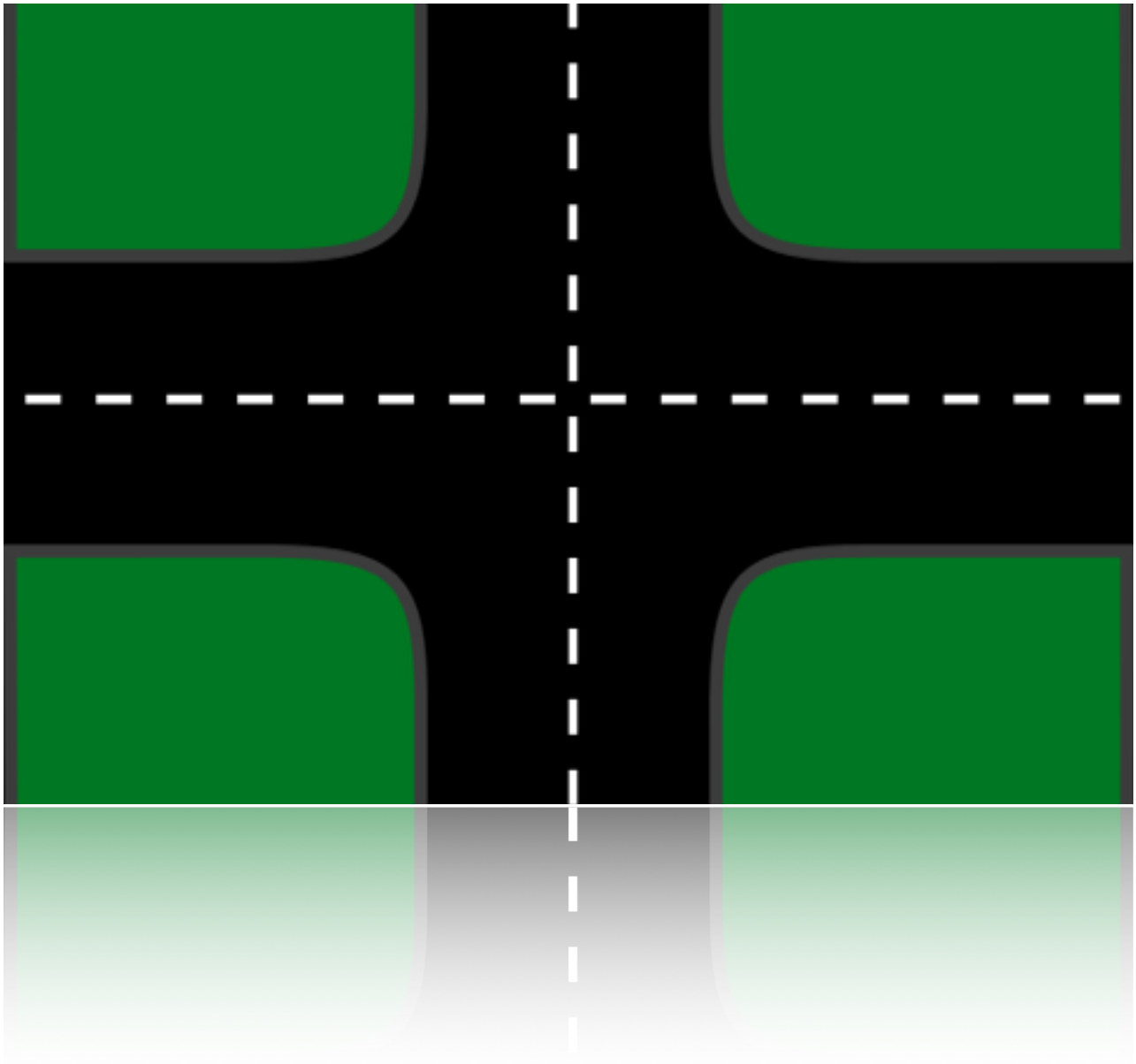
Document descriptif

Crossroad

FAUCHER Alexandre & REZGUI Gada - 27/03/2015



Document descriptif

Crossroad

1. Variables partagées & sémaphores

a) Variables partagées

Nous avons 8 variables partagées, qui sont déclarées dans une structure nommée Shared, se trouvant dans le fichier carrefour.h :

- numberOfCarsInCrossroads : nombre de voitures se trouvant dans le carrefour, cette variable partagé permet aux feux de savoir quand ils peuvent passer au vert. En effet, les feux ne peuvent passer au vert que s'il n'y a aucune voiture dans le carrefour.
- end : est une variable qui permet de détruire (lorsque l'utilisateur le demande) tous les processus, ainsi que de détacher la variable partagée shared qui est reliée à touts les processus.
- numberOfAllCarCreated : nombre total de voitures créées, permet de mettre des indices aux voitures créées. Dans le mode automatique cette variable permet de savoir si nous avons atteint ou non le nombre de voitures maximum que l'utilisateur a choisi.
- firstRoadLights : permet de connaître la couleur du feu[1] dans la route principale. Cette variable est utilisée dans la fonction qui change la couleur du feu, chaque feu a besoin de connaître la couleur de l'autre pour savoir s'il peut passer au vert ou non.
- secondRoadLights : suit le même principe que firstRoadLights, sauf que la variable permet la gestion du feu se trouvant dans la route secondaire.
- nbCarWaitingFirstRoadLights : cette variable partagée est associée à un sémaphore qui permet la gestion des voitures lorsque le feu est rouge (vous aurez plus de détails sur ce sujet par la suite). Elle compte le nombre de voitures qui attendent que le feu passe au rouge dans la route principale.
- nbCarWaitingSecondRoadLights : suit le même principe que nbCarWaitingFirstRoadLights, sauf que la variable compte le nombre de voitures attendant que le feu passe au rouge dans la route secondaire.
- timeToWaitRoadLights : indique le temps d'attente pour le passage au rouge du feu sur la seconde voie (2 * timeToWaitRoadLights, pour la voie principale) . Dans le mode automatique, cette variable est modifiée après que les deux processus permettant la gestion des feux soient créées. Nous aurions pu déclarer une variable globale et créer les processus permettant la gestion des feux avant de modifier la variable, mais nous

[1] Les couleurs du feu sont 2 constantes :

- RED = 0
- GREEN = 1

avons choisi de procéder de cette manière pour avoir une plus grande flexibilité. En effet, si l'utilisateur décide de changer le temps d'attente aux feux en plein milieu du programme (dans l'interface graphique par exemple), cela est tout à fait possible.

b) Sémaphores et mutex

Nous utilisons 4 mutex et deux sémaphores.

Mutex

Les variables partagées qui sont modifiées par les processus « cars » sont protégées, car même si les voitures ne sont pas toutes générées en même temps, les fonctions P() et usleep() peuvent altérer leurs comportements et faire en sorte que 2 processus utilisent les même variables partagées à un même instant.

- crossroadMutex : est un mutex qui protège la variable partagée numberOfCarsInCrossroads, qui est incrémentée lorsque les voitures traversent le carrefour et décrémentée lorsqu'elles quittent le carrefour.
- nbCarWaitingFirstRoadLightsMutex : protège nbCarWaitingFirstRoadLights.
- nbCarWaitingSecondRoadLightsMutex : protège nbCarWaitingSecondRoadLights.
- numberOfAllCarCreatedMutex : protège numberOfAllCarCreated, qui est utilisée lors de la création des voitures.

Les autres variables partagées ne sont pas protégées car elles sont uniquement modifiées par un seul processus. Par exemple, les variables end et timeToWaitRoadLights ne sont modifiées que dans le processus père. Les autres processus les manipulent mais ne modifient pas leurs valeurs.

Sémaphores

Les 2 sémaphores permettent de réguler la circulation du carrefour. Ce sont des tableaux de taille 2 (correspond aux nombres de voies) , ils n'ont pas le même comportement :

- roadLights : permet d'effectuer les changements de couleurs des feux.
- drive : permet d'informer les voitures (qui attendaient devant le feu rouge) que le feu est passé au vert.

2. Structures de données

Deux structures ont été utilisées.

- Une pour les variables partagées :

```
typedef struct {
    int numberOfCarsInCrossroads;
    int end;
    int numberOfAllCarCreated;
    int firstRoadLights;
    int secondRoadLights;
    int nbCarWaitingFirstRoadLights;
    int nbCarWaitingSecondRoadLights;
    int timeToWaitRoadLights;

} Shared;
```

dont les éléments ont été décrits dans la partie 1.a)

- Une autre pour les processus « cars », contenant le pid du processus (permettant entre outre de faire un waitpid, si nécessaire), l'indice de la voiture et la route dans laquelle elle se trouve (principal ou secondaire).

```
typedef struct{
    int pid;
    int route;
    int index;
}Car;
```

3. Algorithmes

Voici les différents algorithmes utilisés. Certains détails seront volontairement omis.

a) Régulation du carrefour

```
fonction genereCar ( paramètre : la route choisit){
        switch(fork()){
                case -1 : envoyer une erreur;
                case 0 : on initialise la nouvelle voiture créée
                        P(numberOfAllCarCreatedMutex); // on vérifie qu'on peut utiliser la
variable partagée (on vérifie si elle n'est pas utilisée par un autre processus)
                        shared->numberOfAllCarCreated ++; // on incrémente le nombre de voitures
                         V(numberOfAllCarCreatedMutex); // on libère la variable pour que les
autres processus puissent l'utiliser
                        la voiture roule vers le carrefour
                        carsCrossroad(car); // la voiture entre dans le carrefour.
                        exit(0); // la voiture disparait
        }
        return car;
}
```

La fonction « genereCar » permet de générer des voitures (processus), une autre version de cette fonction, ayant presque le même contenu existe, seul le mode de génération diffère. Celle qui a était décrite précédemment est faite pour le mode interactif, l'autre est faite pour le mode automatique. Elle ne sera donc pas décrite.

« genereCar » appelle une autre fonction, « carsCrossroad » que nous allons voir à présent.

```
procedure carsCrossroad( paramètre : une voiture){
        switch(car.route){
/** selon la route sur laquelle la voiture se trouve les variables partagées qui seront utilisées ne seront pas les mêmes. Nous allons donc développer uniquement une des voies mais ce qui est valable pour l'une est également valable pour l'autre, seul le nom de certaines variables, sémaphore et mutex change entre les 2 case du switch **/
                case : voie principal
                        if ( shared->firstRoadLights == RED) // si le feu est rouge
                        P(nbCarWaitingFirstRoadLightsMutex);
                        shared->nbCarWaitingFirstRoadLights ++; // on indique qu'il y a une voiture qui attend que le feu passe au vert
                        V(nbCarWaitingFirstRoadLightsMutex);
                        P(drive[PRIMARY_ROUTE]);// on attend que le feu nous indique qu'il est passé au vert et qu'on peux traverser.
                        P(nbCarWaitingFirstRoadLightsMutex);
                        shared->nbCarWaitingFirstRoadLights  - - ; // la voiture n'attend plus, le feu est vert
                        V(nbCarWaitingFirstRoadLightsMutex);
                }
                if(shared->firstRoadLights == GREEN) { // si le feu est vert
                        P(crossroadMutex);
                        shared->numberOfCarsInCrossroads ++; // la voiture est dans le carrefour
                        V(crossroadMutex);
                        la voiture roule dans le carrefour
                        P(crossroadMutex);
                        shared->numberOfCarsInCrossroads —; // la voiture quitte le carrefour
                        V(crossroadMutex);
                }
        case : voie secondaire
/** même comportement que pour la voie primaire **/

}
```

La procédure « carsCrossroad » permet de contrôler la circulation des voitures dans le carrefour, par l'intermédiaire des sémaphores, des mutex et des variables partagées.

Une autre procédure permet également de contrôler la circulation, mais cette fois ci, du côté des feux. Il s'agit de « changeRoadLights ».

```
procedure changeRoadLights( param : la route qu'on veut contrôler){
/** ici aussi même principe que la procédure précédente, le comportement est le même pour les 2 routes. Nous ne décrirons donc qu'une partie du code **/
 if (route == PRIMARY_ROUTE) { // si nous somme dans la voie principale
/** s'il n'y a pas de voitures dans le carrefour, et que le feu de la voie secondaire est rouge **/
   if (shared->numberOfCarsInCrossroads == 0 && shared->secondRoadLights == RED) {
      P(roadLights[PRIMARY_ROUTE]); // le feu attend qu'on l'autorise à passer au vert
         shared->firstRoadLights= GREEN; // le feu est vert
     if(shared->nbCarWaitingFirstRoadLights>0){ // s'il y a des voitures qui attendent que le feu passe au vert
         V(drive[PRIMARY_ROUTE]); // on leurs indique qu'ils peuvent traverser
      }
      usleep(shared->timeToWaitRoadLights*2000); // on attend 2*tempsChoisitParL'utilisateur
      shared->firstRoadLights= RED;// le feu devient rouge
      V(roadLights[SECONDARY_ROUTE]);// on signale à l'autre feu qu'il peut passer au vert s'il le souhaite.
   }
```

Dans cette procédure, nous nous retrouvons dans un inter-blocage, pour passer au vert le feu 1 attend que le feu 2 l'autorise, et le feu 2 attend que le feu 1 l'autorise à passer au vert. Pour remédier à ce problème un V(roadLights[PRIMARY_ROUTE]) a été placé au début du main.

b) Le main

Le main suit les étapes suivantes :

1. Initialisation des mutex, sémaphores et variables partagées
2. Création de deux processus, l'un pour gérer la régulation du feu sur la voie principale, l'autre sur la voie secondaire.
3. Mode interactif : on génère les voitures en fonctions de ce que l'utilisateur choisit.
4. Mode automatique : on génère les voitures en fonctions des arguments que l'utilisateur a indiqué.
5. Libération mémoire des variables partagées, sémaphores et mutex.

c) Autre algorithme

Les données se trouvant dans les fichiers (mode automatique option -f) sont analysées par l'intermédiaire des expressions régulières.