

Projet Francorchamps

Simulation de circuit rédigée en C

François Girondin (2TL2), Louis De Wilde (2TL1), Carlos Emilliano Ruiz Herrera (2TL1), Mathieu Walravens (2TL1)

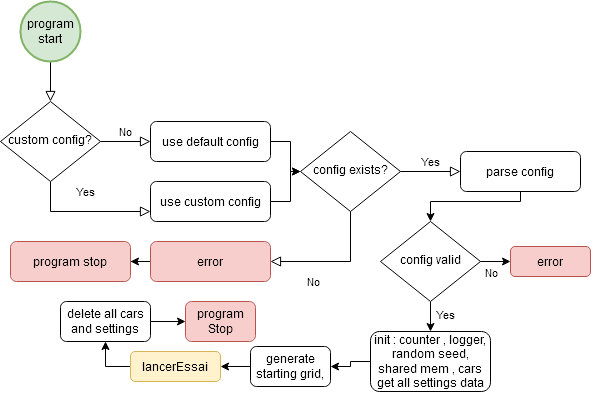
Systèmes d’exploitation II (Pratique) | 13-01-21

# Introduction

Le but de ce projet est de pouvoir simuler, au travers de tableaux de classements rafraîchis à intervalles réguliers, un weekend entier de Grand Prix de Formule 1 sur le circuit de Francorchamps, des séances d’essai à la finale du dimanche.

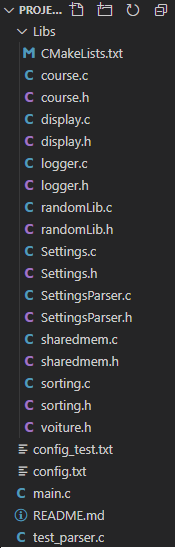
Pour cela, nous avons écrit en langage C un programme utilisant les processus pères et fils, la mémoire partagée et les sémaphores afin de pouvoir faire tourner toutes les voitures en même temps.

# Analyse

1. Plan de l’application.

Lorsqu’on lance le programme, il passe d’abord par plusieurs étapes de vérification des paramètres. Après cela, il initie ses variables en fonction des paramètres qui lui ont été passés via un fichier de config (passé en paramètre lors du lancement du programme, appelé config.txt par défaut), et créée également la mémoire partagée et sa sémaphore ainsi que les voitures. Ensuite, on génère la première frame du tableau, avant de lancer les essais, chacun avec leurs propres paramètres, jusqu’à la finale. Enfin, on détruit toutes les voitures et les paramètres avant de clore le programme.

2. Découpage en modules.

Afin de simplifier le code, le programme a été divisé en 10 modules (dont le main.c) et un fichier de paramètres :

-Le module course : Contient les fonctions responsables de la simulation des tours par les voitures : vérification qu’elles se crashent (ou pas), génération des temps de course, usure des pneus, etc…

-Le module display : Concerne tout ce qui concerne l’affichage du déroulement ainsi que des résultats de la course.

-Le module logger : Contient les fonctions permettant de créer des logs (ou étant utilisées par celles-ci).

-Le module randomLib : Responsable de toute la partie aléatoire du projet par le biais de la création de seeds et de ranges, utilisée pour l’usure des pneus et les temps aléatoires.

-Le module Settings : Contient les fonctions relatives aux paramètres : vérification, traitement de ces derniers.

-Le module SettingsParser : Contient les fonctions de lecture de fichier permettant d’interpréter le fichier des paramètres.

-Le module sharedmem : Contient toutes les fonctions liées à la mémoire partagée et aux sémaphores.

-Le module sorting : Contient toutes les fonctions permettant d’effectuer divers tris sur des arrays.

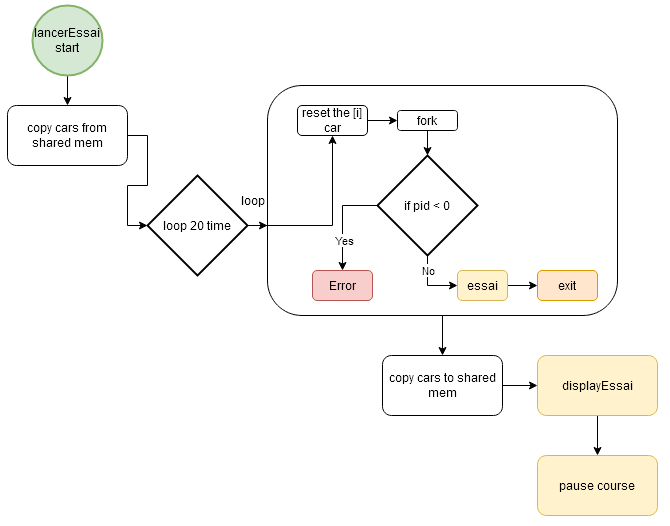
-Le module voiture (.h uniquement) : Contient la définition de la structure des voitures et de leur état (usure et kilométrage).

-Le module main : Le programme en tant que tel, qui créée les voitures, lance les courses, les affiche, gère les accès à la mémoire, vérifie et utilise les paramètres, etc…

-Le fichier config.txt : Il contient les paramètres de course tels que les longueurs des sections, le temps de rafraichissement du tableau ou les noms des voitures en course.

3. Plan des modules principaux.

lancerEssai : la fonction lancerEssai a pour rôle de lancer les 20 voitures dans leur course en créant pour chacune un processus, et lance également la boucle qui gère l’affichage.

Dans un premier temps, la fonction copie les voitures depuis la mémoire partagée. Ensuite, pour chaque voiture, elle réinitialise ses paramètres avant de créer un processus fils qui exécutera essai pour cette voiture. Puis, les voitures sont recopiées dans la mémoire partagée, et on lance displayEssai qui boucle chaque seconde afin d’actualiser son affichage pendant que les voitures dans les processus fils continuent leur parcours.

essai : la fonction essai a pour rôle de faire faire les tours de circuit à la voiture qui lui est attribuée, en simulant ses temps de course, son usure, ses probabilités de crash, etc...

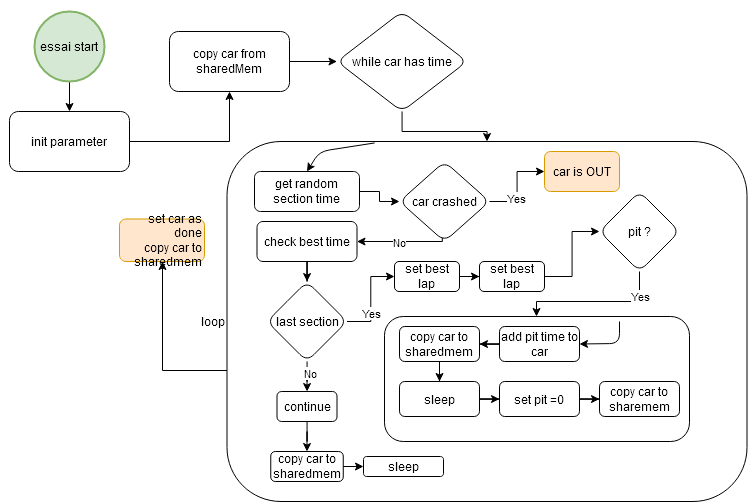
Dans un premier temps, essai récupère les paramètres du config.txt qui lui seront utiles (taux d’usure, longueur des sections, le délai de sleep…). Ensuite, la fonction va copier la voiture depuis la mémoire partagée, avant de lancer une boucle while qui tourne jusqu’à ce que la voiture soit à court de temps.

À chaque passage dans la boucle, la voiture calcule un temps de section aléatoire. Elle passe ensuite un test pour voir si elle se crashe. Si c’est le cas, la voiture est OUT. Dans le cas contraire, on vérifie que le temps de section est bien réaliste.

Si ce n’est pas la dernière section, alors on copie la voiture dans la mémoire partagée et le processus est mis en sleep pendant le temps défini par le paramètre delay du config.txt.

Si c’est la dernière, en revanche, on établit le meilleurs temps de la voiture et on calcule aléatoirement si elle doit aller au pit.

Si oui, alors on ajoute le temps de pit au temps de parcours de la voiture, on recopie la voiture dans la mémoire partagée avant de mettre le processus en sleep, puis on met le pit à 0 et on copie à nouveau la voiture dans la mémoire partagée.

Lorsque la voiture a usé tout son temps de course, son statut est mis à DONE et on la copie dans la mémoire partagée.

# Conclusion

Nous sommes parvenus à réaliser l’entièreté du travail demandé, en utilisant mémoire partagée et sémaphores pour faire tourner les différentes voitures.

Les difficultés rencontrées :

Les difficultés rencontrées ont été nombreuses mais pas insurmontables, et parmi les plus notables, il y avait le fait que lors de la finale, les voitures n’étaient pas synchronisées, bien qu’elles tournaient en même temps. Pour que les voitures restent au même stade lors de l’affichage, il a donc fallu rajouter de la logique.

Il y a également eu des problèmes de « fuite de mémoire », mais nous avons pu régler ce problème à l’aide de Valgrind.

D’un ordre moindre, il y avait le fait qu’il a fallu jongler avec les chiffres pour trouver un équilibre dans la probabilité de crash, afin d’éviter une hécatombe à chaque circuit.

Conclusions personnelles :

Carlos Emilliano Ruiz Herrera : Une fois les premières difficultés surmontées, c’était pas vraiment compliqué, ça demandait beaucoup du temps.

Mathieu Walravens : Le projet était surtout long, mais assez simple malgré les consignes un peu plus floues, ça rappelle au final le projet de l’année passée.

Louis De Wilde : J’ai trouvé les consignes assez vagues, mais une fois qu’on a compris le cœur du projet, il était assez simple.

François Girondin : Tout comme Louis, je trouve que les consignes manquaient de clarté, mais pas pour le projet en lui-même (car tous les éléments pour le comprendre nous ont été donnés en cours) mais pour le rapport. J’ai trouvé difficile de comprendre ce qui était demandé à certains points sans exemples.