Universite de Montpellier

Structures, Protocoles, Objets IPC et Threads

9 décembre 2020

1 Structures de Données

1.1 Structure Ressource :

Une structure qui permet de stocker un entier représentant le nombre de CPU et un autre entier pour stocker les ressources de mémoire.

1.2 Structure Site :

Cette structure de données permet de stocker les informations d'un site donné (une ville) :

— Une chaîne de caractères pour stocker le nom de la ville ou du site.
— Une instance de la structure ressource qui représente les ressources disponibles dans ce site (CPU, memory).

1.3 Structure Cloud :

Une structure qui représente un ensemble de sites.(le système de réservations).

1.4 Structure Commande :

Une structure représentant une commande saisie par l'utilisateur :
— Un type cmd_t prédéfini permettant de stocker le type de la commande.
— Deux tableaux d'entiers pour stocker les entiers saisis de différents sites.
— Un tableaux de chaînes de caractères pour stocker les noms des sites saisis.
— Un entier qui permet de stocker le nombre de sites saisis.

2 Les objest IPC

2.1 Côté Serveur

2.1.1 Segment de mémoire partagé :

— Une mémoire partagée entre les processus du serveur permettant de stocker sous la forme d'une chaîne de caractères le cloud lu à partir d'un fichier.
— un segment de mémoire partagé (un entier) pour stocker le nombre de threads concurrents du serveur et ses fils.
— un segment de mémoire partagé (un entier) pour stocker le nombre de threads qui sont arrivés au point de rendez vous.

2.1.2 Les sémaphores :

Un tableau constitué de quatre sémaphores :
— Un premier sémaphore protégeant la mémoire partagée des ressources des accès concurrents (un verrou).
— Un deuxième pour envoyer un signal lors d'une modification de l'état du système de réservations (le Cloud).
— Un troisième protégeant l'accès aux deux mémoires partagées qui permettent de mettre en oeuvre un système de rendez vous entre threads.
— Le dernier pour envoyer un signal lors que tous les threads sont arrivés au point de rendez vous.

2.2 Côté Client

2.2.1 Les sémaphores :

Un tableau de sémaphores contenant un seul sémaphore, son rôle est de mettre le client en attente lorsque son serveur est en attente et le déverrouiller dès que sa demande soit satisfaite.

3 Les Protocoles d'échange

3.1 Entre le client et le serveur

Le client construit,suite à la saisie de l'utilisateur, une structure **commande** et l'envoi au serveur en utilisant la fonction **sendCommandeTCP()** et le serveur la reçois en utilisant la fonction **recvCommandeTCP()**. Les fonctions fonctionnent ainsi :

- le type de la commande est d'abord envoyer sous forme déntier.
- puis viens le nombre de serveurs qu'il y a dans la commande.
- Enfin, à la suite, le nom du serveur, le nombre de CPU et de mémoire demander sont envoyés.

3.2 Entre le serveur et le client

3.2.1 Entre le thread principal du serveur et le thread du client

Le thread principal du serveur reste en écoute des demandes du client. Une fois la commande reçue (**recvCommandeTCP()**). Le serveur traite la demande et signal toute modification aux autres serveur. Puis envoient la réponse au thread du client qui la reçoit et l'affiche.

3.2.2 Entre le thread du serveur et le thread du client

Lorscequ'un signal de modification de l'état de la ressource est reçu, le thread envoie le nouvel état au thread de son client en utilisant la fonction **sendTCP()**. Cette dernière fonctionne ainsi :

- La taille du message est envoyé.
- puis le message lui mme.

4 Processus et threads

4.1 Côté Serveur

4.1.1 Le Serveur Principal et son threads :

Le serveur principal se charge d'initialiser les objets IPC et de créer son thread qui se charge de récuperer à chaque modification le nouvel état du système pour que le serveur principal soit toujours à jour.

4.1.2 Le Serveur Fils et son threads :

Le serveur fils se charge de créer en premier son thread et ensuite attendre des commandes de la part de son client dédié, il vérifie et exécute la commande envoyée par son client, ensuite en envoyant un signal via un sémaphore déclenche son thread (**wait_thread**) et les threads de tous les serveurs fils (qui sont en attente d'un signal), pour qu'à leurs tour, ils envoient le nouvel état du cloud à leurs du client(chaque thread d'un serveur fils envoie le nouvel état du cloud à son client dédié).

4.2 Côté Client

Le thread principal demande une saisie d'une commande à l'utilisateur pour l'envoyé à son serveur dédié. Le thread **listen_modif()** en boucle attends de recevoir un message de la part du thread de son serveur dédié ou le serveur dédié, que ça le cloud ou la réponse à une demande du thread principal. Ce thread affiche chaque message reçu sur la sortie standard.