**Projet SY40**

**Parking**

NEMO Simon

Robin Alexis

**Objectif du projet :**

L’objectif de ce projet est de simuler le fonctionnement d’un parking sécurisé. Ce parking possède deux types d’usagers :

* Les abonnés : ils possèdent un émetteur de signal qui sert de code pour entrer dans le parking. Si le système reconnait le signal, la barrière s’ouvre automatiquement.
* Les non abonnés : ils ont accès à une borne de distribution de ticket. Ces tickets sont proposés à condition qu’une place soit disponible.

Le parking possède N places de parking privés accessibles aux abonés et M places disponibles pour les non abonnés. Si aucune place abonnée n’est disponible, un abonné à le droit d’occuper une place non abonnée.

A partir de 18h, une zone de débordement est prévue (uniquement pour les places non-abonnés). Par exemple, avant 18h, seulement 400 places sont disponibles puis après 18h, 420 places sont disponibles, après 19h, 450 places… Puis à 8h du matin, le nombre de places disponibles retombe à 400 (si le nombre de personne garé dépasse les 400 à 8h, il faudra attendre que suffisamment de personne partent avant de pouvoir accueillir de nouveaux usagers). Cependant, si une place abonnée est disponible, un abonné pourra entrer.

L’entrée et la sortie du parking se fait par la même voie. Celle-ci est gérer par un feu bicolore permettant d’informer l’utilisateur de la possibilité d’accéder à la barrière. Il ne peut y avoir simultanément deux véhicules à la barrière.

**Solutions techniques :**

Chaque voiture seront des threads. Elles arrivent vers le feu et envoie un message dans la file de message du parking donnant leur action (entrer ou sortir du parking), leur type (abonné / non abonné) et leur thread ID. Ensuite, le thread attend la réponse du parking :

* Le parking lui répond action = 0, la voiture accède à la barrière (ressource critique), le feu repasse au rouge. L’usager passe X secondes à la barrière. Lorsqu’il est sorti de le la voie, il réveille le parking via une sémaphore. L’usager passe ensuite X minutes/heures dans le parking. Lorsqu’il voudra sortir, il enverra une requête dans la file de message avec comme action, la sortie. Il attendra ensuite la réponse du parking pour pouvoir sortir.
* Le parking lui réponde action = 1, il n’y a pas de place dans le parking. L’usager s’en va.

Le parking est un thread, il récupère les messages de la file de message :

* L’usager souhaite entrer dans le parking : Le parking lui répond action = 0 s’il peut accéder à la barrière et le feu passe au vert.
* L’usager souhaite sortir : Le parking lui répond action = 0 pour l’autoriser à sortir.

Si l’usager n’a pas l’autorisation d’accéder à la barrière, le feu reste rouge et le message est émis. Sinon, le feu passe au vert et le message est émis. Le parking s’endort via une sémaphore et sera réveillé lorsque l’usager est sorti de la voie. Puis il traitera le message suivant.

Le programme possède deux compteurs de place disponibles : un compteur pour les places abonnés et un pour les places non abonnées. Ces compteurs sont incrémentés lorsqu’un usager rentre dans le parking et décrémentés lorsqu’un usager sort. Si un abonné accède à une place non abonnée, son thread ID est mémorisé. Lorsqu’un abonné souhaite sortir du parking, son ID est comparé à cette liste afin de décrémenter le bon compteur.

Un thread s’occupe de la gestion du temps. Selon l’heure, ce thread modifie le nombre de place disponible.

Les compteurs sont donc des ressources critiques. Des mutex seront donc utilisés pour réguler leur accès.

Une fonction Attendre sera écrite et permettra de modifier l’accélération du temps.

Un thread générera des voitures toutes les X secondes simulées (0<X<60secondes)

**Digramme (réseau de Pétri) :**