Séance 6
Architecture client/serveur

Processus vs Threads

Exécution en parallèle des processus ou des threads :

- Un seul processus (thread) est exécuté par l'unité centrale à un instant donné.
- Temps partagé : le noyau Linux (module scheduler – ordonnanceur) attribue l'unité centrale à chaque processus (thread) successivement une brève durée de temps.
- Un appel système est atomique : non interruptible par l'ordonnanceur pour donner la main à un autre processus (thread).

Comparaison multithread / multiprocessus :

- Création d'un thread et changement de contexte d'un thread à l'autre plus rapide que pour un processus.
- Les threads partagent des données, les zones de données des processus sont séparées.
- La fin brutale d'un thread entraine la fin du processus et donc de tous les threads, les exécutions des processus sont isolées les unes des autres.

Architecture client / serveur

Serveur :

- Possède les données.
- A la puissance de calcul.
- Accède au matériel.
- Réalise les traitements associées aux requêtes des clients.
- S'exécute en arrière-plan.

Client :

- A l'interface utilisateur.
- Soumet des requêtes au serveur.
- Peut être léger.
- Peut être sur une autre machine.

Serveur monothread et multithread

Serveur monothread (TP3) : une session client à la fois

- Un seul thread (thread principal).
- Attend les connexions des clients (accept).
- A chaque connexion d'un client : démarre une session d'échange avec le client.
- A la fin de la session d'échange avec client, le serveur peut prendre la connexion du client suivant.

Serveur multithread (TP5) : plusieurs sessions client en parallèle

- Le thread principal attend les connexions de clients (accept).
- A chaque connexion d'un client : un thread (un worker) effectue la session d'échange avec le client.

Serveur à workers dynamiques

Modèle dynamique (TP5) = un thread est créé à chaque connexion d'un client.
Fin du thread à fin de la session d'échange.

Avantage : on a juste le nombre de threads nécessaires.
Inconvénient : on n'est pas sûr de pouvoir satisfaire la demande (si on a atteint le nb max de threads du système).

Serveur à workers statiques

Modèle statique : les workers (threads) sont tous créés au lancement du serveur.

- On fixe un nombre de workers, donc un nombre max. de connexions clients.
- Les workers sont en attente, le thread principal en réveille un lorsqu'il y a une session client à gérer.
- A la fin de la session client le worker se remet en attente.

Avantage : on sait dès le démarrage du serveur si on peut assurer le service.
Inconvénient : on a plus de threads que nécessaire et il faut gérer la mise en attente et le réveil des workers.