OpenOfficeBackup

A lightweight OpenOffice backup tool

Andreas Harnack
(ah8 at freenet dot de)

Version 0.1
March 2017

Contents

1 Motivation **2**

2 Introduction **3**

3 Design **4**
3.1 Backup Destinations 4
3.2 Backup Names 5
3.3 Default Backups 6
3.4 Backup Copy Names 6

4 Installation and Setup **6**

5 Configuration, a Tutorial **8**

6 Establishing a Backup Strategy **11**
6.1 The Backup Media 11
6.2 The Media Cycle 12

6.3 Housekeeping 14
6.4 Solving the Removable Device Naming Issue 14
6.4.1 Windows XP 14
6.4.2 Windows 7 and above 15
6.4.3 Linux 16
6.5 File Systems Issues on Flash Memory 17
6.5.1 Journalling File Systems 19
6.5.2 Log-Structured File System 19
6.5.3 UDF – the Universal Disk Format 20
6.5.4 Reformatting FAT32 21
6.5.5 Summary 21

7 Implementation **22**
7.1 Modules 22
7.1.1 `core` 22
7.1.2 `config` 22
7.1.3 `ui` 22
7.1.4 `util` 22
7.2 Dialogues 22
7.2.1 `DocumentBackup` 22
7.2.2 `BackupDestination` 23
7.2.3 `BackupName` 23

8 Wish List **24**

1 Motivation

You are writing up some theses? Or a book? Or some crucial report with a tough deadline? In short, you are writing documents where an unexpected data loss is likely to cause you real pain?

But you wouldn't let that happen, would you? You come prepared. You have backups, haven't you? And you have them configured in a way that restoring a suitable copy would be a matter of seconds rather than minutes, and you won't lose more than an hour or so of your work!

Just in case you're starting to feel slightly uncomfortable now because there might be one statement or another in the previous paragraph that doesn't exactly apply to you – while the first one still does – then carry on reading. The tool we're going to describe is simple. In fact, I'm almost tempted to say trivial. But it might make you feel much better about protecting your precious data from getting lost.

2 Introduction

When talking about backups, many people think of disk failures first. But that's not what's usually seen in reality. Modern disk drives are remarkably reliable and are more likely to be replaced because they became to small or too slow, rather then because – and often long before – they reach the end of there physical live. Of course, they still can fail and you're better prepared for it, but there are threats much more likely to damage your data than that. Software failures for example, or human errors. A program crash scrambling your data, a file accidentally copied in the wrong direction, or a deleted paragraph or chart that after a week or so turns out to be not quite as dispensable after all. In all these cases you'll be happy if you have a backup at hand that's not too old.

Backups, though, are always a bit like insurances: You are paying for something you hope you'll never need. That alone would be annoying enough, but on a PC or Notebook things are even worse. Usually these devices are turned off when you've done with them. Unlike for a server in a data centre, there are no idle hours at the dead of night when you can conveniently schedule a backup, there is no friendly backup team replacing the tapes, and there is no IT-manager bothering you ever other week with the red cells in the OpenOffice Calc backup statistics spreadsheet. Running Backups on a PC is a tedious, annoying, and time consuming work. People tend to avoid such work and, despite of their best intentions, are prone to run backups not quite as regular as they should do, and since only a regular backup is a good backup, they are likely to end up soon with no useful backup at all.

Getting out of this dilemma first and foremost requires backups to be cheap. That doesn't necessarily mean cheap in terms of money, even more import is that they are cheap in terms of time, in both, user time and CPU time. Cheap can also mean to distribute the costs over a longer period of time so that they are less notable. One way to achieve this is to be selective. Data usually changes gradually. Backing up only data that has changed as soon as it has changed is likely to reduce the amount of data to be backed up at any particular point of time. It also ensures that modified data is backed up promptly, and if it could be integrated into the data modifying applications itself, it could even be made to run automatically with very little attention required by the user.

The little project presented here implements a simple backup tool in OpenOffice Basic. It is integrated into OpenOffice and hooked into the OpenOffice file handling system. It runs one or more backups each time a document is saved to disk. Does that sound like an overkill to you? Well, it doesn't to me. I'm using this strategy since more than a decade in all my software development projects, and – believe it or not – this is the one backup that spared me by far the most headaches. I always wanted to

have a similar tool for OpenOffice, *et voilà*, here it is. Well, nearly, since in a well structure source tree you can be much more selective than in an OpenOffice document, at least with a reasonable effort, so the amount of storage required by OpenOffice might be a bit higher. But storage space is cheap nowadays, so please don't try to save at the wrong ends.

3 Design

OpenOffice provides hooks to allow the user to link macros to certain events in the OpenOffice event handling system. These macros then are called each time a particular event occurs. Some of these hooks are dedicated to file saving. The `DocumentSaved` hook, for example, is called each time a document has been successfully saved to disk under its current name. Its sibling, `DocumentSavedAs`, is called each time a document has been successfully saved under a new name. That is true also if a newly created document is saved for the first time.

That's precisely what we're after. The idea of this simple backup tool is to write a macro that each time a document has been saved creates one or more backup copies of the file to one ore more dedicated backup destinations.

A backup destination is nothing but a folder somewhere in the file system hierarchy. What remains to be decided is the way to configure backup destinations, the way to configure individual document backups, which includes the question how backups should be organized and if for a particular document a backup should be performed at all, and how the backup copies shall be named.

3.1 Backup Destinations

As already mentioned, from the backup tool's point of view a backup destination is nothing but a directory somewhere in the file system hierarchy. The tool keeps a list of possible backup destinations. For reasons of simplicity that list lives in a simple configuration file. The list can be maintained in the *OpenOffice Backup Setup* dialogue.

Backups are likely to be made to removable storage media, which might or might not be present at the time a backup shall be performed. Since the backup tool is supposed to run as quietly as possible, it is not considered to be an error if a configured backup destination currently does not exist, so the tool will not complain about it. Instead, the tool checks for available backup destinations and silently ignores those unavailable. The tool will not attempt to create a missing backup destination directory, since it might be supposed to live on a currently missing removable media. Even if there is no backup destination available at all, no error will be reported. It is the user's responsibility to ensure there is a backup destination available. It is

still on the wish list, though, to display at least a warning in the OpenOffice status bar in such a case.

3.2 Backup Names

To organize your backups each backup is assigned a backup name. Technically a backup name is a subdirectory in a backup destination folder where the backup copies are put. So it is not the name of the backup copy itself, but the name of the place, were a set of backup copies belonging together is kept. The backup tool will never place a backup copy directly into a backup destination directory.

Each document can be assigned an individual backup name. OpenOffice allows to define per document custom properties. Such a custom property is used to assign a backup name to a document. Since it is an ordinary custom property, it can manually be added, changed, and removed using the OpenOffice document *Properties* dialogue (*File* → *Properties...* → *Custom Properties*). However, it is not recommended to do that. The backup tool's user interface offers its own dialogue for that purpose. It's best to use that instead.

Since a backup name specifies a directory and directories can contain subdirectories, it is perfectly legal to specify a path as backup name. In the backup tool's user interface you specify the path using the path separator specific to your operating system, while in the document's custom property it is stored using the separator according to the URL syntax, i.e. the forward slash (/).[1]

Missing directories specified in the backup name are going to be created as long as the backup destination exists. That goes for all directories below the backup destination directory.

The same backup name can safely be used for as many documents as you want. Even ambiguous document names usually don't hurt, since the backup copy's name will contain a time stamp.[2] It could be tricky, though, to figure out later which backup belongs to which document, if different documents with the same name are stored to the same backup directory. So be careful if you have documents with the same name in different folders in your file system hierarchy. The document's path does not become part of the backup copy's name and hence will not help you there.

[1] Documents might be moved around from one system to another. Different systems might have different operating systems, so such information has to be stored in an OS-independent way.

[2] Unless in the very unlikely event, that you backup two different documents with the same name to the same directory within the same smallest distinguishable time period.

3.3 Default Backups

The backup tool offers the possibility to define a default backup name. This name is stored in the tool's configuration file an used if a document hasn't got a backup name property specified. If no default backup name is defined, then no default backup will be performed. If a default backup is defined, the backup for an individual document can still be disabled by assigning an empty string as a document's backup name. The *Document Backup Name* dialogue has an option for that. Clearing a document's backup name on the other hand will remove the backup tool's custom property and hence set the behaviour back to default. Clearing the default backup name will disable all default backups.

The default backup is defined along with the backup destinations in the *OpenOffice Backup Setup* dialogue.

3.4 Backup Copy Names

As name for the backup copies the document name is used, augmented by a time stamp. The time stamp contains in that order: 4 digits for the current year, 2 digits respectively for the current month, the day of the month, the hour, minute, and second, and finally 4 digits for the current system time down to milliseconds. The milliseconds were included as a simple mean to minimize they risk of ambiguities in case two backup copies with the otherwise same name are to be stored in a short period of time. Though the procedure is strictly speaking not save, a period of one millisecond should be sufficiently short to avoid problems in all practical situations.

4 Installation and Setup

To work properly the backup tool requires at least one menu or toolbar entry and at least one hook into the OpenOffice event handling system. It would be nice to have an OpenOffice extension packed that does all the setup automatically. While it is possible to have an extension setting up a menu entry, they are restricted to predefined locations. Setting up an event handler doesn't seem to be possible at all. Hence the user has to go through some kind of setup procedure anyway, so it not only seems to be more consistent to abstain from an automatism that would be capable of doing only half the work, it also avoids the restrictions it would impose upon the part it can do and the considerable effort that apparently is required to get there. Also, setting up the tool manually isn't that complicate and has the added advantage for the user to gain some insides into the way the tool works. After all, it's better not to trust software from unknown sources too easily.

To install the software, download the package most suitable to your operating system. Unpack it to a convenient temporary location. Start your OpenOffice an go to the *Basic Macros* dialogue (*Tools* → *Macros* → *Organize Macros* → *OpenOffice.org Basic...*). Open the *Basic Macro Organizer* (*Organizer...*) and choose the *Libraries* tab. Make sure the location *MyMacros & Dialogs* is selected and no `OpenOfficeBackup` library exists yet. In the *Import* dialogue (*Import...*) navigate to the `OpenOfficeBackup` directory you've just unpacked. Select the file `script.xlb` and open it (double click, or select and *Open*). Confirm the import (*OK*). In the library list you should now see the `OpenOfficeBackup` library. The library contains four modules and three dialogues. You can check this in the Organizer's *Modules* and *Dialogs* tab respectively (*My Macros* → *OpenOfficeBackup*). A detailed description of the library components will be given below. For now, close the *Organizer* and the *Basic Macros* dialogue.

To set up the tool open the *Customize* dialogue (*Tools* → *Customize...*). In the *Menus* tab select the menu where you want to place the *Backup* menu entry. I use the *Tools* menu, but you are free to choose any other place. Select the position in the menu where you want to place the entry (click on the entry that shall appear right above the *Backup* entry you are about to add). Open the *Add Commands* dialogue (*Add...*). In the *Category* list navigate to the `ui` module (*OpenOffice.org Macros* → *My Macros* → *OpenOfficeBackup* → *ui*). From the command list select the `Backup` command and add it (double click, or select and *Add*). Close the *Add Commands* dialogue (*Close*). You can still change the position of the newly created entry in the menu using the up and down arrows next to the menu content list. You can also rename it (*Modify* → *Rename...*) or put it in its own group (*Modify* → *Begin a Group*). I use to put it in its own group right above the *Macros* entry keeping the name, but again, you are free to do it any way you want.

You can add a toolbar entry too and even a keyboard short cut from the *Toolbar* and *Keyboard* tab respectively. The procedures are essentially the same. However, you should need these entries only occasionally, so its probably not worth the effort.

The menu content in OpenOffice is context dependent, so the changes you did will appear only when working with the same type of document that was open during the configuration (if any). That implies that you'll have to repeat the process for all document types you're working with (Writer, Calc, etc.). It's probably a good idea to add an menu entry to the desktop as well, i.e. to repeat the process with no documents open (if not already done so).

Finally, to set up the hooks, go to the *Customize* dialogue again (*Tools* → *Customize...*) and choose the *Events* tab. Make sure the *Save In* option is set to *OpenOffice.org*. Select the *Document has been saved* event and open the *Macros Selector* dialogue (*Macros...*). In the Library list navigate

to the `core` module (*My Macros* → *OpenOfficeBackup* → *core*). From the list of available macro names assign the `OnDocumentSaved` macro (double click, or select and *OK*). This hook makes sure a document is backed up when saved under its current name. Repeat the process and assign the `OnDocumentSavedAs` macro to the *Document has been saved as* event as well to ensure a backup is performed if a new name is assigned to a document. Otherwise you might miss a backup if a newly created document is saved for the first time or a changed document is saved under a different name, even though this hook might occasionally run a backup of a renamed but otherwise unchanged document. After all, it is far easier to delete an unnecessary backup copy than to recreate a lost document.

5 Configuration, a Tutorial

Once the backup tool has been installed and set up, it needs to be configured. Nearly all the user interaction with the tool is dedicated to configuration. The `Backup` routine in the `ui` module is the central entry point to the user interface. In *Installation and Setup* is described how to link it to a menu entry. Once you have done so, the time has come to try it out. So just click it.

Calling the user interface will usually launch the *Document Backup* dialogue, summarizing the backups configured for the current document. If, however, the user interface is called from outside a document context, it goes to the *OpenOffice Backup Setup* dialogue instead, since there is no document that could be backed up.

The other occasion, when a call to the user interface leads straight into the *OpenOffice Backup Setup* dialogue, regardless whether there is a document context available or not, is when there are no backup destinations defined yet. That's almost certainly the case when you call the interface for the first time. There is simply no point in worrying about the backup of a document if there is no place defined where to put it. As described above, that's precisely what a backup destination is: a location where a backup copy could be placed and it is just a dedicated folder somewhere in the file system hierarchy. The *OpenOffice Backup Setup* dialogue is the place to turn a directories into a backup destinations.

As a quick start to familiarise yourself with the tool I suggest you create a backup directory at some convenient place on your hard disk. Make it temporary if you want, but experience shows that it's not such a bad idea to keep it permanently. We'll see the reasons in due course. If you want to make it permanent, don't put in your home directory, though. The OpenOffice backup tool can only supplement your other backups, but never replace them. You do have a backup configured for your home directory, haven't you? Without further tuning that backup would also save the backup copies

in your backup directory and backing up the backup copies is typically not the most efficient way of running things.

I have a dedicated backup directory in the root file system respectively the root of the system disk on most of my systems, i.e. something like `/backup` or `C:\backup`. On systems sensitive to permissions and ownership place a user specific `$user` directory with the correct ownership beneath it. On Windows systems used by only one person you can probably do without it. Finally, to avoid conflicts with other applications, create a sub-directory for OpenOffice. That directory shouldn't be used for anything else. So you should end up with something like `/backup/$user/OpenOffice` or `C:\backup\OpenOffice`. Unfortunately, the OpenOffice folder picker used in the dialogue does not allow to create folders,[3] so you'll have to use your operating system tools to do that.

Now go back to the *OpenOffice Backup Setup* dialogue, start the folder picker to add a new backup destination (*New...*) and pick the folder you've just created. This will be your first backup destination.

You should also specify a name for default backups, at least for this tutorial. You can remove it later if you wish (*Clear*). Something like *Default* should do it. Remember that a backup name ends up to be a subdirectory in the backup destinations. If there should already exist any subdirectories in any defined backup destination, they will be displayed in the drop down list to be picked as a backup name. That's one reason you shouldn't have any other stuff in your backup destination directories.

Close the dialogue by confirming your changes (*OK*).

If you've started the user interface with a document open you should now be in the *Document Backup* dialogue. If you haven't, open or create a document and call the interface again. (You already have created a menu entry for that document type, right?) With at least one backup destination defined, that will start the *Document Backup* dialogue.

The *Document Backup* dialogue summarizes the backups the tool would have performed at the moment the dialogue was last updated (if any).

On top of the dialogue you see the current backup name (if any). For a newly created document that should be the default name you have specify in the *OpenOffice Backup Setup* dialogue (if any). If no backup name is displayed, no backup will be performed.

Below the backup name you'll have the list of available backups in form of a tree. The top level entries show the backup destinations currently available to the system. If you've followed this tutorial so far, you should now see (only) the backup destination you've just defined. Below the backup destinations you'll see the subdirectories as specified by the backup name. A backup name can be a path name. How many levels there are obviously depends on the path you've specify as backup name, but there will be at

[3] at the time of writing with OpenOffice version 3.4.1

least one. If any node in the branch of subdirectories is collapsed, then that directory doesn't exist yet and is going to be created, an expanded node indicates an existing directory.[4] The leave nodes show the name of the backup copy to be created.

If your document has already got a name, you can now actually run the backups as displayed in the list (*Run*). This, however, will be necessary only in some special circumstances, since normally backups will be run automatically. If your document is newly created and hence hasn't got a name yet, wild cards will be displayed instead of the backup copy's file name and running the backups will be disabled.

If no backup name is displayed, be it, because neither a document backup name nor a default backup name has been specified, or backups have explicitly been disabled for that document, then only the available backup destinations are shown in the list. In that case, too, running backups manually will be disabled.

You can change the backup name by using the *Change* button next to the backup name field. This will open the *Backup Name* dialogue. In the text field you can enter any valid path name, using your operating system's path separator, or pick one from drop down list containing any existing backup directory on any available backup destination. You can also clear the document's backup name, meaning that the default backup name will be used to backup this document, or you can explicitly disable backups for this document. Confirming or discarding your changes will take you back to the *Document Backup* dialogue.

If anything on your system affecting the backups has changed, you can update the summary displayed in the *Document Backup* dialogue (*Update*). That is useful, for example, if removable media or storage devices have been removed from or connected to the system. Otherwise the content of the dialogue is updated each time the dialogue is entered or a manual backup has been run.

That's it! With at least one backup destination defined and a backup name specified, be it as a default backup name or a document specific one, your tool has everything it needs to run backups. So now is the time to test it. Close the dialogue and go back to your document. Save it. You should see a backup copy of your document put into every defined backup destination currently available under the path specified as backup name. Use your operating system tools to check it.

[4] A backup destination will always be expanded since only existing backup destinations are going to be displayed.

6 Establishing a Backup Strategy

Congratulations! If you have completed the tutorial above, OpenOffice should now silently run a backup for you every time you hit the save button. However, while having a backup of your documents on your system disk can already spare you a lot of headaches, it does not provide real redundancy, so it's time now to think about a proper backup strategy.

How that strategy should look like depends on your individual needs. Most certainly, however, you should back up your data on some removable media and store some copies of them away from your system.

6.1 The Backup Media

Nowadays the backup media of choice is probably the USB memory stick or the SD memory card. They allow to write data in small chunks and are robust, reliable and constantly getting cheaper. As of the time of writing (October 2015), 8 GB USB 2.0 memory sticks are available for around 5 €, that are slightly more than 62 ct/GB, a 16 GB USB 2.0 stick you could get for around 8 € or 50 ct/GB while 32 GB of USB 2.0 memory are currently starting from about 10 € or 31 ct/GB. The prices obviously vary, but are in tendency more likely to fall, rather than to rise.

Personally I prefer to use to use SD Cards, in particular for notebooks. They tend to be slightly more expensive but have the advantage to vanish in the card slot. That is important because a destination media needs to be available all the time while you're working with OpenOffice. You can leave the card in the slot even if you carry the notebook around. A USB stick is easily knocked out of its socket on such occasions, which not only can lead to data loss, but can also physically damage the socket. If you want to use sticks, try to find some with a short casing. They present a much smaller lever to accidentally acting forces. Alternatively, using a short flexible cable might help.

The USB sticks and SD cards available vary in the maximal reachable data transfer rates. However, regarding the size of an average OpenOffice document, for making backups the differences in the available speed ranges are usually of no particular relevance.

If your only feasible option is to backup to an internal disk, then do that. It is still better than no backup at all. If you have more than one disk, one can serve as the backup for the other. Try to created a dedicated partition for your backups. Make it the size of a CD ROM or slightly smaller, and burn your data away to CD every no and then. How often depends on your individual needs. If your backup partition gets full, only delete as many of the oldest copies as free space is needed to lasts to the next burning cycle. That way you use the otherwise unused space on the CD ROM for additional redundancy. The procedure is less convenient then flash memory,

but it works. I used it before memory sticks and cards became available in an appropriate size at affordable prices.[5]

You can use external hard disks for OpenOffice backups, but I feel they are a bit clumsy to handle for that purpose. They are less robust and more expensive then silicon chips and make a notebook less mobile to be carried around. If you plan to use them as permanent backup media, remember that one is not enough. Several small device offer more redundancy than a single big one, and the size of a modern hard disk is huge compared to what you'll need for this kind of backups. So with sticks or cards you'll probably get more value for your money. If, however, you already have a set of disks available as backup media for your other backups and a lack of mobility is not an issue to you, then there is, of course, no reason not to use them for your OpenOffice backups as well.

Be cautious with network devices. While network devices usually make a great backup destination, since they physically separate the backup from the data source, they could cause trouble if there are unexpected network problems. Remember that the backup takes place silently in the background each time a file is saved. It could be rather disturbing if a file save operation suddenly hangs for no apparent reason. That's what you're likely to see if the backup script gets stalled by the network. The backup tool doesn't handle such situations gracefully yet. So, if you use a network device, make sure the network is reliable. If it is, even a memory stick in your internet router is good option for a backup destination.

By the way, don't be tempted to consider a disk mirror as a backup solution. **A mirror never and under no circumstances replaces a backup!** A mirror can protect you from disk failures, but disk failures are not the only, not even the most likely reason for data loss. If you incidentally delete a file or a software crash leaves your data scrambled, then a mirror won't help you at all. A mirror buys you availability, not redundancy! And there is another point easily forgotten: A mirror is useless without a proper monitoring! The mirror software's job is it to hide a disk failure from the user and usually it does a very good job. But on your PC you're not only user, you are also admin, and if your only chance as an admin to notice a disk failure is to wait for the second disk to fail, than there is hardly any point in having a mirror in the first place. However, don't underestimate the effort and pitfalls of setting up a proper and reliable monitoring. That's not something done in a jiffy.

6.2 The Media Cycle

The purpose of backups is to provide redundancy. Redundancy should remove any single point of failure, or at least as many of these points as an

[5] Of course, you can also combine flash memory devices with CD ROM storage if that's convenient for you.

adequate effort will allow. This includes the backup media and implies the backup media to be changed on a regular basis. It also implies, that backup copies are physically separated from the data source and stored, if possible, in different locations. How far you want to take this depends again on the options you have and the value you attribute to your data. Often, however, little things can achieve a great deal.

Assuming, for example, you are a postgraduate student spending your working days in an university office. There you have your own desk with a lockable drawer. You use a notebook that you take home with you, where you continue to write your thesis. Then you could easily establish a backup cycle with, let's say, three SD cards: one you keep in the office, one at home, and one in the card slot of your notebook. During your working day you make your backups onto the card in the card slot. Before you leave the office, you swap the cards, leaving the most recent backup copies at the university while taking the working copy home on the hard disk of your notebook. At home you continue your work, making backups on the card formerly kept in the office. Before you leave home, you swap cards again, leaving the most recent backups there. In your card slot is now the least recently used card as the one next in line for the upcoming day's work, and so on. This way you'll always have two current copies of your work in two different places, so you are save even if your notebook gets lost or damaged on the way, your home is flooded, or your office burns down.[6]

If you don't have any means to lock away things safely, but access to the campus network and a password protected, mountable personal home directory, that could be an option as well. In that case you would keep changing SD cards only at home.

Of course, you also could implement a traditional backup scheme with daily, weekly and monthly backup media, but usually that doesn't make that much sense. While it certainly makes sense to change your media on a regular basis, there is no need to insist on having exactly one change a day and one media per weekday. Change as often and use as many media in the cycle as you feel comfortable with. That might even change with the progress of your work.

There is also hardly any need for weekly or monthly backup media. These are dictated by the fact, that the tapes typically used in traditional backup schemes got overwritten each time they are used. Weekly or monthly tapes are therefore necessary to guarantee a certain retention time. This is not an issue with memory cards or sticks. They can hold many backup copies and are usually large enough to keep them for weeks, months or even years.

If you explicitly need to keep data for longer than that, the simplest way would be to get a new set of backup media. However, in that case

[6]But not necessarily, if two of these events happen at the same time. But then you have probably other things to worry about anyway.

you probably don't have a backup issue, but an archive issue. This backup tool was not designed with archiving in mind, so it's probably not the best choice for the job. If you browse the net for archiving you might find better solutions and advice, how to address your particular problem.

6.3 Housekeeping

The backup tool does not provide any housekeeping, i.e. it will never delete or overwrite any backup copies.[7] It will be your responsibility to ensure there is enough space left on the backup media to back up you data. Use the tools your operating system provides to monitor the available storage space and to delete old backup copies if necessary. Your intervention, however, will only rarely be required. Modern storage media is large compared to the size of an average OpenOffice document, so there is space for a lot of backups.[8] If your documents are going to be too large it is probably a good idea anyway to spilt them up into several small documents, not only reducing the backup space required but also improving you computer's response time.

6.4 Solving the Removable Device Naming Issue

Each storage device in a computer needs some kind of identifier to talk to. On Windows systems this is the drive letter, on Linux systems the device file. For removable devices this identifier is usually assigned when the device is detected and that in increasing order. Hence the identifier for a particular device might vary, depending on the order in which devices are connected. That's bad, since it would be rather annoying to have to configure a device at any possible location it might turn up, let alone the problem, that devices you don't want to use for backups might turn up at places configured as backup destinations. A backup tool shouldn't have to worry about this, it is the operating system's duty to solve this kind of problems, so let's see what options we have.

6.4.1 Windows XP

To permanently assign a drive letter to a device, connect the device to the system. On your desktop, right click on *My Computer*, from the popup menu choose *Manage*. The Computer Management console should start. In the left pane under *Storage*, click *Disk Management*. In the top right pane, select the drive you want to configure and right click it. From the popup menu choose *Change Drive Letter and Paths*.

[7] If it does, it's a bug.

[8] While writing up this document I reached an average backup frequency of about 1 backup every 45 minutes with the document size steadily increasing to the range of 100–150 kB. With the 3x 8 GB USB memory cards I'm using and assuming an 8 hour working day 365 days a year, that would last for about $3 \times \frac{8000MB}{0.15MB} \times \frac{45min}{60\frac{min}{h} \times 8\frac{h}{d} \times 365\frac{d}{y}} \approx 40$ years!

In the dialogue that starts now you can change the assigned drive letter. This letter will be reassigned each time the device is reconnected. However, that could fail if the letter will be already in use, for example for a currently connected network drive. The better option is therefore to choose a mount path instead. Assigning any empty directory on the system disk ensures, that the device's directory tree is accessible through this path regardless of the driver letter assigned. That works even if a permanently assigned drive letter is hidden by a network drive and is the reason I recommend to create a `C:\backup` directory even if you are not planning to store any backups there. It's still a good place to create the subdirectories you need as mount points for all devices you intend to use for backups.

I noticed, that for each USB stick I tried an own exclusive directory was required. It was not possible to mount several USB sticks to the same mount point. After configuring the first SD card, however, any card inserted to the box was mounted to the same path (and assigned to the same driver letter).

I don't know how Windows handles this internally. There are hints on the net, that the identification of USB devices are based on the USB device's serial number and might fail, if a device does not provide one complying to the standard. The identification of the SD cards on the other hand appears to be based on the hardware path of the slot the card is plugged into. So, while in general the scheme works quite well, better be prepared for surprises.

There is also a tool on the net called *USBDLM*, the *USB Drive Letter Manager*. That might do a better job but I haven't tried it out. For the last Windows box I still have, the on-board facilities are good enough.

6.4.2 Windows 7 and above

I can't really tell you. All my productive systems have long been migrated to Linux. I've only one Windows box left running XP – basically for nostalgic reasons – and I do not intend to spend any money for a new licence of an operating system I don't need any more. So, if you're still using Windows, I'm afraid you have to find out on your own.

However, I found the following description on the Microsoft web site for Windows 7:

> "Open Computer Management by clicking the **Start** button, clicking **Control Panel**, clicking **System and Security**, clicking **Administrative Tools**, and then double-clicking **Computer Management**. If you're prompted for an administrator password or confirmation, type the password or provide confirmation."

From that point on it sounds pretty much like the procedure above, so it seems anything said for Windows XP applies to Windows 7 accordingly.

6.4.3 Linux

The key on Linux boxes to solve the removable devices naming issue is `udev`, the Linux device manager. It listens to messages the kernel sends out whenever a device is detected or removed. Based on a set of rules and on the information the kernel provides about the device, it then creates the appropriate device file and mounts the device. The system is extremely powerful and flexible – it's even possible to run user defined programs or query databases to name devices. But there is nothing like a free lunch and like any powerful tool it requires some effort to get the most out of it.

Writing `udev` rules is far beyond the scope of this document. You'll find plenty of information on the net if you have to. Luckily that shouldn't be necessary for most users. A user friendly Linux distribution should provide some reasonable defaults.

Amongst the attributes a device name could be based on are:

- the file system label
- the device class and device serial number
- the bus topology
- kernel name

It is not uncommon for a predefined rule set to consider these attributes in that order. Hence, giving the file system on the volume a unique label usually has a good chance to uniquely identify it. On my Debian systems, a labelled volume will be mounted under `/media/<label>`. If no label exists, it falls back to serial numbers or product ids, mounted respectively at `/media/<serial number>` and `/media/<product id>`. This, however, might vary from distribution to distribution.

How to set up a volume label depends on the file system you are using. Most memory sticks and cards are shipped pre-formatted with a `FAT32` file system, so you might try:

```
# dosfslabel <device file> <label>
```
[9]

You'll need to have the `dosfstools` installed for this to work. However, please be aware that the `FAT32` file system implements volume labels in a rather peculiar way. `FAT32` stores volume labels twice, once in the boot sector and once in a directory entry in the root directory with the volume attribute label set. Unfortunately, these two places are not handled consistently. Windows seems to use mainly the directory entry while non-Windows tools tend to use the one in the boot sector. To make things even worse,

[9] All command lines starting with a hash (`#`) are executed under root privileges. You might prefix them with `sudo`, if you don't want to login as root.

the entry in the boot sector consist of two copies, a primary and a backup copy. Some tools deal with the primary copy (and the directory entry) but leave the backup untouched, causing problems with other tools relying on the consistency of both copies in the boot sector, including the Linux `fsck`. Make sure to use the Linux `dosfstools` version 3.0.16 or above to avoid trouble.

Many Linux desktops create an icon for an inserted removable media, but don't actually mount it. That won't be what you want. Though you can mount such a device with a single mouse click, that's still annoying since you want your backup media present all the time. Check your desktop settings to change that. Usually it's just a missing tick somewhere.

If your desktop doesn't create an icon please check the net how to change that. Your last resort is always the `/etc/fstab`, but that wouldn't be my favourite choice. It could mount your backup media when your system is booting, but you're likely missing a convenient way to unmount and re-mount it, things you'd have to do before and after a media change.

6.5 File Systems Issues on Flash Memory

Flash memory devices up to 32 GB of size, both, USB Memory sticks and SD Cards, are usually shipped pre-formatted with a FAT32 file system. The FAT32 is certainly not the last word wisdom ever has to say on the subject of file systems, yet in most cases it is sufficient for our purpose. It is well support on nearly any device capable of mounting flash memory and there are even technical reasons that make it not such a bad choice after all for these kind of storage technology. So the best option is to keep it if it works for you.

However, above a memory size of 32 GB things are getting trickier. These devices are usually pre-formatted with exFAT, a newer and proprietary file system from Microsoft. Although it is said to be optimized for the use with flash memory, there is a severe portability issue.

Microsoft has never fully released the specification of the exFAT file system. Vendors are required to pay a licence fee which makes the decision harder for them to support it. Parts of it are protected by US software patents which makes it legally difficult to implement it's functionality in open source projects. Due to the terms and conditions of current licence agreements it is virtually impossible that it will ever make it into the Linux kernel.

If you're only working with a recent version of Windows you can probably live with it. On Windows XP you need additional drivers to access exFAT, which is annoying at best, and in the Linux world you're lost anyway. So what are the alternatives?

Flash devices usually emulate the behaviour of a traditional hard drive, so in principle it's possible to format them like any other block device. How-

ever, while technically nearly any file system should work, just picking you favourite one is usually not such a good move. Most file system are designed with traditional hard drives in mind, and that means that they expect the device to work in a way fundamentally different then flash memory does.

A traditional hard drive can be described as a sequence of sectors. A sector is the smallest unit of data than can be transferred. It has a size of typically 512 byte, though you might find disks with sector sizes of 1 KB, 2 KB, 4 KB or 8 KB out there. Each sector can be read, written and overwritten randomly. There is no need to erase an already used sector before rewriting it, and there is no relevant limit to the number of write cycles to a sector. The limiting factor is the seek time, i.e. the time needed to position the head. Consequently, traditional file system try to improve performance by minimizing head seeks.

The smallest transfer unit of a flash memory device, on the other hand, is a *page*. Early flash devices had a page size of 2 KB, which has steadily increased over the years and now reaches up to 32 KB [3]. Pages can be read randomly, but not written at will. A page needs to be erased, before they can be re-written. Erasing can't be done for single pages individually, but only in *erase blocks* of typically 64 pages, i.e. in chunks of 128 KB up to 2 MB. Within an erase block pages need to be written sequentially. Devices often group erase blocks to even larger units, called a *segment*. The most common size for these segments is 4 MB for drives in the multi-gigabyte class [1]. Their will be no data erased in any unit smaller than that.

The number of erase cycles is limited. After a certain number of erase cycles a block becomes permanently unusable. This effect is called *wear out*. Figures range from some 1000 to some 100.000 possible erase cycles per block, depending on the technology used [4]. A way to alleviate the wear out problem is to physically rearrange data, so that repeated writes to the same logical location will not result in repeated erase cycles of the same physical erase block. This job is called *wear levelling* and done by the *Flash Transition Layer* (*FTL*), a software running on the embedded controller of a each modern flash device.

If the host's file system tries to update a single chunk of 512 byte, which would be the natural sector size of a traditional hard disk, the flash device controller internally would have to read a page of up to 32 KB, update it and rewrite it to a new location on a previously erased segment. The page on the old segment is marked as stale. To reclaim that space, all the remaining non-stale pages have to be written to the new segment as well before the old segment can be reused. This is called *garbage collection*. In the worst case it might have to move up to nearly 4 MB. Both, the comparatively large page size and garbage collection lead to an effect called *write amplification*, describing the phenomenon, that internally much more data needs to be written then the intended amount. That can dramatically decrease the performance and the life time of a flash device.

How well the firmware can cope with the problems depends on the effort the manufacturer has put into it. Usually it makes a quite good job, but it is evident that flash devices are extremely sensitive to specific, suboptimal access patterns, and even the best firmware can do little about it. Unfortunately many commonly used file systems tend to produce access patterns we'd like to avoid.

6.5.1 Journalling File Systems

All journalling file systems try to retain data consistency in the event of dramatic failures, like a system crash or a power loss, by keeping a written log of intended changes, which is helpful to recover from such events quickly and to avoid data corruption. This obviously increases the amount of data written to the device, which is bad for flash memory. If a significant portion of this data are in-place changes, which is not unlikely, then things are getting even worse. Wear levelling eases the effect, but the fundamental problem remains. It's best to void journalling file systems on flash memory whenever possible. `NTFS`, `ext3` and `ext4` are all journalling file systems. Whenever I need a Linux specific file system, I stick to `ext2` and mount it with the *noatime* option. Unfortunately the maximal block size of `ext2` – *block* is the Linux word for *sector* – is 4 KB. Apparently there is a better way using `ext4` with no journalling and some raid parameters set to make `ext4` write in in larger chunks then blocks and align write operations to erase segment boundaries [2], but I've never tried that myself and I can't tell you how to do it.

6.5.2 Log-Structured File System

A log-structured file system is a file system in which data and meta data are written sequentially to a circular buffer, called a log [5]. The idea was originally based on the assumption, that read requests would be more and more satisfied by the ever-increasing cache memory available on modern computers, so that performance could be improved by focusing on improving write access, in particular by avoiding seeks during write operations.

This assumption did not hold quite as expected, but it turned out that the design was perfect for flash memory. Treating the storage as a circular log and writing only at its head provides a natural way of wear levelling. The most recently written version of a particular block is the current one. Garbage collection and erasing is done only at the tail of the log. Random read access does not cause a performance penalty and error recovery is very similar to recovering from a journal, so log-structured file systems have in a natural way the same desirable error recovering properties as journalling file system but without the disadvantageous write pattern.

JFFS/JFFS2, UBIFS, LogFS, YAFFS, and F2FS are examples of log-structured file systems. Unfortunately they are not very widely supported, many of them are only available for Linux. If you have some special needs and portability is not an issue for you, then the might be worth a try, otherwise, I'm afraid they are not likely to be a good choice, despite of there technical advantages.

6.5.3 UDF – the Universal Disk Format

Universal Disk Format (*UDF*) is an open, vendor-neutral file system for a broad range of media. In practice, it has been most widely used for DVDs. It is supported on all recent operating systems, does impose little restrictions to the file size, the file name length, and the file name character set and supports permissions and ownership, but is also able to fake ownership, avoiding problems with different user IDs on different systems while still distinguishing between executables and data files. It was designed for early rewritable and write-once optical discs, which have the same page writing and wear levelling requirements as flash media, so it's very well suited.

They main drawback is, that right now there are no tools available to repair a damaged file system. Some people consider this to be a serious flaw. Others say, that due to the journalling character of UDF such a tool isn't even necessary. I can't really comment on that, but since UDF is currently the only portable alternative for devices larger than 32 GB[10] I decided to format a few of my sticks in UDF. So far, that has worked perfectly, but it's too early for any long term experience.

Here is how it's done on a Linux system. You'll have the `udftools` to be installed on your system for this to work:[11]

```
# dd if=/dev/zero of=/dev/sd<x> bs=1M count=1
# mkudffs -b 512 --media-type=hd \
        --lvid=<label> --vid=<label> --fsid=<label> /dev/sd<x>
```

The first line wipes out the remains of any previous file system. UDF doesn't use the first few sectors on a device and anything that's left there might confuse your operating system's file system detection.

In this case, also the partition table is erased. (Note the use of `sd<x>` instead of `sd<x>1` as a device name.) There is usually no need to have partitions on a USB stick. However, don't erase the partition table if there is a chance that you'll ever want to use it again. The boundaries of the partition table and the beginning of the subsequent file system are particular sensitive to proper alignment. This can't be achieved without detailed knowledge of the underlying flash memory's geometry. However, if you know how to find that out you probably don't need to be told how to format a UDF file system

[10] with the only limitation of being only readable on Windows XP systems

[11] Again, all command lines starting with a hash (`#`) are executed under root privileges.

in the first place, so please don't temper with it unless you know exactly what you're doing. Formatting a partition works exactly the same way (just use `sd<x>1` instead of `sd<x>`).

The block size has to be set to 512, despite the fact, that a larger block size would be better for flash devices and that 512 is not even a valid block size according to the man page. It works perfectly well and Windows insists on having this block size. It won't recognize the file system if the block size is set to anything else.

The example also shows how to set a file system label. You'll want to have that for a well defined mount point.

6.5.4 Reformatting FAT32

There is actually an alternative to UDF on large devices that shall not be kept quiet. The FAT32 file system is not really limited to 32 GB, it's just not used for anything larger, apparently for political reasons. You can just reformat any device with:

```
# mkdosfs -n <label> -F 32 <device file>
```

Apparently that works even for devices larger then 32 GB, but I never tried that. However, that's the naive way anyway. To do it properly you should figure out the geometry of the the flash memory. In [1], [2], and particularly in [3] you might find some hints how to do that.

6.5.5 Summary

To keep a long story short:

- When using flash devices as a backup medium, stick to the pre-formatted file system whenever you can.
- On Linux, add a file system label to get the device mounted in a well defined place.
- When using a devices larger then 32 GB and you need portability, use UDF. It's probably the best option you have right now. If you use a different file system, turn of journalling and access timestamps.
- You might try to format large devices with the FAT32 format, but you'll have to figure out the correct file system layout to get optimal performance and life time.
- Avoid re-partitioning the device.

If you keep that in mind, flash memory makes an excellent backup media.

7 Implementation

The OpenOffice backup tool is implemented in OpenOffice Basic. While Basic is certainly not my favourite language, it is the probably best documented and simplest to use language integration OpenOffice currently offers. Since the problem is still rather simple this two advantages out-way the limitations of Basic and make it still a reasonable choice.

The current implementation contains four modules and three dialogues.

7.1 Modules

7.1.1 `core`

This module implements the core functionality, i.e. the actual backup macro.

7.1.2 `config`

The `config` module contains anything that has something to do with configuration. Some hard coded values live here as well as the functions to store and retrieve configuration data from the configuration file or the user document.

7.1.3 `ui`

This module contains the user interface. Anything related to the user interaction lives here. The user interaction is almost only involved with setup and configuration tasks. It provides the `Backup` procedure, which is the entry point to all user interaction.

7.1.4 `util`

Some functionality needed, but not directly related to the backup task, could easily come up in an entirely different project and hence would be a good candidate for a general purpose library. Such stuff lives in the `util` module. Examples are functions to creating, populating, resizing or sorting lists.

7.2 Dialogues

7.2.1 `DocumentBackup`

The *Document Backup* dialogue summarizes the backups configured for the current document. It is usually run whenever the user interface is called from within a document context. It displays the *Backup Name* and a list of all currently *Available Backups* including the full path to each available backup destination, each subdirectory beneath each backup destination as specified by the backup name, and the name of the backup copy, which

consist of the document name and a time stamp. If no document name is specified yet, wild cards will be displayed. If a subdirectory beneath a backup destination already exists, the node will be expanded, otherwise it will be collapsed. Subdirectories beneath a backup destination will be created if needed. Backup destination directories itself will never be created. Backup destinations, that currently don't exist are silently ignored. There is a *Change* button next to the *Backup Name* and the list of *Available Backups* to change the respective configuration. There is an *Update* button to update the summarized information in case something has changed externally, for example if a removable device has become available. When returning from a setup dialogue, the update should happen automatically. If both, the document name and at least on backup destination are available, the *Run* button is enabled an the backup can actually be run. This is mainly there for test purposes. The *Close* button closes the dialogue. Since there is nothing that can directly be changed in this dialogue, there is no need for a cancel button.

7.2.2 `BackupDestination`

The *OpenOffice Backup Setup* dialogue is there to setup a system wide *Default Backup* and the *Backup Destinations*. You enter a valid path name as a *Default Backup* name or pick one from the drop down list. The list contains any existing backup directory on any currently available backup destination. You can *Clear* the *Default Backup* name to disable default backups. You can pick any existing directory as *New* backup destination or *Delete* an existing one. *OK* accepts the changes, *Cancel* rejects them. The *OpenOffice Backup Setup* dialogue can be called from the *Document Backup* dialogue. Alternatively it is run from the `Backup` procedure when called from outside a document context or no backup destinations have been specified yet.

7.2.3 `BackupName`

The *Document Backup Name* dialogue displays and allows you to specify a document specific *Backup Name*. Similar to the default backup name you can specify any valid path name or pick on from the drop down list to assign an already existing backup name. Additionally you can *Disable* backups for the particular document. To *Clear* the backup name will cause the default backup to be run for this document. Again, *OK* accepts the changes, *Cancel* rejects them.

8 Wish List

Here are a few features I would like to have, but couldn't spare the time yet to implement them (in no particular order). Some of them might even be impossible to implement with the current versions of OpenOffice:

- Display a status information in the status bar, especially a warning about failed backups.
- Provide an OpenOffice extension installing the software and setting up menus and hooks automatically.
- Provide network access and encryption to support cloud storage space.
- Provide internationalisation.

If you would like to help out, you're welcome!

References

[1] Bergmann, Arnd: *Optimizing Linux with cheap flash drives* (2011). https://lwn.net/Articles/428584/

[2] blogofterje.wordpress.com: *Optimizing fs on sd-card for Linux/Fedora on Dreamplug* (2012). https://blogofterje.wordpress.com/2012/01/14/optimizing-fs-on-sd-card/

[3] Bradley, Mitch: *How to Damage a FLASH Storage Device* (2013). http://wiki.laptop.org/go/How_to_Damage_a_FLASH_Storage_Device

[4] Wikipedia: *Flash memory* (2015). https://en.wikipedia.org/wiki/Flash_memory

[5] Wikipedia: *Log-structured file system* (2015). https://en.wikipedia.org/wiki/Log-structured_file_system