

Rapport Synchronisation des Processus

Berthet Laurent – Julien Bugnard

TIC-INFO L3

Introduction :

Le but de ce TP était de réaliser la simulation d'un Ordonnanceur. Il devait avoir cinq types d'ordonnancements différents.

I. Fonctionnement

1. Décomposition en classes

Nous avons choisi de créer une classe par type d'objet, c'est à dire que nous avons une classe Ordonnanceur, Tâche, Interface et Mutex qui est le sémaphore.

- Tâche

La classe Tâche contient des entiers correspondant à son temps d'exécution, à son temps de sortie de zone critique, à son type... Elle contient aussi des booléens concernant son état qui évolue pendant l'exécution du programme comme savoir si la tâche est élue, bloquée ou finie.

Une tâche a aussi un compteur personnel quelle incrémente à chaque fois quelle est élue. Cette classe étend la classe thread.

- Ordonnanceur

Cette classe contient un ArrayList de Tâche et une file d'attente qui lui permet de choisir une tâche. Elle a aussi un entier correspondant au type de fonctionnement et un quantum de temps lorsque le type de fonctionnement est le « Tourniquet ». L'ArrayList historique permet d'afficher l'historique d'attribution des tâches.

Une tâche est ajoutée à la file d'attente lorsque le temps d'arrivée de celle ci est égal à la valeur du compteur de l'Ordonnanceur.

La méthode ChoisirTache() choisit une tâche parmi celles qui sont dans la file d'attente, en fonction de son type d'ordonnement.

L'Ordonnanceur s'arrête lorsque toutes les tâches de la liste de tâches sont finies. Cette classe étend de la classe thread.

- Interface

Cette classe permet l'affichage de la valeur du compteur partagé ainsi que l'état des tâches à tous les tops de l'Ordonnanceur.

- Mutex

Cette classe est un sémaphore d'exclusion mutuelle c'est à dire que deux tâches ne peuvent pas prendre le Mutex en même temps. Dans sa zone critique, il y a l'incrémention du compteur partagé des tâches.

2. Utilisation des Threads

Nous avons choisit d'utiliser des threads dans ce TP car nous pensons que la simulation de l'Ordonnanceur est plus réaliste. En effet avec des threads, les tâches s'exécutent en parallèle de l'Ordonnanceur. Ce n'est pas l'Ordonnanceur qui simule le déroulement des tâches.

L'Ordonnanceur gère tous les threads des tâches, c'est à dire qu'il peut suspendre l'exécution d'une tâche et relancer son exécution suivant son type d'ordonnement.

3. Types d'ordonnement

- Premier arrive premier servi (First Come First Serve)

Lorsque l'ordonnanceur est de ce type, toutes les tâches sont exécutées en une seule fois. Lorsqu'une tâche se termine, la tâche qui est arrivée après est exécutée et ainsi de suite jusqu'à que la file d'attente soit vide.

- Le plus petit travail en premier (Short Job First)

Ce type d'ordonnement permet aux tâches qui ont un court temps d'exécution de se dérouler en priorité. Par contre une longue tâche ne sera exécuté que quand il n'y aura plus de petite tâche à exécuter. Ce type d'ordonnement peut amener à une famine.

Pour choisir une tâche, l'ordonnanceur parcourt sa file d'attente et sélectionne celle qui a le plus petit temps d'exécution. La tâche n'est pas interrompue lors de son exécution. Lorsqu'elle est finie, la tâche qui a le « nouveau » plus petit temps d'exécution est élue.

- Le plus court temps restant (Shortest Remaining Time)

Dans notre programme, ce type d'ordonnement aura le même comportement que l'ordonnement de type Short Job First. Une tâche élue n'est pas interrompue jusqu'à la fin de son exécution.

- Tourniquet (Round robin)

Ce type d'ordonnement nécessite un quantum de temps. L'ordonnanceur lance une tâche pendant la durée du quantum de temps, à la fin de ce quantum c'est une nouvelle tâche qui est exécutée. Si pendant le quantum, la tâche se termine, une autre tâche est exécutée pendant le reste du quantum.

- Priorité

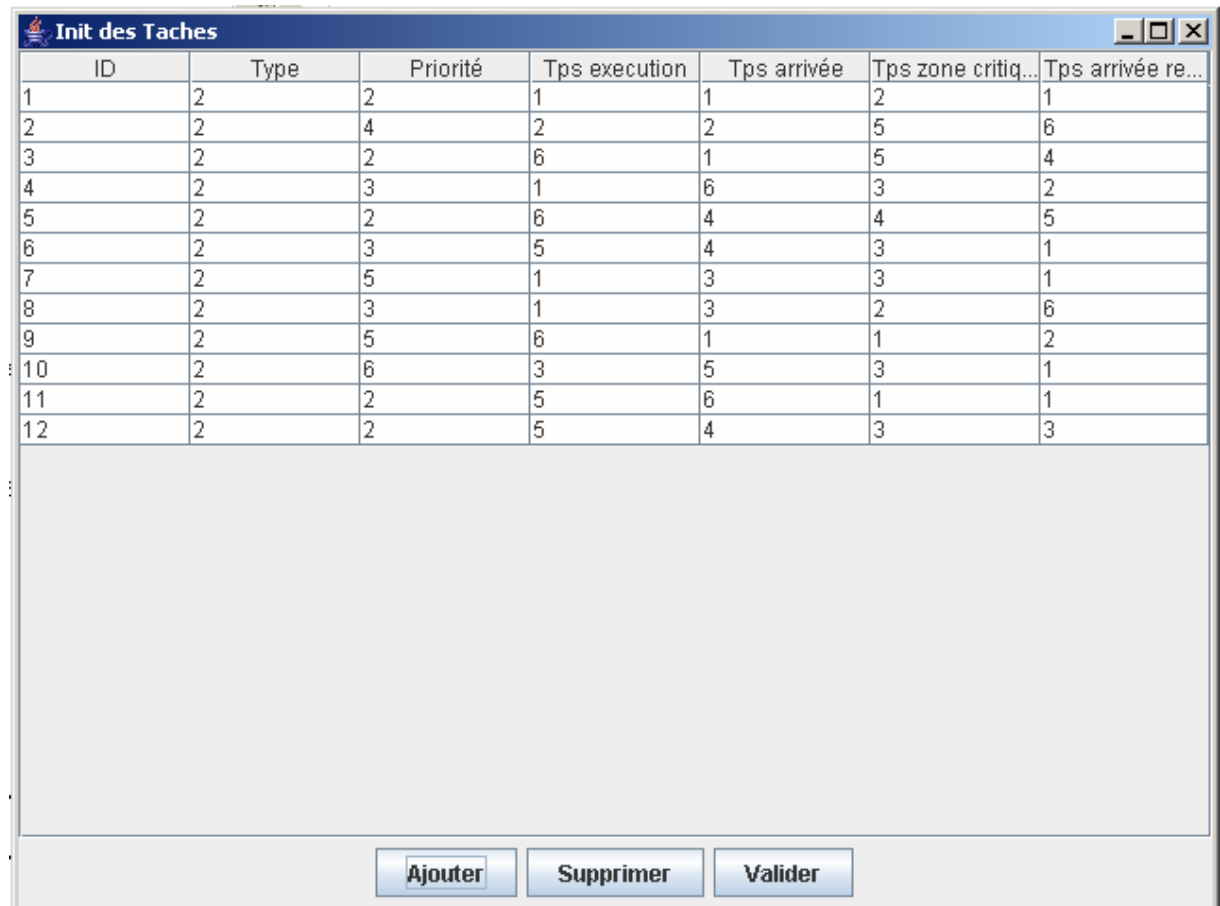
Une tâche peut être interrompue si une autre arrive avec une priorité plus haute. A chaque cycle d'horloge, l'ordonnanceur vérifie la file d'attente pour que la tâche la plus prioritaire soit exécutée en premier

II. Utilisation

Nous allons, dans ce paragraphe vous expliquer comment utiliser notre programme.

1. L'initialisation des tâches :

Lorsque vous lancez le programme, vous vous retrouvez sur la fenêtre d'initialisation des tâches.



ID	Type	Priorité	Tps execution	Tps arrivée	Tps zone critiq...	Tps arrivée re...
1	2	2	1	1	2	1
2	2	4	2	2	5	6
3	2	2	6	1	5	4
4	2	3	1	6	3	2
5	2	2	6	4	4	5
6	2	3	5	4	3	1
7	2	5	1	3	3	1
8	2	3	1	3	2	6
9	2	5	6	1	1	2
10	2	6	3	5	3	1
11	2	2	5	6	1	1
12	2	2	5	4	3	3

Dans cette fenêtre vous pouvez changer les paramètres de chaque tâche. Pour cela il vous suffit de « double-cliquer » dans la case que vous voulez changer et de changer les valeurs. Le champ ID n'est pas éditable car vous n'avez pas le droit de changer le numéro de la tâche. Pour le champ type une liste déroulante vous proposera de choisir le type de la tâche entre 1 et 2.

- Type 1 : La tâche ne touche pas au compteur partagé des tâches, elle s'exécute sans faire attention aux autres tâches. Elle incrémente juste son compteur.
- Type 2 : La tâche incrémente un compteur partagé entre toutes les tâches et son compteur personnel.

Tous les autres champs sont composés des textfield dans lesquels vous pouvez entrer les valeurs désirées. (Nous n'avons pas géré le type de caractère que vous entrez mais nous pensons que ce n'est pas le but du TP).

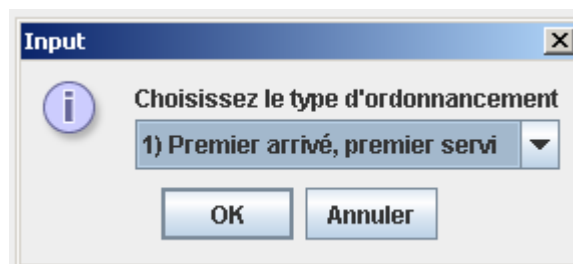
Pour ajouter une nouvelle tâche à la liste, il vous suffit de cliquer sur le bouton « ajouter » afin de voir apparaître la tâche dans la liste. Les tâches sont ajoutées avec des valeurs aléatoires dans les différents champs. Cela nous permet pour tester notre programme, de ne pas avoir à remplir tous les champs et de ne pas perdre trop de temps.

Pour supprimer une tâche de la liste, il vous suffit de sélectionner la tâche voulue et de cliquer sur le bouton « Supprimer » ; une fenêtre de confirmation apparaîtra où vous devrez confirmer la suppression.

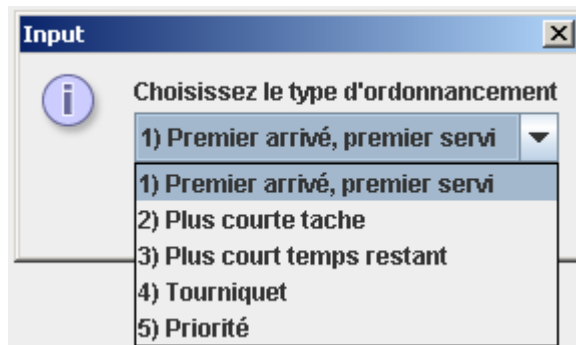
Une fois toutes les tâches configurées, vous pouvez cliquer sur valider afin de passer à l'étape suivante du lancement du programme.

2. Initialisation de l'ordonnanceur :

Vous arrivez maintenant sur la fenêtre de configuration de l'ordonnanceur :



La fenêtre de configuration de l'ordonnanceur se présente sous la forme d'une liste déroulante où 5 choix vous sont proposés :



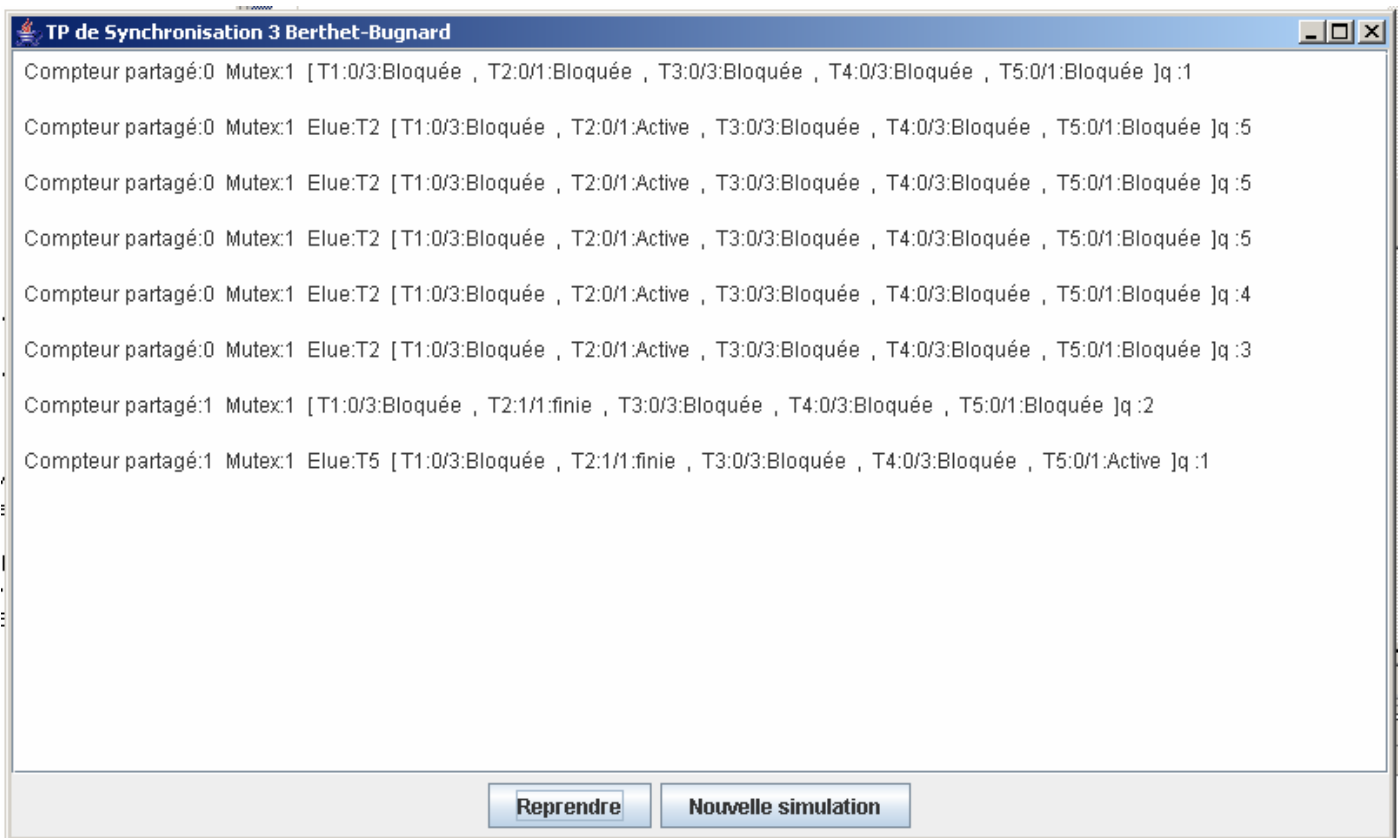
- Premier arrivé, premier servi.
- Plus courte tâche.
- Plus court temps restant.
- Tourniquet.
- Priorité.

Je vous renvoie à l'explication des ordonnancements pour savoir quelles sont les différences entre ceux-ci.

Si vous choisissez l'ordonnement de type tourniquet, vous devrez rentrer nombre de cycles que durera le quantum de temps du tourniquet.

3. Fenêtre principale du programme :

Une fois l'ordonnancement choisi, vous allez arriver sur la fenêtre principale du programme :



Cette fenêtre montre le déroulement de l'exécution du programme.

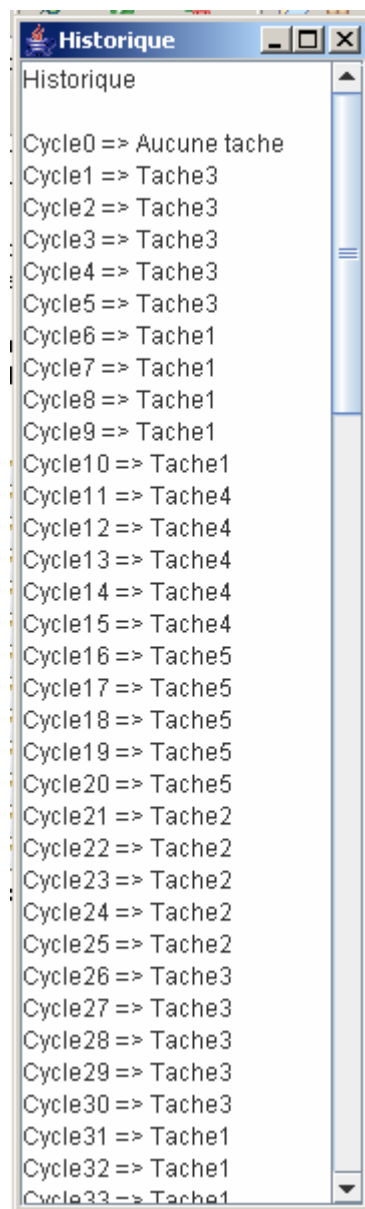
Les informations affichées sont :

- La valeur du compteur partagé par toutes les taches.
- La valeur du mutex (qui est le sémaphore d'exclusion mutuelle pour l'accès au compteur partagé).
- La tache élue lorsqu'il y en a une.
- Un tableau qui affiche les informations de chaque tache. Les informations affichées pour chaque tache sont :
 - Le nom de la tache. (T1,T2,)
 - Le compteur perso de la tache ainsi que son nombre de cycles d'exécution. (Exemple : 3/5 signifie que la tache a déjà fait 3 cycles entiers et qu'elle doit en faire 5 pour se terminer).
 - L'état de la tâche (Bloquée, Elue ou Finie)
- Le quantum de temps. (Lorsque le type d'ordonnancement n'est le tourniquet, le quantum reste toujours à 1).

Remarque sur la fenêtre principale :

Les taches sont toutes lancées en parallèle. L'ordonnanceur affiche toutes les secondes l'état des différents éléments du programme. Il se peut qu'aucun élément ne change car les taches prennent plus d'une seconde pour faire un cycle complet. Le temps que la tache passe dans la zone critique varie entre une seconde et le temps donné à l'initialisation (en seconde) et le temps avant de réserver le mutex varie entre 0 et le temps donné dans lors de l'initialisation (aussi en seconde). Cela veut dire qu'une tache peut mettre plus de 10 secondes pour effectuer un cycle et donc incrémenter le compteur partagé et son compteur personnel.

Une que toutes les taches sont finies, le programme s'arrête en laissant la fenêtre principale visible et affiche un historique de toutes les taches qui étaient exécutées à chaque cycle.



Conclusion :

Nous pensons avoir répondu correctement aux attentes qui étaient dans le TP. Nous avons essayé de réaliser l'ordonnanceur le plus fiable possible. En effet, lors de nos tests le programme n'a jamais présenté d'inter-blocage entre les tâches et il n'y a jamais eu deux tâches élues en même temps.

Pour que notre ordonnanceur soit plus réaliste, il aurait fallu que l'on puisse ajouter de nouvelles tâches pendant l'exécution du programme.