pybackup : Another way of doing backups

Manuel Gutierrez Algaba. May 2003

June 21, 2003

1 Intro

It intends to be a clever and safer way of doing backups, it's comprised by two main python excutables: pyactivity and pybackup.
pybackup is just a recursive tar-zip program , pyactivity is a program that gives a sensible idea of what's up in your hard disk, by sorting into groups those files that have been changed at the same time , more or less.

The idea of pybackup.py is thus. Well, I simply got fed up of getting buggy backup files stored in CD ROMS. I had a large .tar.gz file and then it corrupted, and I lost 50 mb out of 650 mb. I decided to use a safer method, so that if 1 % of the file gets corrupted then I don't lose the 100 % of the file. Yes, I know that bzip2 and probably others can recover from errors, choose the size of block and so on, but *that* doesn't prevent the menace of getting bursts of faulty bytes and then lose full blocks (100k or 900k) because small faults. I know too ,that with DVD-RAM you don't worry about compressing files, just copy the entire filesystem into the DVD-RAM, well, that's not my case. I have old CD burners.

Well, what's the idea of pybackup. Basically there're three ways of backing up:

1. the raw method: no compressing, plaing cp'ing
2. the tar first, the compress secondly : this is the "dangerous" method
3. the compress first and then the tar: this is safer, because a corrupted compressed files does not affect the rest of the tar files. Yes , the tar index can corrupt itself and then you lose all the files. Well pybackup is just compress-tar-recursively in every subdirectory of a given directory. It's a **slow** process. But , I think , much safer in the long term, and that's what I like to have for my backups.

2 Using the program

Currently pyactivity.py and pybackup.py are independent programs, but they'll converge over the time.

2.1 Using pybackup.py

pybackup runs in Linux and with python. Well, it's a python program, I haven't developed yet a fancy interface , but it works compressing and decompressing. You need to write python code to use it, currently, don't worry, you can hack the code without knowing python, just following the following straight forward instructions. If you open pybackup.py with your favourite editor, you'll find at the last lines:

```
'''
os.system("rm -Rf /home/thor/hroyectos/pybackup/e/*")
finddirs().compressdir("test/")
finddirs(). printcontent("test/.xtar", "e" )
finddirs().launch("test/.xtar", "e" )

finddirs().expandzipfiles("e")
finddirs(). storeDirStruct("test", "estructdir" )
'''
finddirs().makeAlternateDir("/tmp/").
  setExcludeddirs("tmpant"). compressdir("test/")
finddirs().launch("test/.xtar", "e" )
finddirs().expandzipfiles("e")
#.expandDirStruct("e", "estructdir")
```

Well, let's explain some of this instructions. first of all, # in the beginning of a line converts that line into a comment. Well , `finddirs().compressdir("test/")` compresses a dir called "test" that is in the *same* directory where pybackup.py is running, you either include a full path or cp pybackup.py into the dir you want to backup (bad method), well, the result of the compression is a file named ".xtar" which is in the test/ directory, beware that is a hidden file , so you must use ls -la to see it, that ".xtar" file is no less than a tar file, composed of zip files and other ".xtar" files of every subdir of test/, on its behalf every ".xtar" of every subdir of test/ comprises other zipped files and other ".xtar" of ... you get the idea?

Next, `finddirs().launch("test/.xtar", "e" )` , this expands the ".xtar" file in all the files comprised in the ".xtar", that is, you get a list of ".zip" files in the "e" subdir. You don't get dirs , you'll get it if you unzip the files, then the dirs are created. You may find this a good or a bad idea, but really it's just an intermediate step .

Next, `finddirs().expandzipfiles("e")` expands all zip files in "e", after this, you got almost a copy of the original "test" hierarchy, except in a couple of details, permissions, you lose them , and the dirs with no files in the "test/" subtree doesn't exist in the "e/" subtree. For getting the same tree structure, empty dirs included, you have: `finddirs(). storeDirStruct("test", "estructdir" )` and `finddirs().expandDirStruct("e", "estructdir")`, the former creates the dir structure of "test/" subtree and the latter expands it.

Well, now let's get sofisticated,

```
finddirs().makeAlternateDir("/tmp/").
     setExcludeddirs("tmpant"). compressdir("test/")
```

This compresses the dir test into test/.xtar, excludes any subdir in the hierarchy whose name is tmpand, and for those copy protected files it uses the /tmp/ dir to work. THIS IS IMPORTANT.

2.2 Using pyactivity

Doing `python yourdirwhereitis/pyactivity.py` will generate "THE LIST" into stdin, sorted by 1 hour "radius", it will generate the list of the files in the dir you're running "python", take a look at the code to modify time or directory.

3 Licensing: GPL

I, Manuel Gutierrez Algaba, am the author, algaba@seul.org. Released under GPL license. I have no responsabilities for misuse, harm, wrongdoing,... that may arise or be caused by the use of this software. If you use it, you take all the risks and responsabilities for it. Read the GPL license before using this program.

4 Todo

1. getopt support, "ala" unix. Better interfaces.
2. Multivolume files
3. Better handling of excluded dirs
4. Incremental copies: Integration with pyactivity.py
5. Other OS'es support
6. Permission support