HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Prof. Yann Thoma

Laboratoire de Programmation Concurrente

semestre printemps 2015 - 2016

Gestion de ressources par moniteurs

Temps à disposition : 6 périodes (travail débutant en semaine 11)

1 Objectif pédagogique

Réaliser la modélisation d'un problème concurrent par moniteur.

2 Énoncé du problème

Une ville comprend un ensemble de sites permettant à ses citadins d'aller de site en site en vélo. Chaque site comporte un nombre fixe de bornes et chaque borne ne peut attacher qu'un seul vélo à la fois. Les habitants prennent un vélo d'un site, se rendent à un autre site puis arriment le vélo à une borne du site pour un prochain usage.

De manière succincte, un habitant a le comportement suivant :

Boucle infinie
1. Attendre qu'un vélo du site i devienne disponible et le prendre.
2. Aller au site $j \neq i$.
3. Attendre qu'une borne du site j devienne libre et libérer son vélo.
4. Faire une activité près du site j.
5. $i \leftarrow j$

Fin de la boucle

Quand plusieurs habitants entrent en conflit pour une attente quelconque, par exemple quand ils attendent qu'une borne se libère ou qu'un vélo deviennent disponible, on supposera que ces habitants se servent selon leur ordre d'arrivée. Initialement, tous les habitants sont sur un site. Les sites de déplacement sont ensuite choisis de manière aléatoire.

Cette ville a aussi une équipe d'employés qui répartissent au mieux les vélos parmi les sites afin de les ramener aux destinations les moins populaires et aussi de laisser des bornes vacantes (éviter la surcharge d'un site). Cette équipe circule de site en site de manière continue à l'aide d'une camionnette pouvant transporter les vélos.

3 Cahier des charges

Réalisez le programme énoncé avec les hypothèses et les contraintes ci-dessous :

— Le nombre de sites, $S \geq 2$, et le nombre d'habitants sont constants durant l'exécution du programme. Ces nombres devront être passés en argument au programme, en donnant d'abord le nombre de sites puis le nombre d'habitants.

— Chaque habitant est modélisé par un thread.

— Le temps que les habitants prennent pour se déplacer entre deux sites, ainsi que la durée de l'activité qu'ils réalisent à un site, seront modélisés par des temps aléatoires.

— Le nombre de bornes par site, $B \geq 4$, est aussi obtenu par le lancement en ligne de commande (troisième argument) et demeure invariable durant toute la durée du programme.

— La camionnette de maintenance peut contenir au plus 4 vélos.

— Le nombre de vélos, V, doit aussi être passé en ligne de commande (quatrième argument) et doit respecter $V \geq S(B-2)+3$. Initialement, chaque site contiendra $B-2$ vélos, et le solde sera déposé dans le dépôt d'où part et revient l'équipe de maintenance.

— L'équipe de maintenance est modélisée par un thread.

— En notant par D le nombre de vélos au niveau du dépôt et par V_i le nombre de vélos au site i, l'équipe de maintenance aura le comportement suivant :

Boucle infinie
1. Mettre $a = min(2, D)$ vélos dans la camionnette
2. Pour $i = 1$ à S faire
   2a. Si $V_i > B-2$ alors
       Prendre $c = min(V_i - (B-2), 4-a)$ et les mettre dans la camionnette.
       $a \leftarrow a + c$
   2b. Si $V_i < B-2$ alors
       Laisser $c = min((B-2) - V_i, a)$
       $a \leftarrow a - c$
3. Vider la camionnette $D \leftarrow D + a$.
4. Faire une pause.

Fin de la boucle

Le temps requis pour se déplacer entre les sites et aussi entre le dépôt et les sites sera un temps aléatoire, mais la durée de la pause de l'équipe sera constante. Afin de simplifier le problème, on supposera que le chargement des vélos dans la camionnette et leur déchargement se fait de manière instantanée ($= 0$).

— Les vélos peuvent disparaître et aussi être remplacés. Pour simuler ce comportement, il devrait être possible d'incrémenter le nombre de vélos en réserve dans le dépôt et d'éliminer des vélos libres des différents sites. L'ajout et l'élimination devront être saisie par clavier par l'utilisateur du programme et à n'importe quel moment. Il est suffisant d'ajouter et d'éliminer un seul vélo à fois. Remarquons que ces interventions modifient dynamiquement des variables partagées et nécessiteront des synchronisations supplémentaires. La boucle de gestion des événements de l'application graphique doit être gérée par le programme principal (main()). La saisie clavier devra donc être gérée par un thread supplémentaire.

— Vos threads ne devront comporter aucune attente active, et toutes les synchronisations devront être faites par des moniteurs.

— Un squelette de programme vous est fourni. Vous y trouverez des exemples sur l'usage de l'interface graphique. Attention, il ne s'agit bien que d'un exemple sans fonctionnalité particulière.

4 Travail à rendre

— Les modalités du rendu se trouvent dans les consignes qui vous ont été distribuées.

— Comme pour les laboratoires précédents, la description de l'implémentation, ses différentes étapes, la manière dont vous avez vérifié son fonctionnement et toute autre information pertinente doivent figurer dans les fichiers sources rendus. Aucun rapport n'est demandé.

— Inspirez-vous du barème de correction pour savoir là où il faut mettre votre effort.

— Vous pouvez travailler en équipe de deux personnes au plus.

5 Barème de correction

Conception, conformité au cahier des charges, structure et simplicité	30%
Exécution et fonctionnement	20%
Codage	20%
Documentation de votre solution	20%
Commentaires au niveau du code	10%