Rapport LO41

Henri Maugendre et Alexis Vallet

22 juin 2011

Table des matières

Résumé

Dans le cadre de l'UV LO41, nous avons à réaliser une simulation réaliste de croisement routier géré par des feux. L'objectif est de faire alterner les feux à intervalle de temps régulier et de faire passer les voitures au feu vert. En plus de cela viennent se greffer les bus qui peuvent arriver à tout moment dans une voie quelconque et demander à passer. Les feux devront alors prioriser le bus arrivé et changer les feux si nécessaire. Nous avons pour traiter la question décidé d'utiliser des files de message, des mémoires partagées concernant la communication entre les différents processus, préférés aux threads, et des sémaphores pour protéger les variables.

1 Fonctionnement

Comme précisé lors de l'introduction, nous avons décidé d'utiliser des processus et non des threads. Le processus main lance un processus de gestion des feux qui change les feux toutes les 7 secondes. C'est également ce processus qui se charge de récupérer les données de la file de message des bus pour les traiter et ainsi faire passer le bus le plus vite possible. En parallèle, un procesus de gestion des directions s'occupe de générer les voitures, les bus et pour chacune de se mettre en fin de file et attendre le passage au vert s'il n'y est pas. Ce procédé de gestion des directions est effectué pour chaque direction, 4 processus sont donc utilisés.

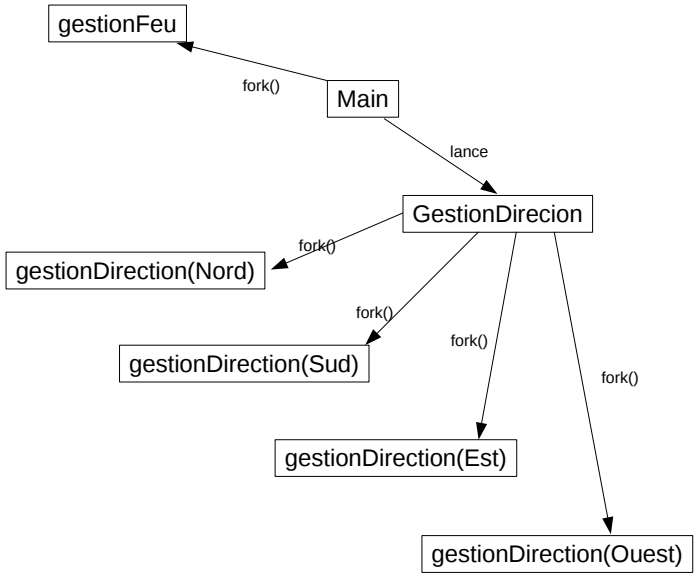

FIGURE 1 – Lancement des processus.

2 Structures utilisées

direction enum {Nord=1,Sud=2,Est=4,Ouest=8} Cette énumération sert à lister toutes les directions possibles.

struct vehicule { direction Depart direction Arrivee } Un véhicule peut être soit un bus soit une voiture. Chacun a une provenance et une destination représentés par des attributs de type direction.

feu enum {VERT_NORD_SUD,VERT_EST_OUEST} Il y a 4 feux mais inutile d'avoir 4 variables car le traitement est symétrique.

struct requeteBus { long type direction contenantLeBus } Cette structure est destinée à la file de message où il y a besoin d'un type et d'une direction.

3 Communication

Dans le but de faire communiquer les voitures et les feux, nous avons mis en place 3 outils de communication. Une file de message a été nécessaire pour stocker les requetes des bus arrivant dans une voie. Cette file contient des structures de type requeteBus. Nous avons également eu besoin d'une mémoire partagée. Elle contient l'état actuel des feux grâce à une structure de type feu. Enfin, une dernière mémoire partagée nous a servi à contrôler le nombre de voitures arrivant dans chaque voie de sortie.

4 Protection des variables

La protection des variables permet de ne pas accéder à une ressource commune à plusieurs processus en même temps. C'est donc tout naturellement qu'on devrait protéger chaque moyen de communication mis en place. En réalité, la file de message contenant les requetes des bus ne nécessite pas de protection car un processus se charge de créer les bus et un autre consomme l'information directement pour changer le feu. Il faut néanmoins protéger les feux. Pour cela, un sémaphore est nécessaire afin de ne pas changer un feu en même temps qu'une voiture consulte son état ce qui donnerait un résultat totalement aléatoire. Un second sémaphore est nécessaire pour gérer les voies de sortie. En effet, à chaque changement de feu, les voies de sortie sont mises à 0 et sont consultées en permanence lors d'un passage de voiture. Encore une fois, pour éviter un résultat aléatoire, celui-ci est nécessaire. Enfin, un dernier mutex est nécessaire pour signaler le changement de feu.

Conclusion

En conclusion, notre projet a atteint ses objectifs, les feux alternent toutes les 7 secondes sauf en cas de demande par un bus, auquel cas ils changent immédiatement. Les voitures passent au vert et jamais plus de voitures qu'une voie de sortie n'est apte à en accueillir ne vont dans celle-ci. Ce projet nous a permis de voir dans un cas concret l'utilisation de bon nombre d'outils vus en cours. Nous avons particulièrement pu être confrontés directement aux problèmes que pose la gestion d'une même ressource accessible par plusieurs processus et les réponses apportées. La gestion de carrefour était un sujet nous permettant d'avoir à faire des choix concernant les structures, les moyens de commmunication et de blocage des variables. C'était en ce sens un sujet riche et intéressant.

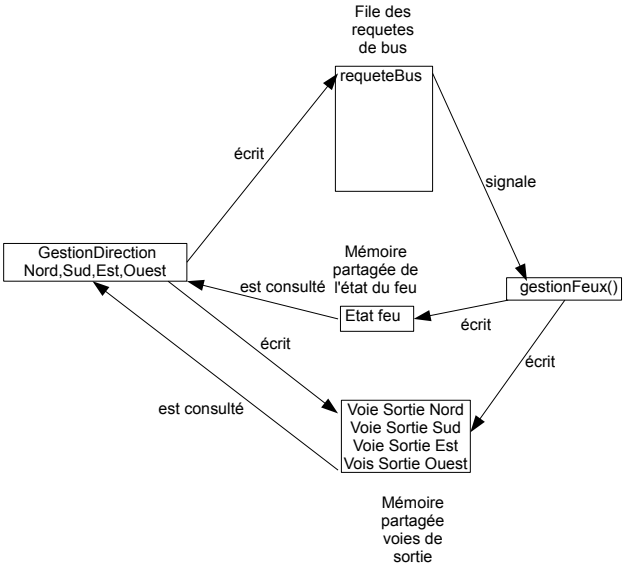

FIGURE 2 – Accès des processus aux mémoires

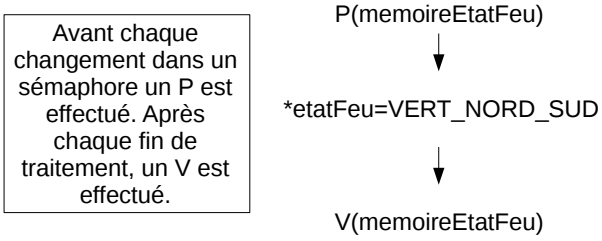

FIGURE 3 – Utilisation des sémaphores