

Report des Algorithmes de Réseaux

ISMAYILOVA Maryam, MAMMADOV Ali

Documentation utilisateur

Fichiers de données

Dans ce projet les données sont gardés en general dans 2 types de fichiers. Ceux sont:

- fichierUtilisateur.txt
- fichierDonnees_pid

Le premier, **fichierUtilisateur.txt** est celui qui va garder toutes les informations nécessaires pour identifier chaque client qui tente de se connecter à la base de données. Le format de stockage de données dans ce fichier, qui est reconnu par le programme est suivant:

```
login:motDePasse:uid [champs:]*  
login:motDePasse:uid [champs:]*  
.....  
login:motDePasse:uid [champs:]*
```

Où login est le nom de l'utilisateur qu'il envoie lors de l'étape de connexion au serveur d'Accès.

Le mot de passe est aussi celui envoyé au serveur d'Accès lors de l'étape de connexion.

Et uid n'est pas connu par le client, celui joue un rôle d'identifiant de l'utilisateur pour le stockage et recuperation de ces données de serveur de Données.

Car notre programme n'est pas considéré pour enregistrer de nouveaux utilisateurs, les informations de chaque nouvel utilisateur doivent être entré à la main.

Après *login:motDePasse:uid* il est obligatoire de laisser un espace vide (même si le client ne possède le droit à aucun champ) pour respecter la propre fonctionnement du programme.

Un 'utilisateur peut avoir permissions aux plusieurs champs, qui sont tous écrites en ligne avec ':' comme séparateur.

Toute l'information concernant un utilisateur doit rester sur une seule ligne.

Concernant le **fichierDonnées_pid**, ce fichier est créé lors de démarrage de chaque nouveaux serveur de données. Pour pouvoir distinguer plusieurs fichiers le nom de chaque fichier contient le numéro de processus qui l'a créé.

Les données sont gardés de manière d'une couple par ligne, où chaque couple est uid:données.

```
uid:données  
uid:données  
.....  
uid:données
```

Où chaque *uid* doit correspondre à un utilisateur dans fichierUtilisateur.txt. Toutes les mises à jour sont effectuées par les serveurs de données auxquels les fichiers sont associés. Quand même lors de la fermeture inattendue de serveur les fichiers restent en mémoire.

Makefile

Pour automatiser la compilation des programmes on a mis un makefile qui compile tous les programmes en respectant leurs relations.

Les possibilités proposées par ce makefile, sont les suivantes.

- make all ou simplement make - pour tout faire compiler.
- make serv - pour compiler le serveur d'accès.
- make sd - pour compiler le serveur de données.
- make client - pour compiler le programme du côté client.
- make free - pour effacer tous les fichiers de données créés par des processus précédents.
- make clean - pour effacer toutes exécutables et fichiers de données.

Execution des programmes

Les noms pour des objets binaires comme prévus par makefile sont sd (serveur de données), serv (serveur d'accès), client.

```
maryam@maryam-Lenovo-B50-70:~/Etudes/AlgoRes/project
/algo_res$ make
cc client.c -o client
cc serveurAccess.c traverserFichier.c -o serv -pthread
ad
cc serveurDonnees.c fichierLectureEcriture.c -o sd
maryam@maryam-Lenovo-B50-70:~/Etudes/AlgoRes/project
/algo_res$ ./serv
En écoute:
Nouveau serveur de donnees connecté
En écoute:
[]

maryam@maryam-Lenovo-B50-70:~/Etudes/AlgoRes/project/
/algo_res$ ./sd age
SUCCESS
[]

maryam@maryam-Lenovo-B50-70:~/Etudes/AlgoRes/project
/algo_res$ ./client 10450
Votre requete: []
```

Comme dans l'image au dessus, c'est un exemple d'exécution de projet.

Pour l'exécution de serveur d'accès il n'y a aucun argument nécessaire, pour le serveur de données il est nécessaire de fournir le nom de colonne qu'il va servir, et pour côté client le client va lui-même fournir le numéro de port où il veut lancer le programme.

Lancer les serveurs et envoyer des requêtes.

1. Au début on lance le **serveur d'accès** qui est en écoute des nouvelles connexions d'utilisateurs ou de serveurs de données. À chaque réception de nouveau message le serveur l'analyse, le traite et renvoie le résultat à l'envoyeur.
2. Suivant à serveur d'accès il faut lancer les **serveurs de données**. Le serveur va recevoir une nouvelle connexion, vérifier si le nouveau serveur a son doublé ou non, et l'envoyer le message de succès ou de données d'autre serveur sur le même champ.
3. Et maintenant on peut lancer les **clients**.

Chaque fois les clients envoient des requêtes qui seront traitées par le serveur.

Le premier message à envoyer c'est la *demande de connexion*, qui se fait comme au dessous:

INIT nomUtilisateur:motDePasse

Le serveur cherche le client et l'envoie le message de succès ou erreur si le client n'existe pas dans le fichier des utilisateurs.

Si le client a été trouvé, le serveur lance une nouvelle connexion entre le client et serveur pour l'échange instantané des messages.

Les requêtes que l'utilisateur peut envoyer au serveur en connexion sont suivantes:

ECRIRE [champs:donnée]*

LIRE [champ:]*

SUPPRIMER

FIN

ECRIRE a pour but d'insérer une ou plusieurs nouvelles lignes sur des champs différents ou de modifier celles existants.

LIRE permet d'obtenir toutes les données correspondantes à la liste de champs (quand même l'utilisateur doit aussi avoir des droits pour les champs, sinon il ne recevra des données).

SUPPRIMER fait effacer toutes les données de l'utilisateur.

FIN est utilisé pour fermer la connexion entre le client et serveur, et après un autre client pourrait lancer une nouvelle connexion entre lui et serveur.

```
Client en connexion sur serveur d'access (0):
En écoute:
Serveur a reçu un requet de connexion
Client a se connecter:mar, 1234
    Client est autorise!

Client en connexion sur serveur d'access (1):
client: mar 3
    age, taille,

Connexion client en écoute:
En écoute:
    Serveur a reçu une requet: ECRIRE
    Envoyer message: "3:17" a serveur sur port 4
4662
    Champs age ont ete modifies !
    Complet!
Envoi de message de reussite
Connexion client en écoute:
    Serveur a reçu une requet: SUPPRIMER
    Envoyer message: "3" a serveur sur port 4466
2
    Champs age ont ete supprimes de serveur sur
port 30382!
    Complet!
Envoi de message de reussite
Connexion client en écoute:
A bientot!
[]

maryam@maryam-Lenovo-B50-70:~/Etudes/AlgoRes/project/
algo_res$ ./sd age
SUCCESS
Requet a ecrire 3:17
ECRIRE a ete effectue!
Requet a supprimer ligne de 3
SUPPRIMER a ete effectue!
[]

maryam@maryam-Lenovo-B50-70:~/Etudes/AlgoRes/project
/
algo_res$ ./client 10300
Votre requete: INIT mar:123
A envoie INIT mar:123
Connection failed
Votre requete: INIT mar:1234
A envoie INIT mar:1234
Port de connexion 6001
Connection Established
Votre requete: ECRIRE age:17
A envoie ECRIRE age:17
OK
Votre requete: SUPPRIMER
A envoie SUPPRIMER
OK
Votre requete: FIN
A envoie FIN
Connexion ferme!
Votre requete: bye
maryam@maryam-Lenovo-B50-70:~/Etudes/AlgoRes/project
/
algo_res$
```

Au dessus vous pouvez observer un exemple d'échange des messages.

À chaque réception des messages et durant la transaction server affiche le statut de traitement.

La liste complete des requêtes possibles pour des clients est au dessous:

INIT login:motDePasse

LIRE champ1 champ2 champN

ECRIRE champ1:donnée1 champ2:donnée2 champN:donnéeN

SUPPRIMER

FIN

