

StoreBackup 3.0

January 10, 2009

Why backing up files?

Simple answer. Two reasons:

1. To restore the last state after eg. a hardware or software crash.
2. To recover old versions of a file or folder because it was deleted / destroyed unnoticed (eg. by a software bug) or you see later it was a mistake to delete it.

Features

- easy restore (also without storeBackup) / transparent format
- copies / compresses files to another disk and generates backups with time stamps
- splits big image files (from eg. TrueCrypt, mbox, Xen, KVM, VMware, etc.) or complete devices into small pieces and saves only differences to existing backups for saving space and time
- uses hard links for files with the same *contents* (recognizes copying, renaming, etc.)
- sophisticated including / excluding possibilities for files / directories
- also fast over slow / high latency lines

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, via NFS via SAMBA, via SSH¹ or whatever). This gives the users the ability to restore files easily and fast. The user only has to copy (and possibly uncompress) to restore the files. There is also a tool for easily restoring (sub) trees for the administrator. Every single backup from a specific time can be deleted without affecting the other existing backups.

StoreBackup can split big image files in little pieces and needs only the space for changes in these splits. Restoring these parts to the full image is also possible easily with simple tools: `cat` and possibly `bunzip2` (or whatever you used for compression). Naturally, storeBackup delivers a tool to restore everything easier. You can also split devices (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 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. You normally don't want 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. The program to be introduced here performs well and saves storage capacity:

- Directories, including their tree structure, may be copied to another location (e.g. `/home => /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 save restoring.
- The content of the files is going to be 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.

¹see FAQ4 for details about making a backup via SSH

- 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 and other criteria.
- Backed up files will be compressed, except they are marked 'exclude'. Compression may be excluded entirely.
- 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. These can be automatically deleted with easy or high sophisticated deletion rules.
- StoreBackup supports a lot of other options. They are described in this document.

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

otes

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

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

Heinz-Josef Claes, January 10, 2009

Contents

1	Installation	3
2	Quick Start for the Impatient; basic Explanations	3
3	The Idea behind it	5
3.1	Another Backup Tool? / Roots of storeBackup	5
3.2	What would be an ideal Backup Tool?	5
3.3	How storeBackup works	5
3.3.1	Reducing Disk Space	6
3.3.2	Performance	7
3.3.3	Example of a Run	9
4	Components / Programs to use	10
4.1	Supported Platforms	10
4.2	storeBackup.pl	11
4.3	storeBackupUpdateBackup.pl	20
4.4	storeBackupRecover.pl	22
4.5	storeBackupVersion.pl	23
4.6	storeBackupSearch.pl	23
4.7	storeBackuppls.pl	24
4.8	storeBackupDel.pl	26
4.9	storeBackupMount.pl	28
4.10	storeBackup_du.pl	29
4.11	storeBackupConvertBackup.pl	29
4.12	llt	29
4.13	multitail.pl	29
4.14	General concepts	30
4.14.1	configuration file and command line	30
4.14.2	Deletion of old Backups	32
4.14.3	Defining rules	33
4.14.4	Saving Image Files / raw Devices / Blocked Files	35
4.14.5	using option lateLinks	39

4.14.6	special files generated and used by storeBackup	41
4.14.7	configuring nfs	42
4.14.8	what's an inode	43
4.14.9	Statistical Output of storeBackup.pl	44
4.14.10	Limitations	45
5	How to use storeBackup (Examples)	46
5.1	Some Information in the Beginning	46
5.2	Example 1, simplest form of a backup	47
5.3	Example 2, backup of multiple directories	47
5.4	Example 3, make a big backup once a week, a small every day	48
5.5	Example 4, backup from different machines, share data	49
5.6	Example 5, different keepTimes for some directories	49
5.7	Example 6, using lateLinks	50
6	FAQ, Frequently asked Questions	50
7	Change Log	53
8	License	65

1 Installation

Installation is straightforward.

Download the source from <http://download.savannah.gnu.org/releases/storebackup/> and go to a place where you want to unpack it:

```
$ cd whereYouHaveWritePermissions
$ tar jxvf path/storeBackup-2.0.tar.bz2
```

This will create a directory storeBackup where you will find three sub directories: bin, lib, and doc. If you do not want to type the whole path every time to start one of the programs in bin, you can set your \$PATH variable (The quotes around the pwd must be back quotes, ascii code 96; some pdf read show them as normal quotes!):

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

Also set \$PATH in your .bashrc or whatever shell you are using. Alternatively, you can make symbolic links from a place where \$PATH is already set. If eg. your \$PATH also points to /usr/local/bin (and you have write permissions), you can do:

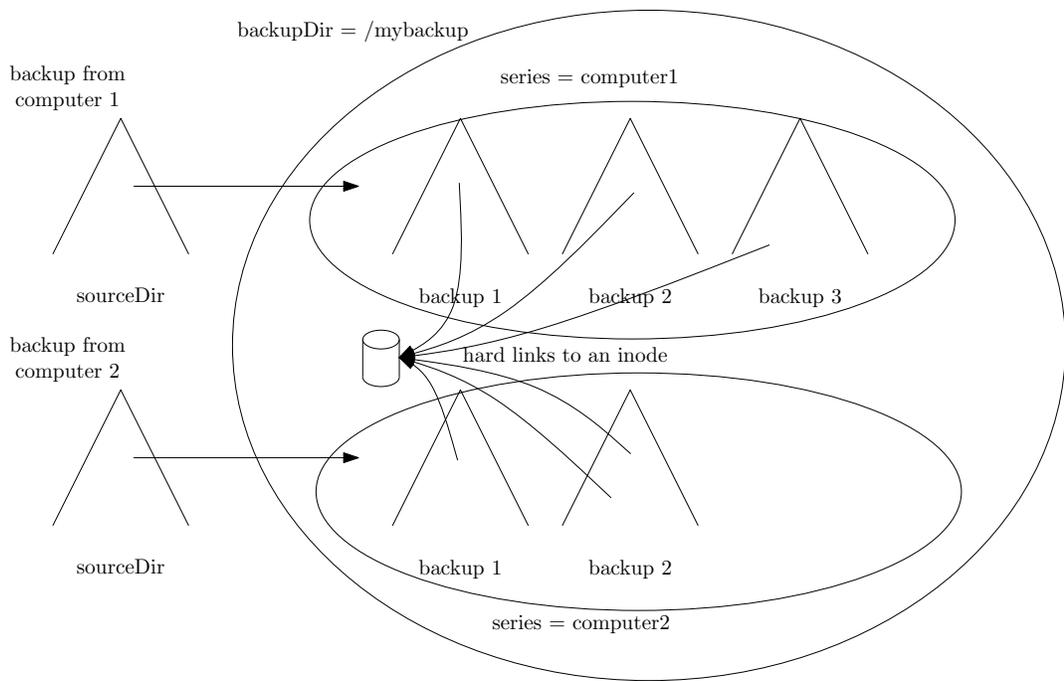
```
$ cd /usr/local/bin
$ ln -s path/storeBackup/bin/* .
```

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

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

2 Quick Start for the Impatient; basic Explanations

As you should have read, 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 on one file server via NFS. If you only want to backup on an external disk (eg. usb or esata), you should also read the following to understand the basic parameters. See section 4.14.7 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.

First of all, let's have a look at the place where you want to store your backup. You have to define a "backupDir" directory, where all your backups will reside. Let's say, this is /mybackup. To separate your different backups from multiple or one computer, you make a separate directory for each; in the example it's computer1 and computer2. We will specify these directories as "series" later. The full path to these directories is /mybackup/computer1 and /mybackup/computer2. In these "series" directories we will generate a *series of backups* for each sourceDir.

If there are files in our backup repository ("backupDir") with the same content, we can tell storeBackup to hard link them to only one inode (see section 4.14.8). The name or path of the files do not matter, only the content is important. 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.

If you want to backup more than one directory at a time, you should have a look at option followLinks in section 4.2. But just to test storeBackup, you should proceed now.

Now lets make your first backup. Let's imagine, you want to backup your home directory to /tmp/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/backup
$ cd
$ storeBackup.pl --sourceDir . --backupDir /tmp/backup
```

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.²

Depending on how much data is in your home directory, this can take a while, because 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.

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/backup
```

You will see, it's much faster now.

²see installation, section 1 if you do not know what this means

Go to `/tmp/backup`. You will see a directory called `default`. As shown in the picture above, you can have multiple series of backups from different sources. The default name for a series is `default`. Inside of the `default` directory you will see two sub directories whose names are build out of the date and time of your backups. 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. Check, 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 it’s the same file only one time on disk.

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 `sourceDir` and `targetDir` are the same.

3 The Idea behind it

3.1 Another Backup Tool? / 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, 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 to 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 or hospitals as well as universities and for general archiving.

3.2 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.

3.3 How storeBackup works

`StoreBackup` tries to accomplish the “ideal backup” and to solve the two problems: storage space and performance.

3.3.1 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 eg. 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 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 sometimes happens when unused areas in an image are blanked.)

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 eg. 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, those may be deleted indiscriminately. Unlike with traditional backups, there is no need to consider if an incremental backup is depending on previous backups. 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).

3.3.2 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 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 4.14.7)
- 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 relevant data to a save (another) place (on the backup server).
- Take this information and restore a “normal” fully linked backup.
- Delete old backups depending on the rules for the deletion.

The following performance measurements only show the direct backup time (without calling storeBackupUpdateBackup.pl (if necessary)³). 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 this were duplicates. The source for the data were my files and the “Desktop” from my Windows XP Laptop, so “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`⁴, which is a performance optimization and not the standard configuration.

command	duration	size of backup
<code>cp -a</code>	28:46	12 GB
<code>tar jcf</code>	01:58:20	9.4 GB
<code>tar cf</code>	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` is 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 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 <code>async</code>			
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

³This 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 3.3.3.

⁴see configuring nfs, section 4.14.7

We can see the following:

- The first run of storeBackup.pl is faster than `tar jcf` (tar with compression.) It's 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, eg. `.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 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
<code>cp -a</code>	37:51	12 GB
<code>tar jcf</code>	02:02:01	9.4 GB
<code>tar cf</code>	25:05	12 GB

storeBackup	1.19, Standard		2.0, Standard		2.0, lateLinks		mount with <code>sync</code>
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 and 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`⁵ 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 use storeBackup (with `lateLinks`) for making a backup over a `vpn` on high latency lines.⁶ 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 4.14.5 how to configure it.

⁵The `vpn` software was `openvpn`, the connection was tunneled trough several firewalls.

⁶You also can exclude too big files with option `exceptRule` of storeBackup.pl from the backup and save them later when you have access to a better line.

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 4.14.4) for the influence of this option to performance and space needed.

3.3.3 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 5.4.) 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
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 + compressed files) have changed.

So only 16% of the files which normally would 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 option lateLinks, I later had to run (via cron) storeBackupUpdateBackup.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.

4 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'.
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.
storeBackupVersion.pl	Analyse the versions of backed up files.
storeBackups.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.
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's more robust than 'tail -f'.

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

4.1 Supported Platforms

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.

4.2 storeBackup.pl

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

1. Directly making a 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 and left the rest to storeBackupUpdateBackup.pl which you have to run later. This is a kind of client / server mode.

```
usage:
storeBackup.pl --help
or
storeBackup.pl -g configFile
or
storeBackup.pl [-f configFile] [-s sourceDir]
    [-b backupDirectory] [-S series] [--print]
    [-T tmpdir] [-L lockFile] [--unlockBeforeDel]
    [--exceptDirs dir1,dir2,dir3] [--contExceptDirsErr]
    [--includeDirs dir1,dir2,dir3]
    [--exceptRule rule] [--includeRule rule]
    [--exceptTypes types] [--cpIsGnu] [--linkSymlinks]
    [--precommand job] [--postcommand job] [--followLinks depth]
    [--ignorePerms] [--lateLinks] [--lateCompress]
    [--checkBlocksSuffix rule] [--checkBlocksMinSize size]
    [--checkBlocksBS]
    [--checkBlocksRule0 rule] [--checkBlocksBS0 size]
    [--checkBlocksCompr0] [--checkBlocksRead0 filter]
    [--checkBlocksRule1 rule] [--checkBlocksBS1 size]
    [--checkBlocksCompr1] [--checkBlocksRead1 filter]
    [--checkBlocksRule2 rule] [--checkBlocksBS2 size]
    [--checkBlocksCompr2] [--checkBlocksRead2 filter]
    [--checkBlocksRule3 rule] [--checkBlocksBS3 size]
    [--checkBlocksCompr3] [--checkBlocksRead3 filter]
    [--checkBlocksRule4 rule] [--checkBlocksBS4 size]
    [--checkBlocksCompr4] [--checkBlocksRead4 filter]
    [--checkDevices0 devs] [--checkDevicesDir0 dir]
    [--checkDevicesBS0 size] [--checkDevicesCompr0]
    [--checkDevices1 devs] [--checkDevicesDir1 dir]
    [--checkDevicesBS1 size] [--checkDevicesCompr1]
    [--checkDevices2 devs] [--checkDevicesDir2 dir]
    [--checkDevicesBS2 size] [--checkDevicesCompr2]
    [--checkDevices3 devs] [--checkDevicesDir3 dir]
    [--checkDevicesBS3 size] [--checkDevicesCompr3]
    [--checkDevices4 devs] [--checkDevicesDir4 dir]
    [--checkDevicesBS4 size] [--checkDevicesCompr4]
    [--saveRAM] [-c compress] [-u uncompress] [-p postfix]
    [--noCompress number] [--queueCompress number]
    [--noCopy number] [--queueCopy number] [--copyBWLimit kbps]
    [--withUserGroupStat] [--userGroupStatFile filename]
    [--exceptSuffix suffixes] [--addExceptSuffix suffixes]
    [--minCompressSize size] [--comprRule]
    [--doNotCompressMD5File] [--chmodMD5File] [-v]
    [-d level][--progressReport number] [--printDepth]
    [--ignoreReadError] [--ignoreTime time]
    [--doNotDelete] [--deleteNotFinishedDirs]
    [--resetAtime] [--keepAll timePeriod] [--keepWeekday entry]
    [--keepFirstOfYear] [--keepLastOfYear]
    [--keepFirstOfMonth] [--keepLastOfMonth]
    [--firstDayOfWeek day] [--keepFirstOfWeek] [--keepLastOfWeek]
    [--keepDuplicate] [--keepMinNumber] [--keepMaxNumber]
    | [--keepRelative] ]
-l logfile
```

```

[--plusLogStdout] [--suppresTime] [-m maxFileLen]
[[-n noOfOldFiles] | [--saveLogs]]
[--compressWith compressprog]]
[--logInBackupDir [--compressLogInBackupDir]
[--logInBackupDirFileName logFile]]
[otherBackupSeries ...]

```

You have to set at least three 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.

The following options can be used on the command line and in the configuration file (see section 4.14.1). 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`).

`--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 `--includeDir`, `--excludeDir` or the recommended option better `--followLinks` (see below).

`--backupDir / -b / backupDir` The repository, where *all* your backups are stored. If you have one series of backups (eg. from one computer), this will normally be similar to 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". Eg. 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 / -t / 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.

--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 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/tmp'
```

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.

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

--includeDirs / -i / includeDirs If this option is set, then only files which are in the 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.

--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 4.14.3.

--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 4.14.3.

--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's 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).

--cpIsGnu / cpIsGnu If you choose this option, you will be able to backup (and restore) file 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, eg.:

```
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
# storeBacup.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 / from your backup.

--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 3.3.2.

--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 4.14.5.

`--lateCompress / lateCompress` This option can only be used if “lateLinks” is set. Compression of files \geq “minCompressSize” will be done later when starting `storeBackupUpdateBackup.pl`. See also section 4.14.5.

`--checkBlocksSuffix` The configuration is similar to `exceptSuffix`, a list of suffixes which are checked for a match, eg. `\.vdmk` for VMware images. They simply mean that the last part of the backup must be similar to what you define here.
The next options described here are only used if `checkBlocksSuffix` is set.
See blocked files (section 4.14.4) for more information about the options with “block” in their name.

`--checkBlocksMinSize` Only files with this minimum size will be treated as blocked files. You can use the same shortcuts as described in defining rules, see section 4.14.3, eg. 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` or `no`, the default value is `no`. On the command line, use `--checkBlocksCompr` to switch to compression mode.
This flag only affects files selected with `checkBlocksSuffix`.

`--checkBlocksRulei` The *i*th 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 4.14.4) for more information about the options with “block” in their name.

`--checkBlocksBSi` The corresponding block size for the blocks in the backup. The default value is 1 megabyte. The minimal value is 10k.

`--checkBlocksCompri` If set to `yes`, the blocks will be compressed. The default value is `no`.

`--checkBlocksReadi` Defines a filter for reading the specified file in `sourceDir`, eg. `gunzip` or `gzip -d`. This parameter 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.)

`--checkDevicesi` List of devices (eg. `/dev/sdd2 /dev/sde1` to backup).

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

`--checkDevicesBSi` 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.

`--checkDevicesCompri` If set to `yes`, the blocks will be compressed. The default value is `no`.

`--saveRAM / saveRAM` Use this option if `storeBackup.pl` runs on a system with very low memory configuration. 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, eg.:

```
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 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, eg.:

```

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 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. Default value is
1000.
--copyBWLimit / copyBWLimit Limits the maximum bandwidth in KBytes for each copy process. Store-
Backup.pl uses rsync if you configure this option. Default is 0, which means no limit (copying is then done
with cp). The needed bandwidth for compressing or setting hard links is not set by this option!
--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. The file will be overridden 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 exceptSuffix)
how to use it in the configuration file.

--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 (eg. you make the first 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 already were a file with the same
content in the first backup.)

--comprRule / comprRule As an alternative to options exceptSuffix and minCompressSize. If this rule
is set, exceptSuffix, addExceptSuffix and minCompressSize are ignored. See defining rules (sec-
tion 4.14.3) to understand how to configure rules. You can eg. define a rule that the data of several users will
not be compressed for easier restore by the users itself.
The default rule generated from option exceptSuffix and minCompressSize above is:

comprRule = '\$size > 1024' and not
           '\$file =~
           /\.zip\Z|\.bz2\Z|\.gz\Z|\.tgz\Z|\.jpg\Z|\.gif\Z|\.tiff\Z|\.tif\Z|\.mpeg\Z|\.mpg\Z|
           \.mp3\Z|\.ogg\Z|\.gpg\Z|\.png\Z/i'

```

(The line break in the pattern is syntactically not correct. It's only printed for readability.)

`--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` needs to be able to read 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).

`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` (time of last modification of file status information). Normally, you will not use this option.

`--doNotDelete / doNotDelete` Do all checks what backups should be deleted, but don't 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 *not* set.

`--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`”, eg. “`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 on the command line.
This backup deletion scheme allows you to specify the relative age of the backups you would like to have rather than 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 don't 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 don't 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.
- `--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.
- `--logFile / -l / logFile` Name of the log file. Default is `stdout`.
- `--plusLogStdout / plusLogStdout` If option `logFile` is set, here you can configure `storeBackup.pl` to also print the log messages to `stdout`.
- `--supresTime / supresTime` 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 overwrites the default value of option `noOfOldFiles`.)
- `--compressWith / compressWith` Specifies the program to compress the saved log files (eg. 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.

`--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 eg.:

```
# 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>/otherSeries
```

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

4.3 storeBackupUpdateBackup.pl

You only need to run this program if you use option `lateLinks` in `storeBackup.pl`. See section 3.3.2 about performance why you perhaps want to use this option and section 4.14.5 how to use it.

`StoreBackupUpdateBackup.pl` does the job of generation 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, eg. it detects if one backup is missing.

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

usage:

```
storeBackupUpdateBackup.pl -b backupDirectory [--autorepair]
  [--print] [--verbose] [--debug] [--lockFile] [--noCompress]
  [--progressReport number] [--checkOnly]
  [--logFile
  [--plusLogStdout] [--suppressTime] [-m maxFilelen]
  [[-n noOfOldFiles] | [--saveLogs]]
  [--compressWith compressprog]]

storeBackupUpdateBackup.pl --interactive -b topLevlDir
  [--autorepair] [--print]
```

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 corrupted backups created with option lateLinks.

--backupDir The repository, where *all* your backups are stored. If you have one series of backup (eg. from one computer), this will normally be similar to the directory where you backups are. The meaning of this parameter is similar to the option backupDir of storeBackup, see section 4.2.

--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 ERROR message in the log file and tells you what it repaired.
If you eg. delete your *last* backup with lateLinks with `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 4.14.6 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 is a very bad idea). The default is `/tmp/storeBackupUpdateBackup.lock`. You should also not run storeBackupUpdateBackup.pl in parallel to storeBackup.pl.

--noCompress / noCompress Maximal number of parallel compression operations. Default value is chosen automatically as the 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.

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

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

--supresstime / supresstime 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 overwrites the default value of option noOfOldFiles.)

`--compressWith / compressWith` Specifies the program to compress the saved log files (eg. 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.

4.4 storeBackupRecover.pl

Restores the backup tree or parts of the backup tree.

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

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 `series`. `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 `series`).

`--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 find 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`).

`--series / -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 “series”. This is only useful if you recover a small number of files.

`--overwrite / -o` Overwrite existing files. Normally not a good idea. It’s 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.

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

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

`--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`, `storeBackupRecover.pl` will not use it’s functionality. There’s no need to do so, because no special files could be backed up.

4.5 storeBackupVersion.pl

storeBackupVersion.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 contents 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 4.6.

```
storeBackupVersion.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 find 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

4.6 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 build 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 4.14.3 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] [backupRoot . . .]
```

This program allows you to set option on the command line and in a configuration file. You have to set 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 4.14.1). 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, where *all* your backups are stored. If you have only one series of backup (eg. from one computer), this will normally be similar to the directory where you backups are. The meaning of this parameter is similar to the option backupDir of storeBackup, see section 4.2.
- searchRule / -s / searchRule rule for searching, see section 4.14.3 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)
- ...backupRoot... / backupRoot On the command line, this is not an option; this is a list parameter. So you have to write on the command line eg.:

```
# 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.

4.7 storeBackups.pl

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

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

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

```
# /opt/test/storeBackup/bin/storeBackups.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/storeBackuppls.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 -> '$file = "/arconis.stib/' 'or' '$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 were used to generate the backup.

```

storeBackuppls.pl -f configFile [--print] [storeBackup-dir] or
storeBackuppls.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:

```

# storeBackuppls.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 option `-print` we see the parameters used from the command line:

```

# storeBackups.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:
<.>

```

4.8 storeBackupDel.pl

`storeBackupDel.pl` deletes old backups regarding the rules you defined. These rules (keep*) are described as options of `storeBackup.pl` in section 4.2. 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 5.7).

```

$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] [--suppresTime] [-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 options in the configuration file (especially changing `backupDir` and unsetting `doNotDelete`) on the command line. See also section 4.14.1.

The following parameters 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`
- `suppresTime`

- maxFilelen
- noOfOldFiles
- saveLogs
- compressWith

4.9 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.

```
storeBackupMount.pl -c configFile [-s server] [-l logFile]
[-d] [-p pathToStoreBackup] [-k killTime] [-m mountPoints...]
```

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,wsiz=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 4.14.7 about the configuration of nfs.

If you have these 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.

You must at least configure option `-configFile`.

The command line options storeBackupMount.pl accepts are:

- `--server / -s` Name or ip address of the of the nfs server. Default is “localhost”. This name is used for pinging.
- `--configFile / -c` Configuration file for storeBackup.pl. StoreBackupMount.pl reads option *logFile* from the configuration file of storeBackup.pl. If this log file is different from the one specified for this program, then the storeBackup.pl log file is read online and the lines are printed into the log file of storeBackupMount.pl. This is especially useful if the log from storeBackupMount.pl is directed to stdout.
- `--logFile / -l` Log file for this process, default is stdout. You can log into the same log file as storeBackup.pl. If storeBackup.pl writes to stdout, this output will be redirected into this log file, *but not till then it finished*.
- `--debug / -d` Generate some extra messages.
- `--pathStbu / -p` Path to storeBackup.pl. Has to be set if storeBackup.pl is not in your environment variable `$PATH`.
- `--killTime / -k` Time until storeBackup.pl will be killed if it didn’t finish before. The time range has to be specified in format ‘dms’, eg. 10d4h means 10 days and 4 hours. Default are 365 days.
- `--keepExistingMounts / -m` If some mounts already exists at starting time of the program, do not umount these mounts after running storeBackup.pl.
- `...mountPoints...` List of mount points needed to perform the backup. This must be a list of paths which have to be defined in `/etc/fstab`.

exit status:

- 0** everything is fine
- 1** error message from storeBackup.pl
- 2** error from storeBackupMount.pl
- 3** error from both programs

4.10 storeBackup_du.pl

storeBackup_du.pl evaluates the disk usage in one or more backup directories.

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

--verbose / -v Print accumulated values for multiple versions (days) of backed up files. Shows the steps when calculating the space used by the specified backups

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

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

4.11 storeBackupConvertBackup.pl

You only have to call this program when storeBackup.pl tells you to do so. converts old backups created with storeBackup.pl to the newest version 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
```

4.12 llt

list 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)

4.13 multitail.pl

multitail.pl reads one or multiple log files. The files read can be written in another log file and saved. This way, you can mix multiple log files.

It's 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] [-t]
              [-o outFile [-m max] [-P]
              [[-n noFiles] | [-s [-c compressprog]] ]
              ] 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 be smaller than 1, eg. 0.2. The default value is 5.

`--position / -p` Read from the begin or end of the file; allowed parameters are `begin` or `end`. Default is `begin`.

`--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 of `multitail.pl` to the log file; default is not to write it

`--maxlines / -l` Maximum number of line 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 (eg. 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.

4.14 General concepts

4.14.1 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 use environ 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 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
```

is equivalent. This also shows an option, which can have exactly *one* parameter (the file name). There are others option 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 define multiple directories not to 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 section4.2, the parameter of this option is parsed as if it were 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.

Sometimes, you can also use list parameters (parameters without an option). Eg. in storeBackup.pl they are called otherBackupDirs.

bringing both together

In some programs (like storeBackup.pl) you can use both command line options and a configuration file. Normally, it's easier to use and more clearly arranged by using the configuration file.

But in some situations it's 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 *otherBackupDirs*, for `storeBackupSearch.pl` it’s *backupRoot*. This is documented for the individual programs (see description).

4.14.2 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 its name, eg:

```
$ mv 2003.07.28_06.12.41 2003.07.28_06.12.41-archive
```

IMPORTANT: If you use option `lateLinks` of `storeBackup.pl`, only do this after a successful run of `storeBackupUpdateBackup.pl`!

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 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 parameter `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 `keepAll` . . . `keepLastOfWeek` and `keepDuplicate`.
- 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` . . . `keepLastOfWeek`. Backups matching an archive flag will never be delete 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 week” 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 than 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 don't 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 don't 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 than 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.

4.14.3 Defining rules

Rules can be defined in `storeBackup.pl`, see section 4.2 (options `excludeRule` and `includeRule`) and in `storeBackupSearch.pl`, see section 4.6 (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`), but the rules are 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.)

This rule will match for all files with more than 1.5GB (1.5 * 1024³) 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's 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':

<code>\$file</code>	file name with relative path from original <code>sourceDir</code>
<code>\$size</code>	size of the file in bytes
<code>\$mode</code>	mode of the file (integer, use 0... to compare with octal value, eg. <code>\$mode = 0644</code>)
<code>\$ctime</code>	creation time in seconds since epoch (Jan 1 1970), see below
<code>\$mtime</code>	modify time in seconds since epoch, see below
<code>\$uid</code>	user id (string if defined in operating systems), eg. <code>\$uid eq "bob"</code>
<code>\$uidn</code>	user id (numerical value), eg. <code>\$uidn = 1001</code>
<code>\$gid</code>	group id (string if defined in operating system), see <code>\$uid</code>
<code>\$gidn</code>	group id (numerical value), see <code>\$uidn</code>
<code>\$type</code>	type of the file, can be one of <code>SbcFpl</code> , see option <code>exceptTypes</code> in <code>storeBackup</code>

If you use `ctime` or `mtime`, it's not pure fun to calculate the number of seconds since epoch 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 format above, you specify "now" minus that period.

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 format above, you 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.

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 message 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 also.

4.14.4 Saving Image Files / raw Devices / Blocked Files

The scope of blocked files

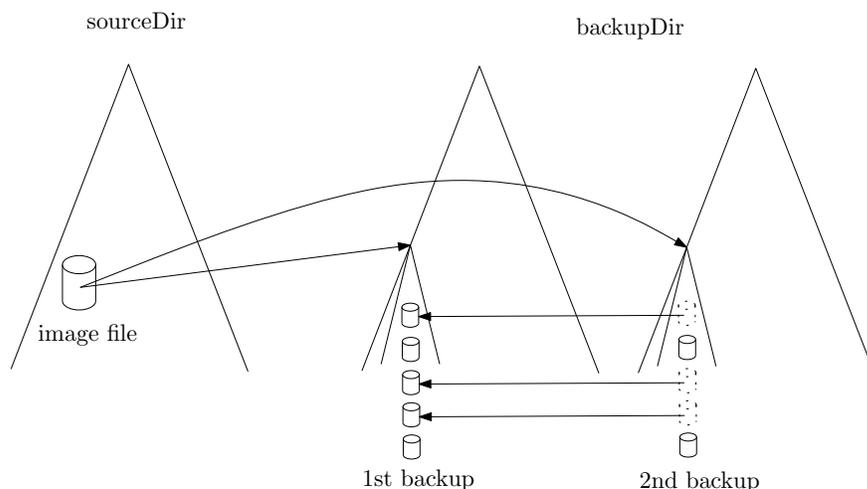
Saving big images files which change only in parts completely every backup is inefficient: very time and space consuming. To give some examples:

- Some mailers use traditional mbox (mailbox) format to save email. This is convenient, because it's 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 portion of it has changed. The same category are the `.pst` files from Outlook. If you have to save these 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 eg. 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 whole (storeBackup doesn't need to now and doesn't know anything about the content it saves).
- Images for Hypervisors like Xen, KVM or VMware are another example which you can save very successful as "blocked files".
- Do not use blocked files which are encrypted as a whole like jpegs or other types of compressed or encrypted files (.gz, .bz2, .pgp, openoffice files, etc.). Changing something in that files result mostly 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 save blocked (see below how to do this), then storeBackup.pl will do the following (first backup):



1. Create a directory with the same combination of path and file name of the original image file in the backup.
2. Split the source file into blocks and save them into the new directory. (If there are duplicates of a block, they will be hard linked.)
3. The md5 sum of all these files will be stored in a special file called `.md5BlockChecksums.bz2` in that directory. They are also stored together with all check sum of those blocked files in `.md5BlockChecksums.bz2` in the root of the backup.
4. storeBackup.pl will also calculate the md5 sum of the whole file and store it in `.md5Checksum`.

In the second and all later runs, storeBackup.pl will do the following:

1. Create the directory as described above.
2. Split the source file into blocks and see, if any of these blocks exist anywhere in a backup (see option `otherBackupDirs` 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 additional information about the blocks (like the md5 sums) is written into the files described above.

Because all files are referenced via hard links, every backup is a full backup. There are no “original” files. If you use option `lateLinks`, see section 4.14.5, the links will be set later. If you also use option `lateCompress`, the compression will be done later also.

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, eg. `\.vmdk` for VMware images. They simply mean that the last part of the backup 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 be treated as blocked files. You can use the same shortcuts as described in defining rules, see section 4.14.3, eg. `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` or `no`, the default value is `no`. On the command line, use `--checkBlocksCompr` to switch to compression mode.

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
checkBlocksCompr0 = yes
```

2. The more flexible way to specify the handling of blocked files is to use rules like described in defining rules, see section 4.14.3. The following options are available five times, so there is a `checkBlocksRule0`, `checkBlocksRule1`, `checkBlocksRule2`, `checkBlocksRule3` and `checkBlocksRule4`:

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

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

`checkBlocksCompri` If set to `yes`, the blocks will be compressed. The default value is `no`.

`checkBlocksReadi` Defines a filter for reading the specified file in `sourceDir`, eg. `gunzip` or `gzip -d`. This parameter 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.)

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=yes
```

You also defined rule 1, which match for all files bigger than 50 megabytes which end with `.pst` or are 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 4.14.3.

You can use `checkBlocksSuffix` and `checkBlocksRulei` at the same time in one configuration file. `StoreBackup` first evaluates `checkBlocksRulei` (in ascending order) 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 `checkDevicesi`, the block size in the backup with `checkDevicesBSi` and switch compression on or off with `checkDevicesCompri`. Additionally, you have to specify the relative path with `checkDevicesDiri` 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 parameters are in detail:

`checkDevicesi` List of devices (eg. `/dev/sdd2 /dev/sde1` to backup).

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

`checkDevicesBSi` 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.

`checkDevicesCompri` If set to `yes`, the blocks will be compressed. The default value is `no`.

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 4.14.5), 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` or `ext3`), 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 show the same relationship regarding the size of the first backup and 10 times the second one. 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 image 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%

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 parallelizing 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's best to make your own tests to get a feeling of useful block sizes in your use cases.

4.14.5 using option lateLinks

You can use storeBackup as one program (storeBackup.pl) which does everything alone or you can split the different task into several pieces. There is mostly one advantage to run different programs for the different task: 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's a good idea to speed up (see section 3.3.2, 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 later what this means.
2. Loading information 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, also the directory structure is generated. For every file, the owner and permissions are set in the same way as in the source directory.
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 option `lateLinks`, then 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 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`).

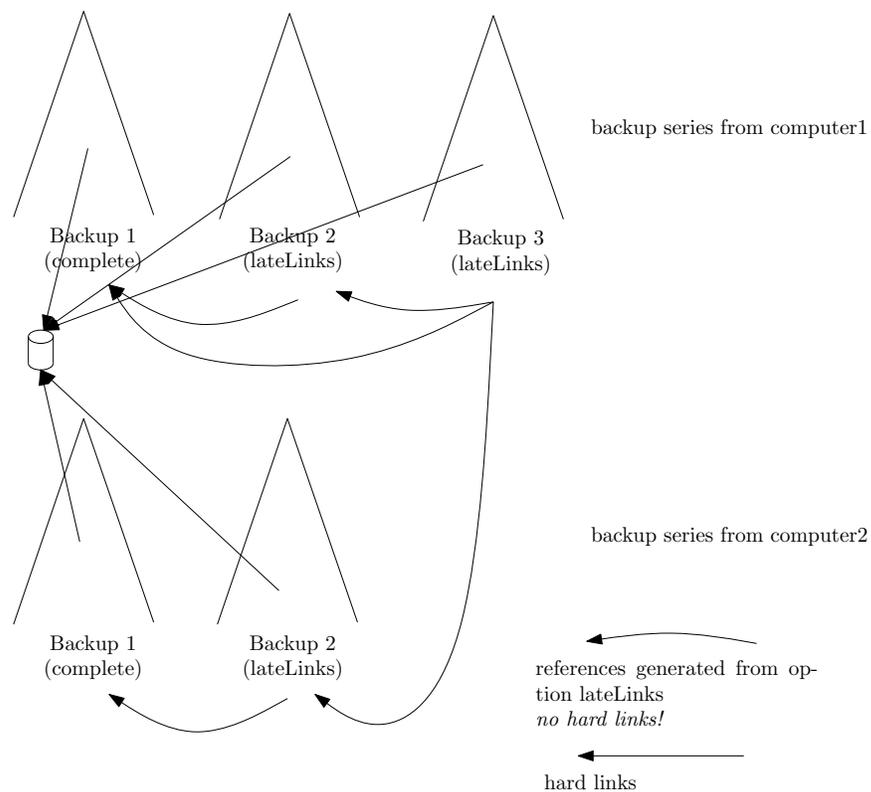
Independent of option `lateLinks` you can always configure `storeBackup.pl` to not perform step 6, the deletion of old backups. Especially if you are writing your backup on an `nfs` mount, this will take some time and lengthen you backup. Use `storeBackupDel.pl` (which can read the configuration file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

- 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.
- But all the information what’s necessary to complete to a full backup are available in the backup. 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, you will destroy *all backups* which are referencing (directly in indirectly) to that backup. **Only delete backups with `storeBackup.pl` or `storeBackupDir.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.

The following picture shows two cross linked backup series (from different computers).⁷ 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 4.14.8). 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.

⁷see section 4.2, option `otherBackupDirs` to see how this can be configured



To complete a backup (make all those nasty linking and compressing and so on), use `storeBackupUpdateBackup.pl`, see section 4.3. It also will analyse your backups (below “`backupDir`”) and find the right order to complete them. *After successful running `storeBackupUpdateBackup.pl` your backups will be in the same state as if you ran your backup without option `lateLinks`. File `.storeBackupLinks/linkFile.bz2` (and some others) are deleted and everything is hard linked (and compressed), also the permissions are set like in the source directory (except if set option `ignorePerms` in `storeBackup.pl`).*

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

4.14.6 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. Don’t delete them. Also make sure you do not have these 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, ... 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 The existence of this file indicates, that this backup was not properly finished.

.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 option write-ExcludeLog of storeBackup.pl) which are *not* stored in the backup.

The following file only exist if you use option lateLinks, see section 4.14.5. After a successful run of storeBackupUpdateBackup.pl, see section 4.3, 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
```

4.14.7 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 then can 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).

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 `nfserver` to `/backup` on your client. This will occur if you boot or if you type:

```
# mount /backup
on the nfs client.
```

There are my other option 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 (eg. `/backup` read only. In the configuration file for `storeBackup`, use:

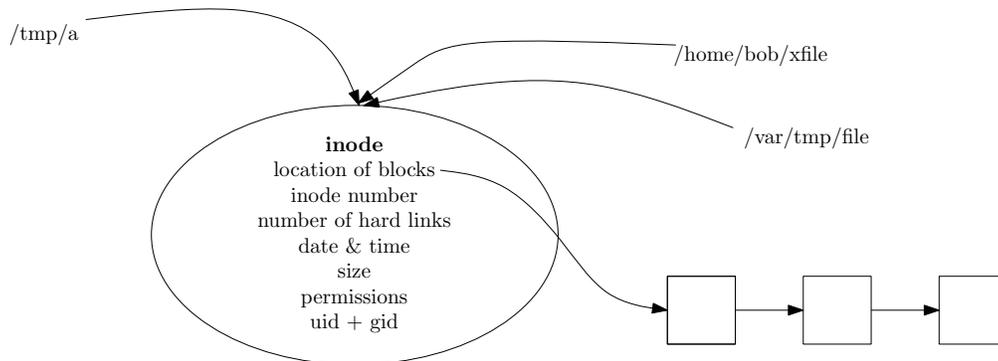
```
precommand = mount /backup -o remount,rw
postcommand = mount /backup -o remount,ro
```

This will give `storeBackup.pl` write access (`rw` = read write) during the backup. Naturally, you can also wrap a script around `storeBackup.pl` what does the same.

2. Simply arrange two mount points to the NFS server: one `rw` and one `ro`. Limit access to the read only (`ro`) 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 4.9 for this.

4.14.8 what's 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, than 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 4.14.10, 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, a 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)!

4.14.9 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 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 backup) (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 a 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 -> 10000 files processed (6.1G, 349M) storeBackup read 10000 files with 6.1GB, 349MB`
`have been copied or compressed up to now`

4.14.10 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.

5 How to use storeBackup (Examples)

5.1 Some Information in the Beginning

Before explaining some examples, it's not too bad if you know what you are doing. Here are some important aspects about how storeBackup works: (The following explains the principle mechanisms, for performance reasons it's 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 (otherBackupDirs) 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.

- This item describes how storeBackup.pl works without sharing files from another backup series (simple backup), see example 1, section 5.2 and example 2, section 5.3.

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 look into dbm(md5sum) for that md5 sum
if it exists there, make a hard link
if it doesn't exist, copy or compress the file
3. write the information of the new file to the corresponding .md5CheckSums[.bz2] file

- This item describes how storeBackup works with sharing of files from another backup series, see example 3, section 5.4 and example 4, section 5.5.

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 contents 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 doesn't exist, copy or compress the file
3. write the information of the new file to the corresponding .md5CheckSums[.bz2] file

- This item describes the usage of Option lateLinks, example 6, section 5.7 below

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.

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 4.14.5, 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's 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 over different partitions.
 - If you want to share data between different backup series, all backups must reside on the same partition.
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 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`.

If you have additional ideas or any questions, feel free to contact me ([hjclaes\(at\)web.de](mailto:hjclaes(at)web.de)).

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).

5.2 Example 1, simplest form of a backup

This is the most simple kind of configuration with `storeBackup`. It will backup source tree `/home/jim` to `/backup`:

```
# storeBackup.pl -s /home/jim --backupDir /backup/jim -S . \
    -l /tmp/storeBackup.log
```

Option `-l` is optional and tells `storeBackup` to log into that file. Otherwise it would log to stdout.

If you do not understand what `backupDir` means, have a look at section 2, Quick Start. If you still do not have any idea why it's there, you should dig into `storeBackup` a little bit deeper, and read section 4.14.5, using option `lateLinks`.

5.3 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 very easily change this later.). First of all, you make a special directory, eg. `/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's 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 you should 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.

5.4 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 option followLinks. Naturally, you can also configure it the other way around, 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 don't want to backup
2. you also don't 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.

5.5 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` to `/disk1/server`
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’s a good idea to use option `lateLinks` if you want to speed 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:client1 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

```

5.6 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.

5.7 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 you backup directory via NFS. You configure all options in the configuration file `stbu.conf` and you set among others:

```
lateLinks = yes
lateCompress = yes
doNotDelete = yes
```

If you have a high bandwidth, there is no need to set `lateCompress` 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 -f stbu.conf -b <backupDirDir> \
    -l /tmp/stbuUpdate.log
```

Where you have to set `backupDirDir` in the command above to the same *location* you specified in the configuration file of `storeBackup.pl`. If it's the same path on the client and on the server, you don't have to overwrite it (you don't have to specify it on the command line of `storeBackupUpdateBackup.pl`). (You can read more about configuration files and command line options in section 4.14.1.)

- Start (via cron) on the server:

```
storeBackupDel.pl -f <cfl> -b <backupDirDir> \
    --unset doNotDelete
```

This will overwrite (unset) also the `doNotDelete` flag in the configuration file.

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.

6 FAQ, Frequently asked Questions

- 1 I don't want to compress any file
- 2 Where is the GUI?
- 3 I do not need that `lateLinks` stuff
- 4 Making a remote Backup with SSH (no NFS)
- 5 I like this blocked file stuff and want to use it for all files bigger than 50 MB
- 6 How do I make a full backup of my GNU/Linux machine?

* * * * *

FAQ 1 I don't want to compress any file

I don't want to compress any file in the backup. How can I configure this?

When configuring `storeBackup.pl`, set option `exceptSuffix` to `.*`, which is the pattern for "match everything".

* * * * *

FAQ 2 Where is the GUI?

Why `storeBackup` does not support any gui (graphical user interface)?

There are several reasons, why `storeBackup` is command line driven:

- If it's possible, you should make your backups on a regular basis via an automatism, eg. via cron.
- If you run storeBackup on a server, there probably is no gui. Or think about the dependencies to different versions of gui libraries.
- If you want to restore data to a somehow corrupted system, perhaps the gui (if you had one running) does not start any more. Then it's 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.

* * * * *

FAQ 3 I do not need that lateLinks stuff

I only want to make my backup to an external usb drive and don't want to use this new option "lateLinks". How can I do this?

Have a look at Example 1. You don't have to care about this "high sophisticated option" (and storeBackupUpdateBackup.pl) if you do not use option lateLinks.

* * * * *

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 mount the target directory, the sshfs program has to be used. It is shipped with most distributions, but can also be obtained from <http://fuse.sourceforge.net/sshfs.html>⁸.

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 only 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 bandwidth 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 remote system:

```
# fusermount -u /mnt/target
```

4. Set hardlinks on remote system:

```
# ssh -T -l backup ebox.rath.org \  
'storeBackupUpdateBackup.pl --backupDir /var/backup'
```

⁸<http://fuse.sourceforge.net/sshfs.html>

5. Delete old backups on remote system:

```
# 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 archive 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, eg. 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 parameters:

```
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's mounted elsewhere. Naturally, you also can save your backups eg. on an nfs mount. If you do so, you can find an explanation how to a remote file system via nfs in section 4.14.7. If you make a backup via nfs, you should read section 4.14.5.

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 a other directories you do not want to save (eg. 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 list. Change the line above to:

```
exceptRule= '$file =~ m#/te?mp/#i' or '$file =~ m#cache.*/#i'
```

But now there have the risk, that perhaps some important files are not saved because the are stored in a directory called `/tmp/`, `/temp/` or a directory with eg. `Cache` in its name.

So 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, especially block and character devices in `/dev`, set:

```
cpIsGnu=yes
```

For making a full backup, you also have to store the boot sector. Make the directory `/backup` and locate the following script (`pre.sh`) in that directory:

```
#!/bin/sh

rm -f /backup/MBR.copy
# 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`.

7 Change Log

```
-----
version 1.0
first official 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
storeBackuppls.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
storeBackuppls.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 programmes with senseless 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
storeBackuppls.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
storeBackuppls.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
storeBackuppls.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
storeBackuppls.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
storeBackuppls.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

storeBackuppls.pl
added option -v for verbose information

storeBackup.pl
- correction of minor errors
- added list parameter(s) otherBackupDirs
 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 possibility to share data via hard link between
 independent backups. See README file for more information (search
 for 'otherBackupDirs').

storeBackupVersion.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
  otherBackupDirs
- 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 otherBackupDirs)

-----
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
  due 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
storeBackuppls.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 'otherBackupDirs' if it's not
  yet ready. this is recognized automatically
- added options --withUserGroupStat --userGroupStatFile

versions are now (overall checksum):
storeBackup.pl -V          => 1.3325
storeBackuppls.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

```

```
storeBackups.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
storeBackups.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
storeBackups.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
storeBackups.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
storeBackuppls.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
storeBackuppls.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 series 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

storeBackuppls.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
storeBackuppls.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
storeBackuppls.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

- otherBackupDirs 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, storeBackuppls.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
storeBackuppls.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
storeBackuppls.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
storeBackuppls.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 effect as debug=1

versions are now (overall checksum):

storeBackup.pl -V	=> 184.4928
storeBackuppls.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
storeBackuppls.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
storeBackuppls.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
storeBackuppls.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
storeBackuppls.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
storeBackuppls.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
storeBackuppls.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
storeBackuppls.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 otherBackupDirs
- the very first backup of a backup series did not hard link
  to another backup series defined with otherBackupDirs
- some temporary files were not deleted
versions are now (overall checksum):
storeBackup.pl -V          => 186.1958
storeBackuppls.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
storeBackups.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

8 License

Copyright (C)

Dr. Heinz-Josef Claes (2000-2008) [hjclaes at web.de]

Nikolaus Rath (2008) [Nikolaus at rath.org] (who made substantial contributions to version 2)

PREAMBLE

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program—to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use,

which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

“This License” refers to version 3 of the GNU General Public License.

“Copyright” also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

“The Program” refers to any copyrightable work licensed under this License. Each licensee is addressed as “you”. “Licensees” and “recipients” may be individuals or organizations.

To “modify” a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version” of the earlier work or a work “based on” the earlier work.

A “covered work” means either the unmodified Program or a work based on the Program.

To “propagate” a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey” a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The “source code” for a work means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.

A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component”, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work’s System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

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.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product” is either (1) a “consumer product”, which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

“Installation Information” for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is

characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

“Additional permissions” are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- (a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- (b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- (c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- (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 under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor’s “contributor version”.

A contributor’s “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor’s essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a “patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant” such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this

particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient’s use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is “discriminatory” if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others’ Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License “or any later version” applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy’s public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR

OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.