Devoir de systèmes d'exploitation

Composition du groupe : *Ryan YALA, Mohamed Aziz Teyeb*

Introduction

Ce rapport présente le travail réalisé dans le cadre du projet de systèmes d'exploitation pour l'année académique 2022-2023 à l'Institut Galilée, filière INFO 1. Le projet consiste à mettre en pratique les concepts vus en cours, tels que la création de processus, la communication et la synchronisation entre les processus. L'objectif principal du projet est de développer un service de messagerie instantanée permettant à plusieurs utilisateurs d'une même machine de se connecter simultanément.

Résumé de la partie 1

Dans un premier temps, nous avons mis en place une structure appelée "User" ainsi qu'une structure "UserList" pour faciliter la gestion des utilisateurs connectés.

```
typedef struct {
    char name[MAX_NAME_LENGTH];
} User;

typedef struct {
    User users[MAX_USERS];
    int numUsers;
    int numConnectedUsers;
} UserList;
```

Par la suite, nous nous sommes focalisés sur l'implémentation du menu et l'interprétation des commandes entrées par les utilisateurs. Dans cette partie, nous avons rencontré une légère difficulté lors du traitement de la commande "p". En effet, nous avons choisi de l'implémenter de la manière suivante : "p User1 User2". Nous avons donc dû récupérer l'argument de cette commande et ensuite le décomposer en deux parties distinctes afin d'établir correctement le dialogue entre "User1" et "User2". C'est le travail effectué par la fonction "découperMots".

```
void decomposerMots(const char* chaine, char** mot1, char** mot2) {
    *mot1 = (char*)malloc(strlen(chaine) + 1);
    *mot2 = (char*)malloc(strlen(chaine) + 1);

    int index = 0;
    while (chaine[index] != '␣' && chaine[index] != '\0') {
        (*mot1)[index] = chaine[index];
        index++;
    }
    (*mot1)[index] = '\0';

    if (chaine[index] == '␣') {
        index++;
    }

    strcpy(*mot2, chaine + index);
}
```

Résumé de la partie 2

Dans la deuxième partie, nous avons simulé un serveur en utilisant un segment de mémoire partagée (SHM) pour stocker la liste des utilisateurs connectés. Deux nouvelles commandes ont été ajoutées :

"l" : pour afficher la liste des utilisateurs connectés. "d" : pour se déconnecter du serveur. La commande "p" a été modifiée pour permettre la communication uniquement avec les utilisateurs enregistrés sur le serveur.

Nous avons également mis en place des mécanismes de synchronisation des accès au segment de mémoire partagée pour éviter les problèmes de concurrence lors de la lecture et de la modification des données grâce au sémaphore.

Nouvelles Fonctions pour la Deuxième Partie

Dans le cadre de la deuxième partie du projet, nous avons ajouté les fonctions suivantes :

- `void listConnectedUsers(UserList * sharedData)` : Cette fonction permet de lister les utilisateurs connectés au serveur. Elle accède au segment de mémoire partagée (SHM) et parcourt la liste des utilisateurs enregistrés, affichant leurs noms à l'écran.
- `void disconnectUser(const char* name,UserList *sharedData)` : Cette fonction permet à un utilisateur de se déconnecter du serveur en retirant son nom de la liste des utilisateurs connectés. Elle effectue la recherche du nom dans la liste, supprime l'utilisateur correspondant et met à jour le segment de mémoire partagée.
- `void enregistrement(const char* nom_utilisateur, UserList* p)` : Cette fonction est responsable de l'enregistrement d'un utilisateur sur le serveur. Elle prend en argument le nom de l'utilisateur et l'ajoute à la liste des utilisateurs connectés dans le segment de mémoire partagée.

Ces nouvelles fonctions nous ont semblé nécessaires pour la gestion des utilisateurs connectés dans la deuxième partie du projet. Elles permettent de manipuler la liste des utilisateurs dans le segment de mémoire partagée de manière sécurisée et synchronisée, en assurant la cohérence des données lors des opérations d'ajout et de suppression d'utilisateurs. Pour la synchronisation, nous avons ajouté un sémaphore afin de nous assurer qu'un seul utilisateur à la fois puisse accéder aux données partagées.

Conclusion

En conclusion, ce projet nous a permis de mettre en pratique les concepts de systèmes d'exploitation étudiés en cours. Nous avons acquis une meilleure compréhension de la gestion des processus, de la communication et de la synchronisation entre les processus. Nous sommes satisfaits des résultats obtenus et considérons que notre service de messagerie instantanée répond aux objectifs fixés au départ.

Nous avons également identifié certaines possibilités d'amélioration pour notre projet. Par exemple, nous pourrions ajouter des fonctionnalités supplémentaires telles que l'envoi de messages privés entre utilisateurs, la gestion des erreurs lors de la connexion et de la déconnexion, ainsi que l'amélioration de l'interface utilisateur.

En somme, ce projet nous a permis d'approfondir nos connaissances en systèmes d'exploitation et de développer des compétences pratiques en programmation. Nous sommes fiers du travail accompli et confiants dans notre capacité à relever de nouveaux défis dans ce domaine.