

PROJET L3 SYSTEME 2015

Auteurs : Thomas Néang et Ghyslaine Ribeiro

Objectif : Synchroniser plusieurs répertoires sur des machines distantes.

Sommaire

1) Langage utilisé et dépendances.....	1
2) Architecture du programme.....	1
a) src/peer.cpp.....	1
b) src/tracker.cpp.....	2
c) include/packet.h.....	2
3) Niveau 1.....	3
4) Niveau 2.....	3
5) Niveau 3.....	3
6) Conclusion.....	4

1) Langage utilisé et dépendances.

Nous avons choisi le langage C++ pour développer ce projet afin de bénéficier de la STL qui contient listes chaînées, maps, strings... Cependant nous avons gardé le caractère impératif (sans programmation objet) afin de garder l'aspect du langage C.

Le programme n'utilise pas de bibliothèques externes et n'a donc pas de dépendances connues.

2) Architecture du programme.

Le projet est composé des répertoires (bleu clair), fichiers (vert) et binaires (bleu foncé) suivants :

/src	/include	/bin	/log	Makefile	doc.pdf
tracker.cpp	tracker.h	tracker	tracker.log		
peer.cpp	peer.h	peer	peer.log		
	packet.h				

a) src/peer.cpp

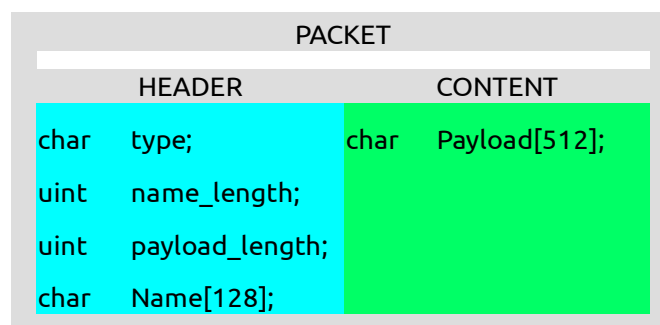
Ce fichier contient le code du binaire "peer". C'est ce programme qui fait la synchronisation des fichiers et répertoires contenus dans le répertoire racine dans lequel il est exécuté avec ses autres instances.

b) src/tracker.cpp

Ce fichier contient le code pour produire le binaire "tracker". Ce programme annexe sert à stocker les adresses ip et port des instances du programme peer afin de faire "vivre" le réseau.

c) include/packet.h

Pour établir une communication tracker ↔ peer et peer ↔ peer nous avons défini un format de requête qui transitera dans le réseau. Celui-ci n'est autre qu'une structure définie dans packet.h dont voici le contenu :



- type est l'identifiant de requête :
 - 'c' = demande de création d'un peer (Create) ;
 - 'l' = demande de déconnexion d'un peer (Leave) ;
 - 'u' = mise à jour de la liste des peers (Update) ;
 - 'f' = demande de création d'un fichier (File create) ;
 - 'a' = demande d'ajout de contenu pour un fichier (Add content) ;
 - 'o' = réponse à la demande d'envoi de contenu (Ordered) ;
 - 's' = demande de suppression d'un fichier (Supprime file) ;
 - 'm' = demande de mise à jour d'un fichier (Mise à jour) ;
 - 'd' = demande de création d'un répertoire (Directory create) ;
 - 'r' = demande de suppression d'un répertoire (Remove directory).
- name_length est la taille en octets du champ name.
- name est une chaîne de caractères qui spécifiera le nom du répertoire ou fichier à créer.

- payload_length est la taille en octets de la chaîne contenue dans le champ payload.
- payload sert à stocker le contenu d'un fichier seulement s'il y'a requête de création de fichier ou une mise à jour.

La taille du "packet" est petite afin d'éviter son partitionnement lors de sa transition dans le réseau.

3) Niveau 1.

Afin de pouvoir faire communiquer plusieurs instances du programme ensemble nous avons décidé d'utiliser la technique des sockets et d'envoyer des requêtes dans le réseau (voir trame ci-dessus). Lors de son lancement, le programme (peer) prend connaissance du répertoire racine dans lequel il s'exécute : il stocke dans des map (C++) les noms des fichiers, liens et répertoires trouvés grâce à la fonction "ftw" qui parcourt récursivement les sous-répertoires depuis la racine. Une fois ce travail fait, le programme envoie à ses instances voisines des demandes de créations de fichier / répertoires lors de la synchronisation.

4) Niveau 2.

La synchronisation récursive est fournie par la fonction "ftw" (voir Niveau 1 ci-dessus).

Pour ne plus avoir à renseigner les instances voisines du programme directement dans le code nous avons développé un système de reconnaissance automatique. Deux solutions nous sont parvenues :

- La première solution était la mise en place d'un système de broadcast. En effet, faire un broadcast dans un réseau local aurait permis à chaque instance de peer de se retrouver, la première instance lancée dans le réseau aurait fait office de "base de données" stockant les adresses ip et les ports de ses voisines. Cependant, le défaut de cette méthode est que si l'application est amenée à être utilisée sur le net, le broadcast n'y sera plus possible donc la reconnaissance automatique ne fonctionnera plus.
- La deuxième alternative (celle que nous avons choisie) est de développer un second programme (tracker) qui stocke les adresses ip et les ports machine de toutes les instances du programme (peer) qui s'y connecte et de renvoyer à ces même instances une liste complète de toutes les autres s'y étant auparavant connectées. Le système ressemble un peu à un réseau déjà existant (le p2p) cependant il n'y a pas été ajouté un système de ping tracker ↔ peer dans le cas où une des instances de peer serait brutalement déconnectée.

5) Niveau 3.

Afin de compléter notre programme nous avons ajouté la gestion des suppressions. Celle-ci a pu être implémentée grâce à la structure unit ci-dessous et plus particulièrement grâce au champ "exist" :

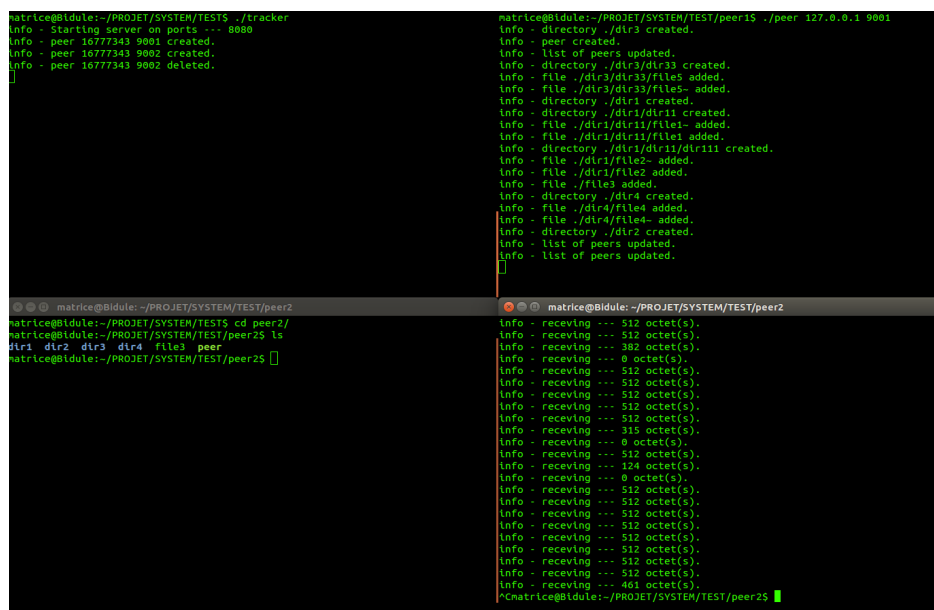
```
typedef struct    _unit_  
{  
    long int      mtime;  
    int           exist;  
    char          state;  
    unit;  
}
```

En effet, avant que "ftw" fasse le parcours récursif des répertoires pendant la synchronisation nous mettons à tous les fichiers déjà stockés dans la map le champ exist à 0 et c'est lorsque "ftw" trouve le fichier en faisant son parcours que nous le mettons à 1. Si le fichier a été supprimé "ftw" ne le trouvera pas, son champ "exist" restera toujours à 0 et nous pourrons envoyer une requête de suppression à toutes les instances de peer.

En ce qui concerne les liens symboliques, nous avons fait un dernier test en créant un lien dans un des répertoires d'une des instances de peer et le fichier pointé par ce lien a directement été copier dans toutes les autres instances.

6) Conclusion

Ce projet a été très intéressant dans la mesure où il a pu nous faire pratiquer la technique des threads, sockets, fork, exec et la connaissance du langage C++. Afin d'améliorer le programme nous espérons pouvoir réussir le transfert d'image voir le téléchargement de vidéo et y implémenter une interface graphique pour une éventuelle migration vers Windows.



```
matrice@bldule:~/PROJET/SYSTEM/TESTS ./tracker  
Info - Starting server on ports --- 8080  
Info - peer 16777343 9001 created.  
Info - peer 16777343 9002 created.  
Info - peer 16777343 9002 deleted.  
  
matrice@bldule:~/PROJET/SYSTEM/TEST/peer1S ./peer 127.0.0.1 9001  
Info - directory ./dir3 created.  
Info - peer created.  
Info - list of peers updated.  
Info - directory ./dir3/dir33 created.  
Info - file ./dir3/dir33/files added.  
Info - file ./dir3/dir33/files- added.  
Info - directory ./dir1 created.  
Info - directory ./dir1/dir11 created.  
Info - file ./dir1/dir11/file1- added.  
Info - file ./dir1/dir11/file1 added.  
Info - directory ./dir1/dir11/dir111 created.  
Info - file ./dir1/file2- added.  
Info - file ./dir1/file2 added.  
Info - file ./files added.  
Info - directory ./dir4 created.  
Info - file ./dir4/file- added.  
Info - file ./dir4/file- added.  
Info - directory ./dir2 created.  
Info - list of peers updated.  
Info - list of peers updated.  
  
matrice@bldule:~/PROJET/SYSTEM/TESTS cd peer2/  
matrice@bldule:~/PROJET/SYSTEM/TEST/peer2S ls  
dir1 dir2 dir3 dir4 files peer  
matrice@bldule:~/PROJET/SYSTEM/TEST/peer2S  
  
matrice@bldule:~/PROJET/SYSTEM/TEST/peer2  
Info - receiving --- 512 octet(s).  
Info - receiving --- 512 octet(s).  
Info - receiving --- 382 octet(s).  
Info - receiving --- 0 octet(s).  
Info - receiving --- 512 octet(s).  
Info - receiving --- 512 octet(s).  
Info - receiving --- 512 octet(s).  
Info - receiving --- 512 octet(s).  
Info - receiving --- 512 octet(s).  
Info - receiving --- 515 octet(s).  
Info - receiving --- 0 octet(s).  
Info - receiving --- 512 octet(s).  
Info - receiving --- 124 octet(s).  
Info - receiving --- 0 octet(s).  
Info - receiving --- 512 octet(s).  
Info - receiving --- 512 octet(s).  
Info - receiving --- 512 octet(s).  
Info - receiving --- 512 octet(s).  
Info - receiving --- 512 octet(s).  
Info - receiving --- 512 octet(s).  
Info - receiving --- 512 octet(s).  
Info - receiving --- 512 octet(s).  
Info - receiving --- 512 octet(s).  
Info - receiving --- 461 octet(s).  
matrice@bldule:~/PROJET/SYSTEM/TEST/peer2S
```