BACKING UP LINUX: GENERAL WARNINGS

This chapter contains some general warnings and guidelines you should keep in mind.

5.1. Incremental backup and mtimes

Some backup utilities support incremental backups, so that they only backup the data that has changed since the last backup. Rdiff-backup is a very good example, as it supports nothing else. My advice is to be careful with incremental backups and investigate how they detect change. The best way is if they support hash checking, but this can be slow. Second best, and very reliable and fast, is ctime checking. The file system has to support ctimes though, but most do. Only the ones which you shouldn't be using, don't (like FAT32).

Some utilities only use the mtime, or the mtime+size combination, to detect changes in files. This is somewhat unreliable. For example, disk images attached to loop devices with *losetup* do not change their mtime when you mount the loop device and write to it. A real-world example I encountered was this: I had just read a disk image with ddrescue, which needed heavy file system correction. I decided to run my daily backup routine first. It ocurred to me to check if the mtime of the image file actually changed when writing to the mounted loop device, because I suspected the file was not accessed through regular file-open routines. And indeed, I was right. To make sure the file is backed up whenever I changed it, I needed to *touch* it first.

This could also be an issue with virtual machine disk images. I always use logical volumes as storage backends, so I can't check (easily), but you may want to check if the mtime of your virtual machine images actually changes when it's

written to. Or, setup a backup routine in the virtual machine itself, of course (which is what I would do).

Another (small) real world example is editing ID3-tags with Easytag. Easytag has an option to preserve the mtime of a file when changing the tag. Should the size of the altered tag be the same, when changing one charachter for example, the mtime and size will be identical, and the change will not be detected.

Rsync has an option to actually check if the file is different. The problem with this, however, is that it's very slow. For example, scanning my */home* for changed files takes longer than doing a complete backup with dar without compression. See the detailed rsync info below for more information.

You can of course decide that the risk of this failure in change detection is not a problem for you (since the chance is small), and benefit from the speed increase. Personally, I just don't really like it if I can formulate a scenario where it is known that the program fails to do what it should, but still accept it and use rdiff-backup on several locations.

5.2. Backing up into a file system

it may not be the smartest choice to simply copy your data to another file system. Using *cp -a* may do a pretty good job of preserving everything you need (but only when copying into a file system which supports everything the source file systems supports, and in case of cp, when not using extended attributes), but my concerns are of a different nature. It's all too easy to accidentally make a change to a file, or it's meta data, by opening and saving it. It is more robust to have the data in archive files, like tar or dar does.Because rsync also simply stores its meta data in the file system, this warning also applies to rsync.

With the information provided in this article, you should be able to decide for yourself if this is an issue or not. It also has definite advantages, such as rapidly being able to find and copy one single file out of the backup.

5.3. Archive size

A lot of file systems (or network protocols) have a very limited maximum file size. When making backups with software that creates archives, the size of these archives has to be considered. A maximum of 2 GB (or a little less, to be on the safe side) is usually a good idea. Files this big can stored on ISO DVD's and FAT32 file systems. My preferred choice would be 650 MB, so that they can be burned on (74 minutes) CD's.

Even if you're not planning to store the backup on such a file system, it's still a good idea split the archive files, so that you still can burn them on CD or DVD when you have to.

5.4. Restoring

Just as important as making the backup, is restoring the backup. Just as you would read the man page to figure out the correct options for backing up, you would for restoring. In my opinion, a good backup program has either well chosen defaults, or stores the options made during backup in the archive or meta data files, to be used upon restoration again.

Dar is good in this respect, as you should not need to specify any special options when restoring. When using tar you have to be somewhat more careful.

Source : http://www.halfgaar.net/backing-up-unix