Francorchamps

Simulation de circuit de course rédigée en C

Tommy Riquet, Quentin Laruelle, Sebastian Dziemianko

Systèmes d’exploitation II (Pratique) 20/01/2022

# **Table des matières**

1. Introduction ........................................................................................................ Page 3
2. Analyse ................................................................................................................ Page 4

1. Plan de l’application ........................................................................................ Page 4

2. Découpage en modules .................................................................................. Page 5

I. Module time …..................................................................................... Page 5

II. Module result ..................................................................................... Page 5

III. Module des courses …....................................................................... Page 6

IV. Module de la finale ........................................................................... Page 7

3. Plan des modules principaux ......................................................................... Page 8

1. Conclusion …........................................................................................................ Page 10

1.Point de vue général ......................................................................................... Page 10

2.Les difficultés rencontrées ............................................................................... Page 10

3.Ce qu’on aurait pu améliorer ........................................................................... Page 10

4.Conclusions personnelles ................................................................................ Page 11

# **Introduction**

Ce projet consiste à simuler un grand prix de Formule 1 se déroulant sur la durée d’un week-end. Pour y parvenir nous employons des tableaux de classement trié sur différents critères en fonction de la course concernée.

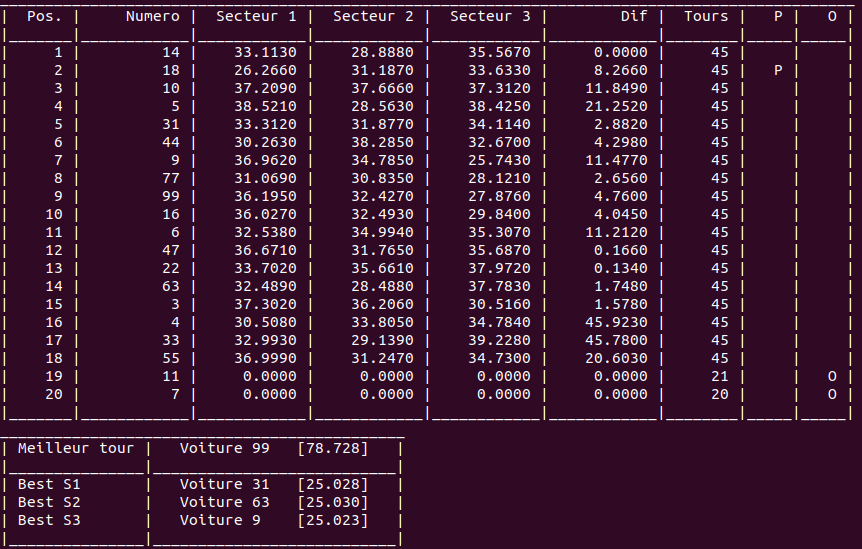
Le week-end est composé des trois séances d’essais, trois séances de qualifications qui sont employées pour générer un classement de départ pour la septième et dernière course, la finale. Les essais et les qualifications sont des courses à durée temporelle limitée alors que la finale est délimitée par un nombre de tours à effectuer.

Pour y parvenir, nous avons écrit un programme utilisant la mémoire partagée afin de parvenir à simuler chaque voiture dans un processus Fils différent, pendant que le père récupère et affiche les résultats en temps accéléré de l’avancement de la course.

# **Analyse**

1. Plan de l’application :

L’application se présente sous plusieurs fichiers, ces derniers servent à générer, faire fonctionner et afficher les résultats dans un fichier externe des différentes sessions du week-end. Il y aura trois séances d’essais (P1, P2, P3), trois séances de qualifications (Q1, Q2, Q3) et une finale. L’exécution du programme se déroule dans l’ordre susmentionné.



Commençons par config qui est le module qui contient les paramètres du programme comme le nombre de tours de la finale ou le temps de chaque course.

Suivi de course qui est le module en charge de la simulation des temps ainsi que des crashs et des arrêts aux stands pour les essais est les qualifications qui s’effectuent sur une durée déterminée. Pendant que final s’en charge pour la dernière course qui s’effectue en fonction d’un nombre de tours prédéterminé.

Ensuite result est le module en charge de la gestion des temps, leur tri, leur récupération et leur sauvegarde.

Le module time sert à générer tous les éléments aléatoires du module que ce soient les temps pour la course ou le stand ou définir si la voiture s’est crashée ou arrêtée au stand.

En ce qui concerne voiture, c’est la création de la structure voiture utilisée à travers l’entièreté de l’application.

Dernièrement, il y a le main qui exécute progressivement les différents modules afin d’aboutir à un résultat organisé chronologiquement depuis les essais jusqu’à la finale en passant par la création de la mémoire partagée, des sémaphores et des forks.

2. Découpage en modules :



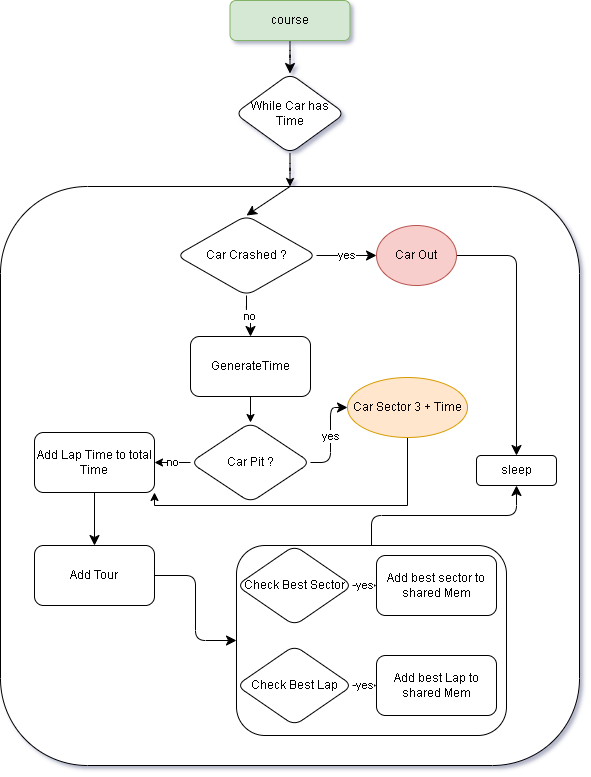
**Module time :**

Ce module est composé de 5 fonctions qui servent à simuler la course. Pour cela, il va générer des temps pour chaque secteur, générer des crashs aléatoires ainsi que des entrées aux stands. Dans le cas de ce dernier un temps d’arrêt aux stands est également généré aléatoirement.

**Module result :**

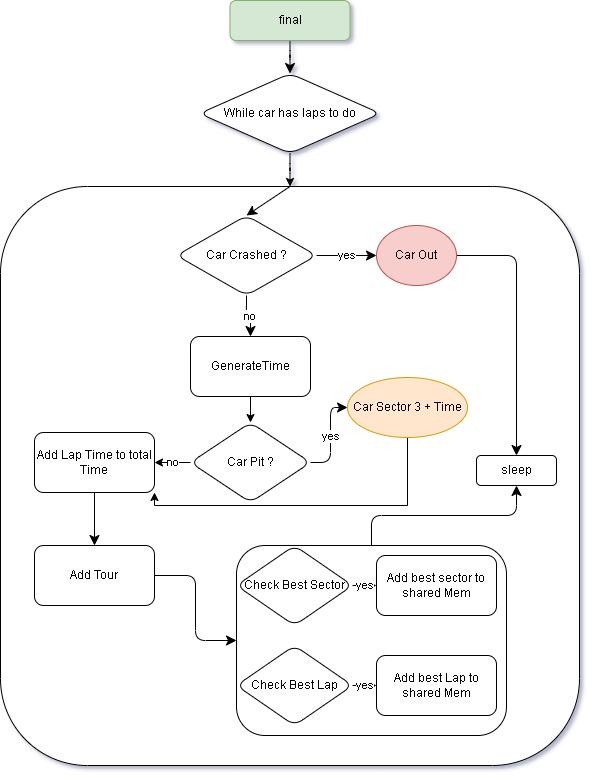
Le module en question est composé de 8 fonctions qui servent à trier de manière adéquate les résultats simulés par les modules course et final pour les courses concernées et afficher les résultats de ces simulations effectuées auparavant. Il va également calculer les différentes informations supplémentaires à afficher lors de la course.

**Module des courses (essais et qualifications) :**



Le module course tourne tant que la limite temporelle prédéfinie de la course concernée n’est pas atteinte. Il vérifie premièrement si un crash n’a pas été généré, si c’est le cas, il modifie le statut de la voiture en out après avoir généré des temps en fonction du secteur dans lequel s’est produit le crash. Ensuite, il endort la voiture concernée pour qu’elle ne génère plus de temps pendant la course. Si le retour de cette vérification est négatif, des temps sont générés pour les secteurs 1 et 2 avant d’entamer la vérification du stand. S'il y a eu un stand un temps de secteur 3 est généré avec l’addition d’un temps de stand. Dans le cas inverse seul un temps de secteur 3 est généré. En poursuivant, il additionne le temps du tour au temps total de la course pour ensuite incrémenter le nombre de tours de la voiture. Ensuite, il vérifie les temps pour chaque secteur effectué durant ce tour et celui du tour par rapport aux meilleurs secteurs et meilleurs tours de la voiture et garde le meilleur des deux en mémoire avant de relancer l’exécution de la boucle. Les résultats renvoyés sont triés sur le meilleur tour de chaque voiture.

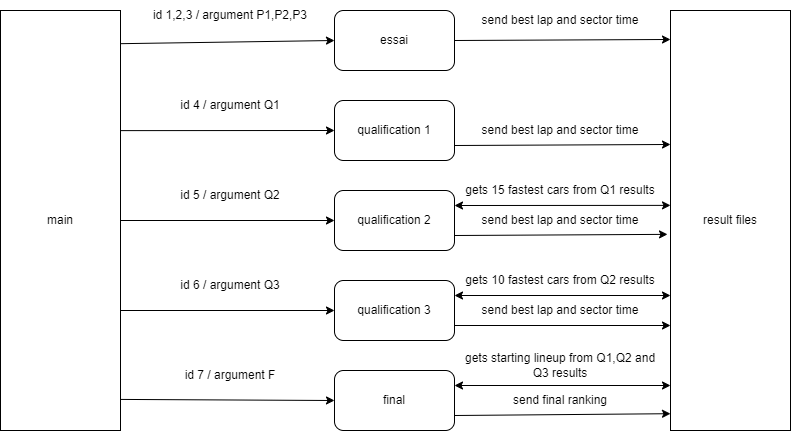
**Module de la finale :**



Le module représenté par le schéma ci-dessus suit une hiérarchie fortement similaire à celui des essais et des qualifications mis à part que celui-ci ne tourne pas en fonction d’un temps prédéfini, mais d’un nombre de tours sur lequel se déroule la course. Les voitures tournent tant qu’elles n'ont pas atteint ce nombre de tours ou que leur statut soit changé à out suite à un crash. Dans ce cas-ci, les résultats sont triés par rapport à la position de la voiture dans le classement. Ce classement est déterminé par la voiture qui a parcouru la distance totale de la course dans le temps le plus court.

3. Plan des modules principaux :

On débute l’exécution du main par l’initialisation des pilotes et l’initialisation des erreurs concernant les arguments que celui-ci reçoit, suivi de la création de la mémoire partagée pour ouvrir un canal de communication entre le processus père et ses fils qui sont les 20 voitures qui vont effectuer les courses. Nous poursuivons avec l’initialisation du fork pour créer lesdits fils.



Celui-ci est directement suivi de la vérification de l’id de la course, les id en question sont définis par l’argument que le programme reçoit avant son exécution, leurs arguments sont définis à la fin du code et renvoient le nombre de voitures qui vont participer à la course ainsi que le temps que celle-ci va durer ou le nombre de tours à effectuer dans le cas de la finale. Dans le cas où l’id est égal à un nombre entre 1 et 6 inclus, le main va faire appel à la fonction course qui va simuler une course en fonction des paramètres reçus. Le module course après avoir simulé l’entièreté de la course va renvoyer et sauvegarder les meilleurs temps de chaque voiture pour chaque secteur et pour un tour complet. Dans le cas des id 1 à 3 qui sont les essais, ces données sont juste sauvegardées dans des fichiers externes.

S'il s’agit des courses avec les ids 4 à 6 ces données sont sauvegardées et réutilisées pour la création du classement pour le départ de la course finale. Les cinq dernières positions sont définies par les cinq dernières voitures de la qualification 1, les cinq positions suivantes (onze à quinze comprises) sont définies par les cinq dernières voitures de la qualification 2, finalement les dix premières positions sont définies par le classement final de la qualification 3. Toutes les 20 voitures participent à la Q1, mais pour la Q2 seules les 15 voitures avec les meilleurs temps de la Q1 pourront participer. Et quand il s’agit de la Q3, les 10 meilleures voitures de la Q2 seront participantes.

Dans le cas où l’id de la course serait égal à 7, c’est le module de la finale qui est exécuté. Celui-ci reçoit le classement de départ créé suite aux qualifications et exécute la course tant que toutes les voitures n’ont pas atteint le nombre de tours demandés ou ne se sont pas crashées pendant la course. Ce module renvoie le classement final en fonction du temps qu’a pris chaque voiture pour effectuer la distance totale de la course, si une voiture se crashe, elle apparaîtra toujours dans le classement, mais elle ne roulera plus. Une fois la course finie, son résultat est sauvegardé dans un fichier externe lui appartenant.

# **Conclusion**

1. Point de vue général :

Nous sommes parvenus à réaliser l’entièreté du programme selon les consignes demandées. La mémoire partagée et les sémaphores ont bien été employée pour gérer le flux des données et la synchronisation entre les différents processus fils et le processus père.

2. Les difficultés rencontrées :

La difficulté majeure dans ce projet a été l’organisation entre les différents membres pour faire coïncider nos calendriers respectifs dans le but d’organiser des réunions. Les impératifs de chacun diminuant les fenêtres disponibles pour s’atteler à la tâche.

D'un point de vue technique, l’enregistrement dans les fichiers a aussi posé quelques problèmes venant de l’utilisation de write qui n’a pas été jugé des plus faciles.

Mais encore, la gestion dynamique des processus, car certaines courses n’ont pas 20 participants. La solution est toute simple, lorsque l’on récupère l’id de la course, on attribue un nombre à une variable NBRVOITURE qui correspond au nombre de participants, cette variable est utilisée par la suite lors du fork afin de créer un nombre prédéfini de processus.

Quelques heures de travail ont été perdues dans l’implémentation de la mémoire partagée de par la complexité de cet ajout, mais aussi le manque d’expérience dans ce domaine.

La majorité des problèmes rencontrés en dehors de ceux cités ci-dessus sont des erreurs de programmation basiques, qui ont nécessité une petite piqûre de rappel du langage C.

3. Ce qu’on aurait pu améliorer :

De manière générale : il serait toujours bon d’améliorer le code pour qu’il corresponde au mieux aux consignes données en début d’année. Certaines parties étant plus importantes que d’autres, la charge de travail a été dirigée en priorité sur ces dernières avant de s’occuper de points considérés plus anecdotiques.

Nous aurions également pu améliorer la synchronisation de l’affichage de la course avec un approfondissement de notre emploi des sémaphores. L’utilisation d’un mutex pour afficher les résultats serait favorable dans notre cas, mais par manque de temps, nous avons préféré nous en passer.

4. Conclusions personnelles :

Tommy Riquet :

Ayant rédigé l’intégralité du code moi-même, je trouve que le programme est assez compliqué à prendre en main au début, mais lorsqu’ on a compris le concept, tout devient plus clair.

La mauvaise organisation du projet a aussi été un gros problème, le manque de participation active du groupe a posé quelques complications au niveau des deadlines.

Mise à part ces quelques problèmes, les nouveaux concepts appris lors de la création de ce projet étaient très intéressants, tant pour l’accomplissement des objectifs que pour le challenge personnel.

Quentin Laruelle :

Mon impression générale sur ce projet une bonne chose pour développer sa maîtrise du langage C malgré une présentation indigeste au début de l’année, beaucoup de nouveaux concepts sont requis pour effectuer l’entièreté du projet. Durant le courant de l’année, le projet fut difficile à entamer et à peaufiner suite à diverses modifications de différentes consignes.

Sebastian Dziemianko :

Je trouve que ce projet est une bonne opportunité de porter un esprit critique sur son avancement personnel et sa compréhension du fonctionnement du langage de développement concerné. Il oblige à utiliser les compétences acquises pendant la progression de l’année pour parvenir à la conception du produit final. J’ai réalisé lors de nombreuses heures de recherche toutes les possibilités que ce langage offrait et les compétences minimes que je possède vis-à-vis de ce que j’ai pu trouver lors de mes recherches pour pouvoir progresser plus loin dans le projet.