A file and directory synchronizer

Dale Miller

`Dale.Miller[at]inria.fr` *Difficulté : moyen (**).*

Votre rapport et votre exposé peuvent être en anglais ou en français.

1 Introduction

It is now common for people to work on several computer in the course of one day: for example, one might frequently move between a computer in the office and one at home and a laptop.

One problem with moving between such computers is trying to keep some of the files and directories on these different machines *synchronized*. For example, after working all night on a paper and some programs at home, you go to your office and you plan to continue working on these same files there. Thus you would like to have the new versions of files you edited at home sent to the office machine to overwrite the corresponding old version of the files there. You can also imagine that some files on both machines have changed and you would like to move files in both directions in order make the two file systems equal again.

One system that can be used to do such file synchronization is Unison. The main part of this project is to design and implement a file synchronizer with a logic similar to that used by Unison. For more on the design philosophy behind file synchronization and Unison, see the following web page, technical report, and slides.

- Web page for Unison: `http://www.cis.upenn.edu/~bcpierce/unison/`.
- Benjamin C. Pierce and Jérôme Vouillon. What's in Unison? A formal specification and reference implementation of a file synchronizer. Technical Report MS-CIS-03-36, Computer and Information Science, Univ of Pennsylvania, 2004. `http://www.cis.upenn.edu/~bcpierce/papers/unisonspec.pdf`
- Benjamin C. Pierce. File synchronization: Theory and practice, 2001. `http://www.cis.upenn.edu/~bcpierce/papers/new-snc-slides.ps`

2 More specifics

The particular goal here is to synchronize two directory structures on the same machine. That is, assume that there are two directories `root1/` and `root2/`. You should write a program that allows them to be synchronized.

1. Since the two roots exist prior to putting them "in sync", an initial phase must be done to initialize the synchronization phase for these roots. This phase will need to create an "archive" file/database that describes the state of the final synchronization.

2. Synchronization will be based on "static" information: that is, one does not assume that there is a log of actions taken on the files and directories. Instead, the previously generated archive file can be used to decide what files and directories have been changed or have been deleted or added. As a consequence of this "static" approach, renaming a file from a.txt to b.txt will look the same as deletion of a.txt and the creation of b.txt.

3. After synchronization, one should be able to change the files and directories in both roots and then re-synchronize again.

4. Sometimes the roots might change in ways for which no simple, automated solution appears (for example, the same file is updated in the two roots in different ways). Make sure to report to a user of the system such conflicts. (See above Unison documents for other such conflicts.) If no conflicts exist, have your system update the two roots in a sensible way.

5. What if someone changes files within the roots *while* synchronization is taking place? How can you make sure that the system does not overwrite new changes and lose data?

Your program solution should be able to: (i) initialize the archive file at the start of synchronization, (ii) interactively tell the user if there are conflicts for synchronization, (iii) what actions will be taken to make the synchronization complete, and (iv) preform those actions and update the archive file. Things should be left in such a state so that synchronization can be done again after some phase when the two root directories might be changed.

Java or OCaml are good choices for implementing this project.