15 février 2021

Rapport - Projet SY40

Julien Audoux

Introduction

Le sujet de ce projet est de modéliser un système de péage pour autoroute, en utilisant une programmation système faisant intervenir des processus et des sémaphores pour synchroniser ces derniers. Le péage devra prendre en compte les cinq catégories différentes de véhicule, la distance parcourue et le type de paiement (télépéage ou tout payment), afin de générer un prix à payer et un temps d'attente adéquat. Le péage devra également gérer des fluctuation dans le flux de véhicule, ouvrir des voies si nécessaire, ou en fermer si besoin. Enfin, une suivie des données devra être possible pour analyser combien rapporte chaque péage, le nombre de voiture traitées, etc…

Solution

Pour réaliser ce projet, j'ai choisie d'utiliser la primitive ***fork()***, pour créer des processus enfants qui seront utilisés pour modéliser chaque véhicule, mais également pour réaliser la fonction d'affichage et la fonction gérant les voies disponibles. Ces derniers doivent fonctionner en simultanés, d'où l'importance des ***fork()***, cependant il est primordial de les synchroniser pour assurer le bon fonctionnement de la modélisation. C'est pourquoi les sémaphores sont importantes, car elle permettent de bloquer un véhicule lorsque qu'il rentre dans un péage, et le débloquer lorsqu'il doit en sortir. Afin d'assurer un suivi de plusieurs données pour un seul véhicule, j'ai réalisé une structure ***vehicule*** avec des données aléatoires pour chaque véhicule généré, qui seront passées à la fonction péage pour déterminer le temps d'attente et le prix à payer par ce dernier. Chaque véhicule est donc représenté par un processus à part entière. Toujours en simultané il existe un processus qui regarde les sémaphores afin de savoir si une ligne supplémentaire doit être rajoutée ou supprimée si inutile. Il existe également un processus qui tourne en continue, se chargeant de récolter toutes les données de l'espace de mémoire partagée afin de les afficher à l'écran.

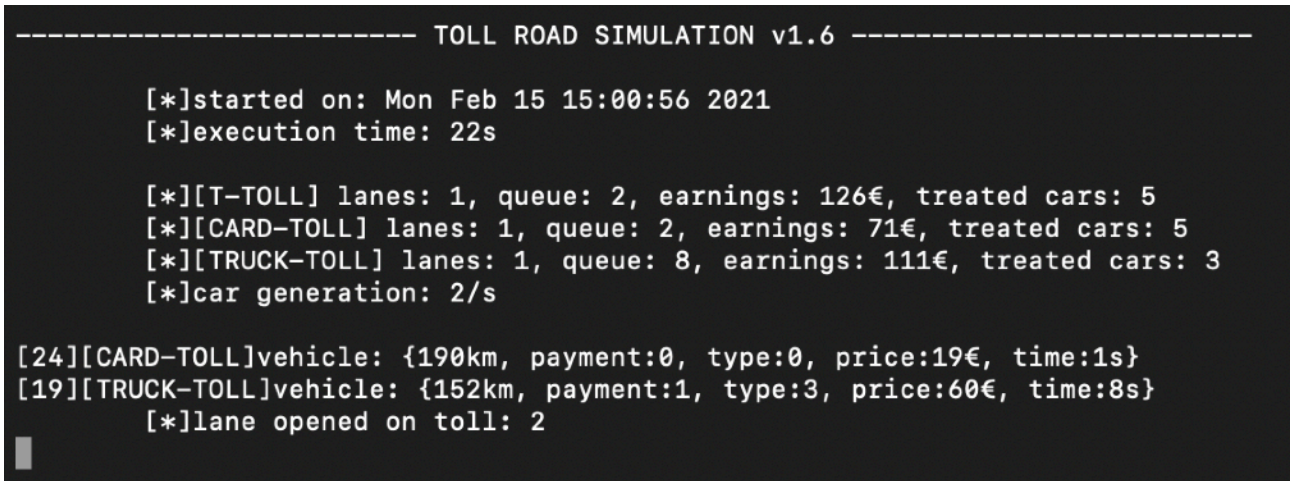
Scénario d’exécution

L'exécution du programme se fait de la manière suivante: au lancement il est possible (mais non obligatoire) de renseigner le paramètre d'exécution ***‘0’*** qui représente l’état d’une variable modifiant l’affichage de la simulation. Ensuite, le flux infinie et non constant de voiture démarre, chaque véhicule est orienté vers le péage adéquat, il en existe trois type : péage camion, péage carte ou espèce et télépéage. S’il existe trop d’attente ou pas assez, le péage se charge de rajouter ou supprimer une voie dans le type de péage qui en a besoin. Le flux étant infini, le programme s’arrête sur un ***‘ctrl-C’*** de l’utilisateur. Ce programme se situe dans un contexte “réel” de péage, c’est à dire que le péage ne s'arrête que lorsque l'autoroute est fermée, autrement il reste disponible pour n'importe quelle voiture. Après l'interruption du programme, les statistiques sont affichées une dernière fois, et tous les processus se terminent correctement.

Fonctionnement détaillé

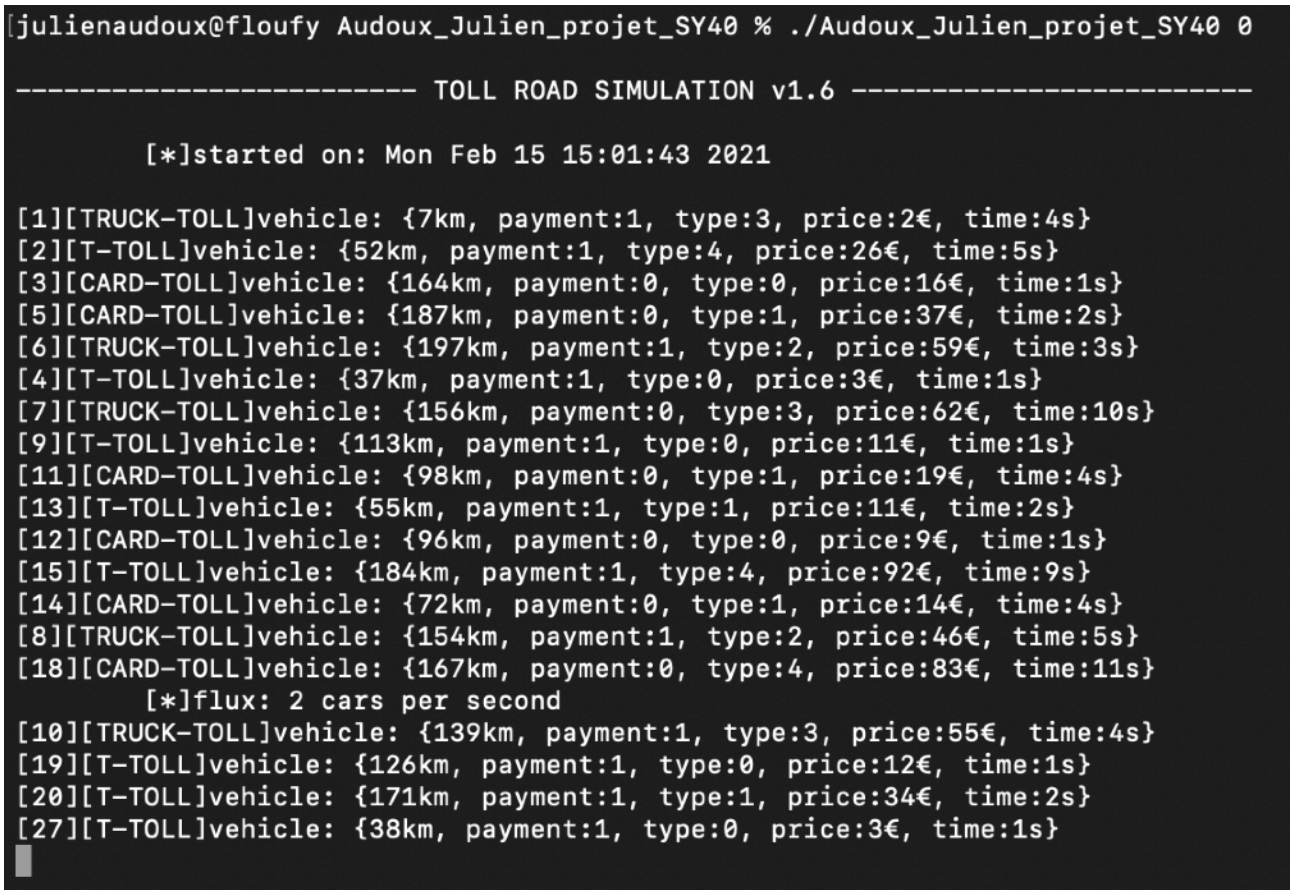
- *Paramètre d’exécution et affichage*

Lors du lancement du programme, il est possible de renseigner l’argument ***‘0’***, il déterminera l’état d’une variable et aura pour effet de changer le mode d’affichage du programme. On peut aussi renseigner ***‘1’***, mais cela n’est pas utile car par défaut cette valeur est déjà à ***‘1’***.



Affichage par défaut

Le mode d’affichage par défaut se compose des informations générales sur chacun des trois types de péages: leur nombre de voies, d’argent gagné, de voitures en attentent et de voitures traitées. Cet affichage est actualisé toutes les 2 secondes, et les informations sur chaque voiture qui passe au péage est affiché brièvement.



Mode d'affichage par véhicule

Le second mode d'affichage, montre uniquement les informations de chaque véhicule, une ligne après l'autre, et affiche les informations statistiques générales en fin de l'exécution.

- *Gestion de données véhicules*

En ce qui concerne les informations relatives à chaque véhicule, j'ai choisi d'utiliser une structure ***vehicule*** qui contient la distance parcourue par le véhicule, le type de ce dernier et son moyen de paiement. A chaque génération, le véhicule contient des informations aléatoires pour le bien de la simulation, il est ensuite orienté vers le type de péage qui lui correspond.

- *Gestion de signaux*

Le programme ne s'arrêtant que par un signal de type ***SIGINT***, on renseigne au début du programme un traitant de signal. Ce traitant se chargera de terminer tous les processus enfants, puis de terminer correctement le processus père en libérant les ressources et en affichant les données statistiques finales de la simulation.

Un signal ***SIGUSR1*** est aussi utilisé pour terminer le programme correctement lors d'une erreur (cf. ***Problème rencontré***).

- *Espace de mémoire partagé*

Pour la communication entre processus, j'ai choisi un espace de mémoire partagé, car la lecture et écriture n'est pas bloquante comme les tubes ou les files de messages. Il s'agit ici d'une variable partagée entre plusieurs processus, ce qui est très pratique pour stocker des valeurs statistiques du péage.

L'espace de mémoire partagé est utilisé pour stocker treize valeurs entières, collectant des données relatives aux statistiques et au déroulement du programme. Elle est modélisée comme suit: ***data[0] data[1] data[2]*** sont respectivement l'argent généré par le télépéage, le tout paiement et le péage camion. Puis, ***data[3] data[4] data[5]*** représentent le nombre de voie pour chaque péage dans le même ordre, ***data[6] data[7] data[8]*** sont le nombre de véhicule en attente dans chaque type de péage. Enfin, ***data[9] data[10] data[11]*** comptent le nombre de véhicule traité pour chacun des péages. De plus, la treizième valeur (***data[12]***) de l'espace de mémoire correspond à la vitesse de génération de véhicule (le flux).

Cet espace de mémoire se résume donc par un tableau de type ***int[]***, sont ***id*** étant une variable globale, cet espace est donc accessible par tout les processus du programme, il est utilisé notamment pour l'affichage en lecture, par le péage en écriture et par la gestion de voie en lecture et écriture.

- *Flux continue de véhicule*

Le flux continue de véhicule est réalisé de manière non constante, c'est à dire qu'il est représenté par un nombre de véhicule qui est amené à varier au cours du temps. En effet, au début de la simulation, 1 voiture par seconde est générée, puis toute les 20 secondes, ce nombre est incrémenté de 1, et ce jusqu'à 6, puis il est décrémenté de 1 jusqu'à 1 toutes les 20 secondes, et le cycle continue indéfiniment.

Chaque voiture générée est donc associée avec des valeurs aléatoires, et est ensuite envoyé au péage correspondant. En fonction de ses paramètres, un prix et un temps d'attente lui sera attribué. Si aucun poste de péage n'est disponible lors de l'arrivée d'une voiture, celle-ci se place en file d'attente. Une fois au poste, lorsque le temps d'attente est effectué, le véhicule libère le poste de péage qui devient libre pour un autre véhicule.

- *Gestion des voies*

La gestion dynamique des voies est assurée par un processus actif en permanence, qui regarde toutes les dixièmes de secondes l'état des files d'attentes pour chacun des péages. Le principe est le suivant : chaque type de péage a un nombre de voie qui lui est propre, on admet que chaque voie peut contenir jusqu'à 10 personnes en attentes en fonctionnement normal.

Ainsi, si sur un péage en particulier, il n'y a plus de place d'attente, on ouvre une nouvelle ligne, et on ajoute donc 10 places d'attentes en plus sur le péage. Au contraire si il y a plus de 20 places d'attentes disponible pour un péage en particulier, on supprime une voie de ce péage et 10 places d'attentes.

- *Sémaphores*

Il existe deux systèmes de sémaphores dans le programme : le premier est un tableau ***int[3]***, dont chaque valeur représente le nombre de poste pour un péage. Autrement dit, au lancement du programme, chaque péage dispose de 1 poste, une voiture à la fois peut donc être traitée par un péage. En pratique, lorsqu'un voiture rentre dans le poste, elle décrémente la sémaphore à l'aide de la fonction ***P(int semnum)***, la sémaphore maintenant à 0 ne peut plus être décrémenté, les véhicules en file d'attentes attendent que celle-ci puissent l'être. Ensuite, en sortant du poste de péage le véhicule incrémente la sémaphore à l'aide de la fonction ***V(int semnum)***, le poste est donc libéré et un autre véhicule peut s'insérer.

Le deuxième système de sémaphore concerne la liste d'attente de chaque péage. De même de précédemment, on utilise un tableau ***int[3]***, dont chaque valeur modélise le nombre de place d'attente disponible pour chacun des trois péages. Ainsi, en pratique juste avant de décrémenter la première sémaphore, le véhicule va décrémenter la deuxième avec ***Pwait(int semnum)***, prenant donc une place dans la file d'attente. Lorsque le véhicule entre dans le poste, il décrémente cette dernière pour libérer une place dans la file d'attente, avec la fonction ***Vwait(int semnum)***.

Ainsi, la première sémaphore compte le nombre de poste pour chacun des trois péages, la deuxième compte le nombre de voiture en file d'attente pour chaque péage. Les véhicules se retrouvent bloqués ou non par ces dernières en prenant et libérant des places, car la valeur d'une sémaphore ne peut être négative.

Fin d’exécution du programme

```
------------------------ TOLL ROAD SIMULATION v1.6 ------------------------

        [*]started on: Mon Feb 15 15:03:04 2021
        [*]execution time: 42s

        [*][T-TOLL] lanes: 2, queue: 8, earnings: 336€, treated cars: 18
        [*][CARD-TOLL] lanes: 2, queue: 8, earnings: 209€, treated cars: 10
        [*][TRUCK-TOLL] lanes: 2, queue: 14, earnings: 309€, treated cars: 10
        [*]car generation: 3/s

        ------------------------ Ctrl-C ------------------------
julienaudoux@floufy Audoux_Julien_projet_SY40 %
```

Ecran de fin d'exécution

En fin d’exécution, après un signal ***SIGINT*** et peu importe le mode d’affichage, on obtient un affichage des données statistiques sur l’exécution du système de péage.

Problème rencontré

Le seul problème rencontré, qui persiste encore, apparaît vers les 194 secondes d’exécution, soit environ vers la 600e voiture générée. En effet l’erreur suivante est rencontrée :

```
[359][TRUCK-TOLL]vehicle: {58km, payment:0, type:3, price:23€, time:10s}
[650][T-TOLL]vehicle: {84km, payment:1, type:1, price:16€, time:2s}
[529][TRUCK-TOLL]vehicle: {81km, payment:1, type:2, price:24€, time:7s}
[609][TRUCK-TOLL]vehicle: {189km, payment:0, type:2, price:56€, time:7s}
[329][TRUCK-TOLL]vehicle: {31km, payment:0, type:2, price:9€, time:7s}
[259][TRUCK-TOLL]vehicle: {54km, payment:1, type:3, price:21€, time:6s}
[295][CARD-TOLL]vehicle: {11km, payment:0, type:4, price:5€, time:11s}
[419][T-TOLL]vehicle: {191km, payment:1, type:0, price:19€, time:1s}
[-]flux fork error: Resource temporarily unavailable

------------------------ TOLL ROAD SIMULATION v1.6 ------------------------

        [*]started on: Mon Feb 15 15:04:51 2021
        [*]execution time: 194s

        [*][T-TOLL] lanes: 5, queue: 38, earnings: 4122€, treated cars: 159
        [*][CARD-TOLL] lanes: 4, queue: 24, earnings: 4492€, treated cars: 159
        [*][TRUCK-TOLL] lanes: 9, queue: 71, earnings: 8183€, treated cars: 235
        [*]car generation: 2/s

        ------------------ [-]flux fork error ------------------
julienaudoux@floufy Audoux_Julien_projet_SY40 %
```

[-]flux fork error: Ressource temporarily unavailable

Cette erreur serait, d'après mes recherches, lié à un nombre de processus trop grand à la fois. Le système semble bloquer la création de nouveau processus au delà d'un certain seuil. En encadrant l'erreur à l'aide d'un signal ***SIGUSR1***, le programme se termine bien et libère les ressources correctement. Cependant l'erreur persiste, et je n'ai encore trouvé aucune solution. En pratique, le système ne tourne donc pas indéfiniment, car l'apparition de cette erreur est systématique.

Amélioration possibles et contexte

Concernant le contexte du programme, il s'agit d'une simulation d'autoroute, mais on peut imaginer le porter sur un système grandeurs nature. Il suffirait de remplacer la partie ***flux()*** par un hypothétique capteur capable de retourner une valeur de débit de véhicule. Les parties paiement et temps d'attente devrait être modifier pour faire intervenir le poste de péage afin de récupérer les informations. Il va de même pour les véhicules, leurs données ne sont plus aléatoires et il faudrait donc les récupérer à l'aide d'autres programmes gérant d'autres machines. Le programme s'exécutant en ligne de commande et se terminant par un signal ***SIGINT***, il pourrait hypothétiquement faire partie d'un système plus vaste gérant un péage réel. Le nombre de voie total devra être limité suivant les possibilités physiques du péage.

Annexes (fournies avec ce rapport)

- Fichier source : Audoux_Julien_projet_SY40.c
- Makefile
- Fichier binaire : Audoux_Julien_projet_SY40