Rapport de LO41
Système de gestion des poubelles

Donatien RABILLER
Quentin SCHULZ

18 Juin 2014

Table des matières

Chapitre 1

Introduction

L'objectif de ce projet est d'écrire un programme simulant la gestion des déchets dans une ville d'un genre nouveau. Pour cela, il s'agit de proposer une solution s'articulant autour de threads. Leur présence impose le partage de ressources et donc de protection des données et de synchronisation. L'utilisation de mutexes est donc nécessaire tout comme les signaux pour pouvoir notifier un thread d'une action à entreprendre.

Chapitre 2

Présentation du problème

On définit un client comme étant un foyer. Suivant le nombre de personnes dans le foyer, le client a le droit de déposer une poubelle plus ou moins volumineuse. Il est donné au client deux possibilités de remplissage des poubelles communes: par clé, qui donne accès à une poubelle commune n'acceptant que des poubelles de 30l, ou par bac, la taille acceptée dépendant du nombre de personnes dans le foyer. Il possède aussi le mode "bac et clé" qui offre les deux possibilités précédentes au choix et il lui est donné l'impensable alternative de déposer illégalement ses poubelles dans le cas où les poubelles communes proposées sont remplies ou n'acceptent plus de poubelles de sa taille.

Quand une poubelle est pleine ou qu'un dépot sauvage est effectué, le centre de tri vient vider la poubelle et nettoyer les dépots sauvages. A côté de cela, le service quotidien est toujours assuré (i.e. tous les jours, le centre de tri va vider les poubelles de toute la ville, qu'elles soient pleines ou non).

Chapitre 3

Principe simplifié du programme

On crée n clients qui déposeront à une fréquence attribuée aléatoirement des poubelles de la taille du foyer qui leur a également été attribuée aléatoirement. S'il n'y a plus de poubelles suffisamment libres pour accueillir la poubelle du client, cette dernière est déposée sauvagement à côté des poubelles communes. A chaque client est affecté le point de tri qui lui est le plus proche.

On crée également m points de tri qui auront chacun un nombre défini de poubelles communes. Dès qu'une poubelle commune dans un point de tri est pleine, ou qu'un dépot sauvage a été effectué, le point de tri demande au centre de tri de venir vider l'ensemble de ses poubelles.

Un unique centre de tri est créé pour gérer l'ensemble des points de tri de la ville. Dès qu'il reçoit l'ordre de vider les poubelles par un de ses points de tri, il s'exécute. Il va également vider les points de tri quotidiennement.

Chapitre 4

Les structures

5 structures sont utilisées en plus de 2 énumérations:
- TrashType
- Mode
- Client
- TriPoint
- TrashBag
- TrashBin
- TriCenter

4.1 TrashType

Une énumération permettant de définir le type de déchets. Cette énumération comprend:
- WASTE
- GLASS
- PAPER
- ANY

ANY n'est présent que pour spécifier le type des dépôts sauvages.

4.2 Mode

Une énumération permettant de définir les possibilités données au client pour déposer ses poubelles. Cette énumération comprend:
- KEY
- BAC
- KEY_BAC

4.3 Client

Une structure définissant un foyer. Elle contient les informations suivantes:

- mode: le mode utilisé pour déposer les poubelles
- trash: la poubelle que va déposer le client
- x, y: les coordonnées du client
- point: le point de tri (TriPoint) associé à ce client
- nbPerson: le nombre de personnes dans le foyer
- period: la période de création des poubelles (création d'une poubelle toutes les "period"-secondes)

N structures seront donc créées et disséminées dans autant de threads.

4.4 TriPoint

Une structure définissant le point de tri. Elle contient les informations suivantes:

- bins: un tableau dynamique des poubelles communes qu'il contient
- nbBins: le nombre de poubelles communes contenues
- free: la "poubelle" commune permettant de stocker les dépots sauvages
- x, y: les coordonnées du point de tri
- mutex: le mutex permettant de protéger les données du point de tri

M structures seront donc créées et disséminées dans autant de threads.

4.5 TrashBag

Une structure définissant une poubelle d'un client. Elle contient les informations suivantes:

- volume: le volume de la poubelle en question
- type: le type de déchets contenus

4.6 TrashBin

Une structure définissant une poubelle commune. Elle contient les informations suivantes:

- volume: le volume maximal de la poubelle commune
- volume_max_trash_bag: le volume maximal accepté pour une poubelle client
- current_volume: le volume actuellement occupé dans la poubelle commune

– type: le type de déchets acceptés par la poubelle commune
– mode: le mode d'utilisation proposée par la poubelle commune
– mutex: le mutex permettant de protéger les données de la poubelle commune (le volume)

4.7 TriCenter

Une structure définissant le centre de tri. Elle contient les informations suivantes:

– x, y: les coordonnées du centre de tri
– triPoints: l'ensemble des points de tri dont il est le responsable
– nbTriPoints: le nombre de points de tri dont il est le responsable
– current_point: le point de tri entrain d'être vidé

Une unique structure sera créée et associée à un thread.

Chapitre 5

Méthodes choisies

Pour pouvoir mener à bien ce projet, nous avons donc utilisé des mutexes et un signal, voici leurs domaines d'application.

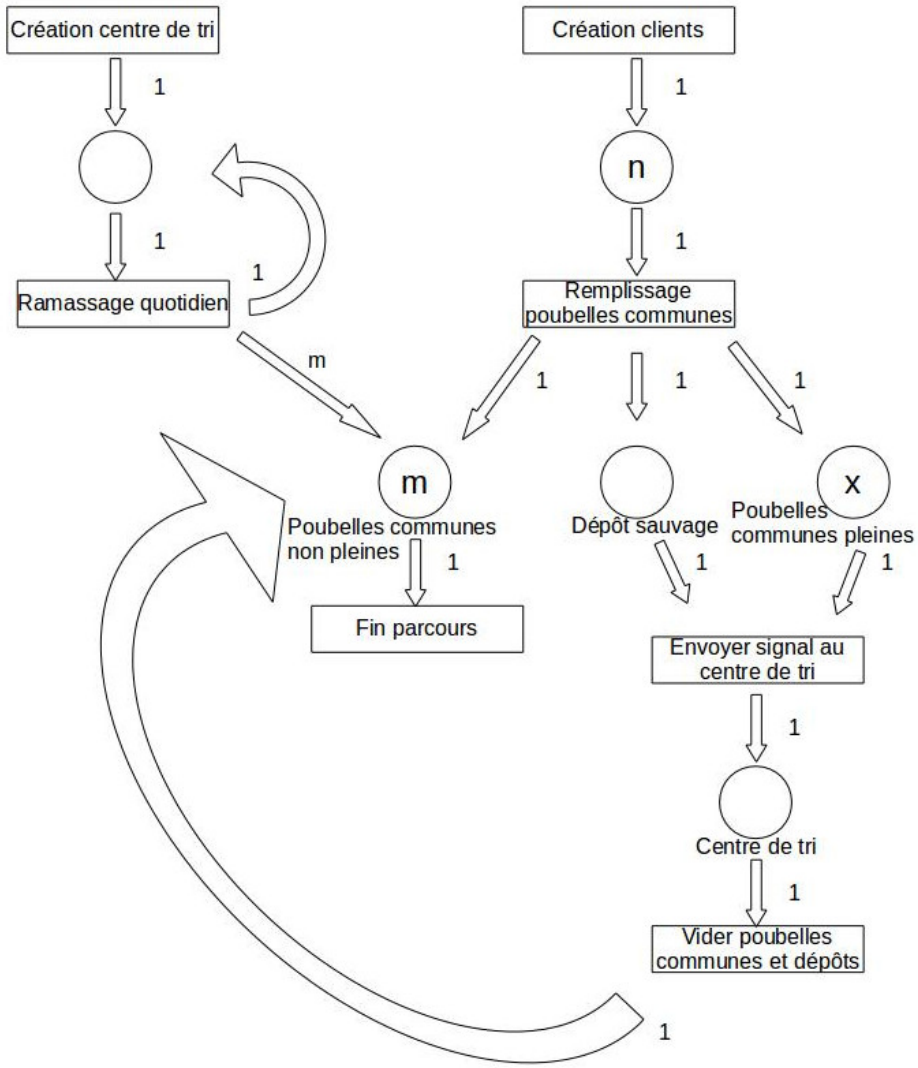
Nous avons un mutex pour protéger chaque poubelle commune. En effet, il faut éviter que deux clients puissent déposer des poubelles en même temps ou qu'un client dépose une poubelle alors que la poubelle commune est en train d'être vidée par le centre de tri. Les accès sont donc protégés en écriture et en lecture.

Nous avons utilisé un signal pour notifier le centre de tri quand une poubelle commune est pleine ou qu'un dépôt sauvage a été effectué. A ce moment là, le centre de tri va vider l'ensemble des poubelles communes appartenant au point de tri qui a envoyé le signal.

Etant donné que l'on n'utilise que des threads, l'utilisation de mémoire partagée n'est donc pas nécessaire.

Chapitre 6

Réseau de Petri



Chapitre 7

Conclusion

Nous avons pu voir en créant ce projet à partir de rien, les étapes de création d'un projet mêlant threads, mutexes, signaux, variables globales et structures. Cela nous a permis de nous former sur les situations d'interblocage, de synchronisation des threads et de protection des données ainsi que de l'utilité des signaux. En effet, nous avons du réécrire une bonne partie de notre solution initiale devant un interblocage inexpliquable. Il nous a de plus permis d'ajouter une part d'application de nos connaissances sur les threads et leur espace mémoire partagé.

Bien entendu, il y a des améliorations possibles dans notre projet. On aurait pu prendre en compte la distance entre le point de tri et le client, le point de tri et le centre de tri pour pouvoir attribuer un temps de parcours relatif pour aller déposer une poubelle ou aller la vider. Il serait aussi plus judicieux de placer les points de tri pour qu'ils aient tous le même nombre de clients et ainsi homogénéiser le remplissage des points de tri. On aurait pu également simuler le déplacement des camions éboueurs pour aller vider les points de tri. Cette solution a été envisagée au début mais abandonnée devant l'inexpliquable interblocage auquel nous faisions face.