  [--absPath] [-w file] [--parJobs number]
      [-d level] [--once] [--print] [-T tmpdir] [backupRoot . . .]
```

This program allows you to set the options on the command line and in a configuration file. You have to set the options backupDir and searchRule.

First, the options which can be used only on command line. There is always a long option (like `--file`) and sometimes also a shortcut (`-f`):

`--generate / -g` Generate a template for a configuration file. After generation, you can edit it with the editor of your choice. It is easier to write rules in the configuration file, because on the command line the shell strips quotes.

`--print` Prints the options used (from command line *and* from the configuration file) and stops after printing the options. In case of difficult quoting (especially on the command line) this gives you the chance to see what's really used in the program.

`--file / -f` Name of the configuration file you want to use.

The following options can be used on the command line and in the configuration file (see section 7.1). There is a long option for the command line (like `--searchRule`), sometimes also a shortcut for the command line (like `-s`) and the name of the term used in the configuration file (like `searchRule`)).

`--backupDir / backupDir` The repository / backup where you want backups to search. You can set this option to your whole backup repository, to a series or to a single backup.

`--searchRule / -s / searchRule` rule for searching, see section 7.4 for how to define rules.

`--absPath / -a / absPath` the files found will be printed with absolute path names

`--writeToFile / -w / writeToFile` write the result of the search to the specified file, default is stdout

`--parJobs / -p / parJobs` Maximum number of parallel search operations. The default value is chosen automatically as the number of cores plus 1.

`--debug / -d / debug` debug level, possible values are 0, 1, 2; default = 0

`--once / -o / once` show every file found only once (depending on the contence respectively the md5 sum of each file)

`--tmpdir / -T /tmpdir` Directory for temporary files, the default value is picked from the environment variable `$TMPDIR`. If it does not exist, `/tmp` is set as the default value.

`...backupRoot... / backupRoot` On the command line, this is not an option but a list parameter. So you have to write on the command line e.g.:

```
# storeBackupSearch.pl <all_options> 2008.08.27_16.59.01 2008.08.30_10.13.38
```

In the configuration file this is similar to:
`backupRoot = 2008.08.27_16.59.01 2008.08.30_10.13.38`
You can define a *relative path* to directories below backupDir where to search. This can be a specific backup itself (like in this example), a whole backup series or a directory in which directories with backup series are stored (and so on). You can configure storeBackupSearch.pl to search in multiple directories.
If you do not specify any directory, then all backups below backupDir are used for the search.
You need read permissions for the `.md5CheckSum.*`-files in the backups.

6.7 storeBackupSetupIsolatedMode.pl

storeBackupSetupIsolatedMode.pl is part of the isolated mode / offline backup functionality of storeBackup. It copies the meta data of the last backup from a series of backups to another filesystem (e.g., on a memory stick or small hard disk). It optionally generates a customized version of the `storeBackup.pl` configuration file.
This can be used to generate incremental backups with `storeBackup.pl` on media with low capacity, e.g., on a memory stick during a travel – without having access to the central backup repository. Later, you can integrate your local backups into the central backup repository by using `storeBackupMergeIsolatedBackup.pl`[15] to replicate your data and `storeBackupUpdateBackup.pl`[16] to complete your incremental backups to full space-efficient backups.

For a general description about how to use isolated mode, have a look at isolated mode / offline backups, see chapter 7.7.
Based on configuration file:

```
storeBackupSetupIsolatedMode.pl -f configFile -t targetDir
                               [-S series] [-g newConfigFile]
                               [-e explicitBackup] [-v] [-F]
```

without configuration file:

```
storeBackupSetupIsolatedMode.pl -b backupDir -t targetDir
                               [-S series] [-e explicitBackup] [-v] [-F]
```

`--existingConfigFile / -f` original configuration file from `storeBackup.pl` used for normal backups on local media. In the configuration file, the letters `#` and `;` are used as comment signs. The customization of the configuration will work correct only if the `;` is used before unused keywords (like in the generated original version from `storeBackup.pl`)!
`storeBackupSetupIsolatedMode.pl` will generate a new key in the newly generated configuration file (see option `--generate`) called `mergeBackupDir=` which points to the original backupDir. This entry is ignored by `storeBackup.pl` and used by `storeBackupMergeIsolatedBackup.pl` to copy the incremental backups from the local media to the master backup repository.
If you want to setup a new isolated backup (after some time) with the same configuration, you can also use the already generated configuration files as the only parameter. This program will then use your old configuration to copy the meta data from the last existing backup to a (clean) stick.

`--targetDir / -t` The new top level backup directory to use by `storeBackup.pl` on the local media for the incremental backups. This directory must exist.

`--backupDir / -b` If you do not want to use a `storeBackup.pl` configuration file, you can specify the path to your master backup repository with this option.

`--series / -S` If more than one series in stored in `backupDir` directory, you have to specify the series you want to use for your local media.

`--generate / -g` Specify the name for the configuration file to generate when the option `--existingConfigFile` is used. If no name is specified, the name for the new configuration file to make backups on the new media will be *isolate–* plus the name specified at option `--existingConfigFile`.

`--explicitBackup / -e` Specify the exact backup which has to be copied. Default value is the last backup of the series chosen.

`--verbose / -v` Generate verbose messages.

`--force / -F` Force usage of last backup (with option `lateLinks` of `storeBackup.pl`), even if that last backup of that series has not been completed to a full backup with `storeBackupUpdateBackup.pl`.

[15]see storeBackupMergeIsolatedBackup.pl
[16]see storeBackupUpdateBackup.pl

6.8 storeBackupMergeIsolatedBackup.pl

storeBackupMergeIsolatedBackup.pl is part of the isolated mode / offline backups functionality of storeBackup. It copies the incremental backups made in isolated mode (see isolated mode / offline backups, chapter 7.7) and the description of storeBackupSetupIsolatedMode, see chapter 6.7 back to the central backup repository.

```
based on configuration file:

storeBackupMergeIsolatedBackup.pl -f isolateConfigFile [-v] [--force]
                                  [-T tmpdir]

no configuration file:

    storeBackupSetupIsolatedMode.pl -i isolateBackupDir -b backupDir
         [-S series] [-v] [--force] [-T tmpdir]
```

`--configFile / -f` The isolated mode config file which contains the new key called `mergeBackupDir=` which points to the original backupDir.

`--force` Force copying of files without listing backups to copy and prompting for acceptance

`--isolateBackupDir / -b` Specifies the backup directory (on local media) where the isolated backups were made

`--series / -S` If more than one series is stored in `backupDir` directory, you have to specify series you want to use for your local media.

`--verbose / -v` Generate verbose messages.

`--tmpdir / -T /tmpdir` Directory for temporary files, the default value is picked from the environment variable `$TMPDIR`. If it does not exist, `/tmp` is set as the default value.

6.9 storeBackupls.pl

storeBackupls.pl gives you information about the age and deletion rules of a backup series.

```
    storeBackupls.pl -f configFile [--print] [storeBackup-dir]
    storeBackupls.pl [-v] [--print] storeBackup-dir
```

There are two possible styles to call it (with examples):

```
# /opt/test/storeBackup/bin/storeBackupls.pl .
  1  Fri Jul 04 2008   2008.07.04_20.17.13   -61
  2  Sat Jul 05 2008   2008.07.05_21.19.09   -60
  3  Sun Jul 06 2008   2008.07.06_17.38.22   -59
  4  Mon Jul 07 2008   2008.07.07_17.31.43   -58
  5  Fri Jul 11 2008   2008.07.11_19.20.14   -54
  6  Sat Jul 12 2008   2008.07.12_18.17.21   -53
  7  Sun Jul 13 2008   2008.07.13_17.07.53   -52
  8  Mon Jul 14 2008   2008.07.14_06.28.29   -51
  9  Tue Jul 15 2008   2008.07.15_07.44.41   -50
 10  Wed Jul 16 2008   2008.07.16_17.56.35   -49
 11  Thu Jul 17 2008   2008.07.17_10.13.47   -48
 12  Fri Jul 18 2008   2008.07.18_14.13.26   -47
 13  Sat Jul 19 2008   2008.07.19_16.03.40   -46
 14  Fri Jul 25 2008   2008.07.25_09.29.39   -40
 15  Mon Jul 28 2008   2008.07.28_19.01.04   -37
 16  Wed Jul 30 2008   2008.07.30_17.25.43   -35
 17  Thu Jul 31 2008   2008.07.31_16.45.56   -34
 18  Fri Aug 01 2008   2008.08.01_16.43.56   -33
 19  Mon Aug 04 2008   2008.08.04_17.26.42   -30
 20  Thu Aug 07 2008   2008.08.07_16.16.21   -27
 21  Fri Aug 08 2008   2008.08.08_20.59.46   -26
 22  Sat Aug 09 2008   2008.08.09_20.48.31   -25
 23  Sun Aug 10 2008   2008.08.10_14.29.18   -24
 24  Mon Aug 11 2008   2008.08.11_19.51.32   -23
 25  Tue Aug 12 2008   2008.08.12_14.13.02   -22
 26  Wed Aug 13 2008   2008.08.13_20.41.43   -21
 27  Thu Aug 14 2008   2008.08.14_16.44.02   -20
 28  Fri Aug 15 2008   2008.08.15_19.47.29   -19
 29  Mon Aug 18 2008   2008.08.18_18.29.06   -16
 30  Tue Aug 19 2008   2008.08.19_17.58.42   -15
 31  Wed Aug 20 2008   2008.08.20_18.53.46   -14
 32  Thu Aug 21 2008   2008.08.21_19.56.03   -13
 33  Fri Aug 22 2008   2008.08.22_23.32.10   -12
 34  Sun Aug 24 2008   2008.08.24_12.57.36   -10
 35  Tue Aug 26 2008   2008.08.26_10.34.06   -8  not finished
 36  Tue Aug 26 2008   2008.08.26_10.59.46   -8
 37  Tue Aug 26 2008   2008.08.26_13.07.08   -8
```

You see that backup number 35 was not finished. Using option verbose results in:

```
# /opt/test/storeBackup/bin/storeBackupls.pl -v .
. . .
 37  Tue Aug 26 2008   2008.08.26_13.07.08   -8
version -> 1.3
date -> 2008.08.26 13.07.08
sourceDir -> '/backup'
followLinks -> 1
compress -> 'bzip2'
uncompress -> 'bzip2' '-d'
postfix -> '.bz2'
exceptSuffix -> '.bz2' '.gif' '.gpg' '.gz' '.jpg' '.mp3' '.mpeg' '.mpg' '.ogg' '.png' '.tgz' '.tif' '.tiff' '.zip'
exceptDirs -> '/backup/home_hjc/nosave'
includeDirs ->
exceptRule -> '$file =~ /acronis.*tib/' 'or' '$file =~ m#/te?mp/#' 'or' '$file =~ m#/\.thumbnails/#'
includeRule ->
exceptTypes ->
preservePerms -> yes
lateLinks -> yes
lateCompress -> no
cpIsGnu -> yes
```

Only the output for the last backup is shown here. You can see which options of storeBackup.pl where used to generate the backup.

`storeBackupls.pl -f configFile [--print] [storeBackup-dir]` or
`storeBackupls.pl --file configFile [--print] [storeBackup-dir]`

Here it will read the configuration file of storeBackup.pl and tells something about the deletion status of each backup:

```
# storeBackupls.pl -f /backup/stbu-gentoo.conf .
. . .
WARNING   backup <./2008.08.26_10.34.06> not finished
analysis of old Backups in <.>:
   Fri 2008.07.04_20.17.13 (61): will be deleted
   Sat 2008.07.05_21.19.09 (60): keepWeekDays(60d)
   Sun 2008.07.06_17.38.22 (59): keepWeekDays(60d)
   Mon 2008.07.07_17.31.43 (58): keepWeekDays(60d)
   Fri 2008.07.11_19.20.14 (54): keepWeekDays(60d)
   Sat 2008.07.12_18.17.21 (53): keepMinNumber30, keepWeekDays(60d)
   Sun 2008.07.13_17.07.53 (52): keepMinNumber29, keepWeekDays(60d)
   Mon 2008.07.14_06.28.29 (51): keepMinNumber28, keepWeekDays(60d)
   Tue 2008.07.15_07.44.41 (50): keepMinNumber27, keepWeekDays(60d)
   Wed 2008.07.16_17.56.35 (49): keepMinNumber26, keepWeekDays(60d)
   Thu 2008.07.17_10.13.47 (48): keepMinNumber25, keepWeekDays(60d)
   Fri 2008.07.18_14.13.26 (47): keepMinNumber24, keepWeekDays(60d)
   Sat 2008.07.19_16.03.40 (46): keepMinNumber23, keepWeekDays(60d)
   Fri 2008.07.25_09.29.39 (40): keepMinNumber22, keepWeekDays(60d)
   Mon 2008.07.28_19.01.04 (37): keepMinNumber21, keepWeekDays(60d)
   Wed 2008.07.30_17.25.43 (35): keepMinNumber20, keepWeekDays(60d)
   Thu 2008.07.31_16.45.56 (34): keepMinNumber19, keepWeekDays(60d)
   Fri 2008.08.01_16.43.56 (33): keepMinNumber18, keepWeekDays(60d)
   Mon 2008.08.04_17.26.42 (30): keepMinNumber17, keepWeekDays(60d)
   Thu 2008.08.07_16.16.21 (27): keepMinNumber16, keepWeekDays(60d)
   Fri 2008.08.08_20.59.46 (26): keepMinNumber15, keepWeekDays(60d)
   Sat 2008.08.09_20.48.31 (25): keepMinNumber14, keepWeekDays(60d)
   Sun 2008.08.10_14.29.18 (24): keepMinNumber13, keepWeekDays(60d)
   Mon 2008.08.11_19.51.32 (23): keepMinNumber12, keepWeekDays(60d)
   Tue 2008.08.12_14.13.02 (22): keepMinNumber11, keepWeekDays(60d)
   Wed 2008.08.13_20.41.43 (21): keepMinNumber10, keepWeekDays(60d)
   Thu 2008.08.14_16.44.02 (20): keepMinNumber9, keepWeekDays(60d)
   Fri 2008.08.15_19.47.29 (19): keepMinNumber8, keepWeekDays(60d)
   Mon 2008.08.18_18.29.06 (16): keepMinNumber7, keepWeekDays(60d)
   Tue 2008.08.19_17.58.42 (15): keepMinNumber6, keepWeekDays(60d)
   Wed 2008.08.20_18.53.46 (14): keepMinNumber5, keepWeekDays(60d)
   Thu 2008.08.21_19.56.03 (13): keepMinNumber4, keepWeekDays(60d)
   Fri 2008.08.22_23.32.10 (12): keepMinNumber3, keepWeekDays(60d)
   Sun 2008.08.24_12.57.36 (10): keepMinNumber2, keepWeekDays(60d)
   Tue 2008.08.26_10.59.46 (8): will be deleted
   Tue 2008.08.26_13.07.08 (8): keepMinNumber1, keepWeekDays(60d)
```

. . . and using the option `--print` we see the parameters used from `storeBackup.pl`:

```
# storeBackupls.pl -f /backup/stbu-gentoo.conf . --print
combined configuration and command line options
options with parameters:
file </backup/stbu-gentoo.conf>
firstDayOfWeek <Sun>
keepAll <60d>
keepDuplicate <7d>
keepFirstOfMonth <undef>
keepFirstOfWeek <undef>
keepFirstOfYear <undef>
keepLastOfMonth <undef>
keepLastOfWeek <undef>
keepLastOfYear <undef>
keepMaxNumber <0>
keepMinNumber <30>
keepRelative <undef>
keepWeekday <undef>
options without parameters:
list parameters:
<.>
```

6.10 storeBackupDel.pl

storeBackupDel.pl deletes old backups according to the rules you defined. These rules (keep∗) are described as options of storeBackup.pl in section 6.2. You should configure storeBackup.pl with a configuration file and read this configuration file with storeBackupDel.pl if you want to delete old backups asynchronously from the backup itself (see also Example 6, section 9.7).

```
      $prog [-f configFile] [--print]
       [-b backupDirectory] [-S series] [--doNotDelete]
       [--deleteNotFinishedDirs] [-L lockFile]
       [--keepAll timePeriod] [--keepWeekday entry] [--keepFirstOfYear]
       [--keepLastOfYear] [--keepFirstOfMonth] [--keepLastOfMonth]
       [--keepFirstOfWeek] [--keepLastOfWeek]
       [--keepDuplicate] [--keepMinNumber] [--keepMaxNumber]
       [-l logFile
        [--plusLogStdout] [--suppressTime] [-m maxFilelen]
        [[-n noOfOldFiles] | [--saveLogs]
        [--compressWith compressprog]]
```

You have to set at least two options: `--backupDir` *and* `--series`*. It doesn't matter if you set them on the command line, in the configuration file or mixed.*

First the options which can be used only on command line. There is always a long option (like `--configFile`) and sometimes also a shortcut (`-f`).

`--print` Prints the options used (from command line *and* from the configuration file) and stops after printing the options. In case of difficult quoting (especially on the command line) this gives you the chance to see what's really used in the program.

`--configFile / -f` Name of the configuration file you want to use.

You can easily overwrite the options saved in the configuration file (especially changing backupDir and unsetting doNotDelete) on the command line. See also section 7.1.
The following options are identical to the ones in storeBackup.pl:

- backupDir
- series
- lockFile
- doNotDelete
- deleteNotFinishedDirs
- keepAll
- keepWeekday
- keepFirstOfYear
- keepLastOfYear
- keepFirstOfMonth
- keepLastOfMonth
- firstDayOfWeek
- keepFirstOfWeek
- keepLastOfWeek
- keepDuplicate
- keepMinNumber
- keepMaxNumber
- keepRelative
- logFile
- plusLogStdout
- suppressTime
- maxFilelen
- noOfOldFiles
- saveLogs
- compressWith

6.11 storeBackupMount.pl

`storeBackupMount.pl` gives you a "script" to mount the needed directories for the backup, start storeBackup.pl and umount the directories. Before trying to mount, it can check via ping if a server is reachable. If these directories are always (or already) mounted, there is no need to use `storeBackupMount.pl`.
It also can run other programs like `storeBackupUpdateBackup.pl` or `storeBackupDel.pl`.
There are different ways to use `storeBackupMount.pl`:

```
        storeBackupMount.pl --help
or
        storeBackupMount.pl -g configFile
or
        storeBackupMount.pl -f configFile
or
        storeBackupMount.pl [-s servers] [-d] [-l logFile
               [--suppressTime] [-m maxFilelen]
               [[-n noOfOldFiles] | [--saveLogs]]
               [--compressWith compressprog]]
            [--storeBackup storeBackup-Params]
            [--storeBackupUpdateBackup storeBackupUpdateBackup-Params]
            [--storeBackupCheckBackup storeBackupCheckBackup-Params]
            [--storeBackupCheckSource storeBackupCheckSource-Params]
            [--storeBackupDel storeBackupDel-Params]
            [--printAndStop] [-k killTime] [-T tmpdir] [mountPoints...]
```

You can generate a configuration file (option `-g`), use a configuration file (option `-f`) or use it by making your settings via command line.

To be able to mount the directories, you need an entry in `/etc/fstab` like the following ones:

```
/dev/sda5 /add reiserfs noatime 0 1
lotte:/disk1 /backup nfs rsize=8192,wsize=8192,user,exec,async,noatime 1 1
```

The first mount point `/add` is on a local device. In this example, it's the device with a file system to be saved. The second one (`/backup`) is located on `/disk1` on the nfs server lotte. The rest are nfs parameters – see section 7.10 about the configuration of nfs.
If you have this kind of entries in `/etc/fstab`, you can mount the file systems with:
`mount /add`
`mount /backup`
and that's exactly what `storeBackupMount.pl` does.

There is one major difference in using the command line or the configuration file to set the options to your needs: As you can define the start of multiple storeBackup related programs, you also have to define the order in which they have to be started. You have to do this:

- on the command line by sorting options like `--storeBackup` or `--storeBackupDel` in same order you want them to be called
- in the configuration file by listing the programs you want to call via keyword `orderOfExecution`.

`storeBackupMount.pl` will stop execution if a program started by it does not start successfully.

You can set the following options:

`--help` Print options and stop processing.

`--printAndStop` Print a summary of the command line options.

`--generate / -g` Specify the name of the configuration file to be generated. This option is available on the command line only.

`--file` / **-f** Specify the name of the configuration file to be used. This option is available on the command line only.

`--servers / -s / servers` List of servers (name or ip address) to be checked via `ping`. This option can be repeated multiple times on the command line.

`--debug / -d` Generate some extra messages.

`--logFile / -l` Log file for this process, default is stdout.

`--suppressTime / suppressTime` Suppress the output of the actual time in the log file.

`--maxFilelen / -m / maxFilelen` Maximal size of a log file. After reaching this size, the log file will be rotated (see option noOfOldFiles) or compressed (see option saveLogs).

`--noOfOldFiles / -n / noOfOldFiles` Number of old rotated log files, default is 5. With default values, it will look like this:

```
$ ls -l /tmp/storebackup.log*
-rw------- 1 hjc  root  328815 30. Aug 12:12 /tmp/storebackup.log
-rw------- 1 root root 1000087 27. Aug 21:18 /tmp/storebackup.log.1
-rw------- 1 root root 1000038 20. Aug 19:02 /tmp/storebackup.log.2
-rw------- 1 root root 1000094 11. Aug 18:51 /tmp/storebackup.log.3
-rw------- 1 root root 1000147 11. Aug 18:49 /tmp/storebackup.log.4
-rw------- 1 root root 1000030 11. Aug 18:49 /tmp/storebackup.log.5
```

Older log files than *.5 have been deleted automatically.

`--saveLogs / saveLogs` Save the log files with a time and date stamp instead of deleting them after rotating. (Setting this option deactivates option noOfOldFiles.)

`--compressWith / compressWith` Specifies the program to compress the saved log files (e.g., with `gzip -9`). Default value is `bzip2`.
On the command line, the parameter to this option is parsed like a line in the configuration file and normally has to be quoted on the command line.

`--storeBackup / storeBackup` Defines that `storeBackup.pl` has to be started and specifies the options/parameters for it. To be able to map this *one* parameter to multiples for `storeBackup.pl`, the quotes on the command line are stripped and you parameter to this option is parsed again.
Example: you set `"-f /backup/stbu.conf"` on the command line, then `storeBackup.pl` will be called with the two parameters `-f` and `/backup/stbu.conf`.
If you use the configuration file, simply set:
`storeBackup = -f /backup/stbu.conf`
to get the same result. If this all sounds strange to you, should read chapter chapter 7.1.

`--storeBackupUpdateBackup / storeBackupUpdateBackup` Defines `storeBackupUpdateBackup.pl` to be called and set its parameters. See option `--storeBackup` for details.

`--storeBackupCheckBackup / storeBackupCheckBackup` Defines `storeBackupCheckBackup.pl` to be called and set its parameters. See option `--storeBackup` for details.

`--storeBackupCheckSource / storeBackupCheckSource` Defines `storeBackupCheckSource.pl` to be called and set its parameters. See option `--storeBackup` for details.

`--storeBackupDel / storeBackupDel` Defines `storeBackupDel.pl` to be called and set its parameters. See option `--storeBackup` for details.

`--killTime / -k / killTime` Time until any of the started programs will be killed. This means each individual program can run up to `killTime`. The format of this option is `dhms`, e.g. `10d4h` means 10 days and 4 hours. See the `keep*` options of `storeBackup.pl` for further examples. The default value is 365 days.

`--tmpdir / -T /tmpdir` Directory for temporary files, the default value is picked from the environment variable `$TMPDIR`. If it does not exist, `/tmp` is set as the default value.

`...mountPoints... / mountPoints` On the command line, this is not an option; it is a list parameter. So you have to write e.g.:

```
# storeBackupMount.pl <all_options> /backupDisk /otherDisk
```

In the configuration file this is similar to:
`mountPoints = /backupDisk /otherDisk`
Here you specify the list of mount points needed to perform the backup. This must be a list of paths which have to be defined in `/etc/fstab`. If you add `ro` or `rw` to the beginning of a mount point, you can overwrite that option in /etc/fstab. Example:

```
ro,/fileSystemToRead
```

will mount `/fileSystemToRead` read only, even if the corresponding entry in /etc/fstab is read/write. But only root is allowed to use this feature!

6.12 storeBackupCheckBackup.pl

If you copy data from one disk to another, you might have unrecognized bit failures (e.g. in main memory, in the cpu, on the transfer lines) resulting in wrong data stored in your backup. This may happen especially over long time periods on the backup disk itself. Because storeBackup also stores check sums in addition to the real data it is possible to compare the stored data in the backup with the check sums generated from the source files to give you a better probability that your backup data is not broken.[17] But don't panic, this doesn't happen all the time.
`storeBackupCheckBackup.pl` Verifies the consistency of one or more backups by using the md5 sums generated during the backup in `.md5CheckSums` (see section 7.9) and compares them with just calculated ones. It also checks all files in the backup and the files considered in `.md5CheckSums` for existence or non-existence.

```
storeBackupCheckBackup.pl -c backupDir [-p number] [-i]
      [-w filePrefix] [--lastOfEachSeries]
      [--includeRenamedBackups] [-T tmpdir]
      [--logFile
       [--plusLogStdout] [--suppressTime] [-m maxFilelen]
       [[-n noOfOldFiles] | [--saveLogs]]
```

`--print` Print the configuration parameters and stop processing

`--checkDir / -c` The repository/backup, where you want backups to search. You can set this option to your whole backup repository, to a series or to a single backup.

`--wrongFileTables / -w` Write file names with detected faults in line based regular files for later (not automated) bug fixing. The parameter for this option is a file prefix, e.g. if the file prefix is `/tmp/bugsB-` the following files are generated:
`/tmp/bugsB-files.missing.txt`
`/tmp/bugsB-md5sums.missing.txt`
`/tmp/bugsB-md5sums.wrong.txt`

`--lastOfEachSeries` Only check the last backup of each series found.

`--verbose / -v` Verbose. Print some extra messages so you see what's happening.

`--parJobs / -p` Maximum number of parallel search operations. The default value is chosen automatically as the number of cores plus 1.

`--includeRenamedBackups / -i` By using this option, renamed backups are checked also If you have renamed backups specified with option `--checkDir`. Renamed backups have to follow the scheme `backupDir−something`, e.g., `2012.01.30_15.21.03−renamed`. Also see section 7.3.

[17] This is not a 100% safety. It is still possible that e.g. a memory fault corrupts reading of the source data, so the check sum *and* the data itself are stored corrupted. In this case storeBackupCheckSource.pl, section 6.13 increases your safety. But in general, the only protection against those issues is completely redundant hardware which is *very* expensive.

`--tmpdir / -T /tmpdir` Directory for temporary files, the default value is picked from the environment variable `$TMPDIR`. If it does not exist, `/tmp` is set as the default value.

`--logFile / -l` Name of the log file. Default is stdout.

`--plusLogStdout / plusLogStdout` If option logFile is set, you can here configure storeBackup.pl to print the log messages to stdout.

`--suppressTime` Suppress the output of the actual time in the log file.

`--maxFilelen / -m` Maximal size of a log file. After reaching this size, the log file will be rotated (see option noOfOldFiles) or compressed (see option saveLogs).

`--noOfOldFiles / -n` Number of old rotated log files, default is 5. With default values, it will look like this:

```
$ ls -l /tmp/storebackup.log*
-rw------- 1 hjc  root  328815 30. Aug 12:12 /tmp/storebackup.log
-rw------- 1 root root 1000087 27. Aug 21:18 /tmp/storebackup.log.1
-rw------- 1 root root 1000038 20. Aug 19:02 /tmp/storebackup.log.2
-rw------- 1 root root 1000094 11. Aug 18:51 /tmp/storebackup.log.3
-rw------- 1 root root 1000147 11. Aug 18:49 /tmp/storebackup.log.4
-rw------- 1 root root 1000030 11. Aug 18:49 /tmp/storebackup.log.5
```

Older log files than *.5 have been deleted automatically.

`--saveLogs` Save the log files with a time and date stamp instead of deleting them after rotating. (Setting this option overwrites the default value of option noOfOldFiles.)

`--compressWith` Specifies the program to compress the saved log files (e.g., with `gzip -9`). Default value is `bzip2`.
On the command line, the parameter to this option is parsed like a line in the configuration file and normally has to be quoted.

6.13 storeBackupCheckSource.pl

If you copy data from one disk to another, you might have unrecognized bit failures (e.g. in main memory, in the cpu, on the transfer lines) resulting in wrong data stored in your backup. This may happen especially over long time periods on the backup disk itself. Because storeBackup also stores check sums in addition to the real data it's possible to compare the stored data in the backup with the check sums generated from the source files to give you a better probability that your backup data is not broken. This is the reason, why `storeBackupCheckBackup.pl` exists.
But such a bit error may also occur in your source data after a long period of time. With this program, you can check files not changed since they were backed up by comparing the check sum in the backup with the source file.

```
storeBackupCheckSource.pl -s sourceDir -b singleBackupDir [-v]
          [-w filePrefix]
          [--logFile
           [--plusLogStdout] [--suppressTime] [-m maxFilelen]
           [[-n noOfOldFiles] | [--saveLogs]]
```

`--sourceDir / -s` This must be the directory you specified as parameter to option `sourceDir` to `storeBackup.pl` when you were running your backup you want to compare with.

`--singleBackupDir / -b` The backup directory you want to use for the comparison. This must be *one single backup directory*.

`--wrongFileTables / -w` When using this option, `storeBackupCheckSource.pl` will write two files beginning with the prefix defined here. If you define this option e.g. as */tmp/faults-* then the files */tmp/faults-*`md5sums.missing.txt` and */tmp/faults-*`md5sums.wrong.txt` will be written. They contain lines with

the names of the files where the md5 sum is missing or wrong (different[18] to `sourceDir`) in the backup. These files are written additionally to error messages and may be used easily to further inspect or repair the backup.
In the files written a \ (backslash) is written with the literals `\5C` and a `\n` (linefeed) with the literals `\0A`.

`--verbose / -v` Verbose. Print additional information:

- always printed: `ERROR` message if the md5 sum differs
- always printed: `WARNING` if permissions, uid or gid differs
- printed with option `-v` only: `MISSING` if file is not in the `source` directory
- printed with option `-v` only: `INFO` if file is identical

`--logFile / -l` Name of the log file. Default is stdout.

`--plusLogStdout / plusLogStdout` If the option logFile is set, you can here configure storeBackup.pl to print the log messages to stdout, too.

`--suppressTime` Suppress the output of the actual time in the log file.

`--maxFilelen / -m` Maximal size of a log file. After reaching this size, the log file will be rotated (see option noOfOldFiles) or compressed (see option saveLogs).

`--noOfOldFiles / -n` Number of old rotated log files, default is 5. With default values, it will look like this:

```
$ ls -l /tmp/storebackup.log*
-rw------- 1 hjc  root  328815 30. Aug 12:12 /tmp/storebackup.log
-rw------- 1 root root 1000087 27. Aug 21:18 /tmp/storebackup.log.1
-rw------- 1 root root 1000038 20. Aug 19:02 /tmp/storebackup.log.2
-rw------- 1 root root 1000094 11. Aug 18:51 /tmp/storebackup.log.3
-rw------- 1 root root 1000147 11. Aug 18:49 /tmp/storebackup.log.4
-rw------- 1 root root 1000030 11. Aug 18:49 /tmp/storebackup.log.5
```

Older log files than *.5 have been deleted automatically.

`--saveLogs` Save the log files with a time and date stamp instead of deleting them after rotating. (Setting this option overwrites the default value of option noOfOldFiles.)

`--compressWith` Specifies the program to compress the saved log files (e.g., with `gzip -9`). Default value is `bzip2`.
On the command line, the parameter for this option is parsed like a line in the configuration file and normally has to be quoted.

6.14 storeBackup_du.pl

storeBackup_du.pl evaluates the disk usage in one or more backup directories. `sumLocal` shows the data local in the specified backup(s) and `sumShared` the data shared with other backups via hard links.

```
storeBackup_du.pl [-v] [-l] backupdirs ...
```

`--verbose / -v` Print accumulated values for multiple versions (days) of backed up files.

`--links / -l` Also print statistic about how many links the files have and how much space this saves.

`...backupdirs...` the backup directories to evaluate

[18]This means that either the md5 sum in the backup meta data is wrong or that you had to bit rot somewhere. (Naturally, it may also be that the data is wrong for any other reason, like faults in the RAM of your box or a fault in storeBackup.)

6.15 storeBackupConvertBackup.pl

You only have to call this program when storeBackup.pl tells you to do so.
This program converts old backups created with storeBackup.pl to the newest version. The current version of the backup format is 1.3.
You can see the version by typing:

```
head -1 < ...<storeBackupDir>/date_time/.md5CheckSums.info
```

Call storeBackupConvertBackup.pl with the backup directories to convert:

```
    storeBackupConvertBackup.pl storeBackup-dir
```

6.16 linkToDirs.pl

Makes a de-duplicated copy of files in defined directories to another location. Utilizes hard links to the full extent possible to avoid wasting storage space.
`linkToDirs.pl` is a general purpose tool. However, it is very helpful if you want to copy a backup made with storeBackup to another disk.
Usage note: whereas many file copy utilities have just two primary parameters (the source and destination), `linkToDirs.pl` allows three primary parameters:

- source(s),
- destination
- and a reference location(s).

The reference location is the place to look for existing content which can be hard linked to (see `--linkWith` option).
The `--linkWith` option is not required. If you use it, you can optionally specify multiple link references for hard linking (i.e., the `--linkWith` option can be repeated).
Files with the same content as the specified link reference(s) and on the same file system will be hard linked. Hard links within the copied files will be maintained or re-created: `linkToDirs.pl` will always hard link identical files, with one exception. That exception is: files in the directories specified by `--linkWith` will never be changed. So if there are two identical files which are not hard linked, they will remain that way (unlinked). `linkToDirs.pl` supports hard linking of symbolic links with at least as much capability as the main storeBackup.pl program does.
(Naturally, if there are no identical files, it will only copy files.)
Hard links on Linux have these rules:

- Hard links cannot link directories.
- Hard links cannot cross file system boundaries.

If it is not possible to create a hard link to the reference file (due to the limitations of hard links) `linkToDirs.pl` will generate a new file copy (on the target file system) and then hard link to that one going forward. In this way, `linkToDirs.pl` can be used to maintain a de-duplicated state of source files when copying them to another filesystem.
`linkToDirs.pl` is a general purpose tool. However, it has a special synergy with storeBackup. As you know, storeBackup eliminates wasted space in the storage location by maintaining a de-duplicated state through the use of hard links (even if the target filesystem supports less hard links per file than the source filesystem). But hard links cannot be maintained across different file systems.
Therefore, when you want to copy an existing backup made with storeBackup to a new disk (or new file system), `linkToDirs.pl` allows you to do so and to maintain all the storage efficiency benefits of the original storeBackup backup.

```
            linkToDirs.pl [--linkWith copyBackupDir] [--linkWith ...]
                          --targetDir targetForSourceDir
                          [--progressReport number[,timeframe]]
                           [--printDepth] [--dontLinkSymlinks]
```

```
                    [--ignoreErrors] [--saveRAM] [-T tmpdir]
                    [--createSparseFiles [--blockSize]]
                    sourceDir ...
```

`--help / -h` Print a help message

`--linkWith / -w` The reference location; consider the files in these directories for hard linking. This option can be repeated. (The directories are recursed, as you would expect.)

`--targetDir / -t` The destination; files from `sourceDir`s will be copied to this directory.

`--dontLinkSymlinks` Do not hard link identical symbolic links (symlinks). The default is to hard link each existing symlink rather than copy the symlink.

`--progressReport / -P / progressReport` Print a progress report after the specified number of files. If you want to get a message at least after a specific time frame, you may add that time frame separated by a comma, eg:
`-P 1000,1m10s` on the command line or
`progressReport = 1000,1m10s` in the configuration file.
There must be no white space in the parameter to that option. The syntax of the time frame is the same as with the keep∗ options.

`sourceDir` the source directory; files (or existing storeBackup backups) from this directory will be copied to `targetDir`. `sourceDir` may be repeated multiple times with different directories. Normal shell file and directory conventions, including wildcards, are acceptable. Copy functionality is recursive into all subdirectories within the listed `sourceDir`.

`--ignoreErrors` Don't stop copying in case of errors during copying / linking.

`--saveRAM / saveRAM` Use this option if storeBackup.pl runs on a system with very low memory. You will then see some dbm files in "tmpDir". This will slow down storeBackup.pl a little bit, so do this only if you run into problems without it. On modern computers, it should only be necessary to use this option if you copy millions of files.

`--tmpdir / -T /tmpdir` Directory for temporary files, the default value is picked from the environment variable `$TMPDIR`. If it does not exist, `/tmp` is set as the default value.

`--createSparseFiles / -s` A mismatch between block size, number of used blocks for a file and the filesize is used to indicate a (possible) sparse file. If this option is set, `linkToDirs.pl` copies the affected file in case of a possible sparse file with the external program `cp` to support sparse files. On Linux systems (and many others) `gnucp` is installed which supports sparse files – on other systems this option may not work (depending on cp on your system).

`--blockSize` The blocksize to indicate a sparse file. The default value is 512.

6.17 llt

Lists create, access and modification times of files.

```
llt [-r] [-i] [-a|-m|-c] [files] [dirs]
```

`--help / -h` print a help message

`--reverse / -r` sort in reverse order according to file names

`--insensitive / -i` sort case insensitively

`--access / -a` sort according to access time

`--modification / -m` sort according to modification time

`--creation / -c` sort according to creation time

`--unixTime / -u` show Unix time (unsigned integer)

6.18 multiTail.pl

multiTail.pl reads one or multiple log files. The files read can be shown on the screen or written into anther log file and saved. This way, you can mix multiple log files.
It is a very robust, so it doesn't care if a file is deleted, moved or newly created. You can also start it with a file name which does not exist at that time.

```
multiTail.pl [-a] [-d delay] [-p begin|end]
    [--print] [-t] [-o outFile [-m max] [-P]
     [[-n noFiles] | [-s [-c compressprog]] ]
    ]
    [-C color=pattern [-C color=pattern ...]]
    [-g expression] files...
```

All options are optional, you simply have to use one or more log file names as parameter.

`--addName / -a` Add the file name from which the line is read to the output.

`--delay / -d` Delay in seconds between checking each file for new data. The value can by smaller than 1, e.g. `0.2`. The default value is 5 (seconds).

`--position / -p` At start of the program, read from the begin or end of the file; allowed parameters are `begin` or `end`. Default is `begin`.

`--print` Print the options used (from the command line *and* from the configuration file) and stop after printing the options. In case of difficult quoting (especially on the command line) this gives you the chance to see what's really used in the program.

`--withTime -t` Add a time stamp to the output.

`--out / -o` write output to a file, default is stdout

`--maxFilelen / -m` Maximal size of a log file. After reaching this size, the log file will be rotated (see option noOfOldFiles) or compressed (see option saveLogs).

`--withPID / -P` write pid (process id) of multiTail.pl to the log file; default is not to write it

`--maxlines / -l` Maximum number of lines to read in one chunk from a log file. Default is 100. If you configure "`--delay 3`" then every three seconds multiTail.pl will read a maximum of 100 lines. The reason for this restriction is to avoid that multiTail.pl will consume too much power if a log file is written too heavily.

`--noOfOldFiles / -n` Number of old rotated log files, default is 5. With default values, it will look like this:

```
 $ ls -l /tmp/storebackup.log*
-rw------- 1 hjc  root  328815 30. Aug 12:12 /tmp/storebackup.log
-rw------- 1 root root 1000087 27. Aug 21:18 /tmp/storebackup.log.1
-rw------- 1 root root 1000038 20. Aug 19:02 /tmp/storebackup.log.2
-rw------- 1 root root 1000094 11. Aug 18:51 /tmp/storebackup.log.3
-rw------- 1 root root 1000147 11. Aug 18:49 /tmp/storebackup.log.4
-rw------- 1 root root 1000030 11. Aug 18:49 /tmp/storebackup.log.5
```

Older log files than *.5 have been deleted automatically.

`--saveLogs / -s` Save the log files with a time and date stamp instead of deleting them after rotating. (Setting this option overwrites the default value of option noOfOldFiles.)

`--compressWith / -c` Specifies the program to compress the saved log files (e.g., with `gzip -9`). Default value is `bzip2`.
On the command line, the parameter to this option is parsed like a line in the configuration file and normally has to be quoted.

`--color / -C` Filter a line to print it in a special color, e.g. `red=ERROR` means *the whole line* containing `ERROR` will be printed in red. You can use pattern matching for the part of the right side of the "`=`". Supported colors are `red`, `green`, `yellow`, `blue`, `magenta` and `cyan`.

`--grep / -g` Use this pattern to filter lines from the log files. No special color is used for output. If this option is missing, all lines are printed.

EXAMPLE:

```
multiTail.pl -p end -a -d 0.5 -C red=ERROR -C yellow=WARNING -g 'END|ERROR|WARNING' *.log
```

This will show only lines in the log file containing `END`, `ERROR` or `WARNING`. Lines containing `WARNING` will be printed in yellow, lines containing `ERROR` in red on the screen.

7 General concepts

7.1 Configuration file and command line

In all these programs a module is used which can handle the combination of command line and configuration file usage. Because in all programs always the same module is used, this description is valid for all of them. Nevertheless, there are some programs which support both interfaces (like storeBackup.pl) and others which support only command line (like storeBackupMount.pl). In general, programs which support a complex configuration have both interfaces, while the ones with a more simple configuration only support command line.

Configuration file
The structure of the configuration file is:

```
keyword = list of parameters
```

There is no difference in writing:

```
keyword=list   of     parameters
```

If you have too much parameters for an optical nice length of the line in the configuration file, you can continue in the next line if you add one or more white space (space or tab) in the beginning of that line:

```
keyword = list of parameters but now really really really
    too much for one single line
```

You can add comments by setting a hash sign at the beginning of a line:

```
# this is a comment
```

You can also make a comment by typing a semicolon (`;`) at the beginning of a line. For better readability, storebackup uses this when writing (not specified) keywords in configuration files. Sometimes, this approach is used to identify commented keywords. So you shouldn't change this convention.
You can use environment variables like in a shell, here it's `$VAR`:

```
keyword = $VAR ${VAR}var
```

If `$VAR` was set to `XXX`, this will be equal to:

```
keyword = XXX XXXvar
```

You can use quotes:

```
keyword = 1 2 "1 2" '1 2' $VAR "$VAR 1" '$VAR' '$VAR 1'
```

This will be expanded (internally) to (the brackets are only used to show the grouping of parameters, they do not exist in reality):

```
keyword = <1> <2> <1 2> <1 2> <XXX> <XXX 1> <$VAR> <$VAR 1>
```

Next thing you can do is masking of special characters. Special characters you can mask with backslash (\) are:

```
$ { } " '
```

It depends on the keyword, how many "words" you can assign to them. There are also underlying rules, that some keywords are only allowed if others are set. And finally, not all characters or words are allowed for all keywords.

Command line
On the command line, you always have a long option and sometimes a shortcut. The long option begins with `--`, while the short one simply begins with a `-`. Example:

```
# storeBackup.pl --file backup.config
# storeBackup.pl -f backup.conf
```

are equivalent. This also shows an option, which can have exactly *one* parameter (the file name).
There are other options which can have more than one parameter:

```
# storeBackup.pl .... -e /proc -e /tmp -e /var/tmp
```

In this example, option `-e` (same as `--exceptDirs`) is used to exclude multiple directories from backup. You can simply repeat the option. It does not matter if you use the long or the short form or a mixture of them.
Now let's look at an option to define the program to uncompress the files in the backup. Let's choose `gzip -d`, a program with a parameter:

```
# storeBackup.pl .... --uncompress "gzip -d"
```

As documented in the description of storeBackup.pl in section 6.2, the parameter of this option is parsed as if it was written in the configuration file (the quoting is stripped by the shell):

```
uncompress = gzip -d
```

In this way, storeBackup.pl will see two parameters (`gzip` and `-d`) of option `--uncompress`. The first one is then used as the program and the rest as parameters for it.
Sometimes, you can also use list parameters (parameters without an option). E.g. in storeBackup.pl they are called `otherBackupSeries`.

Bringing both together
In some programs (like storeBackup.pl) you can use both command line options and a configuration file. Normally, it is easier and more clearly to use the configuration file.
But in some situations it is very convenient to be able to overwrite an option set in the configuration file. You can simply do this by additionally setting that option on the command line. In programs delivered with storeBackup the command line will overrule the settings in the configuration file.
There is one special kind of situation when this normally would not be possible. Imagine, you wrote in the configuration file of storeBackup.pl:

```
doNotDelete = yes
```

This means, storeBackup.pl will not delete any old backup, because you want to do this later with storeBackupDel.pl. But the only thing storeBackupDel.pl has to offer is an option as a flag: `--doNotDelete` which only means *yes* in the configuration file. There's no special option to say *no*. Instead of introducing dozens of options for such cases, these programs use the following syntax:

```
# storeBackupDel.pl -f backup.config --unset doNotDelete
```

This will "unset" the `doNotDelete` option set in the configuration file.
You can also write:

```
# storeBackupDel.pl -f backup.config --unset --doNotDelete
```

which is the name of this option on the command line.

For list parameters, there is a mapping from the list parameters without option to a special option in the configuration file – for storeBackup.pl that's *otherBackupSeries*, for storeBackupSearch.pl it is *backupRoot*. This is documented for the individual programs (see description).

7.2 Selecting Directories / Files to Backup

StoreBackup allows you to use and combine different methods to select directories or individual files to be – or not to be – part of the backup. The following options of `storeBackup.pl` can be combined.

Matching Directories:

- `includeDirs` Allows you to include the specified directories into the backup. You *have to* use the relative path below `sourceDir`. If you use this option, only those directories will be saved – nothing else. You may deselect subdirectories of the directories specified with this option via `exceptDirs`.
- `exceptDirs` Allows you to exclude the specified directories from the backup. You *have to* use the relative path below `sourceDir`. If you use this option, the selected directories will not be saved. You *cannot* select subdirectories to backup from the directories specified with this option via `includeDirs`.
- `followLinks` Allows you to very flexibly map directories into the backup. The way to use this option is to create a special directory for backup only and to map the directories to save into this special backup directory simply via symbolic links. You may want to combine this option with `exceptDirs`. See description of option `followLinks` in section 6.2. Example 2 also shows how to use `followLinks`.
- `stayInFileSystem` Only files and directories in file systems which are specified by option `sourceDir` of via the symbolic links handled as directories via option `followLinks` are saved.

When specifying directories for including or excluding, you can use wildcards. StoreBackup calls a shell to evaluate those wildcards and prints the result in the log file(s). If *wildcard* sounds strange to you and you want to know what it is, use an internet search engine for the term "wildcard shell".

Matching Files:

- `includeRule` If you define a rule for this option then *only* files matching this rule will be included in the backup. This rule is evaluated *before* `exceptRule` (if defined).
- `exceptRule` If you define a rule for this option then files matching this rule will be excluded from the backup. Because this rule is evaluated *after* option `includeRule`, a file matching `includeRule` *and* `exceptRule` will *not* be stored in the backup.
- `exceptTypes` File types specified here will not be saved in the backup. This is useful, if you e.g. do not want to save some type of special files.

Because `includeRule` and `exceptRule` only work on *files*, the backup of the directory structure is not affected by matches. Especially – you cannot match directories itself.
Within a rule, you may use the rule functions `MARC_DIR` and `MARC_DIR_REC` to match *all files* of a directory (maybe including its subdirectories). See EXAMPLE 5 in chapter 7.4, "Defining Rules" about how to use those rule functions.
For better control and examination about excluded files *because of rules*, you may want to use option `writeExcludeLog`. If set, the (relative) files names of all excluded files are written to `.storeBackup.notSaved.bz2` in the top level directory in your actually created backup.

7.3 Deletion of old Backups

The more standard Approach

StoreBackup gives you a lot of possibilities to delete or not delete your old backups. If you have a backup which should never be deleted, the simplest way to achieve this is to rename it by appending a *dash* followed by your desired text string to the existing date_time filename. For example:
`$ mv 2003.07.28_06.12.41 2003.07.28_06.12.41-archive`
Renamed backups must match this naming pattern exactly: `yyyy.mm.dd_dd.mm.ss-(.+)`
IMPORTANT: If you use the option lateLinks of storeBackup.pl, only do this after a successful run of storeBackupUpdateBackup.pl!
Renamed backups are not hard linked by new backups, so please do not rename them until they are not the newest ones any more.
To archive with a simple renaming is possible because storeBackup.pl and storeBackupDel.pl only delete directories which match *exactly* the pattern `YYYY.MM.DD_hh.mm.ss` .
The most simple way to delete a specific directory is to use `rm -rf`. *Do not do this if you use the option lateLinks of storeBackup.pl!* If you want to delete backups which are too old depending on rules, there are several options you can choose. You can specify the time to keep old backups on the basis of weekdays (with a default value for all weekdays in keepAll which can be overwritten with keepWeekday). You can also specify to keep them with keepFirstOfYear, keepLastOfYear, keepFirstOfMonth and keepLastOfMonth, or with keepFirstOfWeek and keepLastOfWeek where you can define the first weekday of your definition of a week. In all of these cases, you have to specify a time period. How to specify a time period is described with the options of storeBackup.pl.
Now imagine you are making your backups on an irregular basis, perhaps from a laptop to a server or you make your backups when you think you have finished an important step of your work. In such cases, it is useful to say "only keep the last backup of a day in a long time range" (with keepDuplicate). If you were in holidays for a month and have set keepAll to `30d` (30 days), then you probably do not want that storeBackup deletes all of your old backups when you start it for the first time when you're back. You can avoid this with the option keepMinNumber. On the other hand, if you have limited space on your backup disk, you want to limit the total number of backups, for this, you can use keepMaxNumber.
With keepDuplicate you specify a time period in which storeBackup keeps duplicate backups of a day. After this time period only the last backup of a day will survive.
With keepMinNumber you specify the minimal number of backups storeBackup (or storeBackupDel) will *not* delete. The logic is as follows:

- Do not delete backups specified with any of the other keep∗ options.
- If this is not enough, do not delete other ones beginning with the newest backups. Duplicates of a day are not affected by this parameter.

With keepMaxNumber you specify the maximal number of backups. StoreBackup will then delete the oldest backups if necessary. To prevent special backups from deletion, you can specify an "archive flag" with keepAll∗ options. Backups matching an archive flag will never be deleted by keepMaxNumber. In this way it is possible that more backups will remain than specified with this parameter, but the archive flag is useful to prevent special backups like "last backup of a month" or "last backup of a year" to be deleted.

Using keepRelative as a Deletion Strategy

This option activates an alternative backup deletion scheme that allows you to specify the relative age of the backups you would like to have rather then the period over which a backup should be kept.
Imagine that you always want to have the following backups available:

- 1 backup from yesterday
- 1 backup from last week
- 1 backup from last month
- 1 backup from 3 months ago

Note that this is most likely *not* what you really want to have, because it simply means that you have to do daily backups and have to keep every backup for exactly 3 months. Otherwise you wouldn't always have a backup that is of *exactly* the requested age.
What you really want to have is therefore probably something like this:

- 1 backup of age 1 hour to 24 hours / 1 day
- 1 backup of age 1 day to 7 days
- 1 backup of age 14 days to 31 days
- 1 backup of age 80 days to 100 days

This is now a very common backup strategy, but you would have difficulties to achieve this with the usual keepFirstOf* options, especially if you do not do backups with perfect regularity. However, you can implement it very easily using keepRelative. All you need to write is:
`keepRelative = 1h 1d 7d 14d 31d 80d 100d`
i.e. you list all the intervals for which you want to have backups. storeBackup will delete backups in such a way that you come as close as possible (if you do not do backups often enough, there is of course nothing that storeBackup can do) to your requested backup scheme.
Note that this may mean that storeBackup keeps more backups that you think it has to, i.e. it may keep two backups in the same period. In this case storeBackup “looks into the future” and determines that both backups will *later* be necessary in order to have a backup for all periods. This is also the reason why in the above example you have somehow implicitly specified the period 7 days to 14 days, although you didn't really want to have a backup in this period – in order to have backups in the next period (14 days to 31 days) you always need to have a backup in the period 7 days to 14 days as well. Therefore the syntax doesn't allow you to exclude some periods.
Finally you should be aware that storeBackup shifts all the intervals if it cannot find a recent enough backup: If your first interval is from 10 days to 20 days, but your most recent backup is actually 25 days old, all subsequent periods will be extended by 5 days. This ensures that if you haven't made any backups over a large period, this period is not taken into account for your backup scheme. To give an example why this is useful: If you wanted to have backups 1, 3, 7 and 10 days old and then went on vacation for 14 days, it is pretty unlikely that you want all your backups deleted when you come back, hence storeBackup ignores these 14 days and keeps the backups appropriately longer.

7.4 Defining rules

Rules can be defined in storeBackup.pl, see section 6.2 (options exceptRule, includeRule and comprRule) and in storeBackupSearch.pl, see section 6.6 (option searchRule). Both support the definition in configuration files and on the command line.

This part of the description shows how to use rules in storeBackup. If you are not familiar with pattern matching and perl you should try to change the examples very carefully a little bit. But you can run easily into error messages you will not understand.
First, all the examples are explained for being written in a configuration file. Mostly I will use the key word from storeBackup.pl (`exceptRule`[19]), but the syntax of rule definition is identical to the ones you can use for `includeRule` and `searchRule`. Later, we will see how to use rules on the command line.
All the values we are talking about now are the ones from the files backed up at the point in time when the backup was performed, *not* from the files in the backup!
In general, rules are a piece of perl with some specialities. We start with some easy and typical examples:

EXAMPLE 1:

```
exceptRule = '$size > 1610612736'
```

(Take care of the quotes. Generate a configuration file with storeBackup.pl or storeBackupSearch.pl and read the comments in the beginning how quoting and environment variables are interpreted.)

[19] In section 7.2“Selecting Directories / Files to Backup” you will get an overview about different options to include / exclude files or directories.

This rule will match for all files with more than 1.5GB ($1.5 * 1024^3$) bytes. `$size` represents the size of each individual file. In this example, all files bigger than 1.5GB will not be saved. This is not very easy to read, and you can write instead:

```
exceptRule = '$size > &::SIZE("1.5G")'
```

(Take care of all quotes.) This will have the same effect as the rule before. &::SIZE is a function which calculates the real value from the string "1.5G". You can use identifiers from 'k' to 'P' with the following meaning:

`1k`	1 kilobyte = 1024 Byte
`1M`	1 Megabyte = 1024^2 Byte
`1G`	1 Gigabyte = 1024^3 Byte
`1T`	1 Terabyte = 1024^4 Byte
`1P`	1 Petabyte = 1024^5 Byte

Eg: `&::SIZE("0.4T")` is valid, while `&::SIZE("1G1M")` is not.

Example 2:

```
exceptRule = '$file =~ m#\.bak$#'
```

(Take care of the quotes.) This rule will match for all files ending with '.bak' which means they will not be saved. `$file` represents the individual file name with the *relative path* below the parameter of option *sourceDir* from storeBackup.pl. If you do not understand the strange thing right to `$file`, it is called pattern matching or regular expression. See *man perlretut* (perl regular expressions tutorial) for detailed explanation. But you should be able to expand this to simple needs:

```
exceptRule = '$file =~ m#\.bak$#' or '$file =~ m#\.mpg$#'
```

(Take care of the quotes and *all* blanks.) This rule will match and therefore not save files ending with '.bak' or '.mpg'.

```
exceptRule = '$file =~ m#\.bak$#' or '$file =~ m#\.mpg$#'
   or '$file =~ m#\.avi$#'
```

It should not be a surprise, that you will not backup files ending with '.bak', '.mpg' or '.avi'.
Now we want to create a rule which will prevent the backup of all files which end with '.bak', '.mpg' or '.avi' and also all files bigger than 500 Megabyte:

```
exceptRule = '$file =~ m#\.bak$#' or '$file =~ m#\.mpg$#'
   or '$file =~ m#\.avi$#' or '$size > &::SIZE("0.5G")'
```

If you set 'debug = 2', you can see if and how the rule matches for individual files. If you set 'debug = 1', you can see if the rule matches for each file. With 'debug = 0' (default), you will not get a message.

You can use the following 'preset variables':

`$file`	file name with relative path from original sourceDir
`$size`	size of the file in bytes
`$mode`	mode of the file (integer, use 0... to compare with octal value, e.g. `$mode = 0644`
`$ctime`	creation time in seconds since epoch (Jan 1 1970), see below
`$mtime`	modify time in seconds since epoch, see below
`$uid`	user id (string if defined in operating systems), e.g. `$uid eq "bob"`
`$uidn`	user id (numerical value), e.g. `$uidn = 1001`
`$gid`	group id (string if defined in operating system), see `$uid`
`$gidn`	group id (numerical value), see `$uidn`
`$type`	type of the file, can be one of `SbcFpl`, see option exceptTypes in storeBackup.pl
`$ENV{XXX}`	use contents of environment variable `XXX`

If you use $ctime or $mtime, it is not pure fun to calculate the number of seconds since epoch (Jan 1st 1970) every time. For this reason, storeBackup supports a special function &::DATE to make your live cosy:

Example 3:

```
searchRule = '$mtime > &::DATE("14d")' and '$mtime < &::DATE("3d12h")'
```

With this search rule (in storeBackupSearch.pl) you will find all files which are younger than exactly 14 days and older than 3 days and 12 hours. The syntax understood by &::DATE is:

1.

`d`	day
`h`	hour
`m`	minute
`s`	second

   So "3d2h50s" means 3 days, 2 hours and 50 seconds. With the function above, you specify "now" minus that period.

2.

`YYYY.MM.DD`	year.month.day
`YYYY.MM.DD_hh.mm.ss`	same format as backup dirs
`2008.04.30`	specifies April 30 2008, 0:00,
`2008.04.30_14.03.05`	specifies April 30 2008, at 2 o'clock, 3 min. and 5 sec. in the afternoon.

   With the function &::DATE, you also specify a fixed point in time.

You already saw some possibilities to group the checking of the "variables" by using `and` and `or`. You can use:

```
and, or, not, (, )
```

Everything is like in perl (to be honest, it is evaluated by the perl interpreter). But you should surround each of these with one (or more) blanks (white spaces) if you want `debug = 2` to work correctly!

Example 4:

```
searchRule = ( '$mtime < &::DATE("14d")' and '$mtime > &::DATE("3d12h")' )
     and not '$file =~ m#\.bak$'
```

Finds all files younger than 14 days and older than 3 days, 12 hours, but only if they do not end with `.bak`. See how `and`, `not`, `(` and `)` have at least one white space surrounding it.

Example 5:

StoreBackup allows you to control its behavior via rules by storing special flag files in the source directory. This can be done for a specific directory or additionally for its subdirectories.
You define in the configuration file of `storeBackup.pl`

```
exceptRule= '&::MARK_DIR($file)'
```

then all files in directories with the default "marker" file `.storeBackupMark` will not be saved because the function `MARK_DIR` returns `1` in that case.
You may also use another marker file, e.g. `.dontBackup`:

```
exceptRule= '&::MARK_DIR($file, ".dontBackup")'
```

Then – naturally – you have to create a file `.dontBackup` in the directories you do not want to backup:

```
$ touch .dontBackup
```

If you do not want to backup all sudirectories below the one with file `.dontBackup` in it, define the rule acting recursively:

```
exceptRule= '&::MARK_DIR_REC($file, ".dontBackup")'
```

Because this is a rule, you can also reverse the result by configuring:

```
exceptRule= not '&::MARK_DIR($file, ".backupThisDir")'
```

As a name for the marker file, you now use e.g. `.backupThisDir`. All backup *without* this marker will *not* be backuped now.
You can do any combination with other rule elements. If e.g. you only want to save files from root below directories marked with `.saveRootOnly`, define:

```
exceptRule= not ( '&::MARK_DIR_REC($file, ".saveRootOnly")'
                  and '$uid eq "root"' )
```

Now think you do not want to save recursively all files in directories with `.dontBackup` *and* you *only* want to save files from user root recursively in directories with `.saveRootOnly`. In this case you can define the following:

```
exceptRule = '&::MARK_DIR_REC($file, ".dontBackup")'
     or
       not ( '&::MARK_DIR_REC($file, ".saveRootOnly")'
                  and '$uid eq "root"' )
```

You can use combinations with different marker files in different or the same rules.
Naturally, you can also use `MARK_DIR` and `MARK_DIR_REC` in other rules, especially for option `comprRule`.

Using rules on the command line
Let's take a look at:

```
exceptRule = '$size > &::SIZE("1.5G")'
```

If we try to use the command line like this:

```
--exceptRule '$size > &::SIZE("1.5G")'               ### WRONG ###
```

we will get some nasty error messages because the shell strips the single quotes and storeBackup tries to interpret the result the same way as in the configuration file (see description in each configuration file at the top). Here, storeBackup will complain about not knowing the environment variable `$size`. (The $-sign is not masked any more because the shell removed the single quote.) So we have to mask the $-sign. We also have to mask the double quotes, because normally, storeBackup will interpret them as grouping quotes and will not bypass them directly to perl. The right way specifying this option is:

```
--exceptRule '\$size > &::SIZE(\"1.5G\")'            ### CORRECT ###
```

We have to write example 4 in the following way:

```
--searchRule '( \$mtime < &::DATE(\"14d\") and \
   \$mtime > &::DATE(\"3d12h\") ) and not \$file =~ m#\.bak\$#'
```

In case of problems, you should read the perl error massage which shows what perl really gets. Beside this, option `--print` will show each parameter after being parsed through shell and storeBackup. You can use `--print` in combination with configuration files, too.

The following predefined functions are available for rules:

`&::DATE(timePeriod)` Calculates the relative time from "now" and lets you compare it with `$ctime` or `$mtime`. Set `timePeriod` to e.g. `"3d2h"`. See example 3 in this section for details.

`&::SIZE(size)` Returns the `size` given in human readable form (e.g. 1.5G) as a plain number (in bytes). See example 1 in this section for details.

`&::MARK_DIR($file [, markerFile])` Checks, if `markerFile` exists in the directory of `$file`. Default for `markerFile` is `.storeBackupMark`. See example 5 in this section for details.

`&::MARK_DIR_REC($file [, markerFile])` Checks, if `markerFile` exists in the directory or any higher level directory of `$file`. Default for `markerFile` is `.storeBackupMarkRec`. See example 5 in this section for details.

7.4.1 How to define if a file should be compressed

You can configure the behaviour of storeBackup about compression with a combination of the following options: `exceptSuffix`, `addExceptSuffix`, `compressSuffix`, `minCompressSize` and `comprRule`. Please note that option `comprRule` is generated from the others so you do not have to care about it. However, if you set `comprRule`, the values of the other options are ignored (only use `comprRule` if you want to do very fancy stuff).

Using default values:

If you use default values, starting with storeBackup version 3.3 nothing changes to previous versions. The default values are:

```
exceptSuffix = \.zip \.bz2 \.gz \.tgz \.jpg \.gif \.tiff \.tif \.mpeg \.mpg \.mp3 \.ogg
   \.gpg \.png
addExceptSuffix =
compressSuffix =
minCompressSize = 1024
```

You can change the value of `minCompressSize`, change the suffixes of `exceptSuffix` or add suffixes to `addExceptSuffixes`. As long as `compressSuffix` is *not* set, storeBackup will internally generate a rule with means (with default values):
Do not compress files less than 1k bytes or having one of the suffixes defined in `exceptSuffix` *or defined in* `addExceptSuffix`*.*
If you do not define `exceptSuffix` or `minCompressSize`, the default values will be taken![20]

No compression at all:

If the rule `comprRule` returns 0, the concerned file will not be compressed. So we can configure simply:

```
comprRule = 0
```

Compress every file:

If the rule `comprRule` returns 1, the concerned file will be compressed. So we can configure:

```
comprRule = 1
```

But that's not really useful. Why compressing files, which cannot be compressed any more and will therefore be bigger after compression!?

Using a white list *and* a blacklist:

In most cases, you know some file types (by suffix) you are using *not* to compress (like `.jpg`) and others where it makes sense to compress (like `.doc`, `.bmp`, `.txt`, etc.).
If you define one or more suffixes at `compressSuffix`, e.g., `.pdf`

```
compressSuffix = \.pdf
```

then storeBackup will behave in the following way:
Do not compress files less the value defined in `minCompressSize`*. Do not compress files with suffixes defined in* `exceptSuffix` *or* `addExceptSuffix`*. Compress files with suffixes defined in* `comprSuffix`*. For the rest of the files, make a decision based on* `COMPRESSION_CHECK`*.*

Let storeBackup decide:

There is a special rule-function which appraises if it is worth to compress a file. This rule-function returns 1 if it thinks the file should be compressed and 0 if not. So we simply define the following comprRule:

```
comprRule = '&::COMPRESSION_CHECK($file)'
```

[20]This is compatible to the behavior before the introduction of `compression Suffix` and `COMPRESSION_CHECK`.

The rule above works pretty well, but often it is not necessary to run this rating and therefore you can simply set the other options and do not have to care about `comprRule` *(which is generated automatically, like described above).*

Recommendation

What you should do and what fits in most cases: Depending on your needs, simply define some suffixes for files which should be compressed (`comprSuffix`*) (like* `.pdf`*) and maybe extend the list of files with suffixes with should not be compressed (*`addExceptSuffixes`*). That's it.*
Remark: If you use the "blocked file"features of storeBackup, you can also use this algorithm by setting e.g., option `checkBlocksCompr` to `check`. See blocked files (section 7.5) for more information.

Do it your own way

If you have very special demands, e.g., configure everything in the way described above, *but* do `not` compress the files for a specific group or some users, you have to (and can) define an individual rule. Do this by using the hints given in section 7.4 and run very small backups with debug level 3 for testing.

7.5 Saving Image Files / raw Devices / Blocked Files

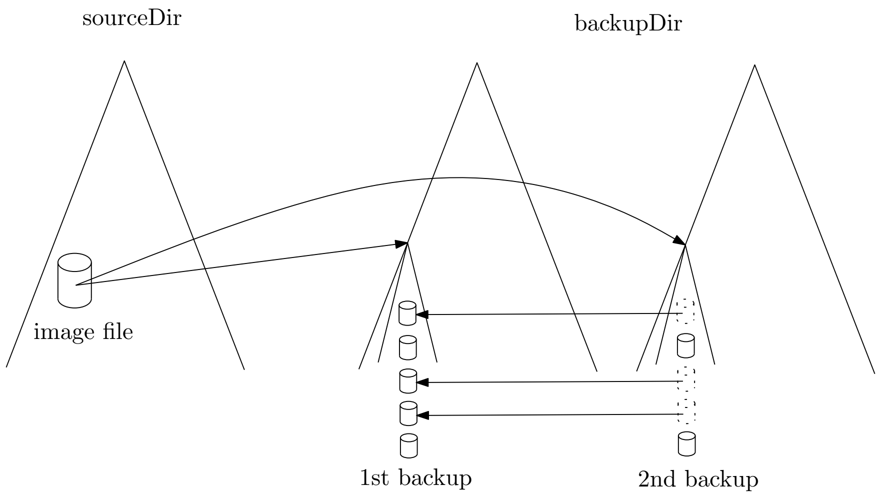
The scope of blocked files

Saving big image files which change only in parts completely with every backup is inefficient, as it is very time and space consuming. To give some examples:

- Some mailers use traditional mbox (mailbox) format to save email. This is convenient, because it is a well supported format. But you will get a big file of perhaps multiple gigabyte with all your mails in it. Backing up such a file means backing up everything despite the fact that only a very little part of it has changed.
  The same category are the `.pst` files from Outlook. If you have to save this kind of files (and if they are big), you should think about using "blocked files".
- If you use an image file with an encrypted file system in it like e.g., TrueCrypt does, you should backup the encrypted data, not the files in it. (If you backup the files in it you need another encrypted container, which means the backup program has to know all passwords to run automatically which is a perfect security hole.)
  For that reason you should backup the binary image data as it is. If you make a simple copy, it will take the size of the image each time (you also cannot compress this data). This is a perfect situation to use the storeBackup blocked files feature (without compression), where you can have lots of historic versions of the image without needing too much space and without a security hole (storeBackup does not need to know and does not know anything about the content it saves).
- Images for hypervisors like Xen, KVM or VMware are another example which you can save as "blocked files" very successfully.
- Do not use blocked files which are compressed as a whole like jpegs or other types of compressed or encrypted files (`.gz`, `.bz2`, `.gpg`, etc.). In most cases, changing something in that files result in a complete change of all blocks.
- The feature of blocked files is also not suitable for database dump files, because storeBackup (up to now) works with fixed blocks. If you add one byte in the beginning of a file, all blocks will be different.

How it works

If you specify a file to be saved in block files (see below how to do this), then storeBackup.pl will do the following:



1. Create a directory with the same combination of path and file name of the original image file in the source directory.
2. Split the source file into blocks and check if any of these blocks exist anywhere in a backup (see option `otherBackupSeries` of storeBackup.pl). If a block already exists, a hard link is generated, if it does not exist, the block will be copied or stored compressed.
3. The md5 sum of all these files will be stored in a special file called `.md5BlockCheckSums.bz2` in that directory.
4. storeBackup.pl will also calculate the md5 sum of the whole file and store it in `.md5CheckSum`.

Because references to existing files are realized via hard links, every backup is a full backup.
If you use the option lateLinks, see section 7.6, the links will be set later. If you also use the option lateCompress, the compression will also be done later.

How to save image files

There are two ways to configure which files storeBackup.pl should treat as blocked files:

1. The easiest way is using the following options:

   `checkBlocksSuffix` The configuration is similar to `exceptSuffix`, a list of suffixes which are checked for a match, e.g., \\`.vdmk` for VMware images. They simply mean that the last part of the file name must be similar to what you define here.
   The next options described here are only used if `checkBlocksSuffix` is set.

   `checkBlocksMinSize` Only files with this minimum size will the treated as blocked files. You can use the same shortcuts as described in defining rules, see section 7.4, e.g., 50M means 50 megabytes. The default value is 100M.

   `checkBlocksBS` Defines the block size in which the files which matches has to be split by storeBackup.pl. The format is equal to `checkBlocksMinSize`. The default value is 1M. The minimal value is 10k.

   `checkBlocksCompr` Defines if the blocks are compressed. Possible values are `yes`, `no` or `check`. On the command line, set `--checkBlocksCompr`.
   This flag only affects files selected with `checkBlocksSuffix`.

   Example:
   You want to backup all your VMware images and you also have to backup some Outlook `.pst` files. The blocked file feature will be chosen from storeBackup for files with a minimum size of 50 megabyte ending with `.vmdk` or `.pst`. The block size chosen is 500k and the resulting blocks in the backup will be compressed:

```
checkBlocksSuffix = '\.vmdk' '\.pst'
checkBlocksMinSize = 50M
checkBlocksBS = 500k
checkBlocksCompr = yes
```

2. The more flexible way to specify the handling of blocked files is to use rules like described in defining rules, see section 7.4. The following options are available five times, so there is a `checkBlocksRule0`, `checkBlocksRule1`, `checkBlocksRule2`, `checkBlocksRule3` and `checkBlocksRule4`:

   `checkBlocksRule`*i* The *i*th rule specifying files to treat as blocked files in the backup.

   `checkBlocksBS`*i* The corresponding block size for the blocks in the backup. The default value is 1 megabyte. The minimal value is 10k.

   `checkBlocksCompr`*i* If set to `yes`, the blocks will be compressed. If set to `no`, they will not be compressed. If set to `check`, storeBackup will decide itself if they will be compressed. This may result in a mix of compressed and copied blocks.

   `checkBlocksRead`*i* Defines a filter for reading the specified file, e.g., *gunzip* or *gzip -d*. This option may be useful if you have to save an already compressed image file. (Using the "blocked file" feature of storeBackup with already compressed files compressed as a whole does not make sense.)

   Example:
   Let's assume, you have a TrueCrypt image on your disk and want to have a backup of it each time you start storeBackup.pl. You chose the unremarkable name `myPics.iso`, block size is 1M, no compression. So you define rule 0:

```
checkBlocksRule0= '$file =~ m#/myPics\.iso$#'
#checkBlocksBS0=
#checkBlocksCompr0=
checkBlocksRule1= '$size > &::SIZE("50M")' and
        ( '$file =~ m#\.pst$#' or '$file =~ m#windows_D/Outlook/#' )
checkBlocksBS1=200k
checkBlocksCompr1=check
```

   You also defined rule 1, which matches for all files bigger than 50 megabytes ending with `.pst` *or* located in the *relative* path `windows_D/Outlook/` in the backup. (I'm using this to backup the data of my dual boot laptop.) If you are not familiar with rules in storeBackup, you should read section 7.4.

You can use `checkBlocksSuffix` and `checkBlocksRule`*i* at the same time in one configuration file. Store-Backup evaluates `checkBlocksRule`*i* (in ascending order) first and then `checkBlocksSuffix`.

how to save mass storage devices

Backing up a mass storage device (like `/dev/sdc` or `/dev/sdc1`) works in the same way as saving an image file with storeBackup. You choose the device(s) with `checkDevices`*i*, the block size in the backup with `checkDevicesBS`*i* and switch compression on or off with `checkDevicesCompr`*i*. Additionally, you have to specify the relative path with `checkDevicesDir`*i* in the backup where the contents of the devices will be stored.
The blocks in the backup resulting from image files or devices are hard linked if storeBackup finds the same contents.
The options are in detail:

`checkDevices`*i* List of devices (e.g., `/dev/sdd2 /dev/sde1`) to backup.

`--checkDevicesDir`*i* Directory where the devices are stored in the backup (*relative* path). The image file will also be restored in that directory if you restore the backup with storeBackupRecover.pl (if you use default parameters). Into this directory storeBackup will create a subdirectory which name is generated from the parameters of option `checkDevices`, e.g., `/dev/sdc` will result in `dev_sdc`.

`checkDevicesBS`*i* Defines the block size in which the devices specified have to be split by storeBackup.pl. The format is equal to `checkBlocksMinSize`. The default value is 1M. The minimal value is 10k.

`checkDevicesCompr`*i* Defines if the blocks are compressed. Possible values are `yes`, `no` or `check`; the default value is `no`.
This option only affects files selected with `checkDevices`*i*. If you set this option to `check`, every block is checked for compression (or not).

Choosing the block size

There is no fix rule about the "best" block size. I made some measurements about the block size and the used space. The second backup was done with lateLinks (see section 7.6), so I could use `df` again to see how much space was really needed. The used file system was reiserfs with tail packing. If you use a file system without tail packing (like ext2, ext3 or ext4), the overhead will be bigger and small block sizes are less attractive (same if you use compression). The results also depend on the application writing to your source image file.
All the examples are done without compression (for performance reasons). They were done with real data. Naturally, I'm using compression in my real backups. The 2nd backup shows the space needed for the changed data. The percentage line below shows the relation between the first and the second backup. The sums line shows the sum of the first and second backup, the next line (1x) the relationship between that sum depending on the last value with 5M (5 megabyte blocks). The last line shows the same relationship regarding the size of the first backup and 10 times the second one (extrapolating 10 backups). So this should be the most interesting value.

The first example shows the results when storing a big Outlook`.pst` file of 1.2GB with the changes I had from one day to the other:

BlockSize	50k	100k	200k	1M	5M
1. backup [kB]	1219253	1172263	1172863	1173801	1173724
2. backup [kB]	7692	13445	22720	73826	240885
	0.63%	1.15%	1.94%	6.29%	20.52%
sum [kB]	1226945	1185708	1195583	1247627	1414609
1x	86.73%	83.82%	84.52%	88.20%	100.00%
10x	36.18%	36.47%	39.08%	53.37%	100.00%

The second example was done with a smaller Outlook file of 117 megabyte. This is the one for the input folder. The numbers show a different behavior than in the first example.

BlockSize	50k	100k	200k	1M	5M
1. backup [kB]	122487	118221	118891	119184	119181
2. backup [kB]	33400	51240	74424	107632	119181
	27.27%	43.34%	62.60%	90.31%	100.00%
sum [kB]	155887	169461	193315	226816	238362
1x	65.40%	71.09%	81.10%	95.16%	100.00%
10x	34.82%	48.10%	65.84%	91.19%	100.00%

The third example shows the results when storing a VMware image of 2.1 GB. Between the first and the second backup the VM was booted, a program for updating my navigational system was updated and I connected the navigational system for an update also.

BlockSize	50k	100k	200k	1M	5M
1. backup [kB]	2162595	2106781	2112547	2117178	2117094
2. backup [kB]	53656	80609	131701	438241	1112652
	2.48%	3.83%	6.23%	20.70%	52.56%
sum [kB]	2216251	2187390	2244248	2555419	3229746
1x	68.62%	67.73%	69.49%	79.12%	100.00%
10x	20.38%	21.99%	25.90%	49.08%	100.00%

In all these examples you can see in the last line, that at some point smaller block sizes will not reduce the space needed. An optimum values seems to be between 50k and 200k (when using tail packing).

There is one additional important aspect about the block size: If you choose a small block size, the performance will also go down. To be able to achieve acceptable performance, the following optimizations are implemented:

- If you do not compress the the blocks within storeBackup.pl (no compression at all or later compression via option lateCompress), no parallelization is used.
- If you compress the blocks within storeBackup.pl and configure a block size of 1 megabyte or more, parallelizing is used.
- If you compress the blocks within storeBackup.pl with bzip2 and configure a block size of less than 1 megabyte, storeBackup.pl tries to use the perl module `IO::Compress::Bzip2`. If it is installed on your system, it will be used.

it is best to make your own tests to get a feeling of useful block sizes in your use cases.

7.6 using option lateLinks

You can use storeBackup as one program (`storeBackup.pl`) which does everything alone or you can split the different tasks into several pieces. There is mainly one advantage to run different programs for the different tasks: the time for backup itself from the perspective of the saved computer (or) data is lower.

It makes sense to use option lateLinks if you store your Backup on an nfs server and if you think it is a good idea to speed things up (see section 5.4.3, performance). Configuring lateLinks is a little bit more complicated than using storeBackup.pl as a standalone programe because you have to manage multiple programs.

StoreBackup.pl as a standalone program does the following tasks:

1. The link consistency of all backups (from all backup series) is checked. We will see late what this means.
2. Loading of meta data from one or more old backups. This task is like an initialisation, where it gets file names, md5 sums, dates, times and some other information from the old backups.
3. Checking for all files to backup if another file with that specific content is already in those old backups from where the file names, md5 sums etc. were loaded.
4. The changed data is transferred to the backup: By copying, by compression or by hard linking. Naturally, the directory structure is also generated.
5. The permissions and owners of the directories are set to the same values as in the source directory.
6. Depending on the rules defined with the keep∗ options of storeBackup.pl, old backups are deleted.

If you start storeBackup.pl with the option lateLinks, the transfer of data (see step 4) and the actions on the remote file system are reduced to the absolutely necessary minimum:

- The changed or new files (including special files) are copied. Changed files which should be compressed are only copied if the option lateCompress is set. It depends on your situation if usage of lateCompress makes sense or not.
- Hard links are not generated in the new backup.
- Directories are only created if they are needed for copying/compressing.
- An additional file is created in the new backup: `.storeBackupLinks/linkFile.bz2`. It contains all the information what should have been done to complete a “full” backup with all hard links, directory entries and compressions. The correct permissions (which are also not set) are stored in the file `.md5CheckSums` in the top level of the backup. This file is also generated in a “full” backup run of storeBackup.pl. It is used for restoring data (storeBackupRecover.pl).

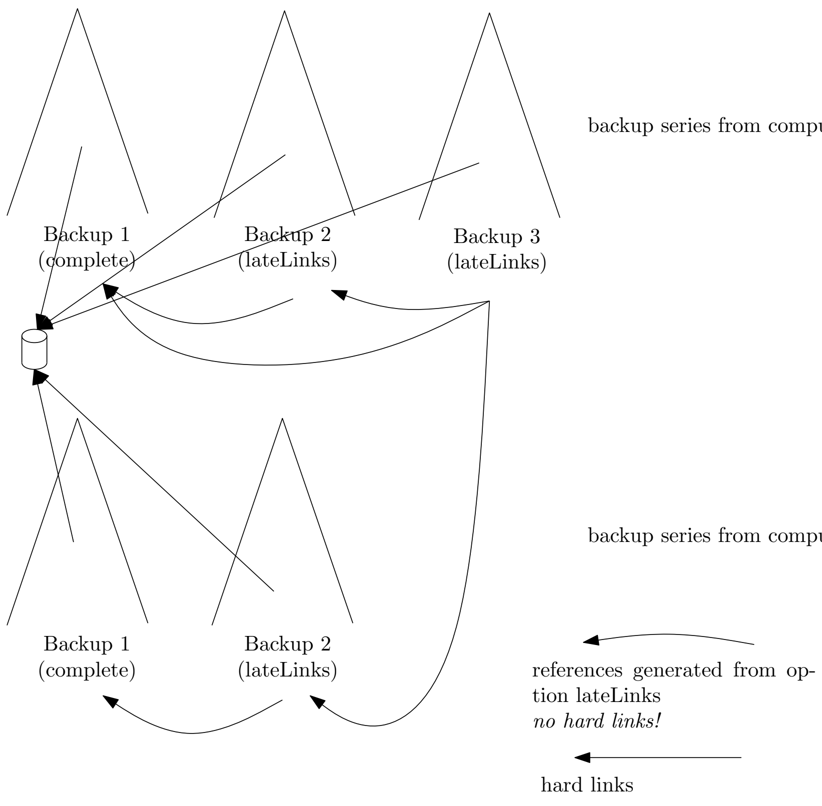
Independent of option lateLinks you can always configure storeBackup.pl to not perform step 6, the deletion of old backups. Especially if you are writing your backup on an nfs mount, this will take some time and lengthen your backup. Use storeBackupDel.pl (which can read the configuration file of storeBackup.pl) to split the deletion of old backups from the direct backup process.

It is important to understand that using option lateLinks creates an unfinished backup. Such backups do contain all the data which was intended to be backed up, so the core backup is complete. But storeBackup’s job is not completed in the following ways:

- Directory entries are missing.
- Files are not compressed (if you use option lateCompress).
- Hard links are not set at all.
- Permissions are not set correctly, neither for files nor for directories.
- Note that all the information that's necessary to complete to a full backup is available! In the case of hard links this means that there is a reference in the file `linkFile.bz2` which points to a file in an older backup. In that older backup, there also can be only a reference in its file `linkFile.bz2`, and so on. Naturally, at one point there *must* (and will) be a real file. But you should be aware: if you delete one of these referenced backups before running `storeBackupUpdateBackup`, you will destroy *all backups* which are referencing (directly in indirectly) to that backup. **Only delete backups with storeBackup.pl or storeBackupDel.pl – never use "rm" or something similar!** These two programs take care of the dependency just described. If you really want to delete files with "rm", then make sure, that storeBackupUpdateBackup.pl completed all backups successfully. When all links are set, there are absolutely no dependencies (beside hard links) between the different backups.

With the option `lateLinks` you create temporary incremental backups. Later you create with `storeBackupUpdateBacku`
full backups out of them (by using older backups).

The following picture shows two cross linked backup series from different computers.[21] You see that the references resulting from the lateLinks option can be complex. The hard links are never a problem, because there is no original or "real" reference – every hard link is an original pointer to the file (or more precise to the inode, see section 7.11). The file will only be deleted if the last hard link has gone. But the references created by lateLinks are just some file names in a file which has nothing to do with the file system.



To complete an unfinished backup (which makes all those nasty linking and compressing and so on), use storeBackupUpdateBackup.pl, (see section 6.3). It also will analyse your backups (below "backupDir") and find the right order to complete them. *After running* storeBackupUpdateBackup.pl *successfully your*

[21]see section 6.2, option otherBackupSeries to see how this can be configured.

backups will be in the same state as if you ran your backup without option lateLinks. Among some others, file `.storeBackupLinks/linkFile.bz2` is deleted and everything is hard linked (and compressed), and the permissions are set like in the source directory (except if the option ignorePerms was set in storeBackup.pl).

If you use option lateLinks, you should run storeBackupUpdateBackup.pl regularly, e.g., every night and check if there were some ERROR messages.

Summary:

- A backup that was made with lateLinks is unfinished and has to be finished via storeBackupUpdate-Backup.pl to be completed.
- You cannot make a new full backup (that is, a backup without using lateLinks) when this new backup refers to a prior unfinished backup. This is simply because you cannot hard link to files which are not there.
- When using storeBackup.pl or storeBackupDel.pl, you cannot delete earlier backups to which an unfinished backup refers.

7.7 Isolated Mode / Offline Backups

Traveling with a laptop and a small storage for backup (e.g., a memory stick) – this is the typical user case for this scenario. The storeBackup `isolatedMode` offers the ideal solution in this use case.

Advantages:

- You can just take incremental backups while not having access to your master backup repository
- You can store these incremental backups on small portable media such as a memory stick.[22]
- You have a history of all changes backed up during your travels.
- You can conveniently integrate the incremental backups into your master backup repository when you return

Limitations:

- It is recommended to use configuration files when using storeBackup in isolated mode [23] (technically, there is no difference with or without configuration files.)
- Naturally, you do *not* have access to all data in your master backup repository via this method.
- If you want to obtain older files from your master backup repository, you need to access to it (or have somebody send you a file per email or whatever).

Imagine, you have a big backup disk (or raid or whatever) at work or in your home containing your master backup repository. Now you have to go away for a week or so. You want to be able to make backups to your master backup repository while not connected (with good enough performance) (we call them offline backups). You want to use the most convenient storage media possible during your travels. And most importantly – you know you need to integrate your travel backup contents into your master backup repository.[24]

With storeBackup `isolatedMode` and a small memory stick, the application initializes your memory stick by copying the meta data from the last backup of your laptop from the master backup. The application will also fine-tune your storeBackup options as needed[25]. Then you go traveling and make incremental backups on your stick. These backups will normally need only very limited memory to store them because

[22]We refer to using a memory stick for traveling, but this storage can be any kind of storage you wish to use, including a large HDD.

[23]This isn't really a limitation, but if you are not using configuration files presently, you should probably begin to use them to take full advantage of this mode because it is easier and more convenient.

[24]Integration could be done with linkToDirs.pl, but `isolatedMode` does not need memory for whole backups, is more convenient and quicker in the present user case.

[25]See section 7.7.1. Depending on the usage of `storeBackupSetupIsolatedMode.pl`, it will generate a new configuration file from your existing configuration.

they contain the (compressed) deltas only. When you're back and you have access to your master backup repository, you'll run `storeBackupMergeIsolatedBackup.pl` which will copy your incremental backups to the right place in the master backup repository, and then `storeBackupUpdateBackup.pl` will automatically do all the necessary tasks to make space efficient full backups from your incremental ones.[26] Naturally, tools from storeBackup will support you in doing this to make these steps easy and reliable.

7.7.1 You can use isolated backup in the following way:

- Prepare your local storage (e.g., the memory stick):
  This step is not strictly specific to storeBackup. This is similar to the basic task you would do to prepare almost any Linux storage media. Make a Linux file system which is able to use hard links (e.g., ext2, reiserfs). You can do this by using graphical tools like gparted or as root on the command line using `fdisk -l` to identify the stick and `mkfs.ext2` (or `mkfs.reiserfs` or whatever) to create the file system. You can also use a stick with NTFS.
  Mount your stick and create a directory as the top level backup directory for `storeBackup.pl`.

- Use `storeBackupSetupIsolatedMode.pl` to automatically copy the meta data from your master backup to the media you want to use for your travel. It will copy just a few megabyte – `storeBackup.pl` with option `lateLinks` only needs these meta data; no real file data is needed in order to generate incremental backups.
  NOTE: This script also changes your configuration file for `storeBackup.pl` – changes depend on your usage of the tool.
  For safety reasons, `storeBackupSetupIsolatedMode.pl` only picks up meta data from full backups. If your last backup of the series you've chosen is not yet a full backup, it asks you to run `storeBackupUpdateBackup.pl`.
  Depending on the usage of `storeBackupSetupIsolatedMode.pl`, it will also generate a new configuration file from your existing ones.

- Use `storeBackup.pl` to run your normal backups. If your configuration file was adjusted by the run of `storeBackupSetupIsolatedMode.pl`, you can use the newly generated one. Otherwise, you have to adjust the options by yourself (e.g., `backupDir`, `lateLinks`).

- After running `storeBackup.pl` to make the isolated backup(s), use `storeBackupMergeIsolatedBackup.pl` to merge your isolated (incremental) backups into your master backup repository once you have access to it again. storeBackup makes each of these steps more convenient and easier if you use configuration files.
  As you can see, it is a good idea to use configuration files when using storeBackup in isolated mode.

- After running `storeBackupMergeIsolatedBackup.pl` your local backups are copied to the central repository; you should now run `storeBackupUpdateBackup.pl` (or wait until it runs via cron, etc.). When everything is fine, you can delete all the affected files (directories) on your local media (e.g., the memory stick).

Naturally, you can use isolated mode together with the replication of backups, see replication of backups, chapter 7.8.
You will find detailed information in storeBackupSetupIsolatedMode.pl, see chapter 6.7. Also, have a look at the description of storeBackupMergeIsolatedBackup.pl, see chapter 6.8.

WARNING: *It is not supported to run your* `lateLinks` *backup onto a file system with fat or vfat format (e.g., on your stick)! This type of file system is not able to distinguish between filenames in upper and lower case. This means, filename* `file.txt` *is similar to* `File.txt` *without any warning or error message. If you have two files or directories with the same name (only different in upper / lower case) in one directory, you will definitely not get all files in your backup.*
But you can use NTFS if you want!

[26]This means it will do compression, hard linking, setting date/time, permissions and ownership.

7.7.2 Setting up isolated mode

To explain what to do, we will go step-by-step through a simple example using configuration files.[27] You can go through this demo and later adopt the paths to your environment.

Running a backup to master backup First of all, I will do something you should already have done: I will create a backup in the master backup directory.

For this demo, I'll create a master backup repository in the directory `isol-test` in my home directory (`backup`), a source directory to backup (`source`), copy a file to backup into it and generate a configuration file:

```
# cd
# mkdir isol-test
# cd isol-test
# mkdir backup source
# ls
# cp -v /bin/ls source
# ls -l source
# storeBackup.pl -g stbu.conf
```

Next, use an editor of your choice and change the following items in `stbu.conf`:

```
sourceDir=source
backupDir=backup
```

We need a full backup so we can copy its meta data to the external media:

```
# storeBackup.pl -f stbu.conf
WARNING   2012.06.09 09:07:57  5647 created directory <backup/default>
...  <snip, deleted output of storeBackup.pl>
```

The warning tells you that `storeBackup.pl` created a subdirectory for the series `default`.

Setting up isolated mode You may want to use `ls` (or maybe a file browser) to see that the backup into directory `backup` was successful.

Now plug in your external media, e.g., a memory stick; it has to be formatted with a Linux file system or with NTFS (*not* FAT). Make sure it is mounted in the same path always. You can do this in several ways, maybe depending on you distribution and / or the GUI you are using. If you have no idea how to do this, search for `"blkid fstab"` with an internet search engine like Google or another one.

In the following settings I assume your external media has been mounted at `/media/stick`. Please adjust the path `/media/stick` to your local settings!
Now it is time to set up the stick after creating a backup directory `/media/stick/stbu` on it:

```
# mkdir /media/stick/stbu
# storeBackupSetupIsolatedMode.pl -f stbu.conf -t /media/stick/stbu
INFO    2012.06.09 09:27:29  5888 ./isolate-stbu.conf: changed <backupDir> to '/media/stick/stbu'
INFO    2012.06.09 09:27:29  5888 ./isolate-stbu.conf: created <mergeBackupDir> as 'backup'
INFO    2012.06.09 09:27:29  5888 ./isolate-stbu.conf: setting <otherBackupSeries> to 0:default
INFO    2012.06.09 09:27:29  5888 ./isolate-stbu.conf: changed <lateLinks> to 'yes'
INFO    2012.06.09 09:27:29  5888 you may want to adjust <./isolate-stbu.conf> to your needs
```

The program `storeBackupSetupIsolatedMode.pl` told you that it had created a new configuration file called `isolate-stbu.conf` with some adjustments: `backupDir` has been set to the directory on the stick and `lateLinks` (`storeBackup.pl`'s option lateLinks, see section 7.6) has been switched on. It also created an entry `mergeBackupDir` which is used later by `storeBackupMergeIsolatedBackup.pl` to integrate your isolated backups on the stick into the central ones in the directory `backup` (in this example). Finally, `otherBackupSeries` is set to this backup series only. Generating references to

[27] I assume all programs of storeBackup to be in `$PATH` so they can be called without a path.

other backup series (which do not exist in this simple example) is not possible when making backups on your stick.[28]
Have a look at the generated configuration file. The adjustment of options will only work if unused options are commented using a semicolon (`;`), *not* a hash sign (`#`).

If you want to use the functionality of isolated mode *later again*, you can take a fresh memory stick where the backup directory on the stick is accessible with the same path as configured here the first time. Then simply take the generated configuration file to copy the meta data to the stick:

```
# storeBackupSetupIsolatedMode.pl -f isolate-stbu.conf -v
```

Option `-v` will produce some output so you can see what's happening.

Run backups on your local media Now lets copy a new file to `source` directory and run a backup on it:

```
# cp /bin/pwd source
# storeBackup.pl -f isolate-stbu.conf
```

That's it. Now let's see what happend:

```
# ls -lh /media/stick/stbu/default/*
/media/stick/stbu/default/2012.06.09_09.07.57:
total 0

/media/stick/stbu/default/2012.06.09_09.56.36:
total 16K
-rw------- 1 root root 13K Jun  9 09:56 pwd.bz2
```

As you can see, there is no saved file in the first backup directory (`2012.06.09_09.07.57`) because meta data was copied by `storeBackupSetupIsolatedMode.pl`. In the second backup you see the new file `pwd` but not `ls` because it wasn't changed. It will be hard linked after being integrated into the master backup. If you want to see some internals, you can look into the command file for `storeBackupUpdateBackup.pl` to see it has to be linked:

```
# bzcat /media/stick/stbu/default/*/.storeBackupLinks/linkFile.bz2
# link md5sum
# existingFile
# newLink
# compress md5sum
# fileToCompress
# dir dirName
# symlink file
# target
# linkSymlink link
# existingFile
# newLink
link 92385e9b8864032488e253ebde0534c3
../2012.06.09_09.07.57/./ls.bz2
ls.bz2
```

You can run as many additional backups as you want, but naturally space on you local media must be sufficient. Use `df -h /media/stick` (adjust the path to your needs) to see how much space is free. You can also run `du` to see how much space has been used so far for your isolated backups:

```
# du -sh /media/stick/stbu/default/*
24K /media/stick/stbu/default/2012.06.09_09.07.57
44K /media/stick/stbu/default/2012.06.09_09.56.36
```

[28]This is true in this simple example. But if you copy multiple backup series via `storeBackupSetupIsolatedMode.pl` to your local backup, you can also adjust this option to cross link between them – but only if both series are available with the same series names (paths) in the master backup also.

Merging your isolated backups back into the central one Merging into the master backup simply means to copy the incremental backups. This job is done by `storeBackupMergeIsolatedBackup.pl`:

```
# storeBackupMergeIsolatedBackup.pl -f isolate-stbu.conf
in directory </media/stick/stbu/default>, copy
<2012.06.09_09.56.36>
to
<backup/default>
?
yes / no -> yes
INFO    2012.06.09 10:15:11  6557 copying data . . .
INFO    2012.06.09 10:15:11  6557 finished copying data
INFO    2012.06.09 10:15:11  6557 please run
INFO    2012.06.09 10:15:11  6557  storeBackupUpdateBackup.pl -b "backup"
```

The program uses the parameter of the option `mergeBackupDir` inserted by `storeBackupSetupIsolatedMode.pl` to get the path to the master backup. For safety reasons, it asks you if you want to copy the presented list of backups (only one in this example) to the master backup at `backup/default`. After answering `yes`, the data is copied.

To get a "normal" full backup, run `storeBackupUpdateBackup.pl -b backup`.

If you use isolated mode in the same way a second time (after re-merging backup to the master backup), you can use the option `--backupDir` of `storeBackupSetupIsolatedMode.pl` (because you already have a valid configuration file) or simply generate a new configuration file with another name (see option `--generate`) and use the old one which you may have adjusted to your needs.

7.8 Replication of backups

NOTE: If you want to use replication in your data center and if you have questions about what is possible in addition to the described scenario(s) in this chapter, please send me an e-mail. With some scripting it is possible to enhance the behavior and the possibilities for replicating, e.g. to other locations. The result of these discussions may result in better documentation and new features.

When you use storeBackup, you normally create new backups by hard linking them to older backups in the same series or maybe also to other series. You should store your backups on another disk (or even another computer) so a failure of the source disk does not destroy your backup. But what happens if the backup disk fails? – You will lose the history of your data. This kind of failure may include the hardware (the disk itself), the filesystem or a when a building burns down. A RAID does not protect against all data loss possibilities. (For more on this topic use a search engine to search the phrase *Why RAID is not a backup solution.*)

7.8.1 Quick start using storeBackup's Replication Wizard

This example is aimed at users who have at least some experience with storeBackup and who have at least one existing backup created by storeBackup. If you have zero experience with storeBackup, learn to make a master backup before you begin considering the replication of your backups. If you are an advanced user, later sections in this document will explain all the details not covered in this quick start.

storeBackup's Replication Wizard (`storeBackupReplicationWizard.pl`) will set up your environment so that you can immediately begin using storeBackup's replication features for the most typical replication scenarios. storeBackup's Replication Wizard is an interactive program. It does lots of checks and it prompts you when needed.

The Replication Wizard creates three configuration files. You can certainly set up storeBackup's replication without using the Replication Wizard and we will give you all the information you need in order to do so – see the sections below. However, in this example we want to show you how to get started with storeBackup's replication features as quick as possible (assuming your replication needs are typical).

If you do *not* run your backups with option `lateLinks` at the moment and want to use replication, *you have to enable option* `lateLinks` when using `storeBackup.pl`. See section 7.8.5 for more information. Also note that in this example, the option `lateLinks` will be set correctly by the wizard.

Now, and in the future, you will set `lateLinks=yes` in the configuration file and use these two commands (which could be put into one executable script):

```
# storeBackup.pl -f stbu.config
# storeBackupUpdateBackup.pl -b <dirOfMasterBackup>
```

In the example below <dirOfMasterBackup> will be `/masterBackup`, but in practice use your actual location. Likewise, use the actual name of your config file instead of `stbu.config`.

This example assumes we have these four different directories involved:

1. `/home` which you want to save
2. `/masterBackup` where your master Backup is located
3. `/extDisk/backupCopy` the location to which you want to replicate your master backup. (This becomes the backup copy.)
4. `/deltaCache` which is a place to keep the deltas until they are delivered to the backup copy

You need write permissions in all of these directories.

Furthermore, we assume the backup series you want to copy is named `homeBackup`. For more information about backup series, have a look at section 3, Quick Start. This example also assumes you already have backups in your master backup repository. We will copy the backups from series `homeBackup` to the replication location to seed the replication process. If you do not have any existing backups, see the other examples such as section 7.8.4 (or make a backup and then return to this section).

Let's begin:

1. copy the existing backup(s) to get a base for the replication. This can take a while.
   `# linkToDirs.pl /masterBackup/homeBackup -t /extDisk/backupCopy`
2. Next, take a quick look at the Replication Wizard help:

   ```
   storeBackupReplicationWizard.pl -h
   ```

3. Now run the Replication Wizard, telling it the location of the master backup, the deltaCache and the location to which you wish to replicate (copy) the master backup. None of those three directories is allowed to be a subdirectory of the others. See section 7.8.3, `Basic concepts to know before using storeBackup's replication` for more information.

   ```
   storeBackupReplicationWizard.pl -S homeBackup -m /masterBackup -c /extDisk/backupCopy/ \
     -d /deltaCache
   ```

   (or

   ```
   storeBackupReplicationWizard.pl --series homeBackup --masterBackupDir /masterBackup \
     --backupCopyDir /extDisk/backupCopy/ --deltaCacheDir /deltaCache
   ```

)

4. At this point you can inspect the content of the three replication-related configuration files if you wish. (They are in `/masterBackup`, `/deltaCache` and `/extDisk/backupCopy` with the extension `.conf`.) For example, check:

   ```
   cat /masterBackup/storeBackupBaseTree.conf
   cat /deltaCache/deltaCache.conf
   ```

5. Now you can run your very first backup which will be replicated:
   `# storeBackup.pl -s /home -b /masterBackup -S homeBackup --lateLinks` 0:homeBackup
   This creates a backup in `/masterBackup`. If you go there, you can see the delta files plus the command file[29] what has to be done to complete the backup. The last parameter (`0:homeBackup`) makes sure, that there are only hard links to older versions of the same backup series. Because we only want to replicate this one series, it is not possible to have cross links to other series! (This is only necessary if you have multiple backup series in your master backup.)

6. Next step you can do is to copy the deltas to the place (`/deltaCache`) where they are kept until you connect the external disk and replicate the deltas. In this step you will also complete the just made backup in the master backup directory. The following command will read the configuration file `/masterBackup/storeBackupBaseTree.conf`:
   `# storeBackupUpdateBackup.pl -b /masterBackup`

7. Now you can finish the replication by completing the backup in the backup copy:
   `# storeBackupUpdateBackup.pl -b /extDisk/backupCopy`
   Have a look into the backup copy at `/extDisk/backupCopy`. This is a complete backup now. Also, have a look `/deltaCache`. The delta was moved to `/deltaCache/processedBackups`.[30]

After you set up the environment (which the wizard did in the steps above), simply do the following steps in the future:

- Run your backups with `storeBackup.pl` like you want – but use option `lateLinks` (and restrict the hard linking references to the series you want to replicate).[31].

- run
  `# storeBackupUpdateBackup.pl -b /masterBackup`
  to complete your backups and copy the deltas to the delta cache. It is the best to do this directly after the run of `storeBackup.pl`. If you have an own server it is very easy to run this command at night via cron on the server. You may not need to include the option `--autorepair`.

- connect your external disk whenever you want (e.g., once a week), mount that drive to `/extDisk` so the path to your backup copy is `/extDisk/backupCopy`. Run
  `# storeBackupUpdateBackup.pl -b /extDisk/backupCopy`
  When it is ready, umount the external disk and disconnect it from you computer and from power. You may not need to include the option `--autorepair`.

Questions not addressed in this Replication Quick Start are covered below.

7.8.2 Why copying backups is not a substitute for replication functionality

We will make this discussion less abstract by providing an example. A common backup strategy is to run your backups to the master backup respository every day (maybe automatically via cron) and to move (or rotate) this backup offsite (or at least to another physically separate media) periodically. For illustration purposes, let's assume you have a separate backup disk stored offsite that you will update weekly.[32] Once each week you will add the newly made backups from your master backup to the backup copy on the external(offsite) disk (which you temporarily bring onsite and connect to your computer).

Copying the new backup only with "`cp -a`" is a bad idea, because the newly copied directories (backups) will not be hard linked to the existing ones on the external disk.[33] You can use `linkToDirs.pl` to link (and copy) the new backups in the master backup to the existing ones in the backup copy on the external disk. Using `linkToDirs.pl` is nice for ad hoc replications, but not the best for planned and automated ones.

[29]That's the file `.storeBackupLinks/linkFile.bz2` inside the root directory of the backup which was just created.
[30]These deltas are not deleted directly for safety reasons. You can set the storage time for this kind of data with the option `--archiveDurationDeltaCache` and `--dontDelInDeltaCache` of program `storeBackupUpdateBackup.pl`.
[31]This restriction may go away in the future
[32]If you want to realize a continuous replication to another location – that's also possible. But for this explanation, we'll assume a single external offsite disk because this is a common and suitable strategy. The mechanisms used are the same for both use cases. You can also use more than one external disk. Let's say you have two external copies and replicate the backups alternating every week to them. Then you can use the same mechanisms as for one disk. The only difference is to describe both disks in the configuration files and to connect another one each week.
[33]This means you will need *much* more space for the backup copy than for your master backup.

Another common way to copy the new backups to the external disk is to use synchronization tools like `rsync`. There are two issues with this approach: 1st, it takes *very* long if you have lots of backups and 2nd, you will replicate every fault on you master backup disk to your backup, and that's really not what you want. Imagine, your disk for the master backup gets a block error in a file from the backup one month ago. So the affected file is broken in your backup. If you now synchronize the disk with e.g. `rsync`, you will copy the broken file. In the worst of all cases, you can destroy your whole Backup by this method (without getting more security). If you use the replication from storeBackup, old data is not affected in the replication.

BUT STOP: What if the *newly copied* data is broken because some sectors of the disk are seriously broken or you have to deal with broken RAM or any other reason which leads to incorrect data in your master backup? Will you ever determine that parts of your data in the backup are broken? The backup program `storeBackup` will tell you the same as `rsync` about that – nothing, because it is not in their control. For this reason you should run `storeBackupCheckBackup.pl` which recalculates check sums for every file periodically on your backup(s). By running this program, you are able to see faults in old backups which you are able to correct manually if you have a replica. And you are able to see in an early stage if your new backups are broken. Therefore, we suggest to run `storeBackupCheckBackup.pl` on new backups every week or so on the master backup *and* on the copy plus to run it on old backups (which may take a long time) every few months.

If you recognize errors on your hard disk, you should investigate deeper into the problem and not hesitate to replace the disk.

The basic idea of storeBackup's replication feature is to solve the issues described above. A replication means we have the same state in two different locations (e.g., in the master backup and in the backup copy). That's what we have done in the description above with the `cp -a` command. Let's say, this was the backup from Monday. After a day we have a change (a new backup on Tuesday) in the master backup. For the replication, we need just *the differences* between the backup from Monday and the backup from Tuesday. If we have some clever algorithm to get all the changes (deltas) from the backup of Monday to the backup of Tuesday, we could transport these changes to the backup copy on the external disk and rebuild the full backup (with all links, permissions and so on) on the external disk. As a result, the backup on the external disk contains exactly the same information as the master backup.

If we want to connect the external disk only once a week[34] we need a place to store the differences. We will have these deltas from Monday to Tuesday, from Tuesday to Wednesday etc. What we are doing, is to rebuild the complete and full backups on the backup copy disk, e.g.:

- Backup from Tuesday $\longrightarrow$ rebuild from full backup from Monday plus Deltas from Monday to Tuesday
- Backup from Wednesday $\longrightarrow$ rebuild from full backup from Tuesday plus Deltas from Tuesday to Wednesday
- Backup from Thursday $\longrightarrow$ rebuild from full backup from Wednesday plus Deltas from Wednesday to Thursday
- ...

This means we need the deltas between two consecutive backups in the master backup. In principle, there are two ways to get these:

1. Calculation of the differences, which means something like a "reverse de-duplication" (storeBackup searches for files (or part of files) with identical contents and hard links them).
2. Identification of the differences directly when creating the backups. At that point, the differences are calculated and known.

The second way is the typical way used for replications, e.g. used in database or LDAP replication. StoreBackup also uses this way to replicate.

StoreBackup generates deltas to (one or more) existing backups with the option `lateLinks` and temporarily stores them in a "delta cache".[35] (See section 7.6 for more information about how to configure it.)

[34]or if we want a replication to another (online) location with no direct routing between the master backup and the backup copy

[35]storing in the delta cache is done only if you use replication

The storeBackup replication functionality provides the following features:

- Replication of backups takes advantage of storeBackup's existing capability to store just the differences to the former backup(s) plus the information needed to reconstruct this to a full backup.
- You can configure which series have to be replicated. At the master backup you do not have to know the number of copies you want to make. This provides decoupling of source (master backup) and target (backup copy)).
- Replication of backups can be fully automated. Along with several configuration files that are needed for replication, you simply use cron (or similar) to run storeBackupUpdateBackup.pl, see section 6.3. You also can run storeBackupDel.pl, see section 6.10 to automatically delete old backups in the replication.
- replica disk(s) do not need to be permanently attached. You can decide if you want to connect one or more of the replica disk(s) permanently or not (so you need manual intervention).) This may depend on the location you replicate to (external via WAN or via LAN) and your backup strategy.
- For security reasons, you can set up the different storage places in a way that the master backup does not have to be accessible from the copies (and vice versa).
- Replication is asynchronous. This means you may connect your disk1 for backup copy number 1 on even weekends and backup copy number 2 on disk2 on uneven weekends. But as a result you get the same data on both backup disks, *regardless of your individual deletion scheme of each backup including the master backup.*
- Deletion of old backups on the replicated backups can be done for replica(s) and master backup with the same tool (`storeBackupDel.pl`, see section 6.10). Especially, you can use different deletion schemes, e.g., to additionally store very long term backups on your (slower) replica medium(s).
- Replicas behave like normal storeBackup backups. Naturally, you can run `storeBackupCheckBackup.pl` (see section 6.12) on the replicas, too.
- It is robust. If something goes wrong with the replication (so the deltas are lost on the way to the replica backup) for whatever reason, you can use `linkToDirs.pl` to generate a new identical status (backup version) to be able to continue with replication of backups. You can do this *without* copying everything (from the master backup or from another replica) by just copying / hard linking missing delta. (But you need direct access between the affected backups / replicas during that time or you have to perform an intermediate copy.) Another advantage of this proceeding is that you do not copy possible faults in older backups from one backup copy to the other one, like you possibly would when using standard synchronization tools.

In short, if you make a "normal" backup (without replication) with `storeBackup.pl`, you typically have one place (see option `backupDir`) where you store your backups. This will be called the *master backup*. (It is the same as what we have called the *master backup repository* in other sections of this document.) If the disk (or e.g., the file) system for this "master backup" fails, you will lose the backup and therefore the *history of your data*. It is a form of data loss that can be prevented with storeBackup's replication feature.

7.8.3 Basic concepts to know *before* using storeBackup's replication

The prior subsection listed some of the main features of storeBackup's replication functionality. In the following subsection we offer a simple and typical example to have a copy of your backup data on another disk (or at another location). In subsection 7.8.6, we also offer a more advanced example.

But **first**, there are a few important conventions and concepts related to storeBackup's replication functionality that you need to be aware of. With storeBackup's replication functionality, there are four important storage locations you need to be conceptually familiar with. These four locations are normal directory trees.[36]

Of these four conceptual locations, one is the original source. The other three are related to backups or replication:

[36]In practice, the four conceptual locations can become more than four physical storage locations because replication is not limited to a single copy of a single backup.

1. “master backup”[37]
2. “backup copy”[38]
3. “deltaCache”.

None of those three directories is allowed to be a subdirectory of the others. These locations are separate directory trees.[39]

You are already familiar with what we are calling the “master backup” if you are doing any kind of backup: it is just the backup of your original data.[40]

The next important storage location for replication is the backup copy. That one is probably obvious – after all, it is the point of replication.[41]

The last of the important storage locations for replication is a cache of deltas (and meta data) used by storeBackup to provide its advanced replication functionality in the most efficient manner. We refer to this location as the “deltaCache”. The reason why there is a deltaCache is thats it allows the masterBackup to be completed (including hard linked) independently of the backup copies.

Another important replication detail to understand is that each of those backup-related directory trees must have its own configuration file in the root of the tree. The reason is that by establishing a fixed location for the configuration files, everything can be handled without additional options (or complications) by `storeBackupUpdateBackup.pl`.

In storeBackup replication, the data flow is always: masterBackup $\rightarrow$ deltaCache $\rightarrow$ (multiple) backup copy / copies.

1. “master backup” contains its own unique `storeBackupBaseTree.conf`
2. each “backup copy” directory tree contains its own unique `storeBackupBaseTree.conf`
3. “deltaCache” contains `deltaCache.conf`

The “master backup” directory tree has to contain the configuration file `storeBackupBaseTree.conf`. This config file defines which backup series has to be copied to the deltaCache.

Each “backup copy” directory tree contains a file named `storeBackupBaseTree.conf` which is its individual configuration file. It defines which backup series has to be copied to this specific backup copy directory tree.

The “deltaCache” directory tree contains `deltaCache.conf` in the root of the tree. The purpose of this configuration file is to provide one central place which denotes which backup series shall be copied to which named backup copy (physical directory paths are not used). This information is needed by `storeBackupUpdateBackup.pl` to decide if a backup can be marked as processed and, later, deleted. `storeBackupUpdateBackup.pl` needs to know who wants to copy a backup and if it has already been copied.

These config files contain some options (e.g., `backupTreeName`) for which you specify a unique identifier. Note that this parameter is simply a named reference to another location. It is not a file system path or an actual directory name. It is a unique identifiers that you can make up. This will be explained further below.

There is no information shared between two different backup copies. For a home user, this is necessary because the external disks used for replication might not always be connected. In the professional admin case it might be related to no routing for security reasons.

However, when understanding the overall concept of storeBackup replication, you might want to understand why the replication configuration uses these unique identifiers (which are not specific directory names). Why not just use the directory name? The reasons can be illustrated with two examples.

[37]It is not supported to have multiple master backups replicating to the same one and only deltaCache. Although it is untested and unsupported, it might work (with different series names).

[38]In practice there may be any number of “backup copy” directory trees.

[39]What this means is that the directory trees are not nested. None of the three replication-specific directories is allowed to be a subdirectory of the others. The deltaCache can be nested under the source tree if it is excluded from the backup via `storeBackup.pl` options.

[40]In storeBackup, a “master backup” is a backup series (or, potentially, a set of backup series). It is called a series because this directory will hold a series of backups (e.g., one each day) for your computer. See section 5.4.1

[41]Replication can be used to produce multiple copies of the master backup at different locations.

First, consider the case of somebody who wants to make two backup copies (replicas) to two different external disks, one on odd weekends and one on even weekends. Assume they would be mounted at the same mount point. The most elegant way for storeBackup has to manage the alternation of these two different copies is via these unique identifiers. In this example, imagine you have unique identifiers named CopyA and CopyB. This allows storeBackup to know whether each one was completed (copied + hard linked) so it can be moved to processedBackups – even if a backup was interrupted, etc. Other implementations would not be as advantageous.

Another example would be a sysadmin who wants to make two replications, one in the same data center and the other one in a remote data center. He sets up a server for that in the same data center which pulls its data from the deltaCache via some mount points. In the remote data center, he sets up another server in the same way. Using unique identifiers in storeBackup's replication configuration (so it is decoupled from the physical directory) makes this administration easier.

The configuration file of deltaCache doesn't know the directory where the backup copy is located. Instead, the configuration file knows only a name (unique identifier), which is more flexible. If you change the directory of the backup copy, you do not have to change the deltaCache configuration file. And, as illustrated in the examples above, you have have two unique identifiers pointing to the same physical path to facilitate rotation of backup copies.

You will probably have at least four separate configuration files with your storeBackup replication setup. These are the three files mentioned above and your normal `storeBackup.pl` config file[42].

The use of replication can affect two options of storeBackup.pl: `--lateLinks` and `--otherBackupSeries`. If you do not run your backups with the option `lateLinks` at the moment and want to use replication, *you have to enable the option* `lateLinks` when using `storeBackup.pl`. However, there is no real disadvantage in using it. It simply splits the full backup process into two steps without otherwise altering anything that would be done without this option.

You also need to be aware of the option `--otherBackupSeries` in the main config file and how this relates to the potential need for using a command line parameter (e.g. `0:homeBackup` as shown in the example below) with `storeBackup.pl`.

If you want to replicate one backup series only, it is not possible to have cross links to other backup series. This restriction only applies, of course, if you have multiple backup series [e.g., different computers] in your master backup. From a series which is replicated, you cannot refer to series **not** being replicated to the same backup copy. (But, conversely, from a series which is not replicated, you can refer to any series being replicated.)

This restriction might go away in the future. (This would mean that the unresolvable files have to be added to the deltas (for deltaCache) when running `storeBackupUpdateBackup.pl` on the master backup.)

In short, to keep it simple and to set up replication the first time, make sure that there are only hard links to older versions of the **same backup series**. Anything where you have links in the master backup you also have to have in the backup copy, so the same links can be established. If you replicate all series, you do not have to change anything about hard linking.

This is all very simple, but only if you understand what's happening. (And naturally, the situation is somewhat more complicated if you replicate different series (overlapping) to different backup copies.)

When running `storeBackupUpdateBackup.pl` on the backup copy, `autorepair` is switched on by default (but generates `INFO` entries only, no `ERROR` messages).

7.8.4 Understanding storeBackup's Replication Wizard via an example

If you have done the Replication Wizard Quick Start, you probably do not need to go through this example in detail. This example is useful if your needs are atypical (which means the Replication Wizard Quick Start wasn't applicable) and you need to become familiar with the Replication Wizard. This example will help you to become familiar with the Replication Wizard quickly so you can move on to more advanced configurations.

The Replication Wizard creates three configuration files. This simple example will help you understand the files created by the Replication Wizard as well as how the Replication Wizard works.

[42]You can also use command line options, but finally that's more complicated.

The following command line example will demonstrate the Replication Wizard by taking you through a complete backup and replication using some temporary files.

To keep this example simple, we will be using the default series. For more information about backup series, have a look at section 3, Quick Start.

The files you will back up will be in `/tmp/repliTest/localDisk/sourceFiles`. The backup will be in `/tmp/repliTest/externalDisk_1/masterBup`. `/tmp/repliTest/externalDisk_2/copyBup` locates the replicated backup. Normally, the replicated backup would be on another disk such as a USB HDD or another server. That is the main way this example differs from what you would do in real life (the other ways it differs are: 1) we are backing up just a few example files, 2) we aren't using `storeBackup.pl`'s main configuration file, and 3) we are not copying an existing backup to "seed" the replica.)

First, set up some temporary files to back up. The contents are not important. This is just an example.

```
mkdir -p /tmp/repliTest/localDisk/sourceFiles /tmp/repliTest/localDisk/deltaCache \
  /tmp/repliTest/externalDisk_1/masterBup /tmp/repliTest/externalDisk_2/copyBup
cd /tmp/repliTest/
cp /bin/ls /tmp/repliTest/localDisk/sourceFiles
touch /tmp/repliTest/localDisk/sourceFiles/test.txt
ls -la /tmp/repliTest/localDisk/sourceFiles
```

We assume `storeBackup/bin` is in your path. If not, create symbolic links as shown in section 1,Super Quick Start. If needed, run these two commands (the 2nd line ends with: space,dot) in a terminal:

```
cd /usr/local/bin
ln -s /opt/storeBackup/bin/* .
cd -
```

Second, do the initial backup using the option `lateLinks`. This gives you something to replicate.

```
storeBackup.pl -s /tmp/repliTest/localDisk/sourceFiles/ \
  -b /tmp/repliTest/externalDisk_1/masterBup/ --lateLinks
```

You can expect one warning during the backup:

```
WARNING   2012.07.21 16:12:11 12580 created directory <backup//default>
```

Next, take a quick look at the Replication Wizard help:

```
storeBackupReplicationWizard.pl -h
```

Now run the Replication Wizard, telling it the location of the master backup, the deltaCache and the location to which you wish to replicate (copy) the master backup. None of those three directories is allowed to be a subdirectory of the others. And normally, the location for the copy (of the master backup) would be an external disk or another server. (The delta cache can be on the same disk where the source resides.) See section 7.8.3, Basic concepts to know before using storeBackup's replication for more information.

```
storeBackupReplicationWizard.pl -m /tmp/repliTest/externalDisk_1/masterBup/ \
  -c /tmp/repliTest/externalDisk_2/copyBup/ -d /tmp/repliTest/localDisk/deltaCache
```

Because you didn't use the series name as an argument of option `-S`, you will be prompted as follows:

```
found series <default>
replicate it?
```

Answer `yes` to the prompt and the wizard completes (if you had specified the `--series` option, the wizard would not have prompted you).

At this point you could inspect the contents of the three replication-related configuration files, if you wish. (They are in `/tmp/repliTest/externalDisk_1/masterBup`, `/tmp/repliTest/localDisk/deltaCache` and `/tmp/repliTest/externalDisk_2/copyBup` with the extension `.conf`.) For example, see:

```
cat /tmp/repliTest/externalDisk_1/masterBup/storeBackupBaseTree.conf
cat /tmp/repliTest/localDisk/deltaCache/deltaCache.conf
```

You can also inspect the current files before your backup is replicated.

```
find /tmp/repliTest/ -print | sort
```

The last step is to finish the backup using `storeBackupUpdateBackup.pl` in the same way you normally would do that with the option `lateLinks`. By virtue of the configuration files the Replication Wizard has created, this step will now replicate (copy) your master backup to the replication location you have specified. Later, you can put the following two commands into a script and you probably set them up to run in a cron job.[43]

```
storeBackupUpdateBackup.pl -b /tmp/repliTest/externalDisk_1/masterBup/
storeBackupUpdateBackup.pl -b /tmp/repliTest/externalDisk_2/copyBup/
```

You can now inspect the files again and see that your backup was replicated (notice `deltaCache/processedBackups` too).

```
find /tmp/repliTest/ -print | sort
```

As we mentioned, storeBackup's Replication Wizard is an interactive program. If you wish to test this, use it with different options or try it with a broken environment. For example, try it with a non-existing master backup in an example similar to the above.

7.8.5 A simple replication example without the Replication Wizard

Now that you are familiar with the Replication Wizard, this section will deepen your understanding so you can move on to more advanced configurations. If your replication needs are simple and typical, you may not need to read this section.

If you do not run your backups with the option `lateLinks` at the moment and want to use replication, *you have to enable it when using* `storeBackup.pl`. This means in practice, that you have previously been running something like:

```
# storeBackup.pl -f stbu.config
```

Now, and in the future, you will set `lateLinks=yes` in the configuration file and use these two commands (which could be put into one executable script):

```
# storeBackup.pl -f stbu.config
# storeBackupUpdateBackup.pl -b <dirOfMasterBackup>
```

Here, `storeBackup.pl` will create the deltas described above which are stored in your master backup. If you look into this data, you will see that there are e.g., no hard links to already existing files in the previous backup. After running `storeBackupUpdateBackup.pl` all the "missing" steps (like linking, changing permissions) are done. It is nothing else than splitting the work `storeBackup.pl` does as an all-in-one application into two different steps.

So the result of these two commands will be exactly the same – a full backup like before (when you didn't use option `lateLinks` and ran one command). The batch above is just a simple example; you can also run `storeBackupUpdateBackup.pl` e.g., on your server at a later time (see section 7.6 for more information). Naturally, if you want to use replication, you have to configure it first. The Replication Wizard can do this for you. However, in this section we demonstrate the manual steps.

If we use the simple example above, replication to the backup copy on the external disk works as follows. I assume we have four different directories involved:

1. `/home` which you want to save
2. `/masterBackup` where your master Backup is located

[43] For the second one, you should double check if the external disk is connected.

3. `/extDisk/backupCopy` where you want to copy your master backup to (the backup copy)
4. `/deltaCache` which is a place to keep the deltas until they are delivered to the backup copy (`/extDisk/backupCopy`)

I also assume the backup series you want to copy is named `homeBackup`. You need write permissions for all of these directories (for `/home` only read permissions are required).

1. copy the existing backup(s) to get a base for the replication:
   ```
   # linkToDirs.pl /masterBackup/homeBackup -t /extDisk/backupCopy
   ```
2. Now you have to create a configuration file in your master backup to tell storeBackup to do the replication:
   ```
   # storeBackupUpdateBackup.pl --genBackupBaseTreeConf /masterBackup
   ```
3. edit the generated configuration file `/masterBackup/storeBackupBaseTree.conf` so it has the following contents:[44]

   ```
   backupTreeName=myMasterBackup
   backupType=master
   seriesToDistribute=homeBackup
   deltaCache=/deltaCache
   ```

   In the master backup configuration file, a value for `backupTreeName` is only needed for error messages, warnings and so on. it is mostly there for future enhancements, so all directories will have a unique identifier.

   You can change the unique identifier for the parameter `backupTreeName` to whatever you want (here, `myMasterBackup` was chosen). But you have to set `backupType` to `master`!
4. Now you can run your very first backup which will be replicated:
   ```
   # storeBackup.pl -s /home -b /masterBackup -S homeBackup --lateLinks 0:homeBackup
   ```
   This creates a backup in `/masterBackup`. If you go there, you can see the delta files plus the command file[45] what's to be done to complete the backup. The last parameter (`0:homeBackup`) makes sure that there are only hard links to older versions of the same backup series. As we only want to replicate this one series it is not possible to have cross links to other series! (This is only necessary if you have multiple backup series in your master backup.)
5. Next step you can do is to copy the deltas to the place (`/deltaCache`) where they are kept until you connect the external disk and replicate the deltas. In this step you will also complete the just made backup in the master backup directory. The following command will read the configuration file `/masterBackup/storeBackupBaseTree.conf`:
   ```
   # storeBackupUpdateBackup.pl -b /masterBackup
   ```
6. Now you need to generate the configuration file for the delta cache. The just started command has copied the deltas from your master backup to this place (you should explore directory `/deltaCache` to see what has happened).
   ```
   # storeBackupUpdateBackup.pl --genDeltaCacheConf /deltaCache
   ```
7. edit the generated configuration file `/deltaCache/deltaCache.conf` so it has the following contents:[46]

   ```
   backupCopy0=myBackupCopy homeBackup
   ;backupCopy1=
   ;backupCopy2=
   ```

   Do not change the other commented keywords `backupCopy1` to `backupCopy9` because we only one replication. (The delta cache is the central distribution place for all defined replications.) The entry `myBackupCopy` is just a name (not a path) for the copied backup on your external disk. You can choose any name you want, but it has to be exactly the same as in the configuration file for your backup copy at `/extDisk/backupCopy`. After "myBackupCopy" you have to enter the list of series you want to replicate. It is only the series `homeBackup` in this example.

[44]The "rules" for the configuration file are the same as for all other configuration files.
[45]That's the file `.storeBackupLinks/linkFile.bz2` inside the root directory of the backup which was just created.
[46]The "rules" for the configuration file are the same as for all other configuration files.

8. Next, you have to tell your backup copy which data it should add to the backup and where this data is located. Generate a configuration file for that:
   `# storeBackupUpdateBackup.pl --genBackupBaseTreeConf /extDisk/backupCopy`

9. Edit the generated configuration file `/extDisk/backupCopy/storeBackupBaseTree.conf` so it has the following content:[47]

   ```
   backupTreeName=myBackupCopy
   backupType=copy
   seriesToDistribute=homeBackup
   deltaCache=/deltaCache
   ```

   The name of the backup must be the same as specified in the configuration file of the deltaCache which is loated at `/deltaCache/deltaCache.conf`. The `backupType` must be '`copy`', so the program `storeBackupUpdateBackup.pl` knows it has to copy the deltas *from* the deltaCache.[48]

10. Now you can finish the replication by completing the backup in the backup copy:
    `# storeBackupUpdateBackup.pl -b /extDisk/backupCopy`

    Have a look into the backup copy at `/extDisk/backupCopy`. It is a complete backup now. Also, have a look `/deltaCache`. The backup was moved to `/deltaCache/processedBackups`.[49]

After you set up the environment, simply do the following:

- Run your backups with `storeBackup.pl` like you want – but use the option `lateLinks` (and restrict the hard linking references to the series you want to replicate).[50]
- Run
  `# storeBackupUpdateBackup.pl -b /masterBackup`

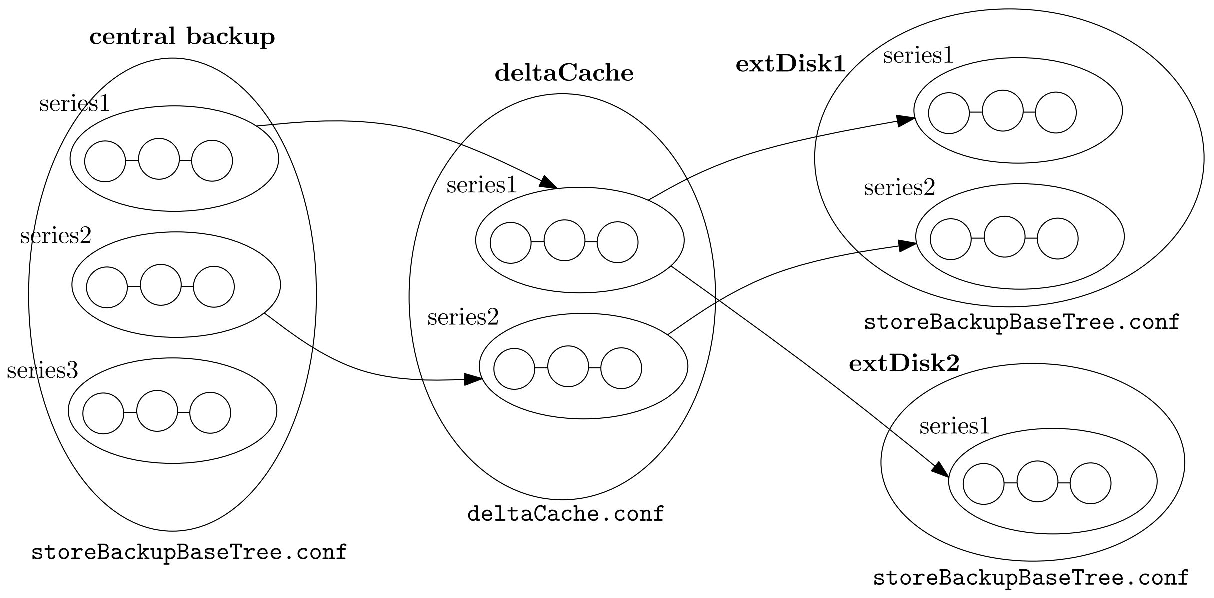
  to complete your backup copy and to replicate the deltas to the delta cache. The best is to do this directly after the run of `storeBackup.pl`. If you have an own server it is easy to run this command at night via cron on the server.
- Connect your external disk whenever you want (e.g., once a week), mount that drive to `/extDisk` so the path to you backup copy is `/extDisk/backupCopy`. Run
  `# storeBackupUpdateBackup.pl -b /extDisk/backupCopy`

  When it is ready, umount the external disk and disconnect it from you computer.

7.8.6 How storeBackup's replication works

If you wish to create advanced replication configurations, this section will be of interest to you. The following picture shows the principle of the data flow when replicating backups:

[47]The "rules" for the configuration file are the same as for all other configuration files.
[48]As opposite to value '`master`' which tells `storeBackupUpdateBackup.pl` to copy the deltas *to* the deltaCache.
[49]These deltas are not deleted directly for safety reasons. You can set the storage time for this kind of data with the option `--archiveDurationCopyStation` and `--dontDelInCopyStation` of the program `storeBackupUpdateBackup.pl`.
[50]This restriction may become obsolete in the future.



Important: Before you can start replicating backups, you have to copy a common version of the backup from the master backup to the locations where the copies will reside. The reason for that is that replication is based on incremental backups (option `lateLinks` of `storeBackup.pl`), and they always have to be linked against the backup version to which these incremental backups were generated. You can use `cp -a` or (better) `linkToDirs.pl` to copy these (old) backups.

If your master backup is located at `/masterBackup` and you want to copy the series `series1`, `series2` and `series3` to `/extDisk1/stbu` (which will be the top level of your backup copy), you can run:[51]

```
# linkToDirs.pl /masterBackup/series1 /masterBackup/series2 /masterBackup/series3 \
  --targetDir /extDisk1/stbu
```

In the example shown in the picture above, there is a master backup where all the backups are stored. In this master backup, there are three different series called `series1`, `series2` and `series3`. In each of these series, there are three incremental backups (shown as circles) created with option `lateLinks` and which have to be replicated.
The configuration file located in the top level directory of the master backup (`storeBackupBaseTree.conf`) has been configured that two (`series1`, `series2`) of the three series have to be copied onto the external disks. The configuration file would look e.g., as follows (without comments):

```
backupTreeName=myMasterBackup
backupType=master
seriesToDistribute=series1 series2
deltaCache=/deltaCache
```

You can generate this type of configuration file by typing
`storeBackupUpdateBackup.pl --genBackupBaseTreeConf` *directory*
where *directory* is the top level directory of your master backup. After its generation, edit the configuration file depending to your needs.

The example configuration above tells storeBackup (to be precise `storeBackupUpdateBackup.pl`) that you are calling your master backup `myMasterBackup` (choose a name whatever you want, it has nothing to do with a directory name – it is a unique identifier you choose and which will point to an actual directory) and that this is your `master` backup. The configuration file also defines, that `series1` and `series2` have to be copied and that the central hub, your "deltaCache" is in directory `/deltacache`. After copying the incremental backup via `storeBackupUpdateBackup.pl` just as described above, the same run of that program starts generating the missing hard links (etc.) in the master backup so the incremental backups become full backups.

In this example, I assume that the server (or generally the box) to save has two internal disks. On `/`, the file system and the user data are located, while the second disk, mounted at `/backup` stores the master

[51] `linkToDir.pl is delivered with storeBackup`

backup (and maybe data you do not want to backup). To locate the "deltaCache" on the first disk which has to be saved makes sense: If the first (operating system) disk is broken, you can use the backup disk to restore it later. If the backup disk is broken, you still have the delta to the external disks on the first one. Naturally, if both are broken at the same time, you only have the external ones and may lose some days (if you do not do a daily sync).

In the deltaCache, you have to generate (see below) and configure a configuration file named `deltaCache.conf` in the top level directory with the following contents:

```
backupCopy0= 'extDisk1' series1 series2
backupCopy1= 'extDisk2' series1
;backupCopy2=
;backupCopy3=
;backupCopy4=
;backupCopy5=
;backupCopy6=
;backupCopy7=
;backupCopy8=
;backupCopy9=
```

You have to define two copy targets because in this example you want to make two copies. The first line (`backupCopy0`) in the configuration file above defines that `series1` and `series2` have to be copied to the backup copy with the unique identifier `extDisk1`. The second line (`backupCopy1`) tells storeBackup to copy `series1` to the backup with the unique identifier `extDisk2`.

You can generate this type of configuration file by typing

`storeBackupUpdateBackup.pl --genCopyStationConf` *directory*
where *directory* is the top level directory of your deltaCache. After its generation, edit the configuration file depending to your needs.

Finally, you have to edit a configuration file for the place where you want to replicate your data. As (in this example) you want to replicate to two different storage systems (called `extDisk1` and `extDisk2`) you have to generate two configurations files in the top level of these replicas:

`storeBackupUpdateBackup.pl --genBackupBaseTreeConf` *directory*

Where *directory* is the top level directory of your backup copy. The generated configuration file with the name `storeBackupBaseTree.conf` will be stored in that top level directory. After its generation, edit the configuration file depending to your needs; in this example it is:

```
backupTreeName=extDisk1
backupType=copy
seriesToDistribute= series1 series2
deltaCache=/deltaCache
```

The Options `backupTreeName` and `seriesToDistribute` must fit to the corresponding entry of `deltaCache.conf` (see above, `backupCopy0`).

When you call:
`storeBackupUpdateBackup.pl --backupDir` *directory*

where *directory* is the top level backup directory (with the series below), it will copy the backups in deltaCache from the configured series to the specified location.

You might want to add some other options to `storeBackupUpdateBackup.pl`, e.g.:

`storeBackupUpdateBackup.pl --progressReport 200 --archiveDurationCopyStation 32d -b` *directory*

This means, it will print a progress report and delete backups in deltaCache after a month, *but only if they were delivered and hard linked successfully to the replica backups.*

Finally, you have to repeat the last step of configuring a configuration file for `extDisk2`, too:

```
backupTreeName=extDisk2
backupType=copy
seriesToDistribute= series1
deltaCache=/deltaCache
```

That's it. Now each run of `storeBackupUpdateBackup.pl` on the master backup will copy the required backup deltas (= backups generated with `lateLinks`) and finalize the backups to be complete ones (not incremental ones). Running `storeBackupUpdateBackup.pl` on the replica backups (extDisk1 and extDisk2) will copy those deltas from the deltaCache to the selected replica. Finally, one of those calls of `storeBackupUpdateBackup.pl` will delete those deltas from deltaCache (depending on option `--archiveDurationCopyStation`).

7.8.7 Using Wildcards for Replication

If you want to use wildcards to configure the replication, be sure you understand the principles the replication is based on before reading this chapter.

Configuring the hard link Option `otherBackupSeries` using Wildcards

To make the result of wildcard usage simple and transparent, `storeBackup.pl` also prints the result of wildcard expansion into the log files.
Imagine you want to hard link all series which we want to replicate in a second step. To make this easy to configure, every series *except* the ones with a name ending with `.norepl` have to be replicated. I assume that each series name is the name of the server (`server1` and `server2`). Therefore, the names of the series to replicate is simply `server1` and `server2`, the name of the series *not* to replicate is `server1.norepl` and `server2.norepl`. You want to hard link each series with each other, but naturally, you do not want to get hard links from a replicated series to a not-replicated ones. This would result in error messages in the replication. You can use the following configurations in the configuration files:

for *server1* → `otherBackupSeries = 0:* -:*.norepl`
for *server2* → `otherBackupSeries = 0:* -:*.norepl`

for *server1.norepl* → `otherBackupSeries = 0:*`
for *server2.norepl* → `otherBackupSeries = 0:*`

Alternatively, you can use also the following syntax:

for *server1* → `otherBackupSeries = +0:* -:*.norepl`
for *server2* → `otherBackupSeries = +0:* -:*.norepl`

for *server1.norepl* → `otherBackupSeries = +0:*`
for *server2.norepl* → `otherBackupSeries = +0:*`

The "`+`" sign is optional. You can also use the plus and minus sign without wildcards, but especially the minus sign doesn't make sense in that case.[52]

Starting `storeBackup.pl` (in this example command line options are used) prints the following log:[53]

```
$ storeBackup.pl -s s -b b -S server1 '0:*' -- '-:*.norepl'
....
INFO      2014.02.22 10:02:20  6822 consider series <*>:
INFO      2014.02.22 10:02:20  6822     consider series <server1>
INFO      2014.02.22 10:02:20  6822     consider series <server1.norepl>
INFO      2014.02.22 10:02:20  6822     consider series <server2>
INFO      2014.02.22 10:02:20  6822     consider series <server2.norepl>
INFO      2014.02.22 10:02:20  6822 avoid series <*.norepl>:
INFO      2014.02.22 10:02:20  6822     avoid series <server1.norepl>
INFO      2014.02.22 10:02:20  6822     avoid series <server2.norepl>
INFO      2014.02.22 10:02:20  6822 resulting series to hard link
INFO      2014.02.22 10:02:20  6822     series <server1>
INFO      2014.02.22 10:02:20  6822     series <server2>
....
```

The only difference between the two different cases (see e.g. *server1* and *server1.norepl*) when configuring `otherBackupSeries` is "`-:*.norepl`". If you want to generate the configuration files, you have to differentiate between the series to replicate and the ones not to replicate. But when generating a configuration file, you have to know if you want to replicate that series or not. So that shouldn't be a problem. The advantage you get by using wildcards is the possibility to group series without having to know (and *complete*) the series names all the time a new series is added.

[52]If you use the minus sign on the command line, do not forget to mask it with `--`, so storeBackup knows it is not an option!
[53]The series directories have to exist before starting the command so the wildcards can be expanded!

Naturally, you can also use more than two different kinds of series to hard link by choosing useful names. This is just a simple example to show the principles.

Configuring Replication options `seriesToDistribute` and `backupCopy*` using Wildcards

If you want to dynamically replicate backup series (and maybe avoid replication for others, like in this example), you can use wildcards when configuring replication. In the example the following directories in `/tmp` are used to explain the usage of wildcards:

`/tmp/a/b` master backup directory

`/tmp/a/d` delta cache directory

`/tmp/a/c` replication ("copy") directory

Now you have to create some backup series directories and to generate the configurations files for the replication:

```
$ cd /tmp/a
$ mkdir server1 server1.norepl server2 server2.norepl
$ storeBackupReplicationWizard.pl -m b -c c -d d -S server1
```

In the next step you have to edit the configuration files with the following results:

```
$ grep -vP '\A\s*\Z|\A[#;]' b/storeBackupBaseTree.conf
backupTreeName='Master Backup'
backupType=master
seriesToDistribute= +* -*.norepl
deltaCache=/tmp/a/d

$ grep -vP '\A\s*\Z|\A[#;]' d/deltaCache.conf
backupCopy0='Backup Copy' +*

$ grep -vP '\A\s*\Z|\A[#;]' c/storeBackupBaseTree.conf
backupTreeName='Backup Copy'
backupType=copy
seriesToDistribute= +*
deltaCache=/tmp/a/d
```

Important: Remember, if you run your backups with `storeBackup.pl`, you *have to* use option `lateLinks` when using replicaton!
As you see, the syntax is very similar to the one used with `otherBackupSeries`. Instead of "+∗" you can also write "∗".
The configuration above may show the following error message when starting the replication program on the replication copy directory:

```
$ storeBackupUpdateBackup.pl -b c
....
ERROR     2014.02.22 11:59:48 10007 c/storeBackupBaseTree.conf series <server1> missing in <Backup Copy>, defined in /tmp/a/d/deltaCache.conf
ERROR     2014.02.22 11:59:48 10007 use option --createNewSeries if you want missing series to be created automatically
....
```

The only chance for storeBackup to compare the series to distribute from deltaCache to the replication copy when using wildcards is to expand the wildcards. But when running the replication for a new series the very first time, the newly replicated series naturally does not exist in the replication copy directory. You can create it manually with `mkdir` (which is probably not what you want) or do, what `storeBackupUpdateBackup.pl` tells you to do – create the new series automatically:

```
$ storeBackupUpdateBackup.pl -b c --createNewSeries
```

7.9 Special files generated and used by storeBackup

Never change the files described below! They are absolutely important for storeBackup to work properly! **Inside a backup, the following entries are always created. Do not delete them.** Also, make sure you do not have them in the top level directory of your source tree:

`.md5CheckSums.info` This file contains meta information about the backup. Example (I cut some lines for better readability):

```
version=1.3
date=2008.09.06 10.23.33
sourceDir='/home/hjc'
followLinks=0
compress='bzip2'
uncompress='bzip2' '-d'
postfix='.bz2'
exceptSuffix='\.bz2' '\.gif' '\.gpg' '\.gz' '\.jpg' '\.mp3' '\.mpeg' '\.mpg' '\.ogg'
exceptDirs='/home/hjc/Mail' '/home/hjc/Maildir' '/home/hjc/nosave' '/home/hjc/tmp'
includeDirs=
exceptRule='$size > &::SIZE("100M")'
includeRule=
exceptTypes=
preservePerms=yes
lateLinks=yes
lateCompress=yes
cpIsGnu=yes
```

`.md5CheckSums[.bz2]` This file contains all information about the files, directories etc. in the backup. A few lines selected as an example:

```
# contents/md5 compr dev-inode inodeBackup ctime mtime atime size uid
gid mode filename
dir 0 2097-386 0 1169342164 1094800914 1200948038 0 1049 100 493 c++
063e5feb114a82059e7f44c5fb0e548c c 2097-1834 1372638 1169343033 1078512595 1125554314 489786 1049 1001 384 mbox
symlink 0 2097-31105 0 1169350675 1169350675 1169350675 0 1049 0 0 .Xresources
```

The permissions (mode) are stored as decimal values (not octal).

`.storeBackupLinks` A directory which is empty if all links are set.

These files may be in the root of your backup directory:

`.md5CheckSums.notFinished` Up to version 3.4.3: The existence of this file indicates that this backup was not properly finished (e.g. if it was stopped by pressing [ctrl]+[c]).

`.storeBackupLinks/backup.Finished` Backups created with version 3.5 or higher: The existence of this file indicates, that this backup was properly finished. If it is missing, the backup was not finished properly (means it was e.g., stopped by pression control-c).

`.storeBackup.log[.bz2]` The log file of storeBackup.pl. This is the default name which you can change using the options of storeBackup.pl (options logInBackupDir and compressLogInBackupDir).

`.storeBackup.notSaved.bz2` If you exclude files with rules, you can generate a list of files (via the option writeExcludeLog of storeBackup.pl) which are *not* stored in the backup.

The following files only exist if you use option lateLinks, see section 7.6. After a successful run of storeBackupUpdateBackup.pl, see section 6.3, these files are deleted:

`.storeBackupLinks/linkFile.bz2` Contains (parts) of the information what has to be done by storeBackupUpdateBackup.pl (beside file `.md5CheckSums[.bz2]`).

`.storeBackupLinks/linkTo` Contains relative paths to the backups where `linkFile.bz2` refers to, eg:

```
../2008.09.05_16.07.23
../../lotte/2008.09.06_02.00.04
```

Here you see a relative path to a previous backup and a link to a backup in another backup series.

`.storeBackupLinks/linkFrom<number>` Each file contains relative paths *from* backups to the actual one. Example:

```
../2008.09.06_10.23.33
```

7.10 Configuring NFS

Let's assume that your server where you want to write your backup via NFS is called 'nfsserver' and the path to the backup is /storeBackup. You can then use the following entry in `/etc/exports` on `nfsserver` (example with GNU/Linux, can differ on other Unix like operating systems):

```
/storeBackup 192.168.1.0/24(async,rw,no_root_squash)
```

`192.168.1.0/24` means, that access from any ip address beginning with `192.168.1` is allowed.
You should run

```
# exportfs -a
```

to make your entry visible to NFS. With

```
# exportfs -v
```

you can see how NFS is configured.

You probably have to change the ip address and the mask to your needs. Using `no_root_squash` is important for the client root user to have root permissions on the mounted file system. Use `async` to get a much better write performance (see `man mount` for further explanations). If you use `async`, storeBackup will not be able to detect if a file systems capacity exceeds.
In /etc/fstab on the NFS client (where you run storeBackup) you should configure a line like

```
nfsserver:/storeBackup /backup nfs user,exec,async,noatime 1 1
```

This will mount the file system `/storeBackup` of `nfsserver` to `/backup` on your client. This will occur if you boot or if you type:

```
# mount /backup
```

on the NFS client.

There are many other options with NFS. This short description only tries to give some helpful hints, not to explain NFS.

Read or write access?

You probably want write access for storeBackup.pl but only read access for the users. There are at least two ways to achieve this:

1. Mount the specific NFS directory for the backup (e.g., `/backup`) read only. In the configuration file for storeBackup, use:

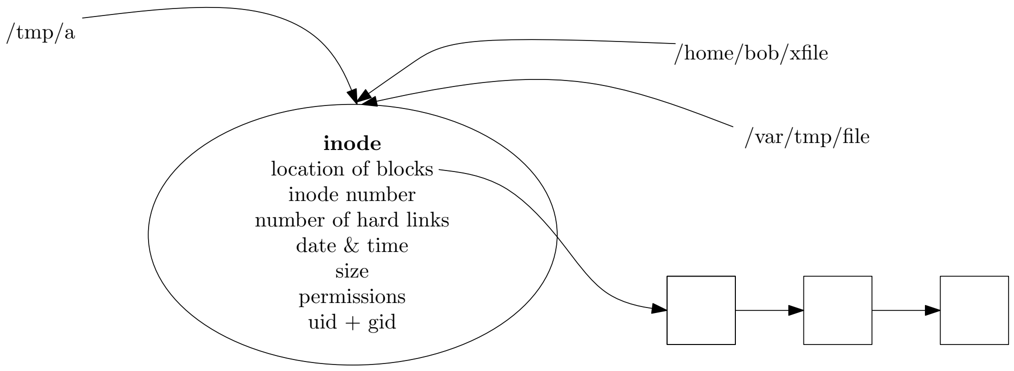
   ```
   precommand = mount /backup -o remount,rw
   postcommand = mount /backup -o remount,ro
   ```

   This will give storeBackup.pl write access (rw = read write) during the backup. Naturally, you can also wrap a script around storeBackup.pl that does the same. The disadvantage of this method is, that the users also will get write access during backup.

2. Simply arrange two mount points to the NFS server: one rw and one ro. Limit access to the read write (rw) mount to root. You can also mount the mount points for storeBackup.pl only for the time during the backup. You can use storeBackupMount.pl, see section 6.11 for this.

7.11 What is an inode

In Unix like system, the central element of a file system entry is an inode (index node):



This inode contains several meta information and the location of that file on the disk (that part of the picture does absolutely not reflect the real situation).
In the figure above, you see three hard links ("file names") for this inode, so the number of hard links to this inode would be 3. We can see this with `ls -li`:

```
$ ls -li /tmp/a /home/bob/xfile /var/tmp/file
114693 -rw-r--r-- 3 hjc hjc 5 Sep 6 13:55 /tmp/a
114693 -rw-r--r-- 3 hjc hjc 5 Sep 6 13:55 /home/bob/xfile
114693 -rw-r--r-- 3 hjc hjc 5 Sep 6 13:55 /var/tmp/file
```

The first digits form the inode number.
Naturally, all hard links to an inode must be in the same file systems. If the different directories in the example above are not in the same file system on your computer, you cannot configure exactly that example. To create a hard link on the shell you have to use the command `ln`.

All these entries for that inode have the same permissions, gid and uid. See section 7.14, Limitations to understand what this means for storeBackup.

Btw., you cannot set a hard link to a directory because this could result in infinite loops. Nevertheless, directory entries also use the feature of having multiple names for one inode: Take a look at the directory "name", "." and ".." (the last perhaps multiple times)!

7.12 Statistical Output of storeBackup.pl

After creating a new backup and possibly deleting old ones, storeBackup will write some statistical output:

`directories` Number of directories storeBackup found in the data source and created in the backup.

`files` Number of files (more exactly: the number of links) storeBackup found in the data source. This includes all types of files storeBackup is able (or configured) to process.

`symbolic links` Number of symbolic links storeBackup found in the data source.

`name pipes` Number of named pipes storeBackup found in the data source.

`new internal linked files` Number of files with the same contents storeBackup found in the actual backup (not in an old one). This is checked first.

`old linked files` Number of files which exists in the previous backup with the same name, same size, same ctime and same mtime.

`unchanged files` Number of files with the same contents storeBackup found in the old backup(s).

`copied files` Files with new contents, copied to the backup directory.

`compressed files` Files with a new contents, compressed into the backup directory.

`excluded files because pattern` Files excluded because of option 'exceptPattern'.

`included files because pattern` Files included because of option 'includePattern'.

`max size of copy queue` Maximum size of copy queue during the backup.

`max size of compress queue` Maximum size of compress queue during the backup.

`calculated md5 sums` Number of files for which an md5 sum was calculated.

`forks total` Total number of forks (number of forks md5 + forks compress + forks copy + forks named pipes).

`forks md5` Number of forks for program md5sum.

`forks copy` Number of forks for program cp.

`forks <compress>` Number of forks for program `<compress>`

`sum of source` Size in bytes of all files in the source directory.

`sum of target all` Size in bytes of all files in the target directory.

`sum of target new` Size in bytes of new copied or compressed files in the target directory.

`sum of md5ed files` Size in bytes of all files for which an md5 sum was processed.

`sum internal linked (copy)` Size of bytes of all files which were internal linked (see: new internal linked files). These files were linked to files which were copied into the backup.

`sum internal linked (compr)` Size in bytes of all files which were internal linked (see: new internal linked files). These files were linked to files which were stored compressed into the backup.

`sum old linked (copy)` Size in bytes of all files which were linked to older backups (see: old linked files). These files were linked to files which were copied into the backup.

`sum old linked (compr)` Size in bytes of all files which were linked to older backups (see: old linked files). These files were linked to files which were stored compressed into the backup.

`sum unchanged (copy)` Size in bytes of all files which existed with the same name, mtime and atime in the previous backup. These files were linked to files which were copied into the old backup.

`sum unchanged (compr)` Size in bytes of all files which existed with the same name, mtime and atime in the previous backup. These files were linked to files which were stored compressed into the old backup.

`sum new (copy)` Size in bytes of all files which were copied into the backup.

`sum new (compr)` Size in bytes of all files which were stored compressed into the backup.

`sum new (compr), orig size` Size in bytes in the source directory of the above files.

`sum new / orig` Percentage of new files in the backup to their original size in the source directory.

`size of md5CheckSum file` Size of the file ¡backupDir¿/.md5CheckSums[.bz2].

`size of temporary db files` Size of the db files generated during the backup in tmpdir.

`deleted old backups` Number of old backups which were deleted.

`deleted directories` Number of directories deleted in old backups.

`deleted files` Number of files truly deleted in old backups (last link removed).

`(only) removed links` Number of links removed in old backups (files not deleted).

`freed space in old directories` Freed space in old directories, does not include meta information.

`add. used space in files` Additionally used space for this backup: difference between new allocated space and freed space in old backups.

`backup duration` Backup duration: time for precommand, backup, postcommand and deletion of old backups.

`over all files/sec (real time)` Number of files divided by real time.

`over all files/sec (CPU time)` Number of files divided by (user and system time).

`CPU usage` Average cpu time for the time period of ”backup duration”.

`PROGRESS 2009.05.09 10:16:43 22774 5000 files processed (324M, 152M) (340234099, 159903981)`
storeBackup read 5000 files so far. The first number (324M or 340234099 bytes) is the total size of new files found on the source. The second number (152M or 159903981 bytes) is the space consumed in the backup destination by those new files. This size represents the files actually copied, the effects of compression to reduce the size, and the effects of linking to identical files already in the backup (so that additional spaced used is essentially near 0).

7.13 Monitoring

If you want to monitor your backups, you can simply grep for `^ERROR` and / or `^WARNING` in the log files. The start of a program (which writes log files) typically starts with a `BEGIN` message in the log and ends with an `END` message if there were no errors which lead to an immediate end of the program.

If you are monitoring your systems with Nagios or Icinga, you can use a plugin from
http://exchange.nagios.org/directory/Plugins/Backup-and-Recovery/storeBackup.

7.14 Limitations

- storeBackup can backup normal files, directories, symbolic links and named pipes. You can backup other file types only with option cpIsGnu (and if gnu cp is installed on your system).
- The permissions in the backup tree(s) are equal to the permissions in the original directory. Under special rare conditions it is possible that a user cannot read one ore more of own his/her files in the backup because files are shared using hard links. With the restore tool – storeBackupRecover.pl – everything is restored with the original permissions.
- storeBackup uses hard links to save disk space. GNU/Linux with ext2 file system supports up to 32000, reiserfs up to 64535 hard links when you use a 32 bit operating system. If storeBackup needs more hard links, it will store a new (compressed) copy of the file. If you use ext2 for the backup, you have to reserve enough (static) inodes! (You will need one inode for each different file in the backup, *not* for every single hard link.)
- Changing the compression program is not supported up to now. In a backup (“backupDir”) you should use the same compression program.

8 Internals

8.1 Remark

I sometimes get questions like “Can I move series around?” or “What happens if the deletion of a backup is interrupted?” and similar ones. Most of the answers to these questions are easy if you understand some internals of storeBackup. Therefore, I think it is time to explain how storeBackup works and on which ideas it depends on.
If you are not familiar with basic Unix / Linux file system concepts and simple command line commands, you might have problems to understand this chapter. In this case, it might be better not to follow the recipes shown or at least to follow them in a test environment first to see what happens.

8.2 Completed Backups

This chapter describes dependencies (or non-existing dependencies) when all backups are complete. This means, there are absolutely no “late-links” to set and all replication jobs finished successfully. To finish backups with “lateLinks” or replications you have to use `storeBackupUpdateBackup.pl`. If you have interrupted backups, they are simply broken and also “finished” in this sense.[54]

A simple Backup (no replication, no lateLinks) is just a directory with several files in it. It is located at *backupDir/series/timestamp*. The files in the backup *might be* hard linked to other files in the same backup or to files somewhere else (normally other backups). This hard linking has *nothing* to do with storeBackup’s functionality. (Naturally, storeBackup tries to generate hard links to save space (deduplication), but that’s it.) Let’s make the following thought experiment: You have two files hard linked in one backup, and you delete the second one of these hard links in the backup and then copy the first one to exactly the path/name of the deleted second one. The result are two files with the same content instead of one file (inode) with two hard links. This (useless) change in your backup means nothing to storeBackup. It is not even able to detect your change.

So it is very easy: storeBackup doesn’t know and doesn’t care in any way if (identical) files are hard linked or not. That’s the reason why you can delete (completed) backups (without lateLinks references to them) like you want (this affects the number of hard links to files also shown in other backups) or why you can copy your backups to another file system with less or more hard links supported per file via `linkToDirs.pl`. Hard links in the backup are managed and controlled by the file system only, not by storeBackup.

But storeBackup has other information about the backup. Within each backup, there is a file called `.md5CheckSums.bz2` (or maybe not compressed). This file has the following structure:

```
# contents/md5 compr dev-inode inodeBackup ctime mtime atime size uid gid mode filename
dir 0 18-139259 0 1376114212 1376114212 1376114500 0 1049 1049 509 sub1
c5f89e40c144b6fb8b61f2ef72e4b556 c 18-141517 148481 1376114209 1376114209 1376114500 31400 1049 1049 493 pwd
c5f89e40c144b6fb8b61f2ef72e4b556 c 18-141521 148481 1376114212 1376114212 1376114500 31400 1049 1049 493 sub1/pwd
b5607b4dc7d896c0fab5c4a308239161 c 18-141513 147606 1376114202 1376114202 1376114500 110088 1049 1049 493 ls
b5607b4dc7d896c0fab5c4a308239161 c 18-141516 147606 1376114205 1376114205 1376114500 110088 1049 1049 493 sub1/ls
```

This is the printout of a small example I made. In the first column, you see the keyword ‘`dir`’ which means the items described in that line are directories. Below, you can see the md5 sums of the files `pwd` and `ls` in different subdirectories. You also see some of them with the same md5 sums, so these have been copies in the `sourceDir`. There’s another file, `.md5CheckSums.info`:

```
version=1.3
date=2013.08.10 08.04.29
sourceDir='/tmp/a/s'
followLinks=0
compress='bzip2'
uncompress='bzip2' '-d'
postfix='.bz2'
comprRule='$size > 1024 and' 'not $file =~ /\.zip\Z|\.bz2\Z|\.gz\Z|\.tgz\Z|\.jpg\Z|\.gif\Z|\.tiff?\Z|\.mpe?g\Z|\.mp[34]\Z|\.mpe?[34]\Z|\.ogg\Z|\.gpg\Z|\.png\Z|\.lzma\Z|\.xz\Z|\.mov\Z/i'
exceptDirs=
includeDirs=
exceptRule=
includeRule=
writeExcludeLog=no
exceptTypes=
archiveTypes=
specialTypeArchiver=cpio
checkBlocksRule0=
checkBlocksBS0=
checkBlocksCompr0=
checkBlocksRead0=
checkBlocksRule1=
checkBlocksBS1=
checkBlocksCompr1=
checkBlocksRead1=
checkBlocksRule2=
checkBlocksBS2=
checkBlocksCompr2=
checkBlocksRead2=
checkBlocksRule3=
checkBlocksBS3=
checkBlocksCompr3=
checkBlocksRead3=
checkBlocksRule4=
checkBlocksBS4=
checkBlocksCompr4=
checkBlocksRead4=
preservePerms=yes
lateLinks=no
lateCompress=no
cpIsGnu=no
logInBackupDir=no
compressLogInBackupDir=no
logInBackupDirFileName='.storeBackup.log'
linkToRecent=
```

[54] Up to version 3.4.3, an interrupted backup is identified by the existence of flag file `.md5CheckSums.notfinished`. Starting with version 3.5, an interrupted backup can be identified by the absence of `.storeBackupLinks/backup.Finished`.

It gives the storeBackup scripts several information needed to do additional tasks with the backup, e.g. how to restore: In this example, the compressed files in the backup have the postfix `.bz2` (keyword `postfix`) and have been uncompressed with the external command `bzip2 -d` (keyword `uncompress`).[55]
There's also an empty directory called `.storeBackupLinks` which contains the information if the backup was created with lateLinks and is not yet finished via `storeBackupUpdateBackup.pl`. But that's explained later.

The entry for "compression" (`c` or `u`) can be set to value `b` also. This means, that the particular file is a "blocked file". At the place the file name points to, storeBackup created a subdirectory with the name of the "blocked file" in the `sourceDir`. Beside the splitted parts of the saved file, there is a file called `.md5BlockCheckSums.bz2` in this directory. This file contains the information about the splitted parts of the "blocked file", eg:

```
66fa1a8f82c35ca87a08aed9701c5d20 c add_VirtualBox/HardDisks/debian.vdi/0000000001.bz2
c522c1db31cc1f90b5d21992fd30e2ab c add_VirtualBox/HardDisks/debian.vdi/0000000002.bz2
c522c1db31cc1f90b5d21992fd30e2ab c add_VirtualBox/HardDisks/debian.vdi/0000000003.bz2
c522c1db31cc1f90b5d21992fd30e2ab c add_VirtualBox/HardDisks/debian.vdi/0000000004.bz2
.....
```

The first column contains the md5 sum, the second one `c` (compressed) or `u` (uncompressed) and the relative path/name combination of that block inside the backup.

Conclusion
Knowing the information shown above, it is easy to understand, that storeBackup is pretty insensible against manipulations you do with the backups – naturally only if you do not touch the integrity of the single backups (backup directories). You can copy backups, move series or even single backups (on the same file system) around. If you do so, you naturally might have to adjust the paths and names (series) in you configuration files according to the changes you made.

8.3 Late Links Backups

Making a backup with option `lateLinks` means to split the creation of backups into two parts. The first part is done with `storeBackup.pl` which identifies and copies all files with new contents, and the second one with `storeBackupUpdateBackup.pl`, which:

- creates symbolic links and compresses files (only if option `lateCompress` was selected),
- sets hard links,
- sets file and directory permissions.

The most important part – regarding storeBackups structure – is the creation of the missing hard links in the backup. This means that the last (unfinished) backup is not a full backup but something like an incremental backup.[56]

Let's make an example:

```
$ mkdir /tmp/a
$ cd /tmp/a
$ mkdir -p s/sub1 b
$ cp /bin/ls /bin/pwd s
$ cp /bin/ls /bin/pwd s/sub1
```

If a directory called `/tmp/a` already exists on you system, select another directory of your choice. We now created a sourceDir `s` with some contents and a backupDir `b`.
Now run the first backup with option `--lateLinks` and `--lateCompress` and look at the result:

[55]The information if a file is compressed is stored in `.md5CheckSums.bz2`: in the second column `c` stands for "compressed" and `u` stands for "uncompressed".

[56]It is not the same as a "real" incremental backup, because there is also information about the missing parts (completed later via hard links).

```
$ storeBackup.pl -s s -b b --lateLinks --lateCompress
.....
$ find b -print | sort
b
b/default
b/default/2013.08.10_10.14.00
b/default/2013.08.10_10.14.00/ls
b/default/2013.08.10_10.14.00/.md5CheckSums.bz2
b/default/2013.08.10_10.14.00/.md5CheckSums.info
b/default/2013.08.10_10.14.00/pwd
b/default/2013.08.10_10.14.00/.storeBackupLinks
b/default/2013.08.10_10.14.00/.storeBackupLinks/linkFile.bz2
```

You can see:

- the series name chosen is `default`
- the two meta data files `.md5CheckSums.bz2` and `.md5CheckSums.info` have been created
- the files `ls` and `pwd` are in the backup, but not compressed – also, the files `sub1/ls` and `sub1/pwd` are missing
- in `.storeBackupLinks` there is a file called `linkFile.bz2`

This “link file” contains, what still has to be done:

```
$ bzcat b/default/2013.08.10_10.14.00/.storeBackupLinks/linkFile.bz2
# link md5sum
# existingFile
# newLink
# compress md5sum
# fileToCompress
# dir dirName
# symlink file
# target
# linkSymlink link
# existingFile
# newLink
dir sub1
compress c5f89e40c144b6fb8b61f2ef72e4b556
pwd
compress b5607b4dc7d896c0fab5c4a308239161
ls
link c5f89e40c144b6fb8b61f2ef72e4b556
./pwd.bz2
sub1/pwd.bz2
link b5607b4dc7d896c0fab5c4a308239161
./ls.bz2
sub1/ls.bz2
```

In the first commented lines the syntax is explained shortly. You can see (all pathes are relative to the actual backup directory):

- create directory `sub1`
- compress file `pwd` (file becomes `pwd.bz2`)
- compress file `ls` (file becomes `ls.bz2`)
- hard link `sub1/pwd.bz2` to `pwd.bz2`
- hard link `sub1/ls.bz2` to `ls.bz2`

There are no dependencies to other directories, because this was a first backup.

After running `storeBackupUpdateBackup.pl` you get:

```
$ storeBackupUpdateBackup.pl -b b
.....
$ find b -ls
232815    0 drwxrwxr-x   3 hjc  hjc    60 Aug 10 10:14 b
240109    0 drwx------   3 hjc  hjc    60 Aug 10 12:18 b/default
305119    0 drwxr-xr-x   4 hjc  hjc   160 Aug 10 12:15 b/default/2013.08.10_11.52.15
249568   52 -rwxr-xr-x   2 hjc  hjc 49915 Aug 10 10:03 b/default/2013.08.10_11.52.15/ls.bz2
249567   16 -rwxr-xr-x   2 hjc  hjc 13395 Aug 10 10:03 b/default/2013.08.10_11.52.15/pwd.bz2
323964    0 drwxrwxr-x   2 hjc  hjc    80 Aug 10 10:03 b/default/2013.08.10_11.52.15/sub1
249568   52 -rwxr-xr-x   2 hjc  hjc 49915 Aug 10 10:03 b/default/2013.08.10_11.52.15/sub1/ls.bz2
249567   16 -rwxr-xr-x   2 hjc  hjc 13395 Aug 10 10:03 b/default/2013.08.10_11.52.15/sub1/pwd.bz2
306983    4 -rw-rw-r--   1 hjc  hjc   300 Aug 10 11:52 b/default/2013.08.10_11.52.15/.md5CheckSums.bz2
305122    4 -rw-------   1 hjc  hjc   950 Aug 10 11:52 b/default/2013.08.10_11.52.15/.md5CheckSums.info
305120    0 drwx------   2 hjc  hjc    40 Aug 10 12:15 b/default/2013.08.10_11.52.15/.storeBackupLinks
```

You can see, that `ls` and `pwd` have been compressed. There are now also hard links from subdirectory `sub1`. Finally, the permissions are corrected and `.storeBackupLinks/linkFile.bz2` does not exist any more. This is now a completed backup.

Let's run `storeBackup.pl` a second time:

```
$ storeBackup.pl -s s -b b --lateLinks --lateCompress
.....
$ find b -print | sort
b
b/default
b/default/2013.08.10_10.14.00
b/default/2013.08.10_10.14.00/ls.bz2
b/default/2013.08.10_10.14.00/.md5CheckSums.bz2
b/default/2013.08.10_10.14.00/.md5CheckSums.info
b/default/2013.08.10_10.14.00/pwd.bz2
b/default/2013.08.10_10.14.00/.storeBackupLinks
b/default/2013.08.10_10.14.00/.storeBackupLinks/linkFrom0
b/default/2013.08.10_10.14.00/sub1
b/default/2013.08.10_10.14.00/sub1/ls.bz2
b/default/2013.08.10_10.14.00/sub1/pwd.bz2
b/default/2013.08.10_11.52.15
b/default/2013.08.10_11.52.15/.md5CheckSums.bz2
b/default/2013.08.10_11.52.15/.md5CheckSums.info
b/default/2013.08.10_11.52.15/.storeBackupLinks
b/default/2013.08.10_11.52.15/.storeBackupLinks/linkFile.bz2
b/default/2013.08.10_11.52.15/.storeBackupLinks/linkTo
```

You can see one change in the first backup: There is now a file called `.storeBackupLinks/linkFrom0`. This file contains:

```
$ cat b/default/2013.08.10_10.14.00/.storeBackupLinks/linkFrom0
../2013.08.10_11.52.15
```

This means that there is at least one unresolved reference from a backup with the relative path from this backup (`2013.08.10_11.52.15`). If references from multiple other backups would exist, there would be more files (`linkFrom1`, `linkFrom2`, ...) of that kind describing those dependencies. Because of one or more of these files, `storeBackupDel.pl` knows that it must not delete that backup, because otherwise the information where the later backup points to (via yet unresolved links) is lost.

In the new backup (`2013.08.10_11.52.15`), the file `.storeBackupLinks/linkTo` lists all other backups where this backup has dependencies to (only one in this case):

```
$ cat b/default/2013.08.10_11.52.15/.storeBackupLinks/linkTo
../2013.08.10_10.14.00
```

As you see in the list of all files generated by the second backup, there are no data files in the new backup, only meta data files. That's fine, because no files have changed since the first backup. Let's look into `linkFile`:

```
$ bzcat b/default/2013.08.10_11.52.15/.storeBackupLinks/linkFile.bz2
# link md5sum
# existingFile
# newLink
# compress md5sum
# fileToCompress
# dir dirName
# symlink file
# target
# linkSymlink link
# existingFile
# newLink
link c5f89e40c144b6fb8b61f2ef72e4b556
../2013.08.10_10.14.00/sub1/pwd.bz2
pwd.bz2
link b5607b4dc7d896c0fab5c4a308239161
../2013.08.10_10.14.00/sub1/ls.bz2
ls.bz2
dir sub1
link c5f89e40c144b6fb8b61f2ef72e4b556
../2013.08.10_10.14.00/sub1/pwd.bz2
sub1/pwd.bz2
link b5607b4dc7d896c0fab5c4a308239161
../2013.08.10_10.14.00/sub1/ls.bz2
sub1/ls.bz2
```

It is easy to see that all missing files have to be hard linked and the subdirectory has to be created. Now you see the changes from running `storeBackupUpdateBackup.pl`:

```
$ storeBackupUpdateBackup.pl -b b
.....
$ find b -ls
232815    0 drwxrwxr-x   3 hjc  hjc    60 Aug 10 10:14 b
240109    0 drwx------   4 hjc  hjc    80 Aug 10 11:52 b/default
305119    0 drwxr-xr-x   4 hjc  hjc   160 Aug 10 12:15 b/default/2013.08.10_11.52.15
249568   52 -rwxr-xr-x   4 hjc  hjc 49915 Aug 10 10:03 b/default/2013.08.10_11.52.15/ls.bz2
249567   16 -rwxr-xr-x   4 hjc  hjc 13395 Aug 10 10:03 b/default/2013.08.10_11.52.15/pwd.bz2
323964    0 drwxrwxr-x   2 hjc  hjc    80 Aug 10 10:03 b/default/2013.08.10_11.52.15/sub1
249568   52 -rwxr-xr-x   4 hjc  hjc 49915 Aug 10 10:03 b/default/2013.08.10_11.52.15/sub1/ls.bz2
249567   16 -rwxr-xr-x   4 hjc  hjc 13395 Aug 10 10:03 b/default/2013.08.10_11.52.15/sub1/pwd.bz2
306983    4 -rw-rw-r--   1 hjc  hjc   300 Aug 10 11:52 b/default/2013.08.10_11.52.15/.md5CheckSums.bz2
305122    4 -rw-------   1 hjc  hjc   950 Aug 10 11:52 b/default/2013.08.10_11.52.15/.md5CheckSums.info
305120    0 drwx------   2 hjc  hjc    40 Aug 10 12:15 b/default/2013.08.10_11.52.15/.storeBackupLinks
241222    0 drwxr-xr-x   4 hjc  hjc   160 Aug 10 10:27 b/default/2013.08.10_10.14.00
249568   52 -rwxr-xr-x   4 hjc  hjc 49915 Aug 10 10:03 b/default/2013.08.10_10.14.00/ls.bz2
249567   16 -rwxr-xr-x   4 hjc  hjc 13395 Aug 10 10:03 b/default/2013.08.10_10.14.00/pwd.bz2
249566    0 drwxrwxr-x   2 hjc  hjc    80 Aug 10 10:03 b/default/2013.08.10_10.14.00/sub1
249568   52 -rwxr-xr-x   4 hjc  hjc 49915 Aug 10 10:03 b/default/2013.08.10_10.14.00/sub1/ls.bz2
249567   16 -rwxr-xr-x   4 hjc  hjc 13395 Aug 10 10:03 b/default/2013.08.10_10.14.00/sub1/pwd.bz2
238098    4 -rw-rw-r--   1 hjc  hjc   300 Aug 10 10:14 b/default/2013.08.10_10.14.00/.md5CheckSums.bz2
241225    4 -rw-------   1 hjc  hjc   950 Aug 10 10:14 b/default/2013.08.10_10.14.00/.md5CheckSums.info
241223    0 drwx------   2 hjc  hjc    40 Aug 10 12:15 b/default/2013.08.10_10.14.00/.storeBackupLinks
```

As you see, the result is a complete backup.

This all was normal behavior. Nothing went wrong. I create a new backupDir (`b1`) and let things go wrong:

```
$ storeBackup.pl -s s -b b1
.....
$ storeBackup.pl -s s -b b1 --lateLinks --lateCompress
.....
$ find b1 -print | sort
b1
b1/default
b1/default/2013.08.10_13.47.41
```

```
b1/default/2013.08.10_13.47.41/ls.bz2
b1/default/2013.08.10_13.47.41/.md5CheckSums.bz2
b1/default/2013.08.10_13.47.41/.md5CheckSums.info
b1/default/2013.08.10_13.47.41/pwd.bz2
b1/default/2013.08.10_13.47.41/.storeBackupLinks
b1/default/2013.08.10_13.47.41/.storeBackupLinks/linkFrom0
b1/default/2013.08.10_13.47.41/sub1
b1/default/2013.08.10_13.47.41/sub1/ls.bz2
b1/default/2013.08.10_13.47.41/sub1/pwd.bz2
b1/default/2013.08.10_13.49.21
b1/default/2013.08.10_13.49.21/.md5CheckSums.bz2
b1/default/2013.08.10_13.49.21/.md5CheckSums.info
b1/default/2013.08.10_13.49.21/.storeBackupLinks
b1/default/2013.08.10_13.49.21/.storeBackupLinks/linkFile.bz2
b1/default/2013.08.10_13.49.21/.storeBackupLinks/linkTo
```

You now see one complete backup (the first one) and the second backup which is not completed.
For whatever stupid reason, the second backup is deleted. Maybe you recognized during the (let's assume long) backup that some option where wrong and pressed control-c and deleted the backup. In this example, I now delete the second backup:

```
$ rm -r b1/default/2013.08.10_13.49.21
```

But the file `b1/default/2013.08.10_13.47.41/.storeBackupLinks/linkFrom` still exists, so the backup series is not consistent.
Running `storeBackupUpdateBackup.pl` shows an error message:

```
$ storeBackupUpdateBackup.pl -b b1
BEGIN     2013.08.10 13:57:46   475 checking references and backup copying in <b1>
VERSION   2013.08.10 13:57:46   475 storeBackupUpdateBackup.pl, 3.4 +
INFO      2013.08.10 13:57:46   475 creating lock file </tmp/storeBackup.lock>
INFO      2013.08.10 13:57:46   475 scanning directory <b1> for existing backups
INFO      2013.08.10 13:57:46   475 scanning directory <b1/default> for existing backups
STATISTIC 2013.08.10 13:57:46   475 found 1 backup series, 1 backups, 0 renamed backups
ERROR     2013.08.10 13:57:46   475 link <../2013.08.10_13.49.21> to non existing dir in
                                    </tmp/a/b1/default/2013.08.10_13.47.41/.storeBackupLinks/linkFrom0>
ERROR     2013.08.10 13:57:46   475 found 1 inconsistencies, please repair and check again
```

Deleting this file may not be a good idea, because in a real world situation, there might be other references to this backup also which means things may be not that simple and it is easy to make something wrong (or worse).

There are two possibilities available with `storeBackupUpdateBackup.pl`:

1. Option `--autorepair` checks the backup and creates the missing references or deletes references which are pointing to nowhere.
2. Option `--interactive` always asks what to do if something's inconsistent.

Because option `--autorepair` does nothing destructive, it is a good choice in 99% of the cases. Therefore I'm using `--autorepair`:

```
$ storeBackupUpdateBackup.pl -b b1 --autorepair
BEGIN     2013.08.10 17:06:59  4569 checking references and backup copying in <b1>
VERSION   2013.08.10 17:06:59  4569 storeBackupUpdateBackup.pl, 3.4 +
INFO      2013.08.10 17:06:59  4569 removing old lock file of process <475>
INFO      2013.08.10 17:06:59  4569 creating lock file </tmp/storeBackup.lock>
INFO      2013.08.10 17:06:59  4569 scanning directory <b1> for existing backups
INFO      2013.08.10 17:06:59  4569 scanning directory <b1/default> for existing backups
STATISTIC 2013.08.10 17:06:59  4569 found 1 backup series, 1 backups, 0 renamed backups
ERROR     2013.08.10 17:06:59  4569 link <../2013.08.10_13.49.21> to non existing dir in
                                    </tmp/a/b1/default/2013.08.10_13.47.41/.storeBackupLinks/linkFrom0>
INFO      2013.08.10 17:06:59  4569 autorepair: deleted
                                    </tmp/a/b1/default/2013.08.10_13.47.41/.storeBackupLinks/linkFrom0>
ERROR     2013.08.10 17:06:59  4569 ----- repeating consistency check -----
INFO      2013.08.10 17:06:59  4569 scanning directory <b1> for existing backups
INFO      2013.08.10 17:06:59  4569 scanning directory <b1/default> for existing backups
```

```
STATISTIC 2013.08.10 17:06:59  4569 found 1 backup series, 1 backups, 0 renamed backups
INFO      2013.08.10 17:06:59  4569 consistency check finished successfully
INFO      2013.08.10 17:06:59  4569 found no references to backups from lateLinks that need
                                    storeBackupUpdateBackup run
INFO      2013.08.10 17:06:59  4569 everything is updated, nothing to do
STATISTIC 2013.08.10 17:06:59  4569                       duration = 1s
STATISTIC 2013.08.10 17:06:59  4569  [sec] |      user|    system
STATISTIC 2013.08.10 17:06:59  4569 -------+----------+----------
STATISTIC 2013.08.10 17:06:59  4569 process|      0.08|      0.00
STATISTIC 2013.08.10 17:06:59  4569 childs |      0.05|      0.00
STATISTIC 2013.08.10 17:06:59  4569 -------+----------+----------
STATISTIC 2013.08.10 17:06:59  4569 sum    |      0.13|      0.00 => 0.13
INFO      2013.08.10 17:06:59  4569 syncing ...
END       2013.08.10 17:06:59  4569 checking references and copying in <b1>
$ find b1 -print | sort
b1
b1/default
b1/default/2013.08.10_13.47.41
b1/default/2013.08.10_13.47.41/ls.bz2
b1/default/2013.08.10_13.47.41/.md5CheckSums.bz2
b1/default/2013.08.10_13.47.41/.md5CheckSums.info
b1/default/2013.08.10_13.47.41/pwd.bz2
b1/default/2013.08.10_13.47.41/.storeBackupLinks
b1/default/2013.08.10_13.47.41/sub1
b1/default/2013.08.10_13.47.41/sub1/ls.bz2
b1/default/2013.08.10_13.47.41/sub1/pwd.bz2
```

The output of the `find` command above shows, that everything is fine now – there’s one completed backup.

Now let’s make a second backup with lateLinks again:

```
$ storeBackup.pl -s s -b b1 --lateLinks --lateCompress
-----
$ find b1 -print | sort
b1
b1/default
b1/default/2013.08.10_13.47.41
b1/default/2013.08.10_13.47.41/ls.bz2
b1/default/2013.08.10_13.47.41/.md5CheckSums.bz2
b1/default/2013.08.10_13.47.41/.md5CheckSums.info
b1/default/2013.08.10_13.47.41/pwd.bz2
b1/default/2013.08.10_13.47.41/.storeBackupLinks
b1/default/2013.08.10_13.47.41/.storeBackupLinks/linkFrom0
b1/default/2013.08.10_13.47.41/sub1
b1/default/2013.08.10_13.47.41/sub1/ls.bz2
b1/default/2013.08.10_13.47.41/sub1/pwd.bz2
b1/default/2013.08.10_17.17.53
b1/default/2013.08.10_17.17.53/.md5CheckSums.bz2
b1/default/2013.08.10_17.17.53/.md5CheckSums.info
b1/default/2013.08.10_17.17.53/.storeBackupLinks
b1/default/2013.08.10_17.17.53/.storeBackupLinks/linkFile.bz2
b1/default/2013.08.10_17.17.53/.storeBackupLinks/linkTo
```

But this time, I delete the *first* backup:

```
$ rm -r b1/default/2013.08.10_13.47.41
```

This means, I now have an uncomplete backup (created with lateLinks) and pointing to nowhere. I try to use `--storeBackupUpdateBackup.pl` with option `--autorepair`:

```
$ storeBackupUpdateBackup.pl -b b1 --autorepair
BEGIN     2013.08.10 17:21:08  5019 checking references and backup copying in <b1>
VERSION   2013.08.10 17:21:08  5019 storeBackupUpdateBackup.pl, 3.4 +
INFO      2013.08.10 17:21:08  5019 creating lock file </tmp/storeBackup.lock>
INFO      2013.08.10 17:21:08  5019 scanning directory <b1> for existing backups
INFO      2013.08.10 17:21:08  5019 scanning directory <b1/default> for existing backups
STATISTIC 2013.08.10 17:21:08  5019 found 1 backup series, 1 backups, 0 renamed backups
```

```
ERROR     2013.08.10 17:21:08  5019 FATAL ERROR: link <../2013.08.10_13.47.41> to non existing dir in
                                    </tmp/a/b1/default/2013.08.10_17.17.53/.storeBackupLinks/linkTo>
ERROR     2013.08.10 17:21:08  5019 found 1 inconsistencies, please repair and check again
```

Now I get a *FATAL ERROR*: No chance for `storeBackupUpdateBackup.pl` to repair this. The only chance would be to copy[57] the missing (first) backup from somewhere else (e.g. a copy from the backup or from replication) to the right location and run `storeBackupUpdateBackup.pl` with `--autorepair` again. If there is no other copy of the missing backup, all backups depending on that backup (and backups depending on that one etc.) will never be complete. You should delete them or move those partly backups outside the `backupDir`, so they do not harm the workflow of storeBackup's tools.

8.4 Replication

It is important to read at least chapter 7.8.6, "How storeBackup's replication works" which gives you an explanation of the basic concepts and configurations.

To understand how replication works in detail, you have to know that it is an extension to lateLinks. The internals of replication are explained with an example. First, create a backup with one replication:

```
$ mkdir /tmp/a
$ cd /tmp/a
$ mkdir -p source backup/default repl delta
$ cp /bin/ls source
$ cp /bin/ls source/ls1
$ storeBackupReplicationWizard.pl -m backup -c repl -d delta -S default
WARNING 2013.08.17 10:12:44 25548 cannot find any existing backup at master backup directory </tmp/a/backup>
INFO    2013.08.17 10:12:44 25548 1 series chosen:
INFO    2013.08.17 10:12:44 25548       <default>
INFO    2013.08.17 10:12:44 25548 wrote master backup configuration file </tmp/a/backup/storeBackupBaseTree.conf>
INFO    2013.08.17 10:12:44 25548 wrote backup copy configuration file </tmp/a/repl/storeBackupBaseTree.conf>
INFO    2013.08.17 10:12:44 25548 wrote delta cache configuration file </tmp/a/delta/deltaCache.conf>
```

Let's see what `storeBackupReplicationWizard.pl` generated:

```
$ cat backup/storeBackupBaseTree.conf | egrep -v '^\s*$|^#'
backupTreeName='Master Backup'
backupType=master
seriesToDistribute='default'
deltaCache=/tmp/a/delta

$ cat delta/deltaCache.conf | egrep -v '^\s*$|^#'
backupCopy0='Backup Copy' 'default'
;backupCopy1=
;backupCopy2=
;backupCopy3=
;backupCopy4=
;backupCopy5=
;backupCopy6=
;backupCopy7=
;backupCopy8=
;backupCopy9=

$ cat repl/storeBackupBaseTree.conf | egrep -v '^\s*$|^#'
backupTreeName='Backup Copy'
backupType=copy
seriesToDistribute='default'
deltaCache=/tmp/a/delta
```

Files available in all directories are:

[57]It is a good idea to use `linkToDirs.pl` for copying because then you can mostly hard link the files against the existing ones. `linkToDirs.pl` knows nothing about storeBackup's logic and its meta files – therefore you can use it with every type of data; also with non-completed backups or whatever.

```
$ find . -print | sort
.
./backup
./backup/default
./backup/storeBackupBaseTree.conf
./delta
./delta/deltaCache.conf
./repl
./repl/storeBackupBaseTree.conf
./source
./source/ls
./source/ls1
```

Now, the first backup with lateLinks:

```
$ storeBackup.pl -s source -b backup --lateLinks
.....
$ find . -print | sort
.
./backup
./backup/default
./backup/default/2013.08.17_10.13.41
./backup/default/2013.08.17_10.13.41/ls1.bz2
./backup/default/2013.08.17_10.13.41/.md5CheckSums.bz2
./backup/default/2013.08.17_10.13.41/.md5CheckSums.info
./backup/default/2013.08.17_10.13.41/.storeBackupLinks
./backup/default/2013.08.17_10.13.41/.storeBackupLinks/linkFile.bz2
./backup/storeBackupBaseTree.conf
./delta
./delta/deltaCache.conf
./repl
./repl/storeBackupBaseTree.conf
./source
./source/ls
./source/ls1
```

Nothing new up to now. A compressed copy of `ls1` from `source` can be found in `backup` now. The file `ls` is missing because of deduplication. Also, the meta data is available in `backup` plus the `linkFile` (see previous chapter).

To finalize the backup, run `storeBackupUpdateBackup.pl`:

```
$ storeBackupUpdateBackup.pl -b backup
BEGIN     2013.08.17 10:14:59 25914 checking references and backup copying in <backup>
VERSION   2013.08.17 10:14:59 25914 storeBackupUpdateBackup.pl, 3.4 +
INFO      2013.08.17 10:14:59 25914 creating lock file </tmp/storeBackup.lock>
INFO      2013.08.17 10:14:59 25914 reading <backup/storeBackupBaseTree.conf>
INFO      2013.08.17 10:14:59 25914 master backup: checking <Master Backup/default>
INFO      2013.08.17 10:14:59 25914     copying <2013.08.17_10.13.41> to </tmp/a/delta/default>
INFO      2013.08.17 10:14:59 25914 scanning directory <backup> for existing backups
INFO      2013.08.17 10:14:59 25914 scanning directory <backup/default> for existing backups
STATISTIC 2013.08.17 10:14:59 25914 found 1 backup series, 1 backups, 0 renamed backups
INFO      2013.08.17 10:14:59 25914 consistency check finished successfully
INFO      2013.08.17 10:14:59 25914 found no references to backups from lateLinks that need
                                    storeBackupUpdateBackup run
INFO      2013.08.17 10:14:59 25914 (1/0) updating </tmp/a/backup/default/2013.08.17_10.13.41>
INFO      2013.08.17 10:14:59 25914 phase 1: mkdir, symlink and compressing files
STATISTIC 2013.08.17 10:14:59 25914 created 0 directories
STATISTIC 2013.08.17 10:14:59 25914 created 0 symbolic links
STATISTIC 2013.08.17 10:14:59 25914 compressed 0 files
STATISTIC 2013.08.17 10:14:59 25914 used 0.0  instead of 0.0  (0 <- 0 ; 0.0%)
INFO      2013.08.17 10:14:59 25914 phase 2: setting hard links
STATISTIC 2013.08.17 10:14:59 25914 linked 1 files
INFO      2013.08.17 10:14:59 25914 phase 3: setting file permissions
STATISTIC 2013.08.17 10:14:59 25914 set permissions for 2 files
INFO      2013.08.17 10:14:59 25914 phase 4: setting directory permissions
STATISTIC 2013.08.17 10:14:59 25914 set permissions for 0 directories
```

```
INFO      2013.08.17 10:14:59 25914 reading <backup/storeBackupBaseTree.conf>
INFO      2013.08.17 10:14:59 25914 scanning directory <backup> for existing backups
INFO      2013.08.17 10:14:59 25914 scanning directory <backup/default> for existing backups
STATISTIC 2013.08.17 10:14:59 25914 found 1 backup series, 1 backups, 0 renamed backups
INFO      2013.08.17 10:14:59 25914 consistency check finished successfully
INFO      2013.08.17 10:14:59 25914 found no references to backups from lateLinks that need
                                    storeBackupUpdateBackup run
INFO      2013.08.17 10:14:59 25914 deleting in deltaCache </tmp/a/delta> processedBackups
INFO      2013.08.17 10:14:59 25914     age for deletion is > 99d (delete backups older than
                                        Fri 2013.05.10 10:14:59)
STATISTIC 2013.08.17 10:14:59 25914                       duration = 1s
STATISTIC 2013.08.17 10:14:59 25914  [sec] |      user|    system
STATISTIC 2013.08.17 10:14:59 25914 -------+----------+----------
STATISTIC 2013.08.17 10:14:59 25914 process|      0.07|      0.03
STATISTIC 2013.08.17 10:14:59 25914 childs |      0.05|      0.00
STATISTIC 2013.08.17 10:14:59 25914 -------+----------+----------
STATISTIC 2013.08.17 10:14:59 25914 sum    |      0.12|      0.03 => 0.15
INFO      2013.08.17 10:14:59 25914 syncing ...
END       2013.08.17 10:14:59 25914 checking references and copying in <backup>

$ find . -print | sort
.
./backup
./backup/default
./backup/default/2013.08.17_10.13.41
./backup/default/2013.08.17_10.13.41/ls1.bz2
./backup/default/2013.08.17_10.13.41/ls.bz2
./backup/default/2013.08.17_10.13.41/.md5CheckSums.bz2
./backup/default/2013.08.17_10.13.41/.md5CheckSums.info
./backup/default/2013.08.17_10.13.41/.storeBackupLinks
./backup/storeBackupBaseTree.conf
./delta
./delta/default
./delta/default/2013.08.17_10.13.41
./delta/default/2013.08.17_10.13.41/ls1.bz2
./delta/default/2013.08.17_10.13.41/.md5CheckSums.bz2
./delta/default/2013.08.17_10.13.41/.md5CheckSums.info
./delta/default/2013.08.17_10.13.41/.storeBackupLinks
./delta/default/2013.08.17_10.13.41/.storeBackupLinks/linkFile.bz2
./delta/deltaCache.conf
./delta/processedBackups
./repl
./repl/storeBackupBaseTree.conf
./source
./source/ls
./source/ls1
```

Lots of things have changed now:

- The master backup in `backup` is completed. In the `backup` directory, the files `ls` and `ls1` have been hard linked and the command file `linkFile` is away.
- In the beginning of the output of `storeBackupUpdateBackup.pl` you can see the following lines:

  ```
  reading <backup/storeBackupBaseTree.conf>
  master backup: checking <Master Backup/default>
  copying <2013.08.17_10.13.41> to </tmp/a/delta/default>
  ```

  The program detected that that lateLinks backup was not yet copied into the delta cache. Before completing the backup, these deltas were copied into the delta cache. Have a look at the output of the `find` command to see that copy in directory `delta`. The delta cache collects all new backups (to be replicated) from the master backup (`backup`) until they are transferred to the final replication directory (`repl`) and also stores them later for a while.
  You also see, that the directory `processedBackups` was created in `delta`.

Now, run `storeBackupUpdateBackup.pl` a second time, but on `repl`:

```
$ storeBackupUpdateBackup.pl -b repl
BEGIN     2013.08.17 10:33:03 28608 checking references and backup copying in <repl>
```

```
VERSION   2013.08.17 10:33:03 28608 storeBackupUpdateBackup.pl, 3.4 +
INFO      2013.08.17 10:33:03 28608 creating lock file </tmp/storeBackup.lock>
INFO      2013.08.17 10:33:03 28608 reading <repl/storeBackupBaseTree.conf>
INFO      2013.08.17 10:33:03 28608 reading </tmp/a/delta/deltaCache.conf>
INFO      2013.08.17 10:33:03 28608 copying </tmp/a/delta/default/2013.08.17_10.13.41> to <repl/default>
INFO      2013.08.17 10:33:03 28608 scanning directory <repl> for existing backups
INFO      2013.08.17 10:33:03 28608 scanning directory <repl/default> for existing backups
STATISTIC 2013.08.17 10:33:03 28608 found 1 backup series, 1 backups, 0 renamed backups
INFO      2013.08.17 10:33:03 28608 consistency check finished successfully
INFO      2013.08.17 10:33:03 28608 found no references to backups from lateLinks that need
                                    storeBackupUpdateBackup run
INFO      2013.08.17 10:33:03 28608 (1/0) updating </tmp/a/repl/default/2013.08.17_10.13.41>
INFO      2013.08.17 10:33:03 28608 phase 1: mkdir, symlink and compressing files
STATISTIC 2013.08.17 10:33:03 28608 created 0 directories
STATISTIC 2013.08.17 10:33:03 28608 created 0 symbolic links
STATISTIC 2013.08.17 10:33:03 28608 compressed 0 files
STATISTIC 2013.08.17 10:33:03 28608 used 0.0  instead of 0.0  (0 <- 0 ; 0.0%)
INFO      2013.08.17 10:33:03 28608 phase 2: setting hard links
STATISTIC 2013.08.17 10:33:03 28608 linked 1 files
INFO      2013.08.17 10:33:03 28608 phase 3: setting file permissions
STATISTIC 2013.08.17 10:33:03 28608 set permissions for 2 files
INFO      2013.08.17 10:33:03 28608 phase 4: setting directory permissions
STATISTIC 2013.08.17 10:33:03 28608 set permissions for 0 directories
INFO      2013.08.17 10:33:03 28608 reading <repl/storeBackupBaseTree.conf>
INFO      2013.08.17 10:33:03 28608 marked <default/2013.08.17_10.13.41> as linked in </tmp/a/delta>
INFO      2013.08.17 10:33:03 28608 scanning directory <repl> for existing backups
INFO      2013.08.17 10:33:03 28608 scanning directory <repl/default> for existing backups
STATISTIC 2013.08.17 10:33:03 28608 found 1 backup series, 1 backups, 0 renamed backups
INFO      2013.08.17 10:33:03 28608 consistency check finished successfully
INFO      2013.08.17 10:33:03 28608 found no references to backups from lateLinks that need
                                    storeBackupUpdateBackup run
INFO      2013.08.17 10:33:03 28608 backup </tmp/a/delta/default/2013.08.17_10.13.41> copied to <Backup Copy>
INFO      2013.08.17 10:33:03 28608     moving backup to </tmp/a/delta/processedBackups/default>
INFO      2013.08.17 10:33:03 28608 deleting in deltaCache </tmp/a/delta> processedBackups
INFO      2013.08.17 10:33:03 28608     age for deletion is > 99d (delete backups older than
                                        Fri 2013.05.10 10:33:03)
INFO      2013.08.17 10:33:03 28608 checking series <default>
INFO      2013.08.17 10:33:03 28608     default -> 2013.08.17_10.13.41 - not old enough to delete
STATISTIC 2013.08.17 10:33:03 28608                       duration = 1s
STATISTIC 2013.08.17 10:33:03 28608  [sec] |      user|    system
STATISTIC 2013.08.17 10:33:03 28608 -------+----------+----------
STATISTIC 2013.08.17 10:33:03 28608 process|      0.09|      0.01
STATISTIC 2013.08.17 10:33:03 28608 childs |      0.05|      0.00
STATISTIC 2013.08.17 10:33:03 28608 -------+----------+----------
STATISTIC 2013.08.17 10:33:03 28608 sum    |      0.14|      0.01 => 0.15
INFO      2013.08.17 10:33:03 28608 syncing ...
END       2013.08.17 10:33:03 28608 checking references and copying in <repl>

$ find . -print | sort
.
./backup
./backup/default
./backup/default/2013.08.17_10.13.41
./backup/default/2013.08.17_10.13.41/ls1.bz2
./backup/default/2013.08.17_10.13.41/ls.bz2
./backup/default/2013.08.17_10.13.41/.md5CheckSums.bz2
./backup/default/2013.08.17_10.13.41/.md5CheckSums.info
./backup/default/2013.08.17_10.13.41/.storeBackupLinks
./backup/storeBackupBaseTree.conf
./delta
./delta/default
./delta/default/2013.08.17_10.13.41.copied
./delta/default/2013.08.17_10.13.41.linked
./delta/deltaCache.conf
./delta/processedBackups
./delta/processedBackups/default
./delta/processedBackups/default/2013.08.17_10.13.41
./delta/processedBackups/default/2013.08.17_10.13.41/ls1.bz2
./delta/processedBackups/default/2013.08.17_10.13.41/.md5CheckSums.bz2
./delta/processedBackups/default/2013.08.17_10.13.41/.md5CheckSums.info
./delta/processedBackups/default/2013.08.17_10.13.41/.storeBackupLinks
./delta/processedBackups/default/2013.08.17_10.13.41/.storeBackupLinks/linkFile.bz2
```

```
./repl
./repl/default
./repl/default/2013.08.17_10.13.41
./repl/default/2013.08.17_10.13.41/ls1.bz2
./repl/default/2013.08.17_10.13.41/ls.bz2
./repl/default/2013.08.17_10.13.41/.md5CheckSums.bz2
./repl/default/2013.08.17_10.13.41/.md5CheckSums.info
./repl/default/2013.08.17_10.13.41/.storeBackupLinks
./repl/storeBackupBaseTree.conf
./source
./source/ls
./source/ls1
```

A look at the file list and the log of `storeBackupUpdateBackup.pl` shows:

- The master backup (directory `backup`) has not changed.
- The deltas have been copied to the replication directory (`repl`) where the copy of the backup should be stored. Additionally, this copy was used to complete the backup in `repl`. A `diff` between `backup` and `repl` shows

  ```
  $ diff -r backup repl
  diff -r backup/storeBackupBaseTree.conf repl/storeBackupBaseTree.conf
  21c21
  < backupTreeName='Master Backup'
  ---
  > backupTreeName='Backup Copy'
  26c26
  < backupType=master
  ---
  > backupType=copy
  ```

  That means that only the configuration file for the replication is different. The backups inside those directory are absolutely identical. You can e.g. run `storeBackupDel.pl` or `storeBackupCheckBackup.pl` on both of them.
- The copy of the deltas in the delta cache directory was moved to its sub directory `processedBackups`. The *status* of that replication is stored in the files:

  ```
  $ cat ./delta/default/2013.08.17_10.13.41.copied
  Backup Copy

  $ cat ./delta/default/2013.08.17_10.13.41.linked
  Backup Copy
  ```

  This means, that the backup from series `default` from `2013.08.17_10.13.41` has been copied and linked to backup copy `Backup Copy`. If you replicate to more than one other directory (e.g. on a usb disk) then you will see the *name* (like `Backup Copy` in this case) of that other backup copy in these files (∗*.copied* and ∗*.linked*) also after it has been copied and linked. By this way, storeBackup can decide if a backup was fully copied and linked to all replication directories. After being sure about this, the deltas for that particular backup are moved to the sub directory `processedBackups` in the delta cache. (We will see later how to use this if somethings goes wrong in the replication process.) These processed backups are stored for some time you can define. The default of 99 days is *very* conservative and far too long.

With the example above, it is easy to see how the concept of replication is realized:[58]

1. run `storeBackup.pl -b backup` and create the lateLinks based delta backup in directory `backup`.
2. run `storeBackupUpdateBackup.pl -b backup` and
   (a) copy the deltas from the backup to the delta cache (directory `delta`)

[58]To make the explanations easy to understand, the directory names in the example above are used.

   (b) complete (make hard links etc.) in the backup in directory `backup`

3. run `storeBackupUpdateBackup.pl -b repl` and
   (a) copy the deltas from the delta cache (directory `delta`) to the replication target directory `repl`
   (b) write the name of the replication (`Backup Copy`) to file *timestamp*`.copied`
   (c) complete (make hard links etc.) the replication target / copy the directory `repl` (in principle the same as in step 2b described above).
   (d) move the just copied backup in the delta cache into the sub directory `processedBackups` and write the name of the replication (`Backup Copy`) to file *timestamp*`.linked`

With the knowledge of the fundamental principles of the replication it is easy to react if something goes wrong. It is always just moving, copying or adding lines to files. Let's simulate an interrupted replication. (New example from the beginning.)

First, create a new backup and replicate it:

```
$ mkdir /tmp/x
$ cd /tmp/x
$ mkdir -p source backup/default repl delta
$ cp /bin/ls source
$ storeBackupReplicationWizard.pl -m backup -c repl -d delta -S default
$ storeBackup.pl -s source -b backup --lateLinks
$ storeBackupUpdateBackup.pl -b backup
$ storeBackupUpdateBackup.pl -b repl
```

Then, add a file to `source`, backup and replicate again.

```
$ cp /bin/pwd source
$ storeBackup.pl -s source -b backup --lateLinks
$ storeBackupUpdateBackup.pl -b backup
$ storeBackupUpdateBackup.pl -b repl
```

I created the following backups:

```
$ ls -1 backup/default/
2013.08.18_09.45.16
2013.08.18_09.49.21
```

No, let's *imagine*, the second backup is broken, maybe because the run of `storeBackupUpdateBackup.pl` was interrupted, had write errors or for whatever reason. Because I wouldn't know the details of the damage and which files may be missing, I'm deleting the newly replicated backup:

```
$ ls -1 repl/default/
2013.08.18_09.45.16
2013.08.18_09.49.21
$ rm -r repl/default/2013.08.18_09.49.21
```

Re-running `storeBackupUpdateBackup.pl` on the replication (*One backup is missing!*) results in:

```
$ storeBackupUpdateBackup.pl -b repl
BEGIN     2013.08.18 09:57:46  6852 checking references and backup copying in <repl>
VERSION   2013.08.18 09:57:46  6852 storeBackupUpdateBackup.pl, 3.4 +
INFO      2013.08.18 09:57:46  6852 creating lock file </tmp/storeBackup.lock>
INFO      2013.08.18 09:57:46  6852 reading <repl/storeBackupBaseTree.conf>
INFO      2013.08.18 09:57:46  6852 reading </tmp/x/delta/deltaCache.conf>
INFO      2013.08.18 09:57:46  6852 scanning directory <repl> for existing backups
INFO      2013.08.18 09:57:46  6852 scanning directory <repl/default> for existing backups
STATISTIC 2013.08.18 09:57:46  6852 found 1 backup series, 1 backups, 0 renamed backups
INFO      2013.08.18 09:57:46  6852 consistency check finished successfully
INFO      2013.08.18 09:57:46  6852 found no references to backups from lateLinks that need
                                    storeBackupUpdateBackup run
INFO      2013.08.18 09:57:46  6852 everything is updated, nothing to do
INFO      2013.08.18 09:57:46  6852 deleting in deltaCache </tmp/x/delta> processedBackups
INFO      2013.08.18 09:57:46  6852     age for deletion is > 99d (delete backups older
                                        than Sat 2013.05.11 09:57:46)
INFO      2013.08.18 09:57:46  6852 checking series <default>
INFO      2013.08.18 09:57:46  6852     default -> 2013.08.18_09.45.16 - not old enough to delete
INFO      2013.08.18 09:57:46  6852     default -> 2013.08.18_09.49.21 - not old enough to delete
STATISTIC 2013.08.18 09:57:46  6852                      duration = 1s
STATISTIC 2013.08.18 09:57:46  6852  [sec] |      user|    system
STATISTIC 2013.08.18 09:57:46  6852 -------+----------+----------
STATISTIC 2013.08.18 09:57:46  6852 process|      0.08|      0.01
STATISTIC 2013.08.18 09:57:46  6852 childs |      0.05|      0.00
STATISTIC 2013.08.18 09:57:46  6852 -------+----------+----------
STATISTIC 2013.08.18 09:57:46  6852 sum    |      0.13|      0.01 => 0.14
INFO      2013.08.18 09:57:46  6852 syncing ...
END       2013.08.18 09:57:46  6852 checking references and copying in <repl>

$ find repl -print | sort
repl
repl/default
repl/default/2013.08.18_09.45.16
repl/default/2013.08.18_09.45.16/ls.bz2
repl/default/2013.08.18_09.45.16/.md5CheckSums.bz2
repl/default/2013.08.18_09.45.16/.md5CheckSums.info
repl/default/2013.08.18_09.45.16/.storeBackupLinks
repl/storeBackupBaseTree.conf
```

So, the result is – nothing. `storeBackupUpdateBackup.pl` thinks everything is fine because there's nothing to replicate anymore. (The control files indicate that everything ran perfectly.)

If this is not fixed, the replication will run into problems with the next backup. To make things clear, let's provoke this:

```
$ storeBackup.pl -s source -b backup --lateLinks
$ storeBackupUpdateBackup.pl -b backup
$ storeBackupUpdateBackup.pl -b repl
$ storeBackupUpdateBackup.pl -b repl
BEGIN     2013.08.18 10:06:06  7050 checking references and backup copying in <repl>
VERSION   2013.08.18 10:06:06  7050 storeBackupUpdateBackup.pl, 3.4 +
INFO      2013.08.18 10:06:06  7050 creating lock file </tmp/storeBackup.lock>
INFO      2013.08.18 10:06:06  7050 reading <repl/storeBackupBaseTree.conf>
INFO      2013.08.18 10:06:06  7050 reading </tmp/x/delta/deltaCache.conf>
INFO      2013.08.18 10:06:06  7050 copying </tmp/x/delta/default/2013.08.18_10.05.54> to <repl/default>
INFO      2013.08.18 10:06:07  7050 scanning directory <repl> for existing backups
INFO      2013.08.18 10:06:07  7050 scanning directory <repl/default> for existing backups
STATISTIC 2013.08.18 10:06:07  7050 found 1 backup series, 2 backups, 0 renamed backups
ERROR     2013.08.18 10:06:07  7050 FATAL ERROR: link <../2013.08.18_09.49.21> to non existing dir in
                                    </tmp/x/repl/default/2013.08.18_10.05.54/.storeBackupLinks/linkTo>
ERROR     2013.08.18 10:06:07  7050 found 1 inconsistencies, please repair and check again
```

The *FATAL ERROR* message found a missing backup, so there is no chance to create the hard links. Looking to the files in the delta cache (`delta`) and the replication target directory (`repl`) shows:

```
$ find delta repl -print | sort
delta
delta/default
delta/default/2013.08.18_09.45.16.copied
delta/default/2013.08.18_09.45.16.linked
delta/default/2013.08.18_09.49.21.copied
delta/default/2013.08.18_09.49.21.linked
delta/default/2013.08.18_10.05.54
delta/default/2013.08.18_10.05.54.copied
delta/default/2013.08.18_10.05.54/.md5CheckSums.bz2
delta/default/2013.08.18_10.05.54/.md5CheckSums.info
delta/default/2013.08.18_10.05.54/.storeBackupLinks
delta/default/2013.08.18_10.05.54/.storeBackupLinks/linkFile.bz2
delta/default/2013.08.18_10.05.54/.storeBackupLinks/linkTo
delta/deltaCache.conf
delta/processedBackups
delta/processedBackups/default
delta/processedBackups/default/2013.08.18_09.45.16
delta/processedBackups/default/2013.08.18_09.45.16/ls.bz2
delta/processedBackups/default/2013.08.18_09.45.16/.md5CheckSums.bz2
```

```
delta/processedBackups/default/2013.08.18_09.45.16/.md5CheckSums.info
delta/processedBackups/default/2013.08.18_09.45.16/.storeBackupLinks
delta/processedBackups/default/2013.08.18_09.45.16/.storeBackupLinks/linkFile.bz2
delta/processedBackups/default/2013.08.18_09.49.21
delta/processedBackups/default/2013.08.18_09.49.21/.md5CheckSums.bz2
delta/processedBackups/default/2013.08.18_09.49.21/.md5CheckSums.info
delta/processedBackups/default/2013.08.18_09.49.21/pwd.bz2
delta/processedBackups/default/2013.08.18_09.49.21/.storeBackupLinks
delta/processedBackups/default/2013.08.18_09.49.21/.storeBackupLinks/linkFile.bz2
delta/processedBackups/default/2013.08.18_09.49.21/.storeBackupLinks/linkTo
repl
repl/default
repl/default/2013.08.18_09.45.16
repl/default/2013.08.18_09.45.16/ls.bz2
repl/default/2013.08.18_09.45.16/.md5CheckSums.bz2
repl/default/2013.08.18_09.45.16/.md5CheckSums.info
repl/default/2013.08.18_09.45.16/.storeBackupLinks
repl/default/2013.08.18_10.05.54
repl/default/2013.08.18_10.05.54/.md5CheckSums.bz2
repl/default/2013.08.18_10.05.54/.md5CheckSums.info
repl/default/2013.08.18_10.05.54/.storeBackupLinks
repl/default/2013.08.18_10.05.54/.storeBackupLinks/linkFile.bz2
repl/default/2013.08.18_10.05.54/.storeBackupLinks/linkTo
repl/storeBackupBaseTree.conf
```

We can see the following:

1. In `repl`

   (a) Backup `repl/default/2013.08.18_09.45.16` has been copied and completed. (`.storeBackupLinks/linkFile.bz2` doesn't exist any more)

   (b) Backup `repl/default/2013.08.18_10.05.54` was copied but *not* completed. (`.storeBackupLinks/linkFile.bz2` still exists)

2. In `delta/default`

   (a) The first backup, `2013.08.18_09.45.16` is marked as copied and linked which is fine, see 1a above.

   (b) The second backup, `2013.08.18_09.49.21` is marked as copied and linked also – this is wrong, because it is missing in `repl`. (I deleted it to simulate an issue.)

   (c) The third backup, `2013.08.18_10.05.54` is marked as copied, but not linked which is correct, see 1b above.

To solve the problem, I could copy the whole completed backups from `backup` to `repl`. But in real life, this would be very time consuming and you may lose data availabe in `repl` but not in `backup` because of different deletion schemes.
Another way to solve the problem is to use `linkToDirs.pl` to copy / hard link the missing backup from `backup` directly to the correct place in `repl`. See section 6.16, "`linkToDirs.pl`" for detailed information.

The most "elegant" and fastest way is to change the delta cache according to the situation in `repl`:

1. The missing backup (see 2b above) is marked as "copied" and "linked", so I remove these to markers. See the first command below.
   ATTENTION: This only works because in this example, only **one** *replication is configured. If more replications are configured, you have to delete the name of the replication in those files. If you simply delete the files, you might probably harm other replications!*

2. But the backup (missing in `repl`) is not at its place in `delta/default`. Fortunately, it is still available in `delta/processedBackups/default`. With the second command below I move it to the right location.

Finally, the output of `find` shows all files at the right location:

```
$ rm delta/default/2013.08.18_09.49.21.copied delta/default/2013.08.18_09.49.21.linked
$ mv delta/processedBackups/default/2013.08.18_09.49.21 delta/default
$ find delta -print | sort
```

```
delta
delta/default
delta/default/2013.08.18_09.45.16.copied
delta/default/2013.08.18_09.45.16.linked
delta/default/2013.08.18_09.49.21
delta/default/2013.08.18_09.49.21/.md5CheckSums.bz2
delta/default/2013.08.18_09.49.21/.md5CheckSums.info
delta/default/2013.08.18_09.49.21/pwd.bz2
delta/default/2013.08.18_09.49.21/.storeBackupLinks
delta/default/2013.08.18_09.49.21/.storeBackupLinks/linkFile.bz2
delta/default/2013.08.18_09.49.21/.storeBackupLinks/linkTo
delta/default/2013.08.18_10.05.54
delta/default/2013.08.18_10.05.54.copied
delta/default/2013.08.18_10.05.54/.md5CheckSums.bz2
delta/default/2013.08.18_10.05.54/.md5CheckSums.info
delta/default/2013.08.18_10.05.54/.storeBackupLinks
delta/default/2013.08.18_10.05.54/.storeBackupLinks/linkFile.bz2
delta/default/2013.08.18_10.05.54/.storeBackupLinks/linkTo
delta/deltaCache.conf
delta/processedBackups
delta/processedBackups/default
delta/processedBackups/default/2013.08.18_09.45.16
delta/processedBackups/default/2013.08.18_09.45.16/ls.bz2
delta/processedBackups/default/2013.08.18_09.45.16/.md5CheckSums.bz2
delta/processedBackups/default/2013.08.18_09.45.16/.md5CheckSums.info
delta/processedBackups/default/2013.08.18_09.45.16/.storeBackupLinks
delta/processedBackups/default/2013.08.18_09.45.16/.storeBackupLinks/linkFile.bz2
```

Running `storeBackupUpdateBackup.pl` on `repl` results in the following output:

```
$ storeBackupUpdateBackup.pl -b repl
BEGIN     2013.08.18 10:39:08  7958 checking references and backup copying in <repl>
VERSION   2013.08.18 10:39:08  7958 storeBackupUpdateBackup.pl, 3.4 +
INFO      2013.08.18 10:39:08  7958 removing old lock file of process <7050>
INFO      2013.08.18 10:39:08  7958 creating lock file </tmp/storeBackup.lock>
INFO      2013.08.18 10:39:08  7958 reading <repl/storeBackupBaseTree.conf>
INFO      2013.08.18 10:39:08  7958 reading </tmp/x/delta/deltaCache.conf>
INFO      2013.08.18 10:39:08  7958 copying </tmp/x/delta/default/2013.08.18_09.49.21> to <repl/default>
INFO      2013.08.18 10:39:08  7958 scanning directory <repl> for existing backups
INFO      2013.08.18 10:39:08  7958 scanning directory <repl/default> for existing backups
STATISTIC 2013.08.18 10:39:08  7958 found 1 backup series, 3 backups, 0 renamed backups
INFO      2013.08.18 10:39:08  7958     1 directory </tmp/x/repl/default/2013.08.18_09.45.16> has no linkFrom entry
INFO      2013.08.18 10:39:08  7958 autorepair: wrote linkFrom from </tmp/x/repl/default/2013.08.18_09.45.16> to
                                    </tmp/x/repl/default/2013.08.18_09.49.21>
INFO      2013.08.18 10:39:08  7958
INFO      2013.08.18 10:39:08  7958 ----- repeating consistency check -----
INFO      2013.08.18 10:39:08  7958 scanning directory <repl> for existing backups
INFO      2013.08.18 10:39:08  7958 scanning directory <repl/default> for existing backups
STATISTIC 2013.08.18 10:39:08  7958 found 1 backup series, 3 backups, 0 renamed backups
INFO      2013.08.18 10:39:08  7958     1 directory </tmp/x/repl/default/2013.08.18_09.49.21> has no linkFrom entry
INFO      2013.08.18 10:39:08  7958 autorepair: wrote linkFrom from </tmp/x/repl/default/2013.08.18_09.49.21> to
                                    </tmp/x/repl/default/2013.08.18_10.05.54>
INFO      2013.08.18 10:39:08  7958
INFO      2013.08.18 10:39:08  7958 ----- repeating consistency check -----
INFO      2013.08.18 10:39:08  7958 scanning directory <repl> for existing backups
INFO      2013.08.18 10:39:08  7958 scanning directory <repl/default> for existing backups
STATISTIC 2013.08.18 10:39:08  7958 found 1 backup series, 3 backups, 0 renamed backups
INFO      2013.08.18 10:39:08  7958 consistency check finished successfully
INFO      2013.08.18 10:39:08  7958 listing references:
INFO      2013.08.18 10:39:08  7958   /tmp/x/repl/default/2013.08.18_09.49.21
INFO      2013.08.18 10:39:08  7958     -> /tmp/x/repl/default/2013.08.18_09.45.16
INFO      2013.08.18 10:39:08  7958   /tmp/x/repl/default/2013.08.18_10.05.54
INFO      2013.08.18 10:39:08  7958     -> /tmp/x/repl/default/2013.08.18_09.49.21
INFO      2013.08.18 10:39:08  7958 (1/2) updating </tmp/x/repl/default/2013.08.18_09.49.21>
INFO      2013.08.18 10:39:08  7958 phase 1: mkdir, symlink and compressing files
STATISTIC 2013.08.18 10:39:08  7958 created 0 directories
STATISTIC 2013.08.18 10:39:08  7958 created 0 symbolic links
STATISTIC 2013.08.18 10:39:08  7958 compressed 0 files
STATISTIC 2013.08.18 10:39:08  7958 used 0.0  instead of 0.0  (0 <- 0 ; 0.0%)
INFO      2013.08.18 10:39:08  7958 phase 2: setting hard links
STATISTIC 2013.08.18 10:39:08  7958 linked 1 files
INFO      2013.08.18 10:39:08  7958 phase 3: setting file permissions
STATISTIC 2013.08.18 10:39:08  7958 set permissions for 2 files
INFO      2013.08.18 10:39:08  7958 phase 4: setting directory permissions
STATISTIC 2013.08.18 10:39:08  7958 set permissions for 0 directories
INFO      2013.08.18 10:39:08  7958 reading <repl/storeBackupBaseTree.conf>
INFO      2013.08.18 10:39:08  7958 marked <default/2013.08.18_09.49.21> as linked in </tmp/x/delta>
INFO      2013.08.18 10:39:08  7958 scanning directory <repl> for existing backups
INFO      2013.08.18 10:39:08  7958 scanning directory <repl/default> for existing backups
STATISTIC 2013.08.18 10:39:08  7958 found 1 backup series, 3 backups, 0 renamed backups
INFO      2013.08.18 10:39:08  7958 consistency check finished successfully
INFO      2013.08.18 10:39:08  7958 listing references:
INFO      2013.08.18 10:39:08  7958   /tmp/x/repl/default/2013.08.18_10.05.54
INFO      2013.08.18 10:39:08  7958     -> /tmp/x/repl/default/2013.08.18_09.49.21
INFO      2013.08.18 10:39:08  7958 (2/2) updating </tmp/x/repl/default/2013.08.18_10.05.54>
INFO      2013.08.18 10:39:08  7958 phase 1: mkdir, symlink and compressing files
STATISTIC 2013.08.18 10:39:08  7958 created 0 directories
STATISTIC 2013.08.18 10:39:08  7958 created 0 symbolic links
STATISTIC 2013.08.18 10:39:08  7958 compressed 0 files
STATISTIC 2013.08.18 10:39:08  7958 used 0.0  instead of 0.0  (0 <- 0 ; 0.0%)
INFO      2013.08.18 10:39:08  7958 phase 2: setting hard links
STATISTIC 2013.08.18 10:39:08  7958 linked 2 files
INFO      2013.08.18 10:39:08  7958 phase 3: setting file permissions
STATISTIC 2013.08.18 10:39:08  7958 set permissions for 2 files
INFO      2013.08.18 10:39:08  7958 phase 4: setting directory permissions
STATISTIC 2013.08.18 10:39:08  7958 set permissions for 0 directories
INFO      2013.08.18 10:39:08  7958 reading <repl/storeBackupBaseTree.conf>
INFO      2013.08.18 10:39:08  7958 marked <default/2013.08.18_10.05.54> as linked in </tmp/x/delta>
INFO      2013.08.18 10:39:08  7958 scanning directory <repl> for existing backups
INFO      2013.08.18 10:39:08  7958 scanning directory <repl/default> for existing backups
STATISTIC 2013.08.18 10:39:08  7958 found 1 backup series, 3 backups, 0 renamed backups
INFO      2013.08.18 10:39:08  7958 consistency check finished successfully
INFO      2013.08.18 10:39:08  7958 found no references to backups from lateLinks that need storeBackupUpdateBackup r
INFO      2013.08.18 10:39:08  7958 backup </tmp/x/delta/default/2013.08.18_09.49.21> copied to <Backup Copy>
INFO      2013.08.18 10:39:08  7958     moving backup to </tmp/x/delta/processedBackups/default>
INFO      2013.08.18 10:39:08  7958 backup </tmp/x/delta/default/2013.08.18_10.05.54> copied to <Backup Copy>
INFO      2013.08.18 10:39:08  7958     moving backup to </tmp/x/delta/processedBackups/default>
INFO      2013.08.18 10:39:08  7958 deleting in deltaCache </tmp/x/delta> processedBackups
INFO      2013.08.18 10:39:08  7958     age for deletion is > 99d (delete backups older than Sat 2013.05.11 10:39:08)
INFO      2013.08.18 10:39:08  7958 checking series <default>
INFO      2013.08.18 10:39:08  7958     default -> 2013.08.18_09.45.16 - not old enough to delete
INFO      2013.08.18 10:39:08  7958     default -> 2013.08.18_09.49.21 - not old enough to delete
INFO      2013.08.18 10:39:08  7958     default -> 2013.08.18_10.05.54 - not old enough to delete
STATISTIC 2013.08.18 10:39:08  7958                     duration = 1s
STATISTIC 2013.08.18 10:39:08  7958  [sec] |      user|    system
STATISTIC 2013.08.18 10:39:08  7958 -------+----------+----------
STATISTIC 2013.08.18 10:39:08  7958 process|      0.12|      0.01
STATISTIC 2013.08.18 10:39:08  7958 childs |      0.04|      0.00
STATISTIC 2013.08.18 10:39:08  7958 -------+----------+----------
STATISTIC 2013.08.18 10:39:08  7958 sum    |      0.16|      0.01 => 0.17
INFO      2013.08.18 10:39:08  7958 syncing ...
END       2013.08.18 10:39:08  7958 checking references and copying in <repl>
```

The following things have happened:

- The missing backup was copied from delta cache (see log above):

  ```
  copying </tmp/x/delta/default/2013.08.18_09.49.21> to <repl/default>
  ```

  The last backup (`2013.08.18_10.05.54`) was not copied, because hat already had been done.
- The program does some auto repair (automatically), which is normal for replications.
- The program detected two uncompleted backups where the hard links (etc.) had to be set:

  ```
  listing references:
    /tmp/x/repl/default/2013.08.18_09.49.21
      -> /tmp/x/repl/default/2013.08.18_09.45.16
  ```

```
/tmp/x/repl/default/2013.08.18_10.05.54
  -> /x/repl/default/2013.08.18_09.49.21
```

Afterwards, it completed both backups.

Finally, I checked with `storeBackupCheckBackup.pl` if the replication is clean:

```
$ storeBackupCheckBackup.pl -c repl
BEGIN     2013.08.18 10:42:50  8042 checking backups in </tmp/x/repl>
VERSION   2013.08.18 10:42:50  8042 storeBackupCheckBackup.pl, 3.4 +
INFO      2013.08.18 10:42:50  8042 scanning directory <repl> for existing backups
INFO      2013.08.18 10:42:50  8042 scanning directory <repl/default> for existing backups
STATISTIC 2013.08.18 10:42:50  8042 found 1 backup series, 3 backups, 0 renamed backups
INFO      2013.08.18 10:42:50  8042 consistency check finished successfully
INFO      2013.08.18 10:42:50  8042 found no references to backups from lateLinks that need storeBackupUp
INFO      2013.08.18 10:42:50  8042 backup directories to check
INFO      2013.08.18 10:42:50  8042   /tmp/x/repl/default/2013.08.18_09.45.16
INFO      2013.08.18 10:42:50  8042   /tmp/x/repl/default/2013.08.18_09.49.21
INFO      2013.08.18 10:42:50  8042   /tmp/x/repl/default/2013.08.18_10.05.54
INFO      2013.08.18 10:42:50  8042 -- checking </tmp/x/repl/default/2013.08.18_09.45.16> ...
INFO      2013.08.18 10:42:50  8042 -- checking </tmp/x/repl/default/2013.08.18_09.49.21> ...
INFO      2013.08.18 10:42:50  8042 -- checking </tmp/x/repl/default/2013.08.18_10.05.54> ...
INFO      2013.08.18 10:42:50  8042 -- no WARNINGS OCCURRED DURING THE CHECK! --
INFO      2013.08.18 10:42:50  8042 -- no ERRORS OCCURRED DURING THE CHECK! --
END       2013.08.18 10:42:50  8042 checking backups in </tmp/x/repl>
```

9 How to use storeBackup (Examples)

9.1 Some Information in the Beginning

Before explaining some examples, it is not too bad if you know what you are doing. Here are some important aspects about how storeBackup works: (The following explains the basic mechanism, for performance reasons it is implemented a little bit different. There are several waiting queues, parallelisms and a tiny scheduler inside which are not described here.)

storeBackup uses at least two internal flat files in each generated backup:

`.md5CheckSums.info`	– general information about the backup
`.md5CheckSums[.bz2]`	– information about every file (dir, etc.) saved

When starting storeBackup.pl, it will basically do (beside some other things):

1. Read the contents of the previous `.md5CheckSums[.bz2]` file and store it in two dbm databases: dbm(md5sum) and dbm(filename) (dbm(md5sum) means, that md5sum is the key). Default is to store these databases in memory.
2. Read the contents of other `.md5CheckSums[.bz2]` files (otherBackupSeries) and store it to dbm(md5sum). Always store the last copied file in the dbm file if two different files (e.g. from different backup series) are identical. This assures that multiple versions of the same file in different backups are unified in future backups.

- Without sharing files from another backup series (simple backup), see example 1, section 9.2 and example 2, section 9.3 storeBackup.pl works as follows:
  In a loop over all files to backup it will do:
  1. Look into dbm(filename) – which contains all files from the previous backup – if the exact same file exists and has not changed. In this case, the needed information are the values of dbm(filename).
     If it existed in the previous backup(s), make a hard link and go to 3.)
  2. Calculate the md5 sum of the file to backup and look into dbm(md5sum) for that md5 sum
     If it exists there, make a hard link.
     If it does not exist, copy or compress the file.

    3. Write the information of the new file to the corresponding .md5CheckSums[.bz2] file

- With sharing of files from another backup series, see example 3, section 9.4 and example 4, section 9.5 storeBackup works as follows:
  In a loop over all files to backup it will do:

  1. Look into dbm(filename) – which contains all files from the previous backup – if the exact same file exists and has not changed. In this case, the needed information are the values of dbm(filename).
     (Now, because there are independent backups, it is possible, that a file with the same content exists in another backup series. So storeBackup.pl has to look into the dbm(md5sum) to ensure linking to the same file from all different backup series.)
  2. Calculate the md5 sum of the file to backup if not known from step 1
     look into dbm(md5sum) for that md5 sum
     If it exists there, make a hard link
     If it does not exist, copy or compress the file
  3. Write the information of the new file to the corresponding .md5CheckSums[.bz2] file.

- The Option lateLinks is used as follows (example 6, section 9.7))
  If you save your backup via NFS to a server, then most of the time will be spent for setting hard links. Setting a hard link is very fast, but if you have many thousands of them it takes some time. You can avoid waiting for hard linking if you use the option lateLinks:

  1. Make a backup with storeBackup and set `--lateLinks` (or set `lateLinks = yes`) in the configuration file. Then storeBackup will not generate any hard links, only a file will be written with the information what has to be linked.
     This newly, just generated backup is initially an incremental backup.
  2. In a separate step, call storeBackupUpdateBackup to set all the required hard links to make full backups out of these incomplete backups. Please also see section 7.6, using option lateLinks for a more detailed explanation.

Conclusions:

1. Do not delete a backup to which the hard links are not yet generated. Use storeBackupUpdateBackup.pl to set the hard links and check consistency. It is a good idea to only use storeBackup.pl or storeBackupDel.pl for the deletion of old backups.
2. All sharing of data in the backups is done via hard links. This means:
   - A backup series cannot be split across different file systems.
   - If you want to share data between different backup series, all backups must reside in the same file system.
3. Every information of a backup in the .md5CheckSums is stored with relative paths. It does not matter if you change the absolute path to the backup or backup with a different machine (server makes backup from client via NFS – client makes backup to server via NFS).
   Unresolved hard links to to other backup series (via option lateLinks) are also stored with relative paths. This means: You can move backupDir around as you like, but you should never change the relative paths between backup series before resolving all the links with storeBackupUpdateBackup.pl.

It is a good idea to use a configuration file instead of command line options. Simply call:

```
# storeBackup.pl --generate <configFile>
```

Edit the configuration file and call storeBackup in the following way:

```
# storeBackup.pl -f <configFile>
```

You can override settings in the configuration file on the command line (see Example 6).

If you have additional ideas or any questions, feel free to contact me (hjclaes(at)web.de).

9.2 Example 1: Very simple backup

This is a simple configuration with storeBackup using only the two required options (the source directory and the backup destination) and a single optional parameter, the name of a log file. This configuration will backup the source tree `/home/jim` to `/backup`:

```
# storeBackup.pl --sourceDir /home/jim --backupDir /backup/jim --logFile /tmp/storeBackup.log
```

The option `--logFile` is optional and tells storeBackup to log into the file /tmp/storeBackup.log. Otherwise it would log to stdout.
The option "backupDir" is the destination – the external USB drive or other place your copied files will reside when the backup is finished. For more info, have a look at section 3, Quick Start. If you still have questions, review subsubsection 6.2.1, storeBackup.pl Options and specifically look at the `--backupDir` option.

9.3 Example 2: Backup of multiple directories

For historical reasons, storeBackup.pl can only handle one source directory. But this drawback transforms to a feature when using option `followLinks`, because everything then becomes very easy and flexible. You can also use the well known mechanism of includeDirs and exceptDirs well-established from other programs. But that is by way of comparison uncomfortable and nasty to handle.

To use lateLinks, execute the following steps (I assume, you will backup `/home/greg/important`, `/home/jim` and `/etc` to `/backup/stbu`. You can change this later very easy.): First of all, make a special directory, e.g., `/opt/stbu`. Let's also assume that you stored storeBackup at `/opt/storeBackup`:

```
# mkdir /opt/stbu
# cd /opt/stbu
# ln -s /opt/storeBackup storeBackup
# ln -s /home/jim home_jim
# ln -s /etc etc
# ln -s /home/greg/important home_greg_important
# ln -s . backup
```

With the first symbolic link we make sure, that storeBackup itself is part of the backup. So it is possible to restore it later with cp and then use storeBackupRecover.pl for the rest.
The last symbolic link is a trick to get an exact copy of `/opt/stbu` in the backup.
Now, write a short script to start storeBackup.pl. Store it at `/opt/stbu/backup.sh`:

```
/opt/storeBackup/bin/storeBackup.pl -s /opt/stbu -b /backup/stbu \
    -S . -l /tmp/storeBackup.log --followLinks 1
```

Option `--followsLinks 1` tells storeBackup to use *the first level* of symbolic links exactly like directories. Therefore, you will find `home_jim` as a directory entry in your backup.
Finally, set the permissions of the script:

```
chmod 700 /opt/backup/backup.sh
```

Whenever you start this script, you will backup the wanted directories and your short script. You need to be root to have the required permissions to read the directories in this example. And naturally you need write permissions in `/backup/stbu`.
Now, you can simply change the directories to save or not to save by deleting or creating symbolic links in this directory.

9.4 Example 3: Make a big backup once a week, a small every day

Now you will configure a big backup of the whole machine with exceptDirs and the small one of some special directories with the option follow links. Naturally, you can also configure it the other way round, only use followLinks for both or use includeDirs.
Let's assume, you want to do:

1. Your machine mounts from other servers directory `/net` which you do not want to backup.

2. You also do not want to save /tmp and /var/tmp.
3. You want to backup the whole machine once a week to /net/server/backup/weekly.
4. You want to backup /home/jim and /home/tom/texts to /net/server/backup/daily more quickly after you finished your work.
5. Naturally, you want to share files between the two backup series
6. If you start both scripts at the same time, then new files will not be shared between these two. But over time, this will come together. But you should not start both backups at the same time when you start them for the very first time! In this case, all your files will *not* be shared!
7. You do not want to use option lateLinks, which would speed up your backups massively, because you cannot run scripts on the nfs server (or for whatever other reason).

To prepare the steps described above, you need to do the following:
For the daily backup, make a special directory (we use followLinks) like described in example 2 (you also stored storeBackup in /opt/storeBackup in this example):

```
# mkdir /opt/small-backup
# cd /opt/small-backup
# ln -s . small-backup
# ln -s /home/jim home_jim
# ln -s /home/tom/texts home_tom_texts
```

and write a backup script byBackup.sh:

```
#! /bin/sh
/opt/storeBackup/bin/storeBackup.pl -s /opt/small-backup
    -b /net/server/backup \
    -S daily -l /tmp/storeBackup.log --followLinks 1 0:weekly
```

Then write a script for the weekly backup:

```
#! /bin/sh
/opt/storeBackup/bin/storeBackup.pl -s / -b /net/server/backup -S weekly \
    -l /tmp/storeBackup.log --exceptDirs net -e tmp -e var/tmp \
    -e proc -e sys -e dev 0:daily
```

The "0" before the paths (like 0:daily) means to take the last backup of the other backup series to check for identical files.
And – naturally – the directories weekly and daily must exist inside of /net/server/backup on the NFS server.
As you can see, using command line options begin to be a little bit confusing. When configuring such examples, you should try to generate a configuration file with storeBackup.pl -g *configFile* and to use that instead.

9.5 Example 4: Backup from different machines, share data

This example shows how to make backups from different machines (not coordinated) and to share the data with hard links.
Imagine, you have defined the following boundary conditions:

1. You have a server called "server" with a separate disk which is mounted at /disk1.
2. You want to backup machine "client1" which mounts disk1 of the server at /net/server/disk1 to /net/server/disk1 and shall save to client1 in that directory.
3. You want to backup machine "client2" which mounts disk1 of the server at /net/server/disk1 to /net/server/disk1 and shall save to client2 in that directory.
4. The backup of the server runs nightly, independent of the other backups.

5. The backups of the clients run uncoordinated, that means perhaps at the same time.
6. You want to share all the data in the backup.
7. You can also make small backups of parts of the source (with data sharing), but that's the same mechanism and not detailed in this example.
8. If you have a client / server architecture like this, it is a good idea to use option lateLinks if you want to speed things up. Example 6 explains how to use it.

Write the following script for the server:

```
#! /bin/sh
<PATH>storeBackup.pl -s / -b /disk1 -S server -l /tmp/storeBackup.log \
    -e /tmp -e /var/tmp -e /disk1 -e /sys -e /dev -e /proc 0:client1 0:client2
```

Write the following script for client 1:

```
#! /bin/sh
<PATH>/storeBackup.pl -s / -b /net/server/disk1 -S client1 \
    -l /tmp/storeBackup.log -e /tmp -e /var/tmp -e /disk1 -e /sys -e /dev \
    -e /proc 0:server 0:client2
```

Write the following script for client 2:

```
#! /bin/sh
<PATH>/storeBackup.pl -s / -b /net/server/disk1 -S client2 \
    -l /tmp/storeBackup.log -e /tmp -e /var/tmp -e /disk1 -e /sys -e /dev \
    -e /proc  0:server 0:client1
```

9.6 Example 5: Different keepTimes for some directories

You can do this very easy and obvious with the following (from the previous examples) known trick. Lets say you want to keep your backup for 60 days and all files in the directory "notimportant" for only 7 days. Simply make two backups, one with `--keepAll 60d` and exclude directory "notimportant". Make the second backup with `--keepAll 7d` for the missing directory. Like described in Example 3, create a relationship between the backups. So, if you move or copy a file between "notimportant"' and the rest of your saved directories, you will not use additional space for the file.

9.7 Example 6: Using lateLinks

After reading the previous example, it should not be a problem for you to understand how to configure multiple source directories (see example 2) and how to configure cross linking between to backup (see examples 3 and 4). I now assume, that you generate a configuration file with

```
# storeBackup.pl -g stbu.conf
```

- Configure storeBackup to make a backup to your backup directory via NFS. Configure all options in the configuration file `stbu.conf` and set:

  ```
  lateLinks = yes
  lateCompress = yes
  doNotDelete = yes
  ```

  If you have a high bandwidth line, there is no need to change lateCompress from `no` to `yes`. Because of `doNotDelete = yes` you will not have to wait for the deletion of old backups.

- Make your backup(s). (Like always, the very first backup will be slow.) You do not have to do anything more from your client (NFS client) side.

- Start (via cron) on the server (NFS server = backup server):

```
storeBackupUpdateBackup.pl -b <backupDirDir> \
       -l /tmp/stbuUpdate.log
```

This will generate the missing hard links (and others).

- Start (via cron) on the server:

  ```
  storeBackupDel.pl -f <cf1> -b <backupDirDir> \
         --unset doNotDelete
  ```

  This will overwrite (unset) also the doNotDelete flag in the configuration file.
  You have to set `backupDir` in the command above to the same *location* you specified in the configuration file of storeBackup.pl. If it is the same path on the client and on the server, you do not have to overwrite it (you do not have to specify it on the command line of storeBackupUpdateBackup.pl).
  You can read more about configuration files and command line options in section 7.1.

You can also make the very first backup with the lateLinks option set. Naturally, you have to run storeBackupUpdateBackup.pl to get a complete backup.
Detailed explanations about option `lateLinks` are available in Using option lateLinks, see chapter 7.6.

10 FAQ, Frequently asked Questions

* * * * *

FAQ 1 I do not want to compress any file
I do not want to compress any file in the backup. How can I configure this?
When configuring storeBackup.pl, set the option exceptSuffix to '.*', which is the pattern for "match everything".

* * * * *

FAQ 2 Where is the GUI?
Why does storeBackup not provide a GUI (graphical user interface)?
There are several reasons why storeBackup is command line driven:

- If it is possible, you should make your backups on a regular basis via an automatic mechanism, e.g., via cron.
- If you run storeBackup on a server, there is probably no gui. Or think about the dependencies to different versions of gui libraries.
- If you want to restore data to a somehow corrupted system, the gui (if you had one running) may not start any more. Then it is fine to have a tool which you can start from any command line or recovery CD. It also makes sense to let storeBackup itself be part of the backup, see example 2.
- If you just want to restore some files, you can use any operating system specific file browser you want. So that is kind of a GUI and you only have to learn the path to the backup.

- If you want to write a separte GUI calling storeBackup, you are welcome!

* * * * *

FAQ 3 I do not need that lateLinks stuff
I only want to make my backup to an external usb drive and do not want to use this new option “lateLinks”. How can I do this?
You do not have to concern yourself with this “highly sophisticated option” (or with storeBackupUpdate-Backup.pl) if you do not use option lateLinks. Have a look at Example 1.

* * * * *

FAQ 4 Making a remote Backup with SSH (no NFS)
Under GNU/Linux, it is also possible to back up data over an SSH connection. This has the advantage that no separate network file system has to be configured (as it is the case for NFS).
In order to NFS mount the target directory, the sshfs program has to be used. It is shipped with most distributions, but it can also be obtained from http://fuse.sourceforge.net/sshfs.html[59].
The command to mount the remote directory /var/backup on the computer chronos as user “backup” to the target directory /mnt/target is:

```
# sshfs backup@chronos:/var/backup /mnt/target
```

Now storeBackup.pl has to be configured to place the backup in `/mnt/target`. After the backup, the target directory can be unmounted with `fusermount -u /mnt/target`.

Speeding up a Remote Backup over SSHFS
sshfs uses an individual network request for each individual hardlink that has to be set and for each single file that has to be deleted. Since the latency for any network operation is generally several magnitudes larger than for any local operation, backing up to a remote system can therefore be very slow even if the network bandwith is as high as for a local harddisk.
For this reason, it is strongly recommended to use the lateLinks and doNotDelete options for remote backups. Their usage allows to perform the hardlinking and deletion operations on the remote system only and generally speeds up backups by a factor of 10 to 75, depending on the amount of changed data and the latency of the network.
The general procedure is as follows:

1. Mount remote system:

   ```
   # sshfs backup@chronos:/var/backup /mnt/target
   ```

2. Do the backup:

   ```
   # storeBackup.pl --backupDir /mnt/target --lateLinks \
        --doNotDelete [other options]
   ```

3. Unmount the remote system:

   ```
   # fusermount -u /mnt/target
   ```

4. Set hardlinks on the remote system:

   ```
   # ssh -T -l backup ebox.rath.org \
       'storeBackupUpdateBackup.pl --backupDir /var/backup'
   ```

5. Delete old backups on the remote system:

[59] http://fuse.sourceforge.net/sshfs.html

```
    # ssh -T -l backup chronos \
        "storeBackupDel.pl --backupDir /var/backup [other options]"
```

Note that this requires that storeBackup is also installed on the remote system.

* * * * *

FAQ 5 I like this blocked file stuff and want to use it for all files bigger than 50 MB
To achive the desired result, simply set:

```
checkBlocksSuffix = .*
checkBlocksMinSize = 50M
```

This configuration will use blocked files for all file with a size of 50 megabyte or more. If you want another size than 50 megabyte, e.g. 800 kilobyte, set the value of `checkBlocksMinSize` to 800k.

Explanation for the experts: storeBackup.pl will generate an internal rule from the configuration above:

```
'$file =~ /.*$/' and '$size >= 52428800'
```

You can also directly use the following rule:

```
'$size >= &::SIZE("50M")'
```

to get the same result.

* * * * *

FAQ 6 How do I make a full backup of my GNU/Linux machine?
First of all, generate a configuration file:

```
storeBackup.pl -g completeLinux.conf
```

Open the configuration file with an editor of your choice and edit the following options:

```
sourceDir = /
```

Set `sourceDir` to `/`, so the whole file system will be saved.

```
backupDir=/media/drive
```

Here, I assume your attached hard disk for the backup uses path `/media/drive`. You have to change this if it is mounted elsewhere. Naturally, you also can save your backups e.g. on an nfs mount. If you do so, you can find an explanation how to back up to a remote file system via nfs in section 7.10. If you make a backup via nfs, you should read section 7.6.

Next, configure the directories you do not want to backup. We have to include `backupDir` in this list to avoid recursion.

```
exceptDirs= tmp var/tmp proc sys media
```

If there are other directories you do not want to save (e.g., nfs mounted home directories), include them into this list.

Now let's say you also want to exclude the contents of all other directories called `tmp` or `temp` (upper or lower case) anywhere in the file system. So add:

```
exceptRule= '$file =~ m#/te?mp/#i'
```

To avoid cached files, add all directories with `cache` in their names (upper or lower case) to that rule. Change the line above to:

```
exceptRule= '$file =~ m#/te?mp/#i' or '$file =~ m#cache.*/#i'
```

But now there is the risk, that perhaps some important files are not saved because the are stored in a directory called `/tmp/`, `/temp/` or a directory with e.g., `Cache` in its name.
Therefore, write all files excluded because of rule `exceptRule` in a file to check these names after the backup:

```
writeExcludeLog=yes
```

In every backup, there will be a file called `.storeBackup.notSaved.bz2` listing all these files.
To copy all file types, expecially block and character devices in `/dev`, set:

```
cpIsGnu=yes
```

For making a full backup, you also have to store the boot sector. The following script assumes your system boots from drive sda. You may need to change this value to match your system. Make the directory `/backup` and locate the following script (`pre.sh`) in that directory:

```
#! /bin/sh

rm -f /backup/MBR.prior
mv /backup/MBR.copy /backup/MBR.prior
# copy the boot loader
dd if=/dev/sda of=/backup/MBR.copy bs=512 count=1 > /dev/null 2>&1

# copy back with:
# dd if=/backup/MBR.copy of=/dev/sda bs=512 count=1
```

Set the permissions:

```
chmod 755 /backup/pre.sh
```

To call the script, set `precommand` in the configuration file:

```
precommand = /backup/pre.sh
```

To see that something is happening during the backup, set:

```
progressReport = 2000
printDepth = yes
```

Look at the `keep*` option and set the appropriate values and set `logFile` to a useful value for you.
Also set the other options to values that fit to your need.

As always, the first backup will take some time because of calculating all the md5 sums and especially because of file compression. The next backups will be *much* faster.
After making your backup, you should control which files were *not* in the backup because of option `exceptRule`.

* * * * *

FAQ 7 How do I install storeBackup on a (Synology) NAS?
The following way leads to success:

Preparation:

- activate `ssh` on the NAS
- install the packet manager link:
  `http://forum.synology.com/wiki/index.php/Overview_on_modifying_the_Synology_Server,_bootstrap,_ipkg_etc#How_to_install_ipkg`

Installation:

- Installation of storeBackup in a directory of your choice, e.g.: `/volume1/homes/admin/storeBackup`. (`/volume1` is the mount point of the disk and the partition you can use; `homes` contains the home directories including the admin home.)

- Log in as `admin`: `ssh admin@`*ip-des-nas*
- Make sure the x-bit is set:
  ```
  # chmod u+x /volume1/homes/admin/storeBackup/bin/*
  # chmod u+x /volume1/homes/admin/storeBackup/lib/stbuMd5*
  ```
- Link to `/usr/bin`:
  ```
  # ln -s /volume1/homes/admin/storeBackup/bin/* /usr/bin
  ```
- Install `md5deep`:
  ```
  # ipkg install md5deep
  ```
- Link `md5deep` as `md5sum` to `/usr/bin`:
  ```
  # ln -s /opt/bin/md5deep /usr/bin/md5sum
  ```

Now storeBackup should run on your NAS box.

* * * * *

FAQ 8 How to run storeBackup on Raspberry Pi
I got reports, that storeBackup is running on RaspberryPi (raspbmc and Raspbian GNU/Linux 7). You should take care about the following:

- set `noCompress` to 1 (more than one compression job doesn't make sense)
- Use option `saveRAM` and take care that there is enough space in your temporary directory. It seems to be best to create an own directory for temporary files and to point to it via the option `tmpdir` of `storeBackup.pl`. At least on raspbmc, temporay space (`/tmp`) seems to be so small, that even without the option `saveRAM` (which means some hash tables are stored on the disk) `storeBackup.pl` crashes with a very strange error message.

It is important to avoid swapping because on Raspberry Pi, swapping is done to something *very* slow (e.g. sdcard). Naturally, swapping depends on how much RAM your Raspberry Pi can use and about the data you are saving.
Like always, the first backup is *very* slow, but the next one is (pretty) fast, naturally depending on the slow cpu (compared ones used in mainstream PC ones) and slow media to write to.

* * * * *

FAQ 9 Can storeBackup run out of hard links?
I remember playing with anti dupe tools (deduplication) that they normaly did not hardlink zero or "small" byte files.
Do we have problems with "hard link count" / too much hardlinks? I did a test on btrfs (rsyncing an ext4 storeBackup series) and it oopsed with "too many hardlinks". I know . . . I should not use rsync and would I have used storeBackup to backup on a filesystem with lower max hardlinks storebackup would have started a new file and restart hardlinks.

For each file you need at least an inode, so hard linking zero byte files saves at least lots of inodes. This may have no result if the file system (ext) has enough statically reserved or really saves some memory if the filesystem allocates them dynamically (reiserfs, btrfs) or you run out of inodes with ext file systems. Setting a new hard link should also be faster than creating a new inode.
For storeBackup, handling of hardlinks is no problem. It tries to create a hardlink, and if this is not successful, it creates a new file and hard links against this in the future. Running out of hardlinks simply means to create a new identical file. Because of this simple and stupid algorithm, the number of hard links a file system supports is nothing storeBackup has to care about. This behavior is different to typical Unix tools like cp, tar or rsync. If you copy a directory with lots of hardlinks to to a filesystem which does not support enough hardlinks, you will get errors (see also explanation to program `linkToDirs.pl` delivered with storeBackup which bypasses this limitation in the way described above).
In case of btrfs, I would not use it for backup at the moment because I think it is not stable enough (2014). But anyway, I made some tests and its behavior seems to be very different from other file systems regarding hardlinks. It has a very limited number of hardlinks *in one directory*. I ran out of hardlinks – all files in one directory – but was able to create additional hardlinks to those inode from another directory. But anyway – because of storeBackups stupid algorithm, it can handle btrfs also efficiently.
Finally, I think there is no reason not to hard link zero byte files.

11 Contributors

Thanks to all people who shared their ideas with me, sent me bug reports and were patient enough to evolve storeBackup to what it is.
I like to list especially

- Francesco Potorti who helped a lot in bug fixing in parts of version 1.x
- Arthur Korn for lots of discussions and his support in Debian.
- Nikolaus Rath (Nikolaus at rath.org) who made substantial contributions to version 2 and rewoke my interest in continuing the development storeBackup
- W. David Shields, ViewMachine Corporation, Florida, USA, (dave at viewmachine.com) who enhanced the documentation, discussed new enhancements and found many bugs during the testing phases of version 3.
- Frank Brungräber for proofreading this documentation.

This should not neglect all the others who helped me.

12 Change Log

```
----------------------------
version 1.0 2002.05.07
first public release

----------------------------
version 1.1      2002.05.18
statistical output 'over all files/sec' was unclear
changed to:
over all files/sec (real time) =
 over all files/sec (CPU time) =
                    CPU usage =

versions are now (overall checksum):
storeBackup.pl -V         => 1.3461
storeBackupls.pl -V       => 1.2583
storeBackupVersions.pl -V => 1.4313
storeBackupRecover.pl -V  => 1.4992

----------------------------
version 1.2      2002.05.19
storeBackup.pl:
with option --exceptDirs you can also use wildcards
added option --contExceptDirsErr

storeBackupRecover.pl:
if you extract a directory (eg. abc) and there exists another
directory with a name with the same beginning (eg. abcd), this
one will also be extracted -> corrected

versions are now (overall checksum):
storeBackup.pl -V         => 1.3471
storeBackupls.pl -V       => 1.2583
storeBackupVersions.pl -V => 1.4313
storeBackupRecover.pl -V  => 1.5145

----------------------------
version 1.3      2002.05.22
all programs:
 the usage of the programms with sensless list parameters
 (like *.h) was ignored -- now an error message is produced

 storeBackupVersions.pl:
 improved performance, checks same inodes before calculating
 md5 sums

 storeBackup.pl:
 when the time for backup was < 1 sec, a division by zero could happen
 (thanks to Joerg Paysen for the report)
 added --keepMinNumberAfterLastOfDay (instead of replacing --keepMinNumber)

 versions are now (overall checksum):
 storeBackup.pl -V         => 1.3491
 storeBackupls.pl -V       => 1.2583
 storeBackupVersions.pl -V => 1.4483
 storeBackupRecover.pl -V  => 1.5159

----------------------------
 version 1.4      2002.05.27
 all programs:
 support recovering of hard links in the source tree of storeBackup.pl

 storeBackupRecover.pl
 fixed some little bugs introduced in version 1.2

 storeBackupConvertBackup.pl
 new program to convert old backup directories (target) to the new
 format of .md5CheckSums[.bz2]
 YOU HAVE TO CALL IT, IF YOU WANT TO USE VERSION 1.4 WITH OLD BACKUPS!

 versions are now (overall checksum):
 storeBackup.pl -V              => 1.3568
 storeBackupls.pl -V            => 1.2583
 storeBackupVersions.pl -V      => 1.4775
 storeBackupRecover.pl -V       => 1.4073
 storeBackupConvertBackup.pl -V => 1.9776

----------------------------
 version 1.5      2002.05.28
 storeBackup.pl
 better statistics about freed/used space on disk

 versions are now (overall checksum):
 storeBackup.pl -V              => 1.3606
 storeBackupls.pl -V            => 1.2583
 storeBackupVersions.pl -V      => 1.4775
 storeBackupRecover.pl -V       => 1.4073
 storeBackupConvertBackup.pl -V => 1.9776

----------------------------
 version 1.6      2002.06.10
 storeBackupVersions.pl
 added flags:
--showAll (same as all below)
--size (shows size of found files)
--uid (show also uid of source file)
--gid (show also gid of source file)
--mode (show also mode of source file)
--ctime (show also creation time of source file)
--mtime (show also modify time of source file)
 storeBackup.pl
```

```
added weekday to INFO output in log file when deleting old dir
via parameter --keepOnlyLastOfDay
ROADMAP is actualized

versions are now (overall checksum):
storeBackup.pl -V              => 1.3617
storeBackupls.pl -V            => 1.2583
storeBackupVersions.pl -V      => 1.4401
storeBackupRecover.pl -V       => 1.4073
storeBackupConvertBackup.pl -V => 1.9776

----------------------------
version 1.7      2002.07.2
storeBackup.pl
added flag --ignoreReadError
added flags --file, --generate, --print: you can now use a
configuration file instead of putting all in command line options

versions are now (overall checksum):
storeBackup.pl -V              => 1.2871
storeBackupls.pl -V            => 1.2972
storeBackupVersions.pl -V      => 1.3795
storeBackupRecover.pl -V       => 1.3280
storeBackupConvertBackup.pl -V => 2.0308

----------------------------
version 1.8      2002.08.17
storeBackupConvertBackup.pl
updated program to convert old backup directories (target) to the new
format of .md5CheckSums[.bz2] and .md5CheckSums.info
YOU HAVE TO CALL IT, IF YOU WANT TO USE VERSION 1.7 WITH OLD BACKUPS!
see file bin/_ATTENTION_ for detailed information

storeBackupls.pl
added option -v for verbose information

storeBackup.pl
- correction of minor errors
- added list parameter(s) otherBackupSeries
  allows you to hard link to older trees from the same backup
  allows you to hard link to backup trees of another backup series
  This gives you the possiblity to share data via hard link between
  independent backups. See README file for more information (search
  for 'otherBackupSeries').

storeBackupVersions.pl + storeBackupRecover.pl
- compatible with new file format

----------------------------
version 1.8.1    2002.08.19
Error fixing:
storeBackup.pl
- didn't build dbm(filename) correctly when first backup with
  otherBackupSeries
- pattern for recognizing of relative part of backup path did not
  work with some strange path names, pattern replaced with substr
  and length
- if the directory to backup was empty, then no .md5CheckSum.bz2
  was created

----------------------------
version 1.9      2002.08.26
```

```
storeBackup.pl
- new option --chmodMD5File
- total internal replacement for handling --onlyMD5Check
  is now handled in ::buildDBMs -> nearly as fast as without
  --onlyMD5Check
- new option --printDepth
- options --onlyMD5Check and --onlyMD5CheckOn are now only needed
  if hard linking with other backups (see otherBackupSeries)

----------------------------
version 1.9.1    2002.08.31
storeBackup.pl
- performance improvement when copying small files (< 100KB)
- error fix: --onlyMD5Check was not as fast as described in v1.9
  du to an error when making the package (but fortunately the
  correct version was in my backup)

versions are now (overall checksum):
storeBackup.pl -V              => 1.3138
storeBackupls.pl -V            => 1.2626
storeBackupVersions.pl -V      => 1.4091
storeBackupRecover.pl -V       => 1.3454
storeBackupConvertBackup.pl -V => 2.0844

----------------------------
version 1.10  2002.10.20
storeBackup.pl
- options --onlyMD5Check and --onlyMD5CheckOn are now obsolete
  storeBackup decides itself, if the functionality is needed
- you do not have to worry when using 'otherBackupSeries' if it's not
  yet ready. this is recognized automatically
- added options --withUserGroupStat --userGroupStatFile

versions are now (overall checksum):
storeBackup.pl -V              => 1.3325
storeBackupls.pl -V            => 1.2966
storeBackupVersions.pl -V      => 1.4295
storeBackupRecover.pl -V       => 1.3709
storeBackupConvertBackup.pl -V => 2.0844

----------------------------
version 1.10.1  2002.10.27
storeBackup.pl + storeBackupRecover.pl
- replaced syscall lchown with fork-exec chown
  because of error messages with perl 5.8 (SuSE 8.1)

versions are now (overall checksum):
storeBackup.pl -V              => 1.3334
storeBackupls.pl -V            => 1.2966
storeBackupVersions.pl -V      => 1.4295
storeBackupRecover.pl -V       => 1.3722
storeBackupConvertBackup.pl -V => 2.0844

----------------------------
version 1.11  2003.03.05
storeBackup.pl
- --exceptSuffix: removed '.bmp', added '.pgp'
- changed default of parameter --logFile
- new parameters:
  --plusLogStdout, --saveLogs, --compressWith,
  --logInBackupDir, --compressLogInBackupDir,
  --logInBackupDirFileName
```

```
- if called with parameter -f ... --print then
  evaluation of wildcards is performed
- correction of litte faults

 versions are now (overall checksum):
 storeBackup.pl -V              => 1.3435
 storeBackupls.pl -V            => 1.3152
 storeBackupVersions.pl -V      => 1.4406
 storeBackupRecover.pl -V       => 1.3862
 storeBackupConvertBackup.pl -V => 2.0844

----------------------------
 version 1.12  2003.04.16
 storeBackup.pl
- exception list was not taken into account when checking
  collisions from options of -t and -s
- added parameter --copyBWLimit (uses rsync for copying)
- in some cases internal linkage of duplicated files did not
  working
- added parameter --postcommand
- added statistical output for used length of queues

 versions are now (overall checksum):
 storeBackup.pl -V              => 1.3537
 storeBackupls.pl -V            => 1.3322
 storeBackupVersions.pl -V      => 1.4518
 storeBackupRecover.pl -V       => 1.4001
 storeBackupConvertBackup.pl -V => 2.0844
 llt -V                         => 1.4294
 multitail.pl -V                => 1.4555

----------------------------
 version 1.12.1  2003.05.01
 storeBackup.pl
- When copying files < 100 KB into the backup, owner and permissions
  were not set correctly. When hard linking in the next backup, this
  was corrected. -> Error fixed
- When problems with forking cp or the compression program occured,
  this was not handled correctly.

 versions are now (overall checksum):
 storeBackup.pl -V              => 1.3545
 storeBackupls.pl -V            => 1.3322
 storeBackupVersions.pl -V      => 1.4518
 storeBackupRecover.pl -V       => 1.4001
 storeBackupConvertBackup.pl -V => 2.0844
 llt -V                         => 1.4294
 multitail.pl -V                => 1.4555

----------------------------
 version 1.12.2  2003.05.18
 storeBackup.pl
- When copying files < 100 KB into the backup, sometimes the
  storeBackup internal scheduler slows down the backup -> fixed
- Files with size zero where not handled correctly -> fixed
- Some complicated if cases where not covered -> fixed
- better internal documentation
- granularity of the internal scheduler is now finer, prog should be
  about 5% faster
- added /etc/cron.daily/storebackup from Arthur Korn for Debian users

 versions are now (overall checksum):
```

```
storeBackup.pl -V             => 1.3554
storeBackupls.pl -V           => 1.3322
storeBackupVersions.pl -V     => 1.4518
storeBackupRecover.pl -V      => 1.4001
storeBackupConvertBackup.pl -V => 2.0844
llt -V                        => 1.4294
multitail.pl -V               => 1.4555

----------------------------
version 1.13  2003.07.28
        - BSD is now supported
storeBackup.pl
- Many new options for managing old backups. New/changed parameters:
  --noDelete changed to --doNotDelete
  --keepAll can now handle the 'archive flag'
  --keepWeekDay can now handle the 'archive flag'
  --keepFirstOfYear is new
  --keepLastOfYear is new
  --keepFirstOfMonth is new
  --keepLastOfMonth is new
  --firstDayOfWeek is new
  --keepFirstOfWeek is new
  --keepLastOfWeek is new
  --keepOnlyLastOfDay changed to --keepDuplicate
  --keepMaxNumber is new
  --keepMinNumberAfterLastOfDay has gone
- Correct error message if you do not have permission to read a
  file (not being root).
- Option --exceptDirs only worked correct when storeBackup was
  started in the source directory (sourceDir)
storeBackupDel.pl
- new programm to only delete old backups with the flags described
  above at storeBackup.pl

versions are now (overall checksum):
storeBackup.pl -V             => 1.3664
storeBackupls.pl -V           => 1.3509
storeBackupVersions.pl -V     => 1.3765
storeBackupRecover.pl -V      => 1.4154
storeBackupConvertBackup.pl -V => 2.0844
storeBackupDel.pl -V          => 1.3606
llt -V                        => 1.2222
multitail.pl -V               => 1.4555

----------------------------
version 1.14  2003.08.26
storeBackup.pl
- most parts of the statistical output were twice when one ore more
  old backups were deleted
- now runs on AIX
- checks, if targetDir has write permissions (better error message)
- replace statistic message:
  additional used space
  with
  add. used space in files
storeBackupDel.pl
- can use the config file of storeBackup.pl to operate
storeBackupls.pl
- can use the config file of storeBackup.pl to show analysis of
  livetime of old backups

versions are now (overall checksum):
```

```
storeBackup.pl -V              => 1.2993
storeBackupls.pl -V            => 1.2102
storeBackupVersions.pl -V      => 1.2949
storeBackupRecover.pl -V       => 1.3134
storeBackupConvertBackup.pl -V => 2.0844
storeBackupDel.pl -V           => 1.2795
llt -V                         => 1.2222
multitail.pl -V                => 1.4555

----------------------------
version 1.14.1  2003.10.25
storeBackup.pl (fixed)
- in some cases, setuid and setgid were not stored in the backup
- depending on the kernel version, permissions in the backup were
  not set correctly
storeBackupRecover.pl (fixed)
- depending on the kernel version, permissions in the backup were
  not set correctly

versions are now (overall checksum):
storeBackup.pl -V              => 1.3001
storeBackupls.pl -V            => 1.2102
storeBackupVersions.pl -V      => 1.2949
storeBackupRecover.pl -V       => 1.3147
storeBackupConvertBackup.pl -V => 2.0844
storeBackupDel.pl -V           => 1.2795
llt -V                         => 1.2222
multitail.pl -V                => 1.4555

----------------------------
version 1.15    2004.02.06
storeBackup.pl
- otherBackupSeries now understands 'from-to' and 'all'
--includeDirs is new
--exceptPattern is new
--includePattern is new
--resetAtime (in the source directory) is new
  - sets atime and mtime in the backup to the same values as in
    the source directory

deleting of old backups (storeBackup.pl, storeBackupls.pl,
                         storeBackupDel.pl)
- fixed bug with options --keepMinNumber and --keepMaxNumber
- set default value of --keepDuplicate to 7d
- result of checking old log files is now write to logfile
  inside of backup (if wanted)

storeBackupRecover.pl
- restores atime and mtime when restoring backups

llt
- output now in format yyyy.mm.dd, no longer in german format

configuration file syntax
- allows now the use of single quotes

storeBackupMount.pl
- pings server, mounts file systems, calls storeBackup and
  umounts filesystems

versions are now (overall checksum):
(these values have changed dramatically because I switched from cvs to svn)
```

```
storeBackup.pl -V              => 157.8243
storeBackupls.pl -V            => 96.8069
storeBackupVersions.pl -V      => 138.2092
storeBackupRecover.pl -V       => 171.4032
storeBackupConvertBackup.pl -V => 178.6868
storeBackupDel.pl -V           => 153.4117
storeBackupMount.pl -V         => 129.1638
llt -V                         => 103.7589
multitail.pl -V                => 62.3245

----------------------------
version 1.15.1   2004.02.08
storeBackup.pl
- fixed a bug when reading the config file
  (affecting exceptPattern, includePattern)
- fixed a bug when using 'sourceDir = /' and exceptPattern
  or includePattern

versions are now (overall checksum):
storeBackup.pl -V              => 183.5295
storeBackupls.pl -V            => 143.9218
storeBackupVersions.pl -V      => 171.1896
storeBackupRecover.pl -V       => 212.6288
storeBackupConvertBackup.pl -V => 178.6868
storeBackupDel.pl -V           => 183.3940
storeBackupMount.pl -V         => 170.2637
llt -V                         => 104.0773
multitail.pl -V                => 116.9386

----------------------------
version 1.16     2004.02.25
storeBackup.pl
- added parameter --exceptTypes
- store data in dbm files with pack / unpack
- better handling if maximum number of hard links is exceeded
- precommand and postcommand now understand single quotes nested
  in double quotes in the commandline (like ...Pattern)
- storeBackup didn't store the uncompress command correctly since
  version 1.15. This means, that storeBackupRecover could not
  restore the original version. This is because of the missing
  option '-d' in file .md5CheckSums.info. Wrong version:
     uncompress=bzip2
  but must be
     uncompress=bzip2 -d
  Change this line with an editor or use the script correct.sh

storeBackupRecover.pl
- storeBackupConvertBackup.pl had a bug, so that storeBackupRecover
  did not work any more. storeBackupRecover is now able to
  handle converted backups (again).

versions are now (overall checksum):
storeBackup.pl -V              => 183.9252
storeBackupls.pl -V            => 144.0733
storeBackupVersions.pl -V      => 171.5950
storeBackupRecover.pl -V       => 213.5498
storeBackupConvertBackup.pl -V => 178.6868
storeBackupDel.pl -V           => 183.7625
storeBackupMount.pl -V         => 170.9166
llt -V                         => 104.0773
multitail.pl -V                => 116.9386
```

```
----------------------------
version 1.16.1    2004.03.07
storeBackup.pl
- better explanations in the configuration file
  and for command line options
- better error messages
- option --print did not work for some values
- fixed a bug in the module for reading the
  configuration file with keepWeekday
- when printing to a log file and to stdout
  simultaneously, a possible error message with exit
  is now also printed to stdout
- option verbose now has the same effekt as debug=1

versions are now (overall checksum):
storeBackup.pl -V              => 184.4928
storeBackupls.pl -V            => 144.6597
storeBackupVersions.pl -V      => 172.0055
storeBackupRecover.pl -V       => 214.0630
storeBackupConvertBackup.pl -V => 178.6868
storeBackupDel.pl -V           => 184.5203
storeBackupMount.pl -V         => 171.2973
llt -V                         => 104.0773
multitail.pl -V                => 117.4461

----------------------------
version 1.16.2    2004.04.04
storeBackup.pl
- exit status is now correct (0) when running successfully
- option --verbose now prints some additionally verbose messages
  it is not similar any more to --debug 1
- the log file written into the backup now contains the
  "delete old backupevaluation"
- unsupported file type didn't generate an error message
  instead, the blew up the backup -> corrected
- integer overrun in the statistical output when saving large
  amounts of data is corrected
storeBackup_du.pl added to the package
versions are now (overall checksum):
storeBackup.pl -V              => 184.6565
storeBackupls.pl -V            => 144.2247
storeBackupVersions.pl -V      => 172.0004
storeBackupRecover.pl -V       => 214.0566
storeBackupConvertBackup.pl -V => 178.6868
storeBackupDel.pl -V           => 184.5157
storeBackupMount.pl -V         => 171.2909
llt -V                         => 104.0773
multitail.pl -V                => 116.9386

----------------------------
version 1.17      2004.09.04
storeBackup.pl
- reduced size of temporary berkeley db files
  this results in better caching (and therefore better performance
  for backups with many files)
- also print size of the berkely db files into the statistical output
- new option --unlockBeforeDel
- various little bug fixes (corrected comments and print outputs)
storeBackupMount.pl
- better exit status, distinguishes between errors in
  storeBackup und storeBackupMount
versions are now (overall checksum):
```

```
storeBackup.pl -V              => 184.9850
storeBackupls.pl -V            => 144.6790
storeBackupVersions.pl -V      => 173.1101
storeBackupRecover.pl -V       => 214.4541
storeBackupConvertBackup.pl -V => 178.6868
storeBackupDel.pl -V           => 184.8048
storeBackupMount.pl -V         => 171.8483
storeBackup_du.pl -V           =>  73.0682
llt -V                         => 104.0773
multitail.pl -V                => 118.2667

----------------------------
version 1.18      2004.06.03
storeBackup.pl
- minor corrections to statistical output
- fixed a bug with options --includePattern and --exceptPattern:
  There had to be brackets around a logical expression.
storeBackupRecover.pl
- restoring of directories with a round bracket in the name did not
  work sometimes, fixed
versions are now (overall checksum):
storeBackup.pl -V              => 185.0688
storeBackupls.pl -V            => 144.6790
storeBackupVersions.pl -V      => 173.1101
storeBackupRecover.pl -V       => 215.1446
storeBackupConvertBackup.pl -V => 178.6868
storeBackupDel.pl -V           => 184.8048
storeBackupMount.pl -V         => 171.8483
storeBackup_du.pl -V           =>  73.0682
llt -V                         => 104.0773
multitail.pl -V                => 118.2667

----------------------------
version 1.18.1    2004.06.08
storeBackup.pl
- fixed a silly bug which occured one did not use option progressReport
versions are now (overall checksum):
storeBackup.pl -V              => 185.1527
storeBackupls.pl -V            => 144.6790
storeBackupVersions.pl -V      => 173.1101
storeBackupRecover.pl -V       => 215.1446
storeBackupConvertBackup.pl -V => 178.6868
storeBackupDel.pl -V           => 184.8048
storeBackupMount.pl -V         => 171.8483
storeBackup_du.pl -V           =>  73.0682
llt -V                         => 104.0773
multitail.pl -V                => 118.2667

----------------------------
version 1.18.2    2004.06.26
storeBackup.pl
- storeBackup calculated too much md5 sums, corrected
- storeBackup had a dependency with perl versions >= 5.8,
  now it does not depend on this new version any more
versions are now (overall checksum):
storeBackup.pl -V              => 185.4812
storeBackupls.pl -V            => 145.1333
storeBackupVersions.pl -V      => 173.1101
storeBackupRecover.pl -V       => 215.5421
storeBackupConvertBackup.pl -V => 178.6868
storeBackupDel.pl -V           => 185.0938
storeBackupMount.pl -V         => 171.8483
```

```
storeBackup_du.pl -V           =>  73.0682
llt -V                         => 104.0773
multitail.pl -V                => 118.2667

----------------------------
version 1.18.3    2004.07.06
storeBackup.pl
- much better performance when used with exceptPattern or
   includePattern
storeBackupls.pl
- if used with option -f, default is to read the the location
   of the backup from the configuration file
   this default can be overwritten (if you have different mount
   points)
versions are now (overall checksum):
storeBackup.pl -V              => 185.8650
storeBackupls.pl -V            => 173.5429
storeBackupVersions.pl -V      => 173.9271
storeBackupRecover.pl -V       => 216.1658
storeBackupConvertBackup.pl -V => 178.6868
storeBackupDel.pl -V           => 185.5475
storeBackupMount.pl -V         => 172.4720
storeBackup_du.pl -V           =>  73.0682
llt -V                         => 104.0773
multitail.pl -V                => 118.2667

----------------------------
version 1.18.4    2004.07.11
storeBackup.pl
- (much) better performance because of reducing the number of
   md5sum calls when using otherBackupSeries
- the very first backup of a backup series did not hard link
   to another backup series defined with otherBackupSeries
- some temporary files were not deleted
versions are now (overall checksum):
storeBackup.pl -V              => 186.1958
storeBackupls.pl -V            => 173.9472
storeBackupVersions.pl -V      => 174.1391
storeBackupRecover.pl -V       => 216.4308
storeBackupConvertBackup.pl -V => 178.6868
storeBackupDel.pl -V           => 185.7402
storeBackupMount.pl -V         => 172.4720
storeBackup_du.pl -V           =>  73.0682
llt -V                         => 104.0773
multitail.pl -V                => 118.2667

----------------------------
version 1.19 2005.08.05
storeBackup.pl
- in some rare cases filenames were stored with a leading slash
   in .md5CheckSum. I could not be simulated by me. But the bug
   should be fixed.
- some fixes in handling of directory paths
- uid and gid were not set correctly for symbolic links in the
   backups (in the files, not the description of the files)
- formatting of file sizes with human readable number (eg. 3.5k)
   didn't work properly in all cases
- check for symbolic links before opening temporary files
- set permissions of backup root directory to 0755
   (independent of umask)
storeBackupRecover.pl
- could not restore directory '.' with option -r
- uid and gid were not set correctly for symbolic links when
  restoring, instead they were changed in the file where the
  symlink pointed to
versions are now (overall checksum):
storeBackup.pl -V              => 186.1958
storeBackupls.pl -V            => 173.9472
storeBackupVersions.pl -V      => 174.1391
storeBackupRecover.pl -V       => 216.4308
storeBackupConvertBackup.pl -V => 178.6868
storeBackupDel.pl -V           => 185.7402
storeBackupMount.pl -V         => 172.4720
storeBackup_du.pl -V           =>  73.0682
llt -V                         => 107.5789
multitail.pl -V                => 118.2667


- changed max args for GNU/Linux to 64*1024 because of possible
  problems when using multibyte character sets

----------------------------
version 1.19.1 2005.10.08
storeBackup.pl
- reduced the lenght of the command line because of problems
  with dual byte characters
- all temporary file names now have a 64 bit random number
  all (randomly generated) file names are checked for existence
  before used

----------------------------
version 1.19.2  2005.11.13
        storeBackup.pl
- when saving with --sourceDir / without using --includeDirs then
  storeBackup calculated useless md5sums

----------------------------
----------------------------
version 2.0  2008.11.09
        all programs:
- changed licence to gpl-3
- backup format is compatible to version 1.19,
  options *have changed*
- fixed several bugs
- introduction of lateLinks (this is the major change)
  - new options lateLinks, lateCompress
- new module for interpreting command line arguments and
  configuration file: a combination is now possible
- better support for files > 2GB on 64 bit operating systems
storeBackup.pl, storeBackupDel.pl:
- arguments in command line can overwrite configuration file
- new option keepRelative
- new option deleteNotFinishedDirs
storeBackup.pl:
- rewrite of core engine
- changed algorithm for linking with old backups
- directories specified with exceptDirs will now be created
  as empty directories
- new option ignorePerms
- new option cpIsGnu (support for special files)
- new option saveRAM (default is now to hold temp. DBs in RAM)
- removal of option exceptDirsSep
- renamed option withTime to suppressTime
- renamed option compressMD5File to doNotCompressMD5File
- exceptPattern has gone, now there is exceptRule (different syntax)
- includePattern has gone, now there is includeRule (different syntax)
- new option writeExcludeLog
- setting time on (absolute) symbolic link resulted in setting time
  in the original file -> corrected
storeBackupUpdateBackup.pl
- new program
- sets links asynchronously after running storeBackup with lateLinks
storeBackupSearch.pl
- new program
- allows searching in backups with a free definition depending on
  filename, size, uid, gid, ctime, mtime and file type

----------------------------
version 2.0.1  2008.12.14
storeBackupDel.pl:
- option keepLastOfWeek wasn't recognized when set in
  configuration file
storeBackup.pl:
- corrected wrong addition for statistical output of
  option progressReport

----------------------------
----------------------------
version 3.0  2009.03.15
- support of ';' as comment sign in configuration files
  (additionally to '#' for better readability)
storeBackupCheckBackup.pl
- new program, checks consistency of a backup
storeBackupDel.pl:
- option keepLastOfWeek wasn't recognized when set in
  configuration file
storeBackup.pl:
- new options for saving files blocked:
  checkBlocksSuffix
  checkBlocksSuffixMinSize
  checkBlocksSuffixBS
  checkBlocksCompr
- new options for saving files blocked:
  checkBlocksRule (0-4)
  checkBlocksBS (0-4)
  checkBlocksCompr (0-4)
  checkBlocksRead (0-4)
- new options for saving devices blocked:
  checkDevices (0-4)
  checkDevicesDir (0-4)
  checkDevicesBS (0-4)
  checkDevicesCompr (0-4)
- new option to hard link symbolic links:
  linkSymlinks
- new option for defining which files to compress:
  comprRule

----------------------------
version 3.1  2009.05.24
storeBackup.pl
- storeBackup did not backup sockets, now it does
- for new files, the md5 sum is now calculated before *and*
  after copying / compressing for safety reasons. The file could
  have been changed during that time. So the md5 sum would not
  match the real one. A file with the firstly calculated
  md5 sum later could be hard linked to the changed file which
  means there is no backup of its content.
  If both md5 sums do not match, an warning is generated and
  the md5 sum is set to ggggg... which is a not possible value.
  This problem does not exist for blocked files in v3.0.
- improved statistic at the end of a run (sum of warnings
  and errors)
- added options checkBlocksParallel and checkDevicesParallel
- added option linkToRecent
- name clashes because of compressing files (eg. add .bz2)
  were not handeld corretly - bug was introduced in 3.0
  corrected
- when making a backup with source=/ while not using
  includeDirs then the md5 sums of all files were calculated
  also after the first backup
- corrected some issues with the statistical output
- option copyBWLimit is now deprecated because
  - of internal performance optimization
  - it is useless
- new option suppressWarning
storeBackupUpdateBackup.pl
- if sourceDir=/, for the very first backup with option
  lateLinks an empty 'linkFrom' file was generated which lead
  to (useless) error messages. corrected.
storeBackupCheckBackup.pl
- now also checks if files in the backup are not listed
  in .md5CheckSum
storeBackupRecover.pl
- the directories in the path to the restored files / directories
  were not set the original permissions, corrected
llt
- added option --epoch to calculate human readable dates from
  epoch based dates
man
- man pages for all programs (Thanks to Ryan Niebur)
all programs
- solved issues with single quotes in path and filenames

---------------------------
version 3.2  2009.07.18
storeBackup.pl
- new option --highLatency, useful on high latency lines
- corrected some typos in print statements to log files
- now also checks for size of files if files with two equal
  md5 sums are detected
- fixed a bug when using *block* options. storeBackup.pl stopped
  with an error message when blocked file was existing with same
  path, filename, contents and times in another series but did
  not exist in the own series of that backup.
- plus some very minor enhancements
all programs:
- if an option in a configuration file is set to nothing, the
  default value (if exists) is used

---------------------------
version 3.2.1 2012.02.12
storeBackup.pl
- replaced File::Copy by own function, because File::Copy did not
  handle strange filenames (eg. with \n) without warnings
- read .md5CheckSum.info with algorithm for configuration file
- changed comments for some options
- new parameter fileNameWithLineFeed to option suppressWarning:
  suppresses warning if a filename contains a line feed
```

```
- write logInBackupDirFileName into .md5CheckSum.info so it can
  be identified by storeBackupCheckBackup.pl
- deletition of old backups is now done before postcommand
- backup of a blocked file (or device) didn't store all md5sums
  for all blocks in the local .md5CheckSum file if two or more
  block in one blocked file were identical
  this means it is possible to restore the data with cat or bzcat,
  but *not* with storeBackupRecover.pl !
- statistics for storage of blocked files corrected
- avoided some (useless) perl warnings about undef'ed variables
- avoided some (useless) perl warnings about gotos (happens in
  new perl versions)
- if a file cannot be hard linked, storeBackup.pl makes a new
  copy of that file. The warning about that fact was shifted to
  debug output because it confused some users
- corrected some confusing code about combinations of compression,
  lateCompression and lateLinks
- solved several possible timing issues (reading of tmp-result files
- masking for file names with \n was missing when writing into
  lateLinks command file
- avoid possibility of division by zero when calculating time
  for run (percentage) in statistics
- corrected calculation of 'sum of target all' in statistics
- directories with \n in their name didn't get right time stamps
  in the backup; corrected
- permissions on directories with \n in their names were not set
  correctly
storeBackupls.pl
- in storeBackupls.pl option keepLastOfWeek, backupDir and series
  was ignored in the configuration file
- option -v didn't work properly
- workaround for timing issue when reading value for inodeBackup
storeBackupMount.pl
- corrected filering of output from mount command
- added 'rw', 'ro' feature to overwrite read only or read write
  from /etc/fstab
storeBackupCheckBackup.pl
- read .md5CheckSum.info with algorithm for configuration file
- add option includeRenamedBackups
- changed option -b to -c (for compatibility to storeBackupRecover.pl
- many error corrections (mostly written new)
storeBackupRecover.pl
- read .md5CheckSum.info with algorithm for configuration file
- backup of a directory / file starting with '.' didn't work
- recovery of blocked files did not work in special cases
  (depending on size of the blocks and compression flag)
- permissions on directories were not restored because if wrong
  order - they are now set after restoring all files
- optimized performance (bigger block size for restoring blocked files)
- mtime of restored files was not set to original values because
  of wrong order of setting permissions (corrected)
storeBackupUpdateBackup.pl
- replaced File::Copy by own function, because File::Copy did not
  handle strange filenames (eg. with \n) without warnings
- read .md5CheckSum.info with algorithm for configuration file
- corrected line number when reporting problems with command file
  (.storeBackupLinks/linkFile.bz2)
- directories didn't get right time stamps when restoring; corrected
storeBackupVersions.pl
- read .md5CheckSum.info with algorithm for configuration file

---------------------------
```

```
version 3.3 2012.09
general
- command line option --unset now also works with list parameters
  set in configuration files
  (you can use eg. --unset otherBackupSeries with storeBackup.pl)
storeBackup.pl
- when saving blocked files or devices with a block size smaller
  than 1M, then always bzip2 is used as compression algorithm -
  doesn't matter if you eg. had chosen gzip2. In the backup, the
  suffix was eg. .gz, but compression algorithm was bzip2.
  storeBackupRecover cannot restore these backups correctly!
  Please restore with eg. zcat manually
- added rule-function COMPRESS_CHECK
- changed option checkDevicesCompr<n> from switch to option with
  parameter. Possible values are yes, no, check
- added option comprSuffix (now there exists a white list and a
  black list to decide if a file should be compressed or not;
  the rest of the files is rated by COMPRESS_CHECK)
- added option checkBlocksParallel (similar functionality
  as eg. checkBlocksParallel0)
- use DB_File now done in eval. This means, that there is now
  error message any more if this extension is not available
  -> should solve problems with several NAS boxes
- ignore option 'mergeBackupDir' used by new program
  storeBackupMergeIsolatedBackup.pl
- added statistical output 'COMPR_CHECK' for blocked files
- added keys to option 'suppressWarning':
  use_DB_File, use_IOCompressBzip2
- option 'ignoreReadError' didn't work - read errors on
  directories always were shown as WARNING only; fixed
- add some default suffixes to exceptSuffix
storeBackup.pl + storeBackupCheckBackup.pl
- storeBackup.pl didn't store correct pathname in .md5CheckSum when
  saving blocked devices, therefore storeBackupCheckBackup.pl couldn't
  check those files
- storeBackupCheckBackup.pl needed small enhancement to be able
  to read correct pathname generated from storeBackup.pl for
  blocked devices
storeBackup.pl + storeBackupUpdateBackup.pl
- fixed bug: backup with block + lateLinks; 1st backup complete;
  2nd backup with *no* changes to blocked file; 3rd backup with
  changes to blocked file (all without UpdateBackup between 1st,
  2nd and 3rd run) -> in 3rd run no blocks were linked to
  existing one
- added option --autorepair (-a) to storeBackup.pl
- fixed bug: storeBackupUpdateBackup.pl now can handle combination
  of lateLinks and not finished backups
storeBackup.pl + storeBackupDel.pl (+ all others reading config files)
- can now read compressed configuration files
  (recognizing suffix .bz2 and .gz)
storeBackupUpdateBackup.pl
- added support for replication, new options:
  --copyBackupOnly, --dontCopyBackup, --archiveDurationCopyStation,
  --dontDelInCopyStation, --genBackupBaseTreeConf,
  --genCopyStationConf
storeBackupMount.pl
- Debian (and Ubuntu) changes all executables to a name without
  the suffix '.pl'. storeBackupMount.pl now looks for
  storeBackup.pl _and_ storeBackup
storeBackupCheckBackup.pl
- corrected help text / man page
- added log file management
```

```
storeBackupDel.pl
- added 'BEGIN' and 'END' to log files for better support
  through NAGIOS plugin
- --plusLogStdout didn't work
storeBackupSetupIsolatedMode.pl
- new program
storeBackupMergeIsolatedBackup.pl
- new program
storeBackupReplicationWizard.pl
- new program
linkToDirs.pl
- new program
storeBackupCheckSource.pl
- new program

---------------------------
version 3.3.1 2013.04
linkToDirs.pl
- added option --saveRAM and --tmpdir
- option --ignoreErrors

storeBackup.pl
- removed generation and reading of file backupDir/.md5BlockCheckSum
  This redundant information is not necessary any more
- blocked files: permissons and owner/group were not set to root (if
  backup ran by root) and not to the real owner/group/permissions
- option --saveLogs was always switched on
- Storing of blocked files didn't work if an existing block had to
  be hard linked when the number of possible hard links was reached.
  (In reality, this happened esp. with big sparse files.) corrected
- added parameter noBackupForPeriod to option suppressWarning
  (thanks to Oliver Okrongli)
- added option --checkCompr / -C  (command line only)
- didn't work when path to storeBackup.pl contained a blank
  (this bug could have been present in other programs also -
  correted in lib)

storeBackupCheckBackup.pl
- also check md5 sum entries in case of already checked files
  which are there for hard links only
- added enhanced logging like in storeBackup.pl
- added option --wrongFileTables
- at the end, more errors than happen where summarizing
- added option --lastOfEachSeries

storeBackupMount.pl
- newly written. Now allows usage of more programs than
  storeBackup.pl only

storeBackupUpdateBackup.pl
- better error correction (option --autorepair)
- added enhanced logging like in storeBackup.pl
- changed useless error message to info
  (repair of 'link from' reference from not replicated series)
- blocked files: permissons and owner/group were not set to root (if
  backup ran by root) and not to the real owner/group/permissions
- added missing function 'cleanup'
- compression now runs natively without calling external program bzip2
  if possible (bzip2 used + module IO::Compress::Bzip2 available)
- changed 'cp -v' to 'cp -a' when copying delta cache information for
  replication. This allows replication on eg. samba shares
```

```
storeBackupCheckSource.pl
- added enhanced logging like in storeBackup.pl

storeBackupSetupIsolatedMode.pl
- added option --explicitBackup

dateTools.pl
- corrected wrong calculation in dateTools::sub
  this affected storeBackupUpdateBackup deletion of old replicas in
  deltaCache for intraday timeframes (not important)

----------------------------
version 3.4 2013.07
storeBackup.pl
- added rule functions MARK_DIR and MARK_DIR_REC
- added options --specialTypeArchiver and --archiveTypes

storeBackupRecover.pl
- now able to restore special files stored with option --archiveTypes
- do not overwrite special files any more if --overwrite is not set

storeBackupCheckBackup.pl
- now able to check backup when special files stored with option
  --archiveTypes

storeBackupSetupIsolatedMode.pl
- when using option --configFile, you can now use an already by this
  progrem generated configuration file to copy the metadata of the last
  backup (like in the past) again.

linkToDirs.pl
- if flag --ignoreErrors is set, also ignore if directories already
  exist

storeBackupRecover.pl
- errors from programs (eg. bzip2, cp) writing data from backup were
  not evaluated

lib/stbuMd5Exec.pl
- error messages in called compression program are now transported
  to log files
- missing waitpid inserted

----------------------------
version 3.4.1 2013.09
storeBackup.pl
- rule functions MARK_DIR, MARK_DIR_REC now work with option saveRAM
- added parameter use_MLDBM to option suppressWarning
- added error message when running out of disk space by copying
  small files (100k)

storeBackupCheckBackup.pl
- added missing entry in file with wrong md5 sums (option -w)

storeBackupRecover.pl
- fixed bug: called non-existing method getSTDERR on class simpleFork

documentation
- new chapter "internals"

----------------------------
version 3.4.2 2013.09
```

```
storeBackup.pl
- fixed bug when reading output files of external programs
  (heuristical bug)
- option --progressReport now accepts additionaly a time frame

storeBackupUpdateBackup.pl
- option --debug now works like -d (typo)

storeBackupRecover.pl
- new option --createSparseFiles

linkToDirs.pl
- new options --createSparseFiles and --blockSize

---------------------------
version 3.4.3
in library for (mostly) all programs
- in ubuntu, starting a program with sudo means $PWD is not set
  changed subroutine absolutePath to avoid issues

multiTail.pl
- changed program name from multitail.pl to multiTail.pl
  to avoid conflicts with other program called multitail
- added options --print, --color, --grep

storeBackup.pl
- changed behavior in case of (non-critical) error messages
  file .storeBackupLinks/linkFrom is written even in case of errors
- speedup through caching of already created directories in backup
  when using --lateLinks -> reduced checking if directory already
  exist on high latency remote line
- sometimes, identical blocks in blocked files were copied instead
  of hard linked (problem with parallelisms)

linkToDirs.pl
- added some error messages in case of not beeing able to read files
  (and therefore to calculate md5 sums)
- option --progressReport now accepts additionaly a time frame
- added option --printDepth

---------------------------
version 3.5
all storeBackup*.pl programs
- depend on file .md5CheckSum.Finished

storeBackupUpdateBackup.pl
- for replication: added support for wildcards in series names
  and option --createNewSeries (-C)
- added option --noWarningDiffSeriesInBackupCopy (-N)

linkToDirs.pl
- /tmp (partly) was used for temp. files instead of using $TMPDIR or
  special option
- changed the file ownership and permissions of files being pointed
at by symlinks, instead of the symlink itself / corrected

storeBackupCheckBackup.pl
- added option --tmpdir

storeBackupMergeIsolatedBackup.pl
- added option --tmpdir
```

```
storeBackupSetupIsolatedMode.pl
- added option --force

storeBackupMount.pl
- added option --tmpdir
- added options --suppressTime, --maxFilelen, --noOfOldFiles,
--saveLogs, --compressWith

storeBackup.pl
- /tmp (partly) was used for temp. files instead of using $TMPDIR or
  special option
- instead series names, now wildcards are also accepted
  (option otherBackupSeries)
- option cpIsGnu is set automatically if Linux system is recognized
- ERROR message "no permissions to read ..." does not enforce an exit
  of storeBackup.pl any more
- added option stayInFileSystem

storeBackupRecover.pl
- /tmp (partly) was used for temp. files instead of using $TMPDIR or
  special option
- library DB_File (better performance) is not a must any more
  necessarry to support some NAS boxes without additional tweaks

storeBackupReplicationWizard.pl
- /tmp (partly) was used for temp. files instead of using $TMPDIR or
  special option

storeBackupSearch.pl
- added option --tmpdir

storeBackupUpdateBackup.pl
- added option --tmpdir

multiTail.pl
- changed option --noOldFiles to --noOfOldFiles
  for better compatibility with other programs
```

13 License

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

(a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

(b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

(c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

(d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

(a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

(b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

(c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

(d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

(e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

(d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

(e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

(f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered “further restrictions” within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

   You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

   However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

   Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

   Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

   You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

    Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

    An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party’s predecessor in interest had or could give