# Key-value Store: Project 2018

Andrea Rar, Ryan Siow, Jonas Epper

IN.4022 Operating System Course, University of Fribourg
andrea.rar@unifr.ch, ryan.siow@unifr.ch, jonas.epper@unifr.ch

**Table des matières**

# 1 Introduction

Ce mini-projet a pour but d'implémenter un stockage clé-valeure avec des accès simultanés via TCP. Ce stockage utilise un tableau associatif comme modèle. Dans ce dernier, les données sont représentées comme une collection de pairs tel que chaque clé est unique dans la liste. L'utilisation de sémaphore ou de mutex étudié en cours permet de pallier au problème de "race condition".

# 2 Description du problème

Plusieurs problèmes doivent être traités durant ce projet. Ce dernier doit permettre d'écrire les clés avec les valeurs via TCP sockets. Lors du processus de lecture, il doit être possible de lire les valeurs via TCP sockets en fournissant la clé (un message d'erreur s'affiche si la clé entrée n'existe pas). Le programme doit utiliser des threads multiples et un accès simultané et sûr doit être garantit aux "readers" et "writers". Enfin, un script permettant un test automatique doit être fournit pour faciliter la tâche de révision du code.
Durant ce projet, diverses décisions d'implémentation ont dû être prise pour rendre la solution unique et proche de l'efficacité optimale.
La première étape consiste à créer une structure permettant de recevoir des paires de clés et de valeurs. Cette dernière doit être dynamique, c-à-d grandir à mesure que des paires clé-valeurs sont ajouté dans le but d'utiliser le moins de mémoire possible. En outre, cette structure doit supporter des actions telles qu'ajouter, modifier, lire ou supprimer des paires.
La deuxième étape se concentre sur la gestion de la communication entre le serveur et ses clients. Le serveur doit être transparent pour les clients, et ceux-ci doivent pouvoir envoyer des requêtes simples au serveur qui se charge de retourner les résultats.
Finalement, les accès à la structure de donnée doit être sécurisé afin qu'aucun conflit ne surgisse. Par exemple, il est interdit que deux clients puissent modifier en même temps la même pair, ceci pouvant générer des comportements inattendus et imprévisibles.

# 3 Solutions et décisions prises

Dans cette partie, les solutions élaborées pour résoudre chaque problème mentionné auparavant sont décrites:

1. Structure KV store (Ryan)
2. Communication serveur-client (Andrea, Jonas)
3. Fonctions: lire, écrire, modifier, supprimer et Regex (Tous)
4. Accès sécurisé des "readers"/"writers" (Andrea)
5. Script de test (Jonas)

### 3.1 KV store: array dynamique de struct

Nous avons opté pour ce projet, de construire un tableau redimensionnable. Les valeurs ainsi que les clés sont ajoutées les unes après les autres dans le tableau. Lorsque les élément atteignent la taille maximale du tableau qui lui est initialement attribuée, le tableau est redimensionné à deux fois sa taille initiale, et ainsi de suite.
Les fonctions pouvant manipuler le tableau permettent d'ajouter, supprimer, modifier et lire les éléments de ce dernier. Lorsque un élément est supprimé du tableau, il laisse un trou qui peut être complété par la prochaine valeur ajoutée, au lieu que cette dernière s'ajoute à la fin du tableau. Cela permet d'optimiser l'espace (trou complété) en n'allouant le moins possible de mémoire.
Voici la structure du tableau:

```
typedef struct{
    int key;
    char *value;
    size_t used; //indicate where we are in the array
    size_t size; //indicate the size of the array
}KVstore;
```

Ce tableau dynamique comporte 3 fonctions permettant d'initialiser, d'ajouter des éléments ainsi que de libérer la mémoire allouée.
Bien que ces fonctions méritent d'être étudiées plus en détail, seule une description de "insertKV()" est donnée. En effet, cette fonction est la plus intéressante car c'est dans celle-ci que l'optimisation de l'espace alloué est faite.
Tout d'abord, la fonction redimensionne le tableau si celui-ci en a besoin grâce à la méthode realloc(). La valeur "newvalue" alors passé en argument, si elle n'est pas nulle, pourra être insérée dans le tableau, soit dans un espace libre s'il y en a, soit à la fin de la série d'éléments déjà présents. Voici un extrait de code explicite:

```
if (newvalue != NULL) {
    size_t length = strlen(newvalue);
    //check if there are holes, if yes add in the hole
    for(int j = 0; j<=kv->used; j++){
        if(kv[j].key == -1 || kv[j].value == NULL ){
            size_t length1 = strlen(kv[j].value);
            kv->used++;
            //clear the memory in this index
            memset(kv[j].value, 0, length1);
            // insert new value
            strncpy(kv[j].value, newvalue,length);
            kv[j].key = newkey; // indicate key
            break;
        }
        //if no holes, just add at the end of the series
        if(j == kv->used){
            kv->used++;
            strncpy(kv[j].value, newvalue,length);
            kv[j].key = newkey; // indicate key
        }
    }
}
```

Le tableau redimensionable a été choisit pour sa simplicité d'implémentation. Toutefois, tel qu'il est implémenté dans ce code, ce tableau comporte un inconvénient: il n'est pas de possibilité de réduire la mémoire alloué (la taille du tableau exacte) pour le tableau si les éléments sont supprimés. Les éléments sont ajouté dans les trous créés par les éléments supprimés afin de pallier au problème et optimiser l'utilisation de la mémoire. Il est aussi déconseillé d'allouer la mémoire à nouveau lorsque la taille du tableau est réduite, car une copie de ce dernier doit être faite à chaque processus de réduction. Des pertes d'éléments peuvent en être la conséquence.
Une meilleure implémentation aurait été une liste lié (linked list), ainsi, le problème de dimensionnement lors d'une suppression d'élément disparaît.

### 3.2 communication server - clients

nombres de client max Andrea : renvoie des messages aux client Jonas tout le reste

### 3.3 lire,écrire et modifier des valeurs

Tous Regex,

### 3.4 Accès sécurisé des lecteurs/rédacteurs

Andrea qui a la priorité? read/write etc nombre de clients max? reader/writer nombres citer la page web pour le problème des lecteurs/rédacteurs

### 3.5 scripts de tests

Jonas
italique: *exemple italique*

## 4 apprentissage

ce que nous avons appris Tous

## 5 Conclusion

## References

[1] ???? *titre de l'article, ref.* Read the ....

[2] ??? *faire un tableau dynamique en C.* `http://www....`, Last visited: ....

## A Manuel d'utilisateur

Le fichier README vous explique de manière détaillée comment compiler et exécuter le programme.