Rapport de projet de LP25

KOYTAVILOGLU Talha - TC03 / MOLIERES Samuel - TC03 / CERHAK-VACELET Maël - TC03 / JACCAUD--GODEFROY Lorenzo - TC03

Automne 2023

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

1. Compétences en C avant l'UV LP25
   - ✓ Maël : Connaissances de base en C acquises dans l'UV IF2A-B à savoir les variables, les conditions, les boucles, les fonctions et la manipulation (implémentation et utilisation) des tableaux, des pointeurs, des chaînes de caractères et des structures et dans L'UV LO21 à savoir les listes chaînées, les arbres binaires, les FILE/PILE, etc.
     D'autres connaissances en programmation générale comme les notions d'arbres binaires, la compilation, les FILE/PILE acquises en terminale avec la spécialité NSI.

   - ✓ Samuel : Connaissance de base en C acquises en IF2A-B : boucles, fonctions, tableaux, pointeurs, structures, …
     Ainsi qu'inscrit en LO21, parallèlement à l'UV LP25 : découverte des listes chaînées, des arbres, des fonctions de tri, …

   - ✓ Talha : Acquisition de connaissances fondamentales en langage C dans le cadre du cours IF2A-B, englobant les concepts tels que les boucles, les fonctions, les tableaux, les pointeurs, les structures, etc.
     Simultanément, inscription au cours LO21, avec exploration des listes chaînées, des arbres, des techniques de tri, et autres sujets associés, dans le cadre de l'UV LP25.

   - ✓ Lorenzo : Apprentissage des notions fondamentales en langage C au cours de l'UE IF2A-B, avec la manipulation de conditions, les fonctions, les boucles, les tableaux et pointeurs, entre autres.
     Également, avec mon inscription à l'UE LO21, découverte du fonctionnement des listes chaînées, de différentes techniques de tri et des arbres, parallèlement abordé en LP25.

1. Compétences tirées de l'UV en C, TD et TP.

Pour l'ensemble des membres du groupe, l'UV LP25 a été l'occasion de découvrir les notions de compilation séparée et d'approfondir nos connaissances sur la gestion de mémoire, les Entrées/sorties sur fichiers et répertoires, les pointeurs et les tableaux.
De plus, nous avons découvert ce qu'est la compilation de code C avec le système d'exploitation linux.

Répartition des tâches : planification

Une réunion a été préalablement organisée au sein du groupe au courant de novembre. Cette dernière a été l'occasion de s'assurer de la bonne compréhension du sujet par tous les membres de l'équipe de projet et de répartir le travail à faire sur la première partie du projet, à savoir produire une version du projet **séquentielle** fonctionnelle qui fera l'objet d'un premier rendu prévu pour le 10 décembre. Nous avons en effet identifié deux étapes majeures sur ce projet. La première étape était la partie de fonctionnement séquentiel. Pour cette partie nous avons alors repéré les fonctions qu'il fallait compléter, et nous avons alors séparé le travail en trois parties :

- ✓ La première partie s'inspire de l'ensemble 1, en regroupant l'analyse des options du programme et la gestion des listes chaînées de fichiers.
- ✓ La deuxième partie s'inspire de l'ensemble 2, en regroupant opérations de tests sur les répertoires, la recherche des informations de fichiers/dossiers (stat) et le calcul de la somme de contrôle des fichiers (MD5).
- ✓ La troisième partie s'inspire de l'ensemble 3, en regroupant l'analyse séquentielle et la copie des fichiers à synchroniser.

Les tâches ont été identifiées et réparties par personnes de la manière suivante :
- ✓ Lorenzo : implémentation de fonctions du fichier *file-properties.c* + implémentation de fonctions du fichier *processes.c* (durée estimée 3 h)
- ✓ Talha : implémentation des fonctions du fichier *configuration.c*, *file-list.c* et partiellement *processes.c* (durée estimée 5h)
- ✓ Samuel : implémentation des fonctions du fichier *file-properties.c* + *main.c* + *message.c* (durée estimée 4h)
- ✓ Maël : Implémentation des fonctions du fichier *sync.c* + message.c + *utility.c (durée estimée 6h)*.

Nous avons fait le choix de séparer le travail avec les fichiers.c à compléter, ce qui permet une compréhension globale de chaque partie traitée par les différents membres du groupe.

Répartition des tâches : réalisation

Dans la pratique, notre planification des tâches a été plutôt bien respectée. Chaque membre du groupe à effectuer le travail qui lui était assigné avec si besoin un peu d'aide de la part des autres membres.

Au final, les tâches suivantes ont été réalisées dans cet ordre par :

- *configuration.c* → Talha (durée réelle : 1h30)
- *file-list.c* → Talha (durée réelle : 1h30)
- *file-properties.c* → Lorenzo et Samuel (durée réelle 4h)
- *sync.c* → Maël *(durée réelle : 6h)*
- *message.c* → Samuel et Maël *(durée réelle : 3h)*
- *utility.c* → Maël *(durée réelle : 20 min)*
- *processes.c* → Lorenzo et Talha (durée réelle : 4h)

Pour ce qui est du RETEX, il a été complété petit à petit au cours du projet.
Enfin, le rapport a été écrit par l'ensemble des membres du groupe.

Difficultés rencontrées

Les difficultés rencontrées ont été multiples. La première difficulté a été de comprendre le projet, qui semblait long et compliqué, et ensuite de saisir ce qui était demandé pour chaque fonction. Ensuite, il a fallu comprendre au maximum chaque fonction pour pouvoir réutiliser des exemples du cours ou des TD/TP. La difficulté a également été très présente lorsque nous n'avons pas trouvé d'exemple concret de codes similaires utilisés en cours ou en TD/TP pour pouvoir les réutiliser. Pour certaines fonctions, nous avons donc dû effectuer des recherches pour pouvoir réaliser pleinement ce qui nous était demandé.

Nous avons également remarqué une difficulté au niveau de l'organisation. Il a été en effet très difficile de nous organiser en fonction de nos emplois du temps et de nos temps libres pour mener à bien ce projet.

Enfin, il nous a été très difficile de tester nos programmes, car certaines fonctions dépendent de valeurs d'autres fonctions. En plus de cette particularité, ayant réalisé globalement ce projet sur nos PC personnels, il nous a été très difficile de tester les fonctionnalités des fonctions directement sur LINUX ou même de tout compiler sur LINUX directement. En effet, certains membres de notre groupe étaient habitués à réaliser les TD et TP directement sur les machines de l'UTBM et non sur leur propre machine avec des Virtualbox, qui sont parfois compliqués à utiliser.

Compréhension des objectifs

Concernant la partie sur les tests des répertoires et la recherche des informations de fichiers/dossiers, il était écrit dans le fichier *main.c* qu'il fallait contrôler si la destination existait et pouvait être modifiée, ou s'il n'existait pas, auquel cas il fallait la créer. Or, il n'y avait qu'une seule fonction qui vérifie l'existence d'un répertoire. De plus, cette fonction était appelée presque en même temps pour le répertoire source et la destination. Ainsi, cette fonction ne contrôlait uniquement que l'existence des dossiers, tandis qu'une autre fonction vérifiait si la destination pouvait être modifiée. Il manquait donc la condition de création de la destination si elle n'existait pas.

Pour remédier à ce problème, il a fallu rajouter une partie de code dans le fichier main.c, entre la configuration des valeurs et le contrôle de l'existence des répertoires. Cette partie de code vérifie uniquement si la destination existe, et dans le cas où elle n'existe pas, essaie de la créer. Ainsi, toutes les conditions de l'énoncé seront vérifiées.

Réalisation des objectifs

Problème : Gestion des listes de fichiers
Symptômes : Erreurs dans l'ordre des listes, incohérences dans la comparaison entre listes source et destination.
Solutions : Révision de la logique de construction des listes, ajout de vérifications supplémentaires.
Anticipation : Analyse approfondie lors de la conception des fonctions pour éviter les erreurs.

Problème : Implémentation multiprocessus
Symptômes : Problèmes de synchronisation entre processus, difficultés dans la communication inter-processus.
Solutions : Réorganisation de la création et de la terminaison des processus, ajout de signaux pour gérer les opérations.
Anticipation : Planification détaillée du flux des opérations entre processus pour éviter les problèmes de synchronisation.

Problème : Pertes ou mauvaise gestion de messages.
Symptômes : Messages manquants ou mal interprétés, impactant les résultats attendus.
Solutions : Revoir la logique de gestion des files de messages, implémenter des vérifications pour assurer la réception et le traitement adéquat des messages.
Anticipation : Planifier une analyse minutieuse des échanges de messages, établir des protocoles de communication clairs pour éviter les problèmes de transmission.

Proposition d’une amélioration

Pour ce qui est du problème du contrôle de l’existence du répertoire destination et de sa création s’il n’existe pas, pourquoi ne pas séparer la fonction *directory_exists* en deux fonctions : une pour le contrôle du répertoire source qui vérifiera simplement l'existence du répertoire, et une nouvelle fonction pour le contrôle de l’existence du répertoire destination et la création de ce répertoire s’il n’existe pas encore. Ainsi, la fonction *main.c* serait plus courte

Analyse des résultats

N’ayant pas une compilation finale et globale du projet, nous n’avons pas de résultats spécifiques analysable. En revanche, certaines fonctions ont été testé séparément ce qui nous a permis d’en attester leur bon fonctionnement. Des tests systématiques et un temps pour compiler et débuguer le projet auraient été nécessaire pour obtenir les résultats finaux.

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).

Pour répondre à la question « si je devais refaire le projet maintenant avec ma compétence acquise par le projet, que ferais-je différemment ? » nous avons remarqué que nous changerions surtout notre organisation. En effet, nous avons réussi à travailler en groupe mais nous pensons que si nous nous étions vus plus tôt et plus régulièrement, pour discuter du projet et de nos incompréhensions communes, nous aurions pu avancer plus rapidement et avec plus d'efficacité. Le travail a bien été réparti, mais il aurait peut-être fallu plus communiquer entre nous pour nous expliquer mutuellement à quoi servent exactement les fonctions sur lesquelles chacun d'entre nous travaillait. Ainsi la compréhension globale du sujet aurait pu être améliorée.

Pour continuer sur cet aspect d'organisation, nous pensons également que si c'était à refaire nous aurions commencé le projet le plus tôt possible en avançant un maximum dès le début de sorte à pouvoir bénéficier d'un feedback complet. Avoir l'opportunité d'avoir un feedback plus conséquent est une chance, car ce dernier nous a permis d'avoir une vérification de notre code, ainsi que des retours sur ce dernier. Ces différents retours étaient tous très précieux et nous permettaient de corriger nos fonctions.

Enfin, le dernier point à améliorer serait le test des différentes fonctions au fur et à mesure qu'on avançait sur nos fonctions respectives. En effet, ce projet était composé de nombreuses fonctions à compléter et il aurait fallu tester chacune des fonctions au préalable pour faciliter l'implémentation finale sur LINUX.

Conclusion

Ce projet a été pour nous une réelle expérience. Il nous a permis de découvrir une certaine organisation à avoir lorsqu'on réalise un projet en groupe de 4. C'était pour nous l'occasion de découvrir également l'utilisation de *github*, qui était nouveau pour nous et qui s'est révélé être très utile.

Au niveau du groupe nous n'avons pas eu de soucis particuliers. Chaque membre a joué le jeu et réalisait les tâches qui lui était attribué. Comme déjà dit précédemment, nous aurions peut-être pu communiquer davantage sur l'avancement de chacun et sur notre compréhension mutuelle du projet et des différentes fonctions.

Malgré nos efforts pour finir toute la partie codage des différentes fonctions nous avons tout de même rencontrer des problèmes lors de l'implémentation des différentes fonctions sur LINUX. En effet, c'est une partie du projet que nous avions sous-estimé et nous reconnaissons qu'il nous a été très difficile de tester notre code sur l'interface LINUX dans une VirtualBox. C'est pourquoi nous n'obtenons pas réellement de résultats cohérents lors des tests du code.

Mais globalement en plus d'avoir amélioré nos connaissances sur le codage C et sur l'interface LINUX, ce projet nous aura également appris la rigueur à faire preuve lors de la réalisation de projet de ce genre, de l'importance de l'organisation et de la communication au sein d'une équipe et de l'importance du travail d'équipe.