Prof. Yann Thoma

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

# Laboratoire de programmation concurrente (PCO)

semestre printemps 2014 - 2015

## Gestion de ressources

Temps à disposition : 8 périodes (travail débutant en semaine 4)

## 1 Objectifs

— Réalisez un programme temps réel où il y a des situations de compétition et de gestion de ressources, et ce à l'aide de sémaphores.

## 2 Enoncé

Sur l'une des quatre maquettes Märklin au laboratoire, implémentez un programme en C++ avec utilisation de sémaphores qui réalise la gestion et le contrôle de deux locomotives. Les locomotives suivent des tracés circulaires. Vous êtes libre de choisir le tracé des locomotives, toutefois il y a une contrainte à respecter : il doit y avoir au moins un tronçon commun aux 2 parcours. Deux programmes vous sont demandés, mais seul le dernier sera corrigé.

### 2.1 Programme 1

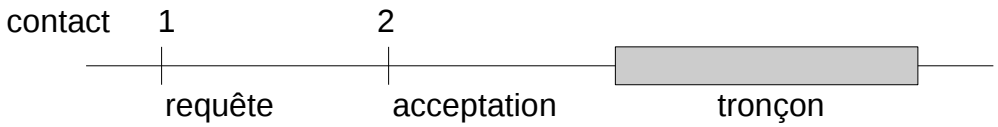
Les locomotives partent d'un point particulier de leur tracé. Chaque fois que les locomotives réalisent 2 tours complets, elles inversent leurs directions et repartent pour 2 tours supplémentaires dans le sens contraire, et ceci de manière infinie. Les locomotives formulent une requête pour obtenir un tronçon partagé et, si celui-ci n'est pas disponible, la locomotive s'arrête devant le tronçon demandé. Si au contraire le tronçon demandé est immédiatement disponible, les locomotives ne devront pas s'arrêter (pas même une micro-seconde). Les locomotives ont la même priorité d'être choisies lorsqu'il faut les départager.

Pour les groupes du mercredi, votre programme devra avoir un arrêt d'urgence actionné par l'interface graphique. Cette fonctionnalité permettra d'arrêter toutes les locomotives immédiatement, et ce de manière propre. Deux heures supplémentaires seront donc consacrées au laboratoire pour ce groupe (10 périodes au total).

### 2.2 Programme 2

Modifiez le programme 1 pour y introduire un traitement prioritaire. Toutes les locomotives ont une priorité distincte qui régit l'ordre de passage sur le tronçon commun. Cette priorité doit être définie dans votre code, et demeure constante. La demande du tronçon commun se réalise en 2 étapes délimitées par des contacts qui précèdent le tronçon :

contact 1 : formulation de la requête avec la priorité de la locomotive en paramètre ;
contact 2 : arrêt de la locomotive ou passage de celle-ci en cas d'acceptation.




Au moment du passage sur le contact *d'acceptation*, si une locomotive de priorité supérieure a demandé le tronçon, alors la locomotive courante doit s'arrêter. Elle doit évidemment aussi le faire si le tronçon est déjà occupé.

## 3 Remarques

— Le code du simulateur de maquettes ainsi que la documentation du simulateur et des maquettes est disponible sur le site du cours.
— Il est interdit d'utiliser la fonction `try_acquire()`.
— L'arrêt d'urgence est une nécessité pour tout système critique où un arrêt peut être réalisé. Dans le présent laboratoire, une manière simple et triviale de réaliser cet arrêt consiste à faire un appel à la procédure `mettre_maquette_hors_service`. Ceci équivaut à couper toute l'alimentation du réseau ferroviaire. Dans le contexte de ce laboratoire, il est demandé d'arrêter les 2 locomotives sans utiliser cette fonction.
— Une démonstration des 2 programmes devra être faite sur les maquettes. De plus, vous devrez nous rendre un listage complet du programme 2, sur papier et par courrier électronique. Aucune correction du programme 1 ne sera faite.
— La description de l'implémentation, ses différentes étapes, la manière dont vous avez vérifié son fonctionnement et toute autre information pertinente doivent figurer dans le programme rendu. 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.

## 4 Barème de correction

| | |
|---|---|
| Conception et conformité à l'énoncé | 25% |
| Exécution et fonctionnement (10% pour programme 1 et 15% pour programme 2) | 25% |
| Codage | 15% |
| Documentation et en-têtes des fonctions | 25% |
| Commentaires au niveau du code | 10% |