KeepBackup

# What is KeepBackup?

KeepBackup is a simple yet smart tool to back up your files. It was mainly developed with photos in mind, but course it can be used for documents of any kind of any files you might want to backup.

# Why another backup program?

Well, I couldn’t find a program which was simple, smart and free. I’ll describe my requirements and what makes KeepBackup special in the following text.

I used robocopy for backups a long time. My main concern was, that I delete over overwrite a file by accident and then refresh my backups. After that I notice the file is gone, but I can’t restore it, since it was already removed from my backups. So, the restore of deleted and overwritten files was an issue for me.

Also, I sometimes rename or move a large number of files (photos) and it was very annoying that the files were backed up all over again.

# What’s the status?

You might call it a working prototype. I’m using it for my backups. However, there are many things that can be improved. And it’s currently a command line tool, there is no GUI.

# Technology?

I’s written in C# based on .NET Framework.

It uses the Apache log4net library to write logfiles and make console output. And it contains source code from the LZMA SDK (7-zip) for compression.

More information about these components can be found here:

<https://logging.apache.org/log4net/>

<https://logging.apache.org/log4net/license.html>

<http://www.7-zip.org/sdk.html>

As I understand their licensing terms it can be freely used and redistributed. If there are any licensing issues, please contact me.

# Key issues for backups?

I’ll try to explain what I want from a backup tool. KeepBackup was designed with these requirements in mind.

1. The tool should backup a directory (“origin”) to another directory (“storage”), which is maybe on a USB disc or a NAS.
2. The tool would be run regularly, maybe daily or weekly.
3. It should never backup the same file twice. Every file that has been backed up before and is unchanged should not be backed up to the same storage again. This must be true, even if the file was renamed or moved. There is no need to backup it again, it just should be restored to its new path - in case you ever need to restore a backup.
4. If there are multiple copies of the same file at origin, only one backup is needed in storage.  
   Of course while restoring the multiple copies should be restored as they were.
5. Each backup to the same storage must be restorable, so also the old ones (maybe 2 weeks ago or a year ago, or yesterday).
6. Each backup should be equally important. No distinction between full backup, incremental backups or whatsoever.
7. It must be possible to restore just a subdirectory or a single file.
8. It must be possible to clean up space on storage by removing old backups. But there has be a preview, what will be removed indefinitely before doing it.
9. Is should be possible the track and analyze changes over time. So, it should be possible to track changes from one backup to the next, but also from an arbitrarily old backup to a arbitrarily newer one. Which files were deleted? Moved? Overwritten? Renamed?

# How is this achieved?

Most people have a path and a filename in mind, if they think about a file. I addition there is some metadata, especially the date and time of creation and last modification. For most programs, the filename and path is the key to the file.

For KeepBackup however, the path and filename is also just metadata. The files are managed and identified by their checksum.

So, if the same file has already been backed up before, KeepBackup would not backup it again, because the checksum is already present in storage.

# Some more terms

Origin is the source directory you want to backup. Storage is the target.

An inventory is a xml file containing the directory structure of origin to a specific point in time. If you open an inventory in a xml editor or text editor you’ll find the directories, file names, checksums and their date and time of creation and last modification.

An inventory is created each time you make the first backup of an origin and thus create a new storage, or if you make a new backup to an existing storage.

The manifest is an xml file in the storage which contains information about all files in storage from all backups. Strictly this file isn’t needed. However, it’s nice for testing and so check consistency of the storage.

# Some more nice features

1. You can blacklist files and directories via configuration. I use this to blacklist the previews Adobe Photoshop Lightroom creates, for example.
2. All files are compressed and encrypted. You can change password in the configuration. Inventories and manifest are currently plain xml, however.
3. There is a not well tested “hidden feature” called “partitions”. In case you don’t want to change your storage, you can tell KeepBackup to store new files to a different location. This is handy if you uploaded your storage to a cloud drive and don’t want is to change.

# Inventories

The current version of KeepBackup stores the inventory at origin and in the storage.

I thought it’s a good reminder to see at the origin when the last inventory was created. It might show, that the next backup is overdue. The previous inventory at origin is also used to recover the checksums of all unmoved and unchanged files.

You might not want a backup program to add files to your origin. You can delete the inventories at origin with no risk, since for restoring files the inventories at the storage is used.

However, currently the next backup will be a lot slower, since all checksums will be calculated again. In the current version, the checksums of unchanged files are not used from storage, but from previous inventories at origin. This might change in future versions.

# Let’s get started!

Remember that it’s currently just a command line tool.

First open the configuration and change the password ald salt to something only you know!

KeepBackup.exe backup “d:\somepath\” “g:\storage\”

This is the command to back up the origin “d:\somepath\” to the storage “g:\storage\”.

If the storage directory does not exist, it will be created. If it is already present a new backup (and a new inventory) will be created and added to the storage.

KeepBackup.exe createinventory “d:\somepath\”

Just creates an inventory at the origin, without backing up anything. This is more or less useful just for testing.

KeepBackup.exe restore “g:\storage\KeepBackup-2017-07-07 09-01-35.inventory” “d:\restore\”

Restore a complete backup of origin based in an inventory file.

KeepBackup.exe restore “g:\storage\KeepBackup-2017-07-07 09-01-35.inventory” “d:\restore\” “\subdir\”

Restore just the directory “subdir” from the complete backup of origin based in an inventory file.

KeepBackup.exe restorefile “g:\storage\KeepBackup-2017-07-07 09-01-35.inventory” “d:\restore\” “/somedir/1.jpg”

Restore a single file.

# More cool features

KeepBackup.exe validatestorage “g:\storage\”

This is just for diagnostics, but it gives you peace of mind.

Compare manifest with all inventories and the present files to see if everything matches up.

This reads all files and compares their checksums to the manifest, which might take some time. You can disable this by using a third parameter “fast”. Then the presence and sizes of the files will be checked only. Alternatively with the third parameter “decompress” each file will be decrypted, decompressed and the checksum of the content will be checked. This might take even longer.

KeepBackup.exe compare “g:\storage\KeepBackup-2017-07-07 09-01-35.inventory” “g:\storage\KeepBackup-2017-07-08 10-11-37.inventory”

Compares two inventories and tells you what changed in your origin in the time between them.

KeepBackup.exe largefiles “g:\storage\”

Look for the largest files in your storage. Per file you get on overview in which inventories this is present.

KeepBackup.exe analyzeduplication “g:\storage\KeepBackup-2017-07-07 09-01-35.inventory”

This tells you something about your origin. Have you multiple copies of the same files? How much space do you waste by file duplication? Are there directories, where every single file is present somewhere else?

Keep in mind that duplication doesn’t blow up your storage. But you might consider cleaning up your origin.

KeepBackup.exe garbageanalysis “g:\storage\”

Tells you for every inventory how much space would be freed up, if it would be removed from storage. Someday you might want so remove old inventories.. Or maybe you made a backup with a very large file, which you certainly won’t need in the future.

KeepBackup.exe garbagecollect “g:\storage\”

After using garbageanalysis or largefiles command, you might have decided to delete some inventories. You can just move, delete or rename the inventory file in storage. After that validatestorage will tell you, there are unused files in your manifest and in the file storage. Garbagecollect will move these files to a folder “garbage”. After that you can delete them. In the manifest these files will be moved to a <Garbage> element for future reference. You can delete this Element manually, if you want to.

# Thoughts on Performance

KeepBackup is very fast, at least in my opinion, when it comes to adding new files (e.g. photos). The new files will be added to storage, renaming files doesn’t cause any files in storage to be deleted and backed up again.

Initial Backups take some time. First of all, every file needs to be hashed. I’m using Sha256 which is very secure, but maybe there are faster algorithms?

Then each file must be compressed and encrypted, which takes time.

After that the result is hashed again. This could be avoided,, but the checksum of the compressed and encrypted file is stored in the manifest and is used for validatestorage.

I got it faster by using parallelism and handling small files with memory streams, larger files with temporary files.

It is probably possible so get is a lot faster.

# Future plan

* More tests
* Cleaning up code and commenting
* Performance (see above)
* Better configurability of blacklisted files and folders
* Removing a combination of inventories might free up a lot of space in storage. Garbage analysis might be extended to look for this. Also, it could be enhanced to give an overview which files would be removed and if there is new version of the file in later inventories (same path, other hash).

# Compatibility

There are already a lot of storages with backups in existence. Changes to the storage format are somewhat critical. Any new version of KeepBackup needs to be able to restore the current structure and format of the storage. In addition, an automated upgrade command to a potentially new storage format would be great. Since a lot of backups are uploaded to cloud drives the current format must remain restorable. The partition feature mentioned above might be a good solution for this. New partitions in the storage get a new version, old partitions stay as they are.