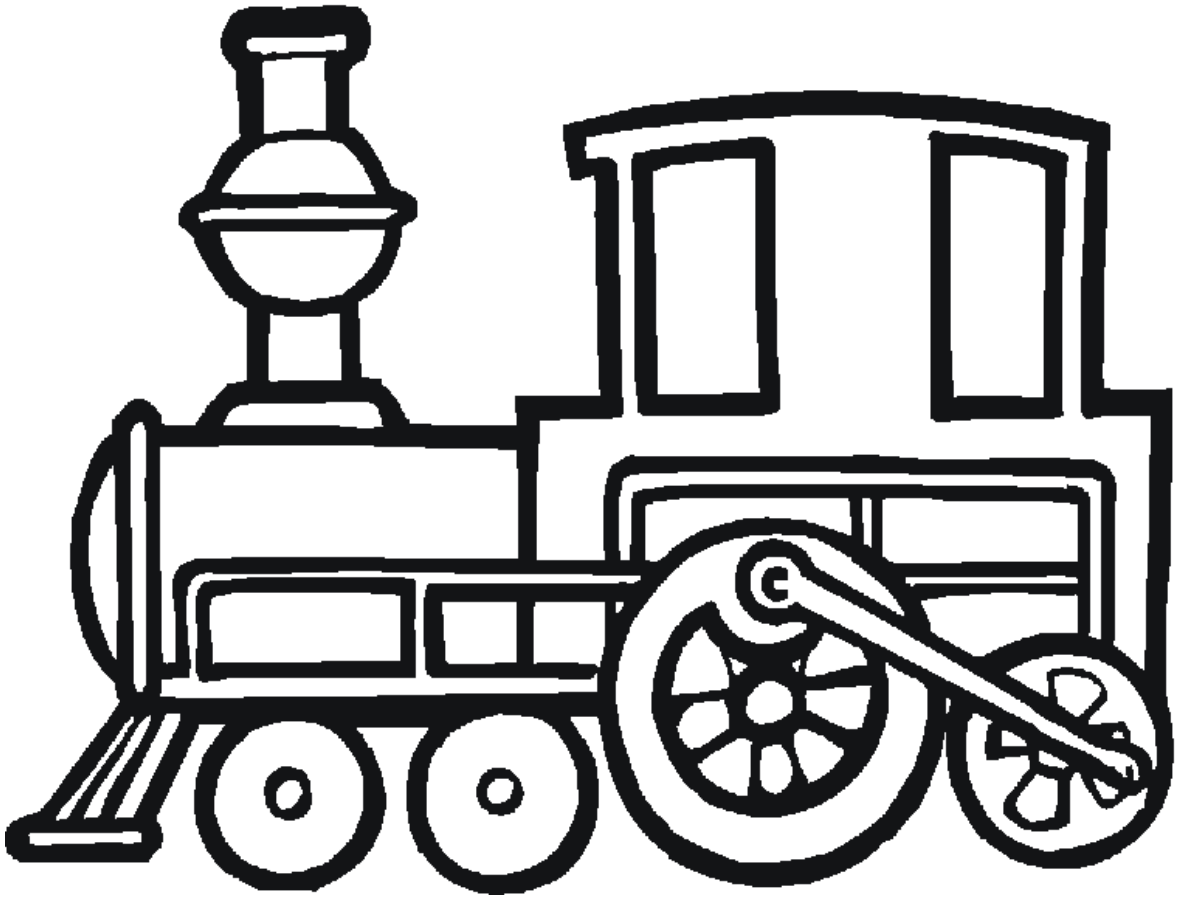
Nassim TRIKI
Fadil TCHAKPIDE

Rapport Programmation Temps Réel

PROJET TRAINS



Master 1 – Informatique **2017/18**

***NOTE**: Le projet a été codé sous Mac par erreur. En effet, alors que nous avions déjà réalisé 3 parties du projet, nous avons remarqué qu'il fallait réaliser le projet sous Linux. Ce qui fait que les résultats obtenus sont incorrects que lorsque notre code est exécuté sous Linux. Nous ne savons pas si notre travail sera accepté mais par manque de temps nous n'avons pas d'autres alternatives. Merci de votre compréhension.*

I) Présentation et description générale du code

Afin de représenter les trajets de chaque train nous avons utilisé des chaines de caractères. Chaque caractère de cette chaîne représente une gare. Nous avons d'une autre part déclaré des variables qui représentaient pour chaque train, les gares de départ et d'arrivée à l'instant t.

Nos threads représentent les déplacements sur les liaisons ferrées des trains 1,2 ou 3.

Dans chaque partie du projet, le code est organisé comme suit :

- Une fonction (train) utilisée en argument pour les 3 threads qui sont créés.
- 3 threads symbolisant le déplacement des 3 trains, appelée dans la fonction (train).
- 3 fonctions utilisées et représentant le déplacement de chaque train (**train1_deplacement, train2_deplacement, train3_deplacement**)avec les messages qui témoignent des déplacements de chaque train. Ces fonctions sont appelées selon l'ID du thread. Dans chacune de ces fonctions, des conditions sont présentes pour détecter si les trains sont sur la même liaison, vont se croiser ou bien circulent chacun sur des liaisons différentes. C'est dans ces fonctions que nous avons établies les zones critiques.

II) Outils et modèle de synchronisation

A)Verrous-(partie_3.c)

Pour ce qui est de la synchronisation avec les verrous, on a déclaré le verrou (lock) en statique afin qu'il puisse être utilisé dans nos 3 fonctions correspondant au déplacement des 3 trains. Nous nous sommes inspirés du modèle lecteur-écrivain. Nous avons uniquement utilisé le verrou d'écriture qui nous a permis d'empêcher les dépassements et le croisement des trains. En effet l'entrée dans une zone critique ne pouvant être faite que par un train à la fois, cela permettait de monopoliser la liaison ferrée, le temps nécessaire à son déplacement et ainsi d'éviter les dépassements ou croisement.

B) Mutex-(partie_1.c)

Pour ce qui est de la synchronisation avec les mutex, nous avons initialisé le mutex en statique au début du fichier. Le comportement du mutex est semblable au verrou d'écriture. Nous avons donc donc établi la zone critique aux même endroit que celle établie avec les verrous. Les explications pour cette implémentation sont identiques à l'implémentation avec les verrous.

C)Sémaphores-(partie_2.c)

Le sémaphore déclaré dans cette partie est un sémaphore nommé. Il est initialisé à zéro lors de sa création puis incrémenté afin de permettre le déplacement d'un train et la mise en mouvement des autres trains. La zone critique est établie de la même manière qu'avec les implémentations précédentes.

III) Banc de test

Afin de mesurer le temps moyen de trajet de chaque train, nous nous sommes servis de la fonction **time()** de la bibliothèque time.h. En effet, introduite dans les fonctions de déplacement de chaque train (**train1_deplacement, train2_deplacement, train3_deplacement)** elle permet de récupérer le temps d'exécution d'une fonction, c'est ce qui nous a permis de récupérer le temps pris par le train pour effectuer 1000 trajets. Il nous suffisait ensuite de diviser par 1000 pour obtenir le temps moyen de trajet . Les temps moyens de trajet des trains pour chaque implémentation sont écrits dans un fichier.

IV) Compilation

Partie_1 (mutex) : $ gcc -Wall partie_1.c -lpthread -lm

Partie_2 (sémaphores) : $ gcc -Wall partie_2.c -lpthread -lm

Partie_3 (verrous) : $ gcc -Wall partie_3.c -lpthread -lm

V) Répartition du travail

Nous avons implémenté ensemble la partie 2 sur les sémaphores tandis que Nassim s'est occupé de la partie 3 sur les verrous et Fadil de la partie 1 sur les mutex.