Sivatarsiban Steeve SIVANANTHAM
Damien Chesneau

Projet de système

Pour ce projet de système voici la structure que nous avons choisi :

- synchronizer.c (contenant le main)
- makefile
- module parser
- module copy
- module connect

Choix des modules :

Pour réaliser ce projet, nous avions plusieurs choix à faire.

Tout d'abord, nous avons choisi de séparer le projet sur deux parties (sur des thread différents). Une partie du programme va attendre qu'une instance externe se connecte et une partie qui sera exécuté pour se connecter aux instances actives se trouvant aux adresses IP données en argument au programme.

La première partie s'exécutera toujours. La deuxième n'est seulement utilisé quand des arguments du programme sont identifié. L'intérêt de cette séparation est de pouvoir faire plusieurs synchronisation en même temps.

Ensuite, nous avons choisi que l'échange de l'intégralité des données se ferait d'un seul coup. Les échanges sont bidirectionnelle.

Cette méthode à un défaut majeur: la RAM peut vite se saturer pour peu qu'il y ait plusieurs instances sur une même machine. Ou qu'il y ait des grosses quantités de données.
Cependant, cette méthode facilite grandement la synchronisation des dossiers.

Enfin, nous avons choisi le protocole de synchronisation suivant :

- Une instance A récupère les données de son répertoire.
- L'instance A se connecte à un ensemble d'instance B.
- Pour chaque instance de B, l'instance B récupère les données de son répertoire, les encode et les envoie à A. A récupère les données encodé de cette instance, les décode et enrichie ses propres données en fonction des timestamp et des nouveaux fichiers des données décodé.
- A encode ses données.
- Pour chaque instance de B, A envoie ses données encodé, l'instance B récupère les données de A, les décode et met à jour son répertoire en fonction des données décodé.
- A met à jour son répertoire en fonction de ses données.

Le module parser :

Dans ce module, nous avons regroupé toutes les fonctions utilisées pour encoder et décoder les données à envoyer sur le réseau. Pour plus de détails, voir parser.h

Le module copy :

Dans ce module, nous avons regroupé toutes les fonctions utilisées pour la lecture et l'écriture de données sur le disque. Pour plus de détails, voir copy.h

Le module connect :

Dans ce module, nous avons regroupé toutes les fonctions utilisées pour l'envoie et la réception des données sur le réseau. Pour plus de détails, voir connect.h

Difficultés:

Durant la réalisation de ce projet nous avons fait face à différentes difficultés. Tout d'abord, notre projet inclut le niveau un et la synchronisation récursive du niveau deux. Par manque de temps, nous avons malheureusement pas pu faire plus que cela.

Nous avons en premier lieu eu des difficultés a nous mettre d'accord sur l'implémentation de ce projet. Il nous a donc fallu en priorité avoir la même vision du projet. Ensuite, nous avons pris beaucoup de temps à mettre en place le protocole d'échange des données. Beaucoup de protocoles ont été envisagé. Chacun avait leurs avantages et leurs défauts, mais nous ne savions pas lequel mettre en avant.

Ensuite, nous avons rencontré quelques soucis au niveau de l'encodage et du décodage plus précisément au niveau de la gestion de la mémoire.

Enfin, nous avons eu beaucoup de mal à faire en sorte que le programme fonctionne correctement. Ce n'est probablement pas encore parfait, mais la gestion des erreurs aura été l'une des plus grosse difficulté rencontré durant la réalisation de ce projet.