Université Catholique de Lille

Projet Architecture Système C

Gestionnaire de tâches différées

FACULTÉ DE
GESTION,
ÉCONOMIE
& SCIENCES

Johann De Almeida | Raphael Bauvin
15/01/2020

Le projet

Le projet portait sur le développement d'un gestionnaire de tâches différées.

Le repository git est disponible à cette adresse : https://github.com/Rafyb/Gestionnaire_taches

Les principaux acteurs de ce projet ont été Raphael BAUVIN et Johann DE ALMEIDA.

Description du projet

Pour commencer nous avons défini le squelette du projet en complétant le header.

Nous avons récupéré les fonctionnalités principales demandées telles que la possibilité de lancer une commande de manière répétée avec un chemin donné.

Mais nous avons aussi innové en incluant des options supplémentaires telles qu'un lancement différé ou encore la possibilité de généré un log d'exécution.

Tout le travail et le déroulement du projet a été fait via GIT, plus précisément GitHub qui permet le partage de repository distant et la notion de pull request et de merge auto-géré.

Nous avons pris soin de créer une architecture pour le projet afin que les différents sources ne se trouvent pas au même endroit que l'application ou encore les fichiers de log.

Nous avons divisé les tâches par fonctionnalité, exemple : Raphael a pu développer le fonctionnement de base tel que le lancement du programme ainsi que les répétitions associées alors que Johann a pu développer certaines options telles que la gestion des fichiers de log ou encore l'affichage des résultats.

Nous avons en effet repris certains principes agiles tels que l'intégration continue via GitHub ou encore la mise en place de Pull Request soumises à validation.

Certaines User Story ont été pensées afin de se recentrer sur le besoin ainsi que des brainstorming afin de mettre en commun les divers avancements mais aussi les difficultés rencontrées.

La plupart de ces principes ont été repris dans le but de nous faire gagner un temps précieux et nous permettre également d'éviter de perdre trop de temps dans la gestion des tests de non-régression.

C'est aussi pour cette raison que nous n'avons pas repris d'autres aspects tel que les phases de Sprint, qui nous auraient ajouté davantage de pression sans pour autant garantir un meilleur résultat.

Il est difficile d'évaluer le temps passé par tache/fonctionnalité car chacune a évolué en même temps que d'autres se créaient.

Nous avions commencé par créer une branche par fonctionnalité mais nous nous sommes vite rendu compte qu'il s'agissait d'une actions supplémentaire inutile car le code impacté par chacun de nous n'était jamais identique à la suite de la répartition des tâches.

Les fonctionnalités qui ont été les plus chronophages resteront surement la mise en place d'un log claire et objectif ainsi celle intégrant le lancement de commande de manière répétée.

Une des difficultés rencontrées a été la gestion des options passées en arguments.

En effet il n’a pas été simple de mettre en place un système qui permette à une option passée à l’application de contrôler l’intégrité de sa variable associée.

Afin de surmonter cette difficulté, nous avons créé une structure options qui contient une variable par option, le programme parcours tous les arguments une première fois, si une option est présente : il contrôle qu'elle est bien utilisée.

Si c'est bien le cas, elle est alors stockée et le compteur d’option créé pour l’occasion est incrémenté. Une fois le traitement terminé le compteur sert d'index pour savoir où commence la ligne de commande sans les options passées en paramètres.

Concernant les limites du projet, nous aurions surement pu avec plus de temps et de moyen à disposition créer une application beaucoup plus complète, intégrant notamment un système de menu mais aussi d’autres fonctionnalités telles que la possibilité de donner à l’application un fichier de script qu’elle aurait pu suivre où encore la possibilité de passer une commande par timestamp ce qui permettrai le lancement de celle-ci un jour précis et non par délais décrémenté.

Le projet nous a tout de même permis de valider certains acquis et de nous renforcer dans l’idée que le C est un langage permettant beaucoup de choses si on se donne les moyens de trouver l’algorithme qui permettra d’atteindre la fonctionnalité recherchée.

Il s’agit d’un langage complet qui reste proche du langage machine, il permet un apprentissage optimal dans le milieu de la programmation car il nécessite un contrôle parfait de l’objet développé.

Ce projet m’a permis (Johann) de créer un outil que je pourrai reprendre en entreprise pour automatiser des tâches plus ou moins rébarbatives et ce sans les erreurs que j’aurais pu rencontrer si je passais par un batch ou autre langage d’interprétation.

Ce projet m'a permis (Raphael) de me familiariser un peu plus avec la programmation sur système Unix via le langage C et de rattraper certaines lacunes que j'avais dans ce langage.

Exemple de commande pouvant être passée

```
./PROGCOMMANDE -L -M -D 1 -P "/SNAP/BIN/EMACS" 3 2 EMACS -T TOTO
```

D’autres exemples de commande sont disponibles dans le README.md.

Quelques chiffres clés


38 commits


4 branches


2 contributors


16 functions