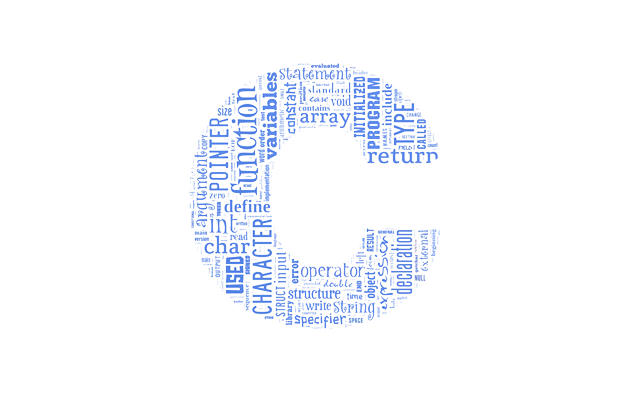
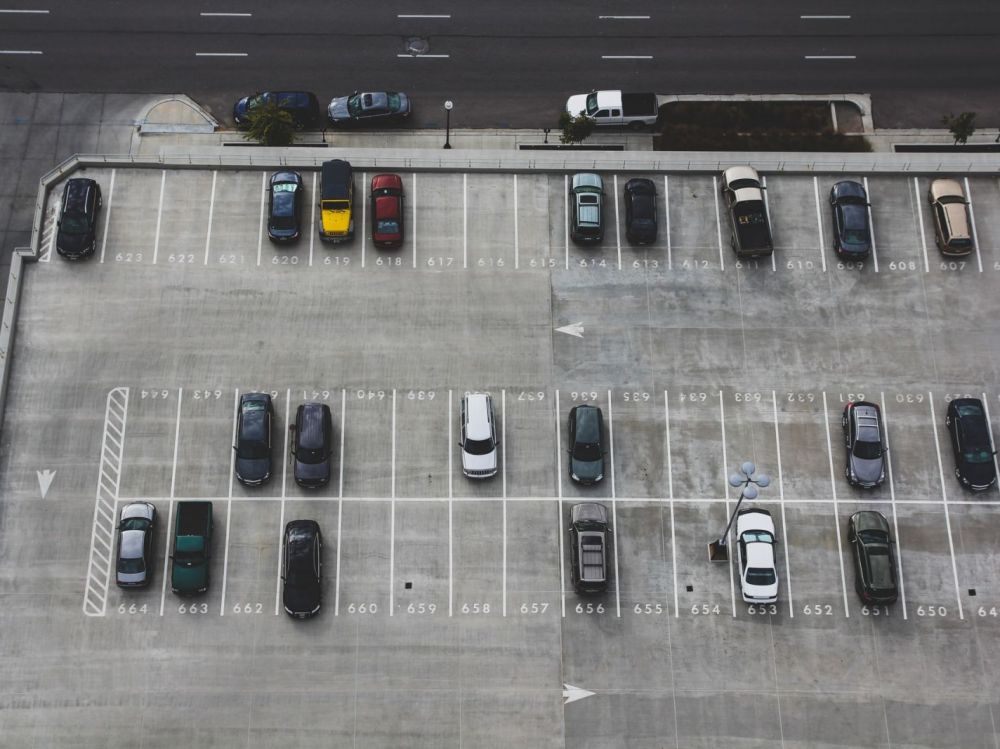
**Semestre printemps 2022**

Rapport de projet - SY40



PROJET LANGAGE C

SIMULATION D’UN PARKING

Projet réalisé par

**Jérémy KIEFFER**

**Titouan BURGY**

Projet encadré par

**Philippe DESCAMPS**

Sommaire

[**I.** **Présentation du projet** 3](#_Toc106143053)

[1. Description du projet 3](#_Toc106143054)

[2. Analyse des demandes 3](#_Toc106143055)

[**II.** **Les méthodes utilisées** 4](#_Toc106143056)

[1. Choix de conception – Les Threads 4](#_Toc106143057)

[2. Mise en place des structures 5](#_Toc106143058)

# **Présentation du projet**

## Description du projet

Dans le cadre de l’UV SY40 (Architecture des systèmes d’exploitation) en première année d’Informatique à l’UTBM, il nous a été demandé de réaliser un projet pour mettre en pratique les différentes connaissances acquis durant le semestre avec comme finalité la mise en place d’une simulation d’un parking public à l’aide du langage de programmation C. Les différentes consignes pour ce projet peuvent être téléchargé ici :



## Analyse des demandes

Après une première analyse des différentes demandes du projet, nous avons pu mettre en place une liste de problèmes à résoudre pour mener à bien ce projet :

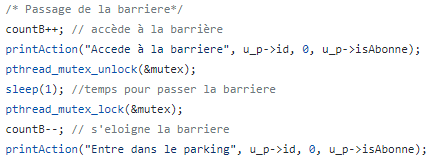
* Quelle(s) méthode(s) utiliser pour la simulation des usagers, du parking etc … ?
* Quelles sont les différentes structures à mettre en place ?
* Comment gérer la création de processus non-stop ? Comment gérer le temps (primordial pour la mise en place de la zone de débordement) ?
* Quel affichage ? Intérieur du parking ou activités à la barrière ?

# **Les méthodes utilisées**

## Choix de conception – Les Threads

Notre choix pour la simulation des processus s’est assez vite tourné vers les threads. En effet, nous avons privilégié cette méthode pour plusieurs raisons :

* Les différentes fonctions vues en cours nous paraissaient adaptée aux différents problèmes que présente ce projet. Le thread étant un processus léger, la communication entre threads est beaucoup moins lourde et coûteuse que la communication entre processus système (dit lourds, comme l’utilisation de fork). De plus, étant donné que les threads se lancent en même temps que le processus main, la multiplication des threads et la synchronisation des différentes tâches pour la simulation du parking est largement simplifiée.
* D’un point de vue personnel, nous avons tous les deux bien compris et assimilé la notion de thread. Lors des différents travaux pratiques de l’UV, nous étions à l’aise avec l’utilisation des threads là où certains aspects d’autres méthodes nous paraissaient beaucoup plus compliqués pour effectuer la même action.
* La protection des variables ou des parties de code dont la synchronisation est très importante (comme le nombre d’usager dans le parking ou le nombre d’usager qui attende à la barrière) pouvait très simplement être résolue à l’aide des mutex :



Dans l’exemple ci-dessus, on remarque en effet que le principe de « passer la barrière » est très important. Dans la consigne il est en effet précisé qu’un seul utilisateur peut passer de l’autre côté et le mutex nous permis de respecter ce principe-là en rajoutant simplement deux fonctions (implémenter avec la librairie des threads).

## Mise en place des structures

* 1. Abonné et non-abonné

Structure d'un usager

Description générée automatiquement

* 1. Parking

Une image contenant texte

Description générée automatiquement

* 1. Chronomètre

Une image contenant texte

Description générée automatiquement

## Génération des Threads