Travail pratique de programmation système avancée

Logging

1 Objectifs

- I/O asynchrone
- Syntaxe async/await
- Partage de données asynchrones

2 Énoncé

2.1 Logging dans un fichier

Dans cet exercice, on suppose qu'on a prgramme (serveur, démon) qui reçoit des messages. Ces messages doivent être loggés dans un fichier pour inspection ultérieure. Ces messages arrivent dans un ordre indéterminé et donc il semble raisonnable d'avoir une tâche qui envoie le message à notre exécuteur pour que le message soit écrit dans le fichier de façon asynchrone quand le fichier est disponible.

On peut imaginer que le logging fait partie d'une application plus conséquente et qu'on ne veut pas bloquer toute l'application pendant qu'on écrit le fichier de logs.

Pour simplifier, on va uniquement logger les logins/logout d'utilisateurs fictifs. Leurs noms sont les suivants:

```
let names = vec!["Orestis", "Paul", "Florent", "Andres", "Fabien", "Quentin"];
```

3 Cahier des charges

1. On va logger le nom des utilsateurs se loggant/déloggant du système dans deux fichiers `login.txt` et `logout.txt`. Il faut donc créer une fonction créant un fichier en écriture s'il n'existe pas ou en mode `append` s'il existe déjà et retournant son identifiant (un type `File`) comme ressoruce partagée.
2. Créer un type `AsyncWriteFuture` sur lequel vous implémenterez le trait `Future` permettant d'écrire le nom de la personne se logant/délogant dans un fichier. En cas d'échec on affiche un message d'erreur avec l'erreur qui a eu lieu. Sinon on retourne `Ok(())`.
3. Créer une fonction permettant d'écrire le log à proprement parler à partir du fichier et du nom à logger.

Finalement, écrivez un programme créant un itérateur contenant avec les logins/logout à logger dans un futur chacun et `join_all()` le tout à la fin de l'exécution.

3.1 Indications

Pour cette application vous devez utiliser la librairie `futures-util` et la fonction `join_all()`.