haute école neuchâtel berne jura arc ingénierie www.he-arc.ch

Certifié ISO 9001

Labo

Programmation concurente 24-25

Temps à disposition : 6 périodes

1 Objectifs

- Réalisation d'un programme utilisant la librairie Pthread
- Réalisation d'un problème concurrent à travers les variables de condition et les moniteurs
- Simulation de la gestion des bornes de location de vélo dans une ville intéligente

Enoncé du problème

Dans cette ville intéligente, nous avons plusieurs sites qui permettent à ses habitants d'aller d'un site à l'autre en vélo.

Un habitant peut prendre un vélo d'un site pour se rendre à un autre site. A l'arrivée l'habitant doit fixer le vélo sur une des bornes du site, pour que le vélo puisse se charger d'un côté et de l'autre côté pour qu'il soit à disposition. D'une manière simple le comportement d'un habitant est le suivant :

Boucle infinie

1. Attendre qu'un vélo du site i devienne disponible et le prendre.
2. Aller au site j != 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 <- 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 devienne disponible, c'est l'ordre d'arrivé qui prime.

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 >= 2, et le nombre d'habitants sont constants durant l'exécution du programme.
- Ces nombres devront être entrés au clavier 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 >= 4, est aussi obtenu grâce à une entrée clavier de l'utilisateur (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 est aussi entré au clavier (quatrième argument) et doit respecter V >= 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 Vi 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 Vi > B - 2 alors

   Prendre c = min(Vi - (B - 2); 4 - a) et les mettre dans la camionnette.

   a <- a + c

   2b. Si Vi < B - 2 alors

   Laisser c = min((B - 2) - Vi; a)

   a <- a - c
3. Vider la camionnette D <- 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).

Vous avez le choix de partir sur la version console pour afficher les résultats ou une version avec une interface graphique en Qt(voir videos).

- Un squelette de programme vous est fourni. Vous y trouverez des exemples sur l'usage de l'interface graphique ou mode console. Attention, il ne s'agit bien que d'un exemple sans fonctionnalité particulière.
- Attention la partie graphique permet les interactions nécessaires au bon fonctionnement du programme, ces interactions sont implémentées et atteignables grâce à des signaux Qt :
  - initHabitant(int habId,int initSiteId), place l'habitant (habId) sur le site (siteId)

- initCamion(), place le camion de maintenance au dépôt
- setHabitantState(int habId,int state), met l'habitant (habId) dans l'état (state) 3 états disponible WAIT (en attente), ACTION (action sur site), BIKE (vélo)
- setSiteVelo(int idSite,int nbVelo), met nbVelo sur le site (siteId)
- startDeplacement (int habId,int initSite,int destSite,int parcourTime), démarre le deplacement de l'habitant (habId) de initSite à destSite en parcourTime seconde
- setDepotVelo (int nbVeloDep), met nbVeloDep dans le dépôt
- setCamVelo(int nbVeloCam), met nbVeloCam dans la camionnette
- startCamionDeplacement(int initSite,int destSite,int parcourTime), démarre le deplacement de la camionnette de initSite à destSite en parcourTime

- Une partie du squelette de l'algorithme est faite dans la classe AlgoThread.cpp, les deux pthreads (habitants et maintenance) sont créés et comporte des exemples d'utilisation des signaux Qt, mais c'est ici que le travail de concurrence commence.
- Si l'utilisation de l'interface graphique pose un problème, une sortie correcte de votre programme fonctionnel en console est plus importante.

4 Travail à rendre

- Le projet fonctionnel avec une description de l'implémentation, et la manière dont vous avez vérifié son fonctionnement dans le code. Pas de rapport
- Travail peut se faire en équipe de deux personnes au plus.
- A rendre le **vendredi 23 mai 2025** un fichier zippé avec le nom des étudiants par message sur Teams au groupe aicha.rizzotti@he-arc.ch et julien julien.senn@he-arc.ch

5 Barème de correction

Conception	20%
Exécution et fonctionnement	20%
Codage	40%
Documentation et commentaires	20%