Note : les entrées en gris sont à remplacer par votre propre texte

Rapport de projet de LP25

Diebolt Matthieu TC3

Baptiste Pierson TC3

Belmonte Enzo TC3

Hajeri Ali TC3

Automne 2022

Contexte du projet

Le projet de LP25 est un projet en C ayant pour but d'appliquer les notions de C système vues en cours, TD et TP sur un cas d'usage concret. Pour l'édition de cette année du projet, l'objectif est de fournir une implémentation simplifiée de l'utilisateur *rsync* utilisé pour faire des sauvegardes de fichiers. L'implémentation proposera un fonctionnement séquentiel et un fonctionnement parallèle.

État des lieux des compétences du groupe

Avant la LP25, nous étions tous dans la même classe l'année dernière. Nous avions donc tous à peu près le même niveau en termes de compétences en C, puisque nous avons fait les mêmes projets et suivi les mêmes cours. Nous avions acquis les fondamentaux du C sans trop rentrer dans les détails. Nous connaissions les pointeurs, les structures abstraites, la gestion de fichiers… Cependant, nous connaissions uniquement les bases et n'avions pas vraiment appliqué ces points dans un projet, ou très peu.

Répartition des tâches : planification

La répartition des tâches dans notre groupe a été plutôt compliquée, puisque de base, nous n'étions que 2 dans le groupe (Matthieu et Baptiste). Nous avions donc prévu de nous répartir le travail en deux. Nous avons donc commencé le travail à deux, puis une ou deux semaines avant le premier rendu, Enzo, qui n'avait pas de groupe, nous a rejoint. Nous avons donc dû lui expliquer tout ce que nous avions déjà fait, tout en essayant de finir le premier rendu, ce qui était assez compliqué. Enfin, après le premier rendu, Ali, qui n'avait lui aussi pas de groupe, nous a rejoint et il a donc à nouveau fallu essayer de lui expliquer nos différentes fonctions déjà codées.

Au final, en termes de réelle planification, nous avions prévu, pour la première partie, de faire le code à deux. La moitié des fonctions était attribuée à Matthieu et l'autre moitié à Baptiste. Matthieu avait pour rôle de coordonner le projet. Nous n'avions pas vraiment défini qui ferait quelle fonction, mais chacun faisait la fonction qui lui plaisait ou dans l'ordre qui faisait avancer le projet. Nous pensions passer 2-3 heures par semaine chacun sur le projet. Puis, quand Enzo est arrivé, nous avons essayé de lui donner les 2/3 des fonctions qui restaient, tout en l'aidant.

Pour la seconde partie, nous avions prévu qu'Ali, qui n'avait donc pas pu participer à la première partie, fasse un peu plus de code que les autres, et que Matthieu, qui en avait fait beaucoup sur la première partie, en fasse beaucoup moins. Nous n'avions à nouveau pas attribué des fonctions précises par personne, mais devions communiquer ensemble avant/après de les coder. Nous pensions tous travailler sur le projet la première semaine des vacances de Noël et finir le projet.

Nous avions décidé de ne pas attribuer de réels blocs de fonctions à chacun, car selon nous, les fonctions avaient une certaine logique entre elles et il fallait les coder dans l'ordre. Par exemple, dans certaines fonctions, nous avions besoin d'autres fonctions, donc il fallait coder celles d'avant et nous ne pouvions pas attendre que l'autre fasse sa partie pour commencer.

Il était prévu que Baptiste se charge du rapport.

Répartition des tâches : réalisation

Au final, dans la première partie du code, Matthieu a fait la plus grande partie. Baptiste s'est occupé du fichier « files-properties.c » et Enzo s'est occupé des fonctions copie_entry_to_destination, synchronize, et de l'amélioration de make_list avec pas mal d’aide de Matthieu. Matthieu s'est occupé de tout le reste des fonctions sur la première partie. Il s'est également chargé de modifier les petits bug qui bloquaient la compilation et des tests. Cela nous a pris plus de temps que prévu. Nous estimons cela à 5-6 heures par semaine.

Pour la deuxième partie, au final, Matthieu s'est occupé de la modification des fonctions en fonction des commentaires du feedback. Il a également réalisé les fonctions du fichier «

message.c ». Mais également “make_process”, “clean_process” et le début de “lister_process_loop” et “analyzer_process_loop” dans “processes.c”. Ali a fait le début de la fonction”prepare”.

Au final, c'est Matthieu qui s'est occupé du rapport.

Difficultés rencontrées

Compréhension des objectifs

Nous avons rencontré des difficultés au début pour comprendre ce qui était attendu pour le premier rendu, notamment quelles fonctions il fallait coder ou non. Nous pensions devoir coder les deux premiers "ensemble". Pour résoudre cela, nous avons demandé à d'autres groupes qui nous ont répondu qu'il fallait faire les trois premiers "ensemble". Nous aurions pu communiquer avec les autres groupes plus rapidement.

Nous avons également eu du mal à comprendre le système de liste chaînée de fichiers et de répertoires. D’autre part, nous avons eu du mal à comprendre qu'il fallait rajouter les propriétés des fichiers après avoir fait la liste. Pour résoudre cela, nous avons essayé de coder fonction par fonction sans trop réfléchir à la suite. De plus, nous avons essayé de demander des conseils aux autres groupes. Nous aurions pu remédier à ce problème plus tôt si nous avions lu plus attentivement toutes les fonctions avant de les coder.

Enfin, nous avions du mal à comprendre ce qu'est la somme MD5 d'un fichier. Nous avons donc fait des recherches sur internet pour comprendre. Nous pensons que nous n'aurions pas pu résoudre ce problème plus rapidement.

Réalisation des objectifs

Dans la réalisation du projet, nous avons tout d'abord eu du mal à comprendre la façon de créer les listes. Initialement, nous construisions les listes en ajoutant directement les propriétés, puis nous avons compris qu'il existait une fonction spécialement dédiée à cela. Il a donc fallu recommencer les deux fonctions.

Nous avons également rencontré des problèmes lors de la réalisation de la copie des fichiers, n'ayant pas compris exactement comment procéder. Nous avons donc essayé plusieurs méthodes différentes jusqu'à trouver une manière qui fonctionne.

D’autre part, nous avons eu du mal à comprendre comment modifier la date de modification du fichier. Après quelques recherches, nous avons mieux compris la bibliothèque, ce qui nous a permis de modifier correctement la date du fichier.

Dans la seconde partie, nous avons eu du mal à comprendre les fonctionnements de messages entre processus. Nous avons donc relu plusieurs fois les objectifs attendus pour ces fonctions ainsi que les structures qui les définissent. Nous avons également relu plusieurs fois le sujet du projet qui expliquait ce point. Malgré cela, nous ne comprenions toujours pas parfaitement leur fonctionnement, mais comprenions les bases.

Proposition d’une amélioration

.

Nous pensons qu'un ordre plus précis des fonctions à coder, ainsi que la possibilité de tester les fonctions une par une, faciliterait la compréhension du code attendu. Nous avons cette opinion, car dans le code actuel, nous ne savions pas vraiment ce que la fonction devait précisément accomplir et ce qu'elle devait vraiment retourner. Nous avons essayé de coder notre propre "main" dans chaque fichier pour tester les fonctions une par une. Nous croyons que le fait de coder toutes les fonctions sans les tester directement et de les évaluer dans le "main" final à la toute fin du projet, risque d'entraîner trop d'erreurs de compréhension de fonction.

Nous trouvons également que la compréhension de la seconde partie est compliquée. Nous n’avions encore jamais utilisé des processus en pratique à part dans 2-3 fonctions en tp-td. Nous pensons donc que cette partie manque un peu d’explication ou d’exemples.

Enfin, en termes d’organisation, cela a été compliqué pour nous. Nous pensons que pour améliorer cela, il faudrait composer les équipes beaucoup plus tôt et avoir des équipes définies avant de donner le sujet.

Analyse des résultats

En termes de résultats, nous sommes plutôt fiers de notre rendu de la 1ere partie, nous ne pensions pas être capables de rendre quelque chose qui marche à 2 (puis 3). Ainsi, la 1ere partie du code, selon nous, fonctionne plutôt bien, il manquait quelques détails, mais le code fonctionnait mieux qu’attendu.

En ce qui concerne la seconde partie, les résultats sont beaucoup moins bons. Le groupe n’a pas fonctionné et nous n’avons pas pu finir le code. Ainsi, pour le rendu final, nous n’avons pas de programme complet fonctionnel et n’avons que la partie itérative qui fonctionne.

Retour d’expérience

Le retour d’expérience (REX ou RETEX) d’un projet est un des enseignements majeurs de ce projet. Il consiste à reprendre le déroulement réel du projet, à le comparer au plan, puis à identifier les principaux points où les deux divergent afin d’en examiner les causes, les conséquences et les procédures à mettre en œuvre pour éviter que cela ne se reproduise. Il est fréquent que le RETEX soit pratiqué de manière informelle, ou par un manager n’ayant pas réellement contribué au projet. Certaines structures, en revanche, ont une approche bien plus formelle et enrichissante du RETEX, on pourra par exemple citer les forces armées et l’industrie électro-nucléaire. En effet, procéder à un RETEX complet, sans concession, dont les conclusions impliqueront des mises à jour des doctrines est un processus garant du maintien d’une structure en conditions opérationnelles. Il arrive parfois même de faire le RETEX d’un événement ou d’un projet d’une entité externe (EDF, l’IRSN et l’ASN ont fait un RETEX de l’accident de Fukushima, bien qu’il ait eu lieu sur des réacteurs de nature différente opérés par un autre fournisseur d’électricité, TEPCO).

Notre groupe a très mal fonctionné. Nous n'avions pas la même vision du projet ni les mêmes objectifs, et certains membres travaillaient moins que d'autres à la réalisation du projet. Nous aurions peut-être dû fixer des blocs de code précis à coder pour chaque personne, avec des dates limite. Si ces délais n'étaient pas respectés, les autres membres du groupe auraient pu continuer malgré tout, même avec un morceau manquant. Cependant, dans ce cas, nous aurions peut-être eu un code non fonctionnel. Il aurait peut-être fallu une personne supplémentaire pour coordonner le groupe, une personne plus ferme et capable de motiver davantage les autres (mais il n'est pas certain que nous avions ce profil dans le groupe).

Nous avons un peu négligé la planification, ce qui nous a conduit à coder une plus grande partie du code la dernière semaine avant le premier rendu plutôt que pendant les autres semaines.

Nous avons peut-être eu du mal à nous lancer dans le projet, car nous entendions d'autres dire que le projet était très compliqué, alors qu'en faisant les fonctions une par une, cela n'était pas si compliqué que cela.

Peut-être n'aurions-nous pas dû intégrer de nouveaux membres au groupe en plein projet, car cela rend la compréhension du code déjà réalisé plus difficile pour la personne qui rejoint le groupe.

Au final, peut-être que nous aurions dû nous repartir dans d’autres groupes plus adaptés à la motivation de chacun.

Conclusion

En conclusion, la première partie du code c'est mieux passée que la seconde. Nous avons été capables de rendre quelque chose de correct pour la première partie, mais pas pour la seconde. Nos difficultés ont résidé davantage dans l'organisation et la motivation du groupe que dans le code lui-même.