Les processus Windows

Différences entre threads légers et threads lourds

Moyen mnémotechnique :

- Un processus lourd, c'est chiant : pas de variables communes : on est obligé de passer par les variables partagées -> zone mémoire séparée
- Un processus léger, ça gagne de la place en mémoire : tous les processus légers partagent la mémoire du père -> zone mémoire commune (+ locale à chaque fils…)

Aujourd'hui, nous allons voir comment faire des threads légers grâce à Windows.

Prise de contact

Le principe n'est pas tout à fait le même… Alors qu'avec Linux, il faut utiliser `fork()` pour dupliquer le processus, avec Windows, il faut créer une fonction (ce que devra faire le fils), puis lancer cette fonction avec la fonction **`CreateThread`**.

Tout d'abord la fonction :

```
DWORD WINAPI maFonction(LPVOID param){
	cout<<"wait..."<<endl;
	Sleep(2000);
	cout<<"bye..."<<endl;
	return 0;
}
```

Le type de la fonction est un peu spécial : `DWORD` c'est pour dire un `double`, et `WINAPI` pour dire que ce sera un thread…
Ensuite, dans le main il est possible de lancer cette fonction ainsi :

```
	CreateThread(NULL,NULL, maFonction,NULL,NULL,NULL);
```

Les différentes valeurs des paramètres sont à découvrir de vous-même…

Exercice 1

Reprendre l'exercice du TP1 où il fallait faire un père qui dise 10 fois « Ping », et un fils qui dise 10 fois « pong ». Exécuter plusieurs fois l'exercice pour s'assurer que l'ordre d'apparition des « Ping » et des « Pong » n'est pas toujours le même. Cela prouve que quand on utilise les threads, on ne peut pas prédire l'ordre d'exécution !

Exercice 2

Pour partager de l'information, c'est plus simple. Alors que sous linux, il fallait utiliser de la mémoire partagée, sous Windows, c'est plus simple.

Première solution : variable globale

Expliquer ce que c'est qu'une variable globale.

Créez un programme dans lequel vous créerez une variable globale de type tableau de 5 cases de type **`int`**. Le père essaye de mettre la combinaison suivante dans la variable globale : 1 2 3 4 5. Le fils essaye quant à lui de mettre la combinaison suivante : 5 4 3 2 1.C'est le même principe que lors du TP précédent : tant que la combinaison est différente de 5 4 3 2 1, le père reste dans une boucle tant que ; pour le fils, tant que la combinaison est différente de 1 2 3 4 5, il reste dans une boucle.

Seconde solution : Les sockets

Sans la tester, expliquer comment on pourrait faire pour passer à un processus X une valeur Y. Faites le schéma de connexion complet (en utilisant par exemple un diagramme de séquence !)

TP MSN

On s'est rendu compte lors du TP sur les sockets que notre MSN maison n'était pas très fonctionnel. En effet, notre précédente solution imposait un ordre dans la conversation de type question/réponse. Nous allons maintenant améliorer ça en utilisant les threads (processus…)

Voila le principe, à vous de trouver la solution algorithmique :

Lors de la connexion au serveur, il faut créer un fils qui s'occupera d'envoyer ce que l'utilisateur tapera, et un autre fils qui s'occupera de récupérer ce que l'autre utilisateur enverra.

Testez votre solution…

Sections critiques

Comme vous avez du le constater, il y a des problèmes… En effet, si pendant que vous écrivez l'autre vous envoie un message, vous ne serez plus où vous en êtes. Il va donc falloir utiliser les sémaphores. En C++ sous Windows, il existe le type `CRITICAL_SECTION` qui est la version Microsoft des sémaphores. Donc, je vous conseille de faire une variable globale de ce type (que vous appellerez `Semaphore`), et dans le main de l'initialiser ainsi :

```
InitializeCriticalSection(&Semaphore);
```

Grace à cette initialisation, le sémaphore est initialisé à 1. Les fonctions P et V sont les suivantes :

```
EnterCriticalSection(&Semaphore); -> P(S)
```

```
LeaveCriticalSection(&Semaphore); -> V(S)
```

Donc nous allons décider que nous sommes en zone critique quand on affiche ou quand on saisi des valeurs (en gros quand on utilise l'écran…). Utilisez ces fonctions pour protéger cette zone et empêcher qu'il y ait deux processus qui utilisent l'écran.