IFT2245 – Travail pratique #3 – 2009-11-23

SYNCHRONISATION DE RÉPERTOIRES

Marc Feeley

Le TP3 se fera sur le système d'exploitation Linux. Il consiste à écrire un programme client/serveur pour synchroniser le contenu de deux répertoires sur des ordinateurs distincts. Cela demande d'utiliser les "sockets" UNIX pour créer et gérer une connexion réseau et à concevoir un protocole de synchronisation de répertoires.

---

1 Synchronisation de répertoires

Souvent un utilisateur a accès à plusieurs ordinateurs, un à la maison et l'autre au travail, ou encore un ordinateur portable et un desktop, etc. Dans un tel environnement, il est intéressant de maintenir des copies de tous les fichiers, ou les fichiers de certains répertoires, sur tous les ordinateurs pour qu'on puisse passer d'un ordinateur à l'autre de façon transparente.

Supposons que nous cherchions à maintenir les fichiers du répertoire R synchronisés sur les ordinateurs A et B. S'ils sont synchronisés le répertoire R de A et le répertoire R de B ont la même structure et le même contenu (incluant les permissions et dates de dernier accès et modification) pour tous les fichiers et sous-répertoires contenus dans R. Si l'utilisateur commence à travailler sur l'ordinateur A et modifie, crée ou élimine certains fichiers du répertoire R de A, il y aura maintenant des différences avec le contenu du répertoire R de B. Plus tard, mais avant que l'utilisateur commence à travailler sur le répertoire R de l'ordinateur B, il faudra synchroniser le répertoire R de A et le répertoire R de B pour qu'ils aient à nouveau le même contenu. Vous devez écrire un programme (client et serveur) qui effectuera cette synchronisation à travers le réseau.

Mais comment peut-on synchroniser deux répertoires? Il faut qu'un processus sur l'ordinateur A communique avec un processus sur l'ordinateur B afin de comparer le contenu de leurs répertoires R. Là où il y a des différences il faudra faire des modifications soit sur A ou sur B pour éliminer les différences. Pour éviter de transférer tous les fichiers entre A et B pour vérifier s'ils sont identiques, on peut faire la simplification suivante : les contenus de deux fichiers sont différents si et seulement si leur "hache MD5" est différent. Quelles sont les différences possibles entre deux répertoires et que faut-il faire pour les corriger?

1. **Le fichier F existe sur A et sur B, mais n'a pas le même contenu, ou bien pas les mêmes permissions ou dates de dernier accès ou de dernière modification.** Il faut regarder la date de dernière modification de F. Le fichier qui est le plus récent doit être copié sur l'autre machine en préservant les permissions et dates de dernier accès et de dernière modification. Évidemment, si le fichier a le même contenu sur A et B, il n'est pas nécessaire de transférer le fichier, il suffit de mettre à jour les dates.

2. **Le fichier F existe sur A et n'existe pas sur B.** On ne sait pas si c'est parce que l'utilisateur vient de créer le fichier F sur A (et donc qu'il faut le copier sur B) ou bien c'est qu'il vient d'éliminer le fichier F sur B (et donc qu'il faut l'éliminer sur A).

   Pour résoudre ce problème nous allons utiliser un truc : les *fichiers fantômes*. Le fichier fantôme du fichier F c'est un fichier vide dans le même répertoire que F mais avec le nom "`.synchro.`F". Les fichiers fantômes ne sont jamais éliminés. La procédure de synchronisation crée les fichiers fantômes et met à jour leurs dates de dernier accès et de dernière modification. Au moment d'une synchronisation, si le fichier F existe sur A et n'existe pas sur B on fait ceci :

(a) Si "`.synchro.`F" n'existe pas sur B, ou sa date de dernière modification est plus ancienne que celle du fichier F sur A, on transfère le fichier F de A à B, et on met à jour les dates du fichier F sur B et "`.synchro.`F" sur A et B pour qu'elles soient les mêmes que le fichier F sur A.

(b) Sinon on élimine le fichier F sur A.

Cette procédure de synchronisation fonctionne correctement à condition que les horloges sur les machines A et B soient synchronisées. Pour ce travail vous devez supposer que c'est le cas. Cependant, pour ne pas perdre d'information s'il y a une erreur de communication réseau ou une mauvaise synchronisation des horloges, on ne doit jamais véritablement éliminer les fichiers. Donc dans le cas 2b ci-dessus, on va plutôt déplacer le fichier F dans un sous-répertoire "poubelle" du répertoire qui contient F. Le nom du répertoire poubelle est "`.trash.`D" où D est la date du début de la procédure de synchronisation (au moins précise à la seconde). La procédure de synchronisation ne doit pas tenter de synchroniser les fichiers "`.synchro.`XXX" et "`.trash.`XXX" comme les fichiers normaux.

2 Programme `synchro`

Vous devez écrire un seul programme dont le nom est `synchro` (fichier source "`synchro.c`") qui peut être utilisé comme serveur et comme client. Vous devez également écrire un fichier "`makefile`" qui permet de compiler `synchro.c` pour produire `synchro` en exécutant la commande "`make`".

Pour synchroniser un répertoire il faut démarrer le serveur sur une machine et le client sur une autre machine. L'utilisateur doit choisir un numéro de port libre qui sera utilisé par le serveur et le client (c'est le premier argument de ligne de commande de `synchro`). Du côté client il faudra en plus spécifier le nom de l'autre machine et le répertoire à synchroniser (vous pouvez supposer que le répertoire est relatif au client et au serveur). Par exemple, pour synchroniser le répertoire `ift2245/tp3` sur les machines `frontal01.iro.umontreal.ca` et `portable.iro.umontreal.ca` on pourra faire :

- Sur `frontal01.iro.umontreal.ca` : `% ./synchro 8544`
- Sur `portable.iro.umontreal.ca` : `% ./synchro 8544 frontal01 ift2245/tp3`

Le client doit imprimer sur la sortie standard la liste de tous les fichiers qui ne sont pas identiques sur les deux machines. La synchronisation de deux répertoires ne doit pas changer les dates de dernier accès et modification des fichiers qui sont identiques sur les deux machines.

C'est à vous de concevoir un protocole pour l'échange d'information entre le client et le serveur. Le protocole n'a pas besoin d'être sécure. Cependant, il y a jusqu'à 20 points bonis pour ceux qui implantent un protocole sécure (c'est-à-dire où il est impossible, ou très difficile, pour un fraudeur de connaître quoi que ce soit sur les fichiers échangés). Comme point de départ vous pouvez vous familiariser avec les principes de la cryptographie à clé publique (http://fr.wikipedia.org/wiki/Cryptographie_asymétrique), ou bien allez voir la bibliothèque SSL ("`man ssl`"). Vous pouvez supposer l'existence du programme `md5sum` qui calcule le hache MD5 du fichier spécifié sur la ligne de commande, par exemple :

```
% md5sum -b ~dift2245/tp2/minos2.tar.gz
ed933e608aa028eb154759bac46242c6  /u/dift2245/tp2/minos2.tar.gz
```

Le nombre hexadécimal de 32 chiffres qui précède le nom de fichier est le hache MD5 du fichier.

3 Évaluation

Il n'y a pas de rapport à rédiger pour le TP3. Il suffit de faire la remise des fichiers "`makefile`" et "`synchro.c`". Il n'y a pas de test à faire en présence du démonstrateur.

- Pondération:
  - 10%: Établissement et gestion de la connexion (serveur)
  - 10%: Établissement et gestion de la connexion (client)
  - 20%: Traitement des erreurs et robustesse
  - 30%: Protocole de synchronisation
  - 30%: Accès et modification du système de fichier
  - 20%: Bonis pour la sécurité du protocole
- L'élégance et la lisibilité du code, l'exactitude et la performance sont des critères d'évaluation.
- **Vous devez faire le travail par groupes de 2 personnes.** Indiquez vos noms en commentaire en haut des fichiers "`makefile`" et "`synchro.c`".
- Vous avez jusqu'à 23h59 le 7 décembre pour remettre votre programme. Vous serez pénalisés de 30% par jour de retard.
- Vous devez remettre vos programmes par remise électronique grâce à la commande suivante:

```
remise ift2245 tp3 makefile synchro.c
```