Examen SEM 6 décembre 2018

Il s'agit de simuler des joueurs qui se lancent un unique ballon les uns aux autres. A la fin de l'examen, chacun doit envoyer [1] un mail [2] avec en fichier attaché un répertoire [3] nommé avec prénom et nom et contenant pour chaque exercices les codes en C des programmes [4], ainsi que le cas échéant la réponse détaillée de l'exercice 6.

Dans chaque exercice, il est important que :

— Toute fonction susceptible d'émettre un jeton récupérable par la fonction `GetLastError()` doit être testée.
— La gestion de l'héritage des Handle doit être bien faite.
— Tous les Handle créés doivent être fermés dès qu'ils ne sont plus utilisés. Il convient cependant de prendre en compte qu'un Handle fermé ne peut plus être hérité et qu'un objet dont à un instant donné tous les handle associés sont fermés est un objet qui n'existe plus et qui ne peut donc plus être ouvert sans être à nouveau créé.

Exercice 1 *Ce premier programme simule* 10 *joueurs.*

Le programme `ex1` *réalise les tâches suivantes :*

— *Lancement de* 10 *unités d'exécutions* [5] *en informant chaque unité d'exécution de son numéro entre* 0 *et* 9.
— *Attente que l'utilisateur appuie sur une touche.*
— *Suppression des* 10 *unités d'exécutions.*

Chaque unité d'exécution fait une boucle infinie constituée des tâches suivantes :

— *Affichage du message* `joueur numéro` *suivi de son numéro.*
— *Attente d'une seconde.*

Exercice 2 *Le précédent programme est modifié de façon à simuler l'échange de ballon à travers un événement non-nommé : la réception du ballon est modélisé par le fait de se se mettre automatiquement en position non-signalée dès réception de l'événement, le fait d'envoyer le ballon est modélisé par le fait de mettre en position signalé l'événement.*

Le programme `ex2` *réalise les tâches suivantes :*

— *Création d'un événement non-nommé*
— *L'événement est signalé.*
— *Lancement de 10 unités d'exécutions en informant chaque unité d'exécution de son numéro entre* 0 *et* 9.
— *Attente que l'utilisateur appuie sur une touche.*
— *Suppression des 10 unités d'exécutions.*

Chaque unité d'exécution fait une boucle infinie constituée des tâches suivantes :

— *Attente de la réception de l'événement*
— *Affichage du message* `réception du ballon par le joueur numéro` *suivi de son numéro.*
— *Attente de* 100*ms.*
— *L'événement est remis en position signalée.*
— *Attente de* 100*ms.*

Exercice 3 *Le précédent programme est modifié de façon à simuler l'échange de ballon à travers un mutex non-nommé.*

Le programme `ex3` *réalise les tâches suivantes :*

— *Création d'un mutex non-nommé.*
— *Lancement de 10 unités d'exécutions en informant chaque unité d'exécution de son numéro entre* 0 *et* 9.
— *Attente que l'utilisateur appuie sur une touche.*
— *Suppression des 10 unités d'exécutions.*

Chaque unité d'exécution fait une boucle infinie constituée des tâches suivantes :

— *Attente de la prise en main du mutex*
— *Affichage du message* `réception du ballon par le joueur numéro` *suivi de son numéro.*
— *Attente de* 100*ms.*
— *Relachement du mutex.*
— *Attente de* 100*ms.*

1. à l'adresse : gabriel.dauphin@univ-paris13.fr
2. Le mail doit contenir la liste des exercices réalisés et le prénom et le nom.
3. Le répertoire doit être nommé avec le format suivant `prenom_nom`.
4. `ex1.c` pour le premier exercice, `ex2.c`, pour le deuxième exercice, `ex3.c` pour le troisième exercice, `ex4a.c` et `ex4b.c` pour le quatrième exercice et `ex5a.c` et `ex5b.c` pour le cinquième exercice. La première ligne de chaqu'un de ces fichiers doit contenir le prénom et le nom.
5. `thread`

Exercice 4 *Le précédent programme est modifié de façon à considérer 100 joueurs au lieu de 10 joueurs. Chaque dizaine de joueurs est regroupée en une entité correspondant à un processus. Le mutex est transmis aux différents processus par héritage.*

Le programme `ex4b` *réalise les tâches suivantes :*

- *Création d'un mutex non-nommé héritable* [6].
- *Lancement de 10 processus associés au programme* `ex4a` *avec en paramètre un chiffre entre* 0 *et* 9 *pour indiquer le numéro du processus et un nombre correspondant à la valeur du Handle du mutex, (pensez à l'héritage).*
- *Attente que l'utilisateur appuie sur une touche.*
- *Suppression des 10 processus.*

Le programme `ex4a` *réalise les tâches suivantes :*

- *Récupération du chiffre associé au processus et du Handle du Mutex.*
- *Lancement de 10 unités d'exécutions en informant chaque unité d'exécution de son chiffre entre* 0 *et* 9 *et l'informant aussi de son chiffre associé au processus et au Handle du Mutex.*

Chaque unité d'exécution fait une boucle infinie constituée des tâches suivantes :

- *Attente de la prise en main du mutex.*
- *Affichage du message* `réception du ballon par le joueur numéro` *suivi de son numéro entre* 0 *et* 99*, le premier chiffre de ce numéro indique le processus et le deuxième indique l'unité d'exécution.*
- *Attente de* 100*ms.*
- *Relachement du mutex.*
- *Attente de* 100*ms.*

Exercice 5 *Le précédent programme est modifié de façon à remplacer le fait de tuer les processus par le fait d'envoyer un signal nommé qui lors de leur réception par les différentes unités d'exécutions déclenchent l'arrêt de la boucle infinie. Chaque processus s'arrête lorsque les processus ou les unités d'exécutions lancés sont terminés.*

Le programme `ex5b` *réalise les tâches suivantes :*

- *Création d'un événement nommé* `"arret"`.
- *Création d'un mutex non-nommé héritable avec un attribut de sécurité transmis par adresse sur le premier argument de* `CreateMutex`.
- *Lancement de 10 processus associés au programme* `ex5a` *avec en paramètre un chiffre entre* 0 *et* 9 *pour indiquer le numéro du processus et un nombre correspondant à la valeur du Handle du mutex, (penser à l'héritage).*
- *Attente que l'utilisateur appuie sur une touche.*
- *L'événement nommé* `"arret"` *est mis en position signalé.*
- *Attente de la fin des processus lancés.*

Le programme `ex5a` *réalise les tâches suivantes :*

- *Récupération du chiffre associé au processus et du Handle du Mutex.*
- *Lancement de 10 unités d'exécutions en informant chaque unité d'exécution de son chiffre entre* 0 *et* 9 *et l'informant aussi de son chiffre associé au processus et au Handle du Mutex.*
- *Attente de la fin des unités d'exécutions lancées.*

Chaque unité d'exécution fait une boucle infinie constituée des tâches suivantes :

- *Récupération* [7] *du Handle de l'événement nommé* `"arret"`.
- *Attente* [8] *de la réception du signal ou de la prise en main du mutex.*
- *En cas de réception du signal nommé* `"arret"`*, sortie de la boucle infinie et fin d'exécution de l'unité d'exécution.*
- *Affichage du message* `réception du ballon par le joueur numéro` *suivi de son numéro entre* 0 *et* 99*, le premier chiffre de ce numéro indique le processus et le deuxième indique l'unité d'exécution.*
- *Attente de* 100*ms.*
- *Relachement du mutex.*
- *Attente de* 100*ms.*

Exercice 6 *Expliquez en détail dans le corps du mail, mais sans le réaliser, comment on pourrait faire en sorte que chaque unité d'exécution transmette les informations au processus qui a lancé les autres processus afin de pouvoir afficher le numéro à deux chiffres du joueur qui a le plus reçu le ballon et combien de fois il a reçu le ballon. Il existe naturellement plusieurs solutions.*

---

6. cela se fait avec un attribut de sécurité transmis par adresse sur le premier argument de `CreateMutex`.
7. avec `CreateEvent` ou `OpenEvent`.
8. Utilisation de `WaitForMultipleObject` en veillant à ce que le premier élément du tableau corresponde au Handle du signal nommé `"arret"`.