Rapport - Projet

LP25

JANVIER 2025
LP25 - Automne 2024

WALLE – JOURNET – LABBI
Armand – Célia – Marouane

Contexte du projet

Le projet consiste à construire une solution de sauvegarde incrémentale d'un répertoire source (le répertoire à sauvegarder) vers un répertoire cible (le répertoire de sauvegarde). L'objectif est de développer un programme en langage C pour créer un outil de sauvegarde inspiré de Borg Backup, avec un accent sur la déduplication des données et la possibilité de réaliser des sauvegardes sur un serveur distant via des sockets.

Il est important de noter que nous n'avons jamais vu ce genre de projet ou de manipulation en cours ou dans quelque contexte que ce soit. Ce projet représente donc une nouvelle exploration pour nous, nécessitant une compréhension approfondie des concepts de sauvegarde incrémentale, de déduplication des données et de communication réseau via des sockets.

Compétences du groupe

Avant de commencer le projet LP25, notre groupe avait une expérience variée en programmation en langage C. Les cours, travaux dirigés (TD) et travaux pratiques (TP) nous ont permis d'acquérir une base solide en syntaxe et en concepts fondamentaux du langage C.

Célia :	Avant le projet, j'avais une bonne maîtrise du langage C, mais je n'avais pas encore travaillé sous Linux, un système que je n'ai pas sur mon PC. De plus, je manquais d'expérience avec la manipulation de fichiers et de répertoires en C.
Marouane :	Avant le projet, je n'avais aucune idée de la gestion des fichiers sous Linux, étant donné que je travaille principalement sous Windows. Cependant, j'avais une connaissance de base du langage C, bien que je n'aie pas encore exploré la gestion des fichiers en mode binaire.
Armand :	Je code depuis presque 10 ans, ce qui me donne une solide expérience en programmation. Avant LP25, j'avais déjà travaillé avec le langage C dans différents projets, ce qui me permet d'aborder ce projet avec une certaine aisance.

Objectifs du Projet

Sauvegarde Incrémentale

L'un des objectifs principaux de ce projet est de développer un mécanisme de sauvegarde incrémentale. Contrairement aux sauvegardes complètes qui copient l'intégralité des données à chaque fois, la sauvegarde incrémentale permet de ne sauvegarder que les fichiers et les parties de fichiers qui ont été modifiés depuis la dernière sauvegarde. Cela réduit considérablement le volume de données à transférer et à stocker, optimisant ainsi les ressources disponibles. En outre, il est crucial d'assurer une gestion efficace des versions des fichiers. Cela permet de restaurer précisément les fichiers à un état antérieur, offrant une flexibilité et une sécurité accrues en cas de besoin.

Déduplication des Données

Un autre objectif clé est l'implémentation d'algorithmes de déduplication des données. La déduplication consiste à identifier et à éliminer les données redondantes, de sorte que chaque élément de données soit stocké une seule fois. Cela permet d'optimiser l'utilisation de l'espace de stockage, ce qui est particulièrement important lorsque l'on traite de grandes quantités de données. En évitant de stocker des doublons, nous pouvons non seulement économiser de l'espace, mais aussi améliorer les performances globales du système de sauvegarde. La déduplication nécessite des algorithmes sophistiqués capables de détecter efficacement les similitudes entre les fichiers et les parties de fichiers.

Sauvegarde sur Serveur Distant

Enfin, le projet vise à permettre la sauvegarde sur un serveur distant via des sockets. Les sockets sont des interfaces de communication qui permettent l'échange de données entre deux machines sur un réseau. En utilisant des sockets, nous pouvons établir une communication fiable entre le client (la machine locale) et le serveur (la machine distante). Cela permet de transférer les données de sauvegarde de manière sécurisée et fiable, même sur des réseaux potentiellement instables. Assurer la sécurité des données pendant la transmission est essentiel pour protéger contre les accès non autorisés et les pertes de données. Des mécanismes de chiffrement et d'authentification peuvent être mis en place pour renforcer cette sécurité.

Répartition des tâches

Planification - Réalisation

Planification

Pour mener à bien notre projet de sauvegarde incrémentale, nous avons identifié plusieurs tâches clés et les avons réparties en fonction des compétences et de l'expérience de chaque membre de l'équipe. Voici la description des tâches, leur affectation et les estimations de durée pour chacune.

Tâche	Affectation	Durée estimée (heures)
Analyse du modèle de projet	Toute l'équipe	2
Évaluation de la difficulté des fichiers	Toute l'équipe	2
Développement du module Main	Toute l'équipe	5
Développement du module File Handler	Armand	5
Développement du module Deduplication	Armand	10
Développement du module Backup (sauvegarde)	Célia	15
Développement du module Backup (restauration)	Marouane	12
Total		51

- Le module **main** est le point d'entrée du programme. Il initialise les autres modules et coordonne les opérations de sauvegarde et de restauration
- Le module **Backup (sauvegarde)**gère la sauvegarde des fichiers en identifiant et copiant uniquement les fichiers modifiés depuis la dernière sauvegarde.
- Le module **deduplication** élimine les données redondantes en utilisant des algorithmes de déduplication pour optimiser l'espace de stockage.
- Le module **file handler** gère les opérations de base sur les fichiers et les logs, fournissant des fonctionnalités pour lister les fichiers et lire les logs.
- Le module **Backup (restauration)**restaure les fichiers à partir des sauvegardes incrémentales, effectuant le travail inverse du module **backup**.

Ces estimations de durée sont basées sur notre évaluation initiale et peuvent être ajustées en fonction de l'avancement du projet et des défis rencontrés.

Réalisation

La réalisation du projet s'est déroulée différemment de la planification initiale, avec une répartition des tâches qui a évolué selon les difficultés rencontrées et les compétences de chacun.

Tâche	Affectation	Durée estimée (heures)
Développement du module Utilities	Armand	10
Développement du module Main	Toute l'équipe	10
Développement du module File Handler	Armand	15
Développement du module Deduplication	Armand	20
Développement du module Backup (sauvegarde)	Célia	25
Développement du module Backup (restauration)	Marouane	15
Débogage	Toute l'équipe	15~20
Total		115

- Le module **Utilities** contient des fonctions utilitaire permettant par exemple la gestion des chemins et dates avec test d'intégration.
- La partie **débogage** intervient lors de la Correction des fuites de mémoire, résolution des problèmes d'intégration et tests du système en additionnant tout nos scripts.
- Le module **réseaux** avait été prévu comme étant un simple module de transfert d'un client à un serveur. Cependant en raison du manque de documentation et d'exemples surtout avec le temps qu'on avait qui coïncidait avec les dernières 2 semaines ou nous nous préparions pour nos finaux, nous n'avons pas été en mesure de l'implémenter complétement. Un première implémentation a cependant eu lieu mais sans aller plus loin que les bases.

Le temps total réel du projet s'est élevé à 115 heures, soit plus du double de l'estimation initiale. Cette augmentation est principalement due à la complexité du système de déduplication, qui a nécessité plusieurs modifications au cours de la réalisation du projet, et aux difficultés rencontrées dans la gestion de la mémoire

Difficultés rencontrées

Compréhension - Réalisation

Compréhension

Système de déduplication

Instruction incomplète
La notion de "chunks" et leur utilisation dans la déduplication n'étaient pas clairement définies. On s'est longtemps interrogé sur la manière de gérer les chunks et leur stockage.

Compréhension complétée
On a pu éclaircir ce point grâce à l'exemple donner dans le README qui nous a aidé à visualiser concrètement le concept tout en discutant entre nous afin de clarifier le fonctionnement des chunks et leur stockage.

Sauvegarde incrémentale

Instruction incomplete
La logique de comparaison entre les fichiers sources et les sauvegardes existantes n'était pas évidente.

Compréhension complétée
Clarification à travers nos discussions sur l'utilisation des dates de modification et des MD5 avec des tests pratiques pour comprendre quand un fichier devait être sauvegardé ou non.

Remédiation
Un diagramme de flux décisionnel pour la sauvegarde incrémentale aurait facilité la compréhension initiale.

Coordination des modules

Instruction incomplète
Les interactions entre les différents modules (file_handler, deduplication, backup_manager) n'étaient pas clairement définies.

Compréhension complétée
On a dû définir clairement les responsabilités de chaque module et faire des tests d'intégration pour valider les interactions.

Remédiation
Un diagramme d'architecture initial aurait facilité la compréhension des interactions entre modules.
Ces difficultés de compréhension ont significativement impacté le temps de développement, ce qui nous a obligé à faire plusieurs itérations pour certaines fonctionnalités.
Et aussi pour la partie de réseau où on n'a pas pu progresser puisque les spécifications pour l'implémentation réseau étaient très limitées, sans documentation ni exemples. Une documentation détaillée des exigences réseau dès le début du projet aurait été appréciée.

Réalisation

Gestion de la mémoire

Symptômes
Fuites de mémoire importantes détectées et erreurs de segmentation lors de l'exécution.

Résolution
Révision des allocations et libérations de mémoire dans build_full_path et on a corrigé la gestion de mémoire dans list_files et create_backup avec implémentation des fonctions de nettoyage (free_file_list, free_log_list)

Gestion des chemins de fichiers

Symptômes
Problèmes avec les slashs dans les chemins, erreurs de la construction des chemins complets et difficultés avec les chemins relatifs vs absolus.

Résolution
On a créé les fonctions build_full_path ainsi que remove_trailing_trash en plus de l'implémentation de cut_after_first_slash et remove_source_dir avec une standardisation de la gestion des chemins.

Déduplication des fichiers

Symptômes
La déduplication des fichiers a posé le plus de défis techniques, notamment des problèmes avec les bits de marquage, une taille de chunks incorrecte et des erreurs lors de la restauration des fichiers.

Résolution
Ces difficultés étaient principalement liées à la gestion binaire des données. Par exemple, dans la fonction deduplicate_file, la gestion des types de chunks et la création de sub-chunks de référence ont nécessité une attention particulière pour éviter les erreurs de mémoire et les références incorrectes. De plus, la fonction undeduplicate_file a dû être soigneusement révisée pour corriger la gestion des types de chunks et améliorer la gestion des données et des références. Ces ajustements ont permis de résoudre les problèmes de bits de marquage et de taille de chunks, tout en assurant une restauration correcte des fichiers.

Proposition d’améliorations

Interface de gestion des sauvegardes

Le modèle actuel présente plusieurs limitations, notamment dans la gestion et la manipulation des sauvegardes. Une amélioration significative serait l'implémentation d'une interface de gestion des sauvegardes plus robuste. Cette interface pourrait inclure des fonctionnalités telles que l'amélioration de la déduplication grâce à un cache de chunks fréquemment utilisés et l'optimisation de la recherche des doublons, réduisant ainsi l’empreinte mémoire. De plus, l'intégration de statistiques et de monitoring permettrait un suivi précis du taux de déduplication et une mesure des performances avec une journalisation détaillée. Enfin, une meilleure gestion des ressources, incluant une gestion d’erreurs améliorée, la libération automatique des ressources et la réduction des fuites mémoire, contribuerait à une utilisation plus efficace et fiable du système de sauvegarde.

Changement de code

Création d'un nouveau module "backup_controller" avec de nouvelles fonctionnalités

```c
typedef struct {
    char *backup_root;     // Répertoire racine des sauvegardes
    char *current_backup;  // Sauvegarde en cours
    BackupConfig config;   // Configuration de la sauvegarde
    BackupStats stats;     // Statistiques de sauvegarde
} BackupController;

// Gestion du cycle de vie des sauvegardes
void init_backup_controller(BackupController *ctrl, const char *root);
void cleanup_backup_controller(BackupController *ctrl);

// Manipulation des sauvegardes
BackupResult start_backup(BackupController *ctrl);
BackupResult pause_backup(BackupController *ctrl);
BackupResult resume_backup(BackupController *ctrl);
BackupResult cancel_backup(BackupController *ctrl);

// Gestion des erreurs et reprise
```

Ces améliorations permettraient de rendre le code plus robuste et plus facile à maintenir, tout en offrant de nouvelles possibilités pour la gestion des sauvegardes.

En conclusion, les améliorations proposées pour l'interface de gestion des sauvegardes visent à surmonter les limitations actuelles en matière de manipulation et de gestion des sauvegardes. La création d'un nouveau module "backup_controller" avec des fonctionnalités avancées, telles que l'optimisation de la déduplication via un cache de chunks et la réduction de l'empreinte mémoire, permettra d'améliorer significativement l'efficacité du système. L'intégration de statistiques et de monitoring offrira un suivi précis des performances, tandis qu'une meilleure gestion des ressources et des erreurs assurera une utilisation plus fiable et sécurisée. De plus, l'ajout de fonctionnalités réseau, bien que partiellement implémentées en raison du manque de documentation, d'exemples et du temps limité disponible, constitue une étape importante pour permettre le transfert de données entre un client et un serveur. Ces améliorations rendront le code plus robuste et facile à maintenir, tout en ouvrant de nouvelles possibilités pour la gestion des sauvegardes.

Retour d’expérience (RETEX)

Planification et Gestion du Temps

Au départ, on pensait boucler ça en 51 heures. Au final, on en a mis 105. C'est le genre de chose qui arrive quand on manque d'expérience sur des projets complexes. La prochaine fois, on sait qu'il faudra découper les tâches plus finement et prévoir une marge pour les imprévus et le temps d'apprentissage.

Compréhension Technique

On a eu une certaine difficulté avec le concept de déduplication ce qui nous a donné un retard de 20h sur le développement du module et c’était à cause du manque de recherche approfondi préalable au sujet (ce qu’on devait faire depuis le début). De plus, le fait que seulement un membre de l’équipe puisse avoir accès à linux pour le projet a beaucoup impacté notre retard.

Tests et Validation

Les fuites mémoire, on les a découvertes sur le tard, du fait que la compilation tardive du travail de chacun, ce qui nous a fait perdre beaucoup de temps de débogage en fin de projet. On devrait intégrer des tests dès le début pour être plus sure.
Cette expérience nous a permis d'identifier plusieurs axes d'amélioration cruciaux pour nos futurs projets :

- Adopter une approche plus structurée dans l’estimation des taches et intégrer des tests dès le début du développement.
- Documenter les décisions techniques importants.
- Prototyper les fonctionnalités complexes et investir plus de temps dans la phase de recherche.

Ce projet nous a montré l'importance d'une approche plus professionnelle et méthodique dans le développement logiciel, compétences essentielles pour notre future carrière d'ingénieur.

Communication et Délégation

Nous avons constamment dû demander au responsable de projet pourquoi certaines choses ne fonctionnaient pas. Dans l'attente de ses réponses, nous ne pouvions rien faire, ce qui a considérablement ralenti notre progression. Malgré ses réponses, le projet en lui-même n'était pas prêt du tout pour être donné aux élèves, étant donné que les consignes ont continué à apparaître jusqu'à trois jours avant le rendu final. Il est regrettable de constater qu'un projet dans un tel état d'inachèvement ait été proposé aux élèves, ce qui souligne l'importance d'une meilleure préparation et d'une communication plus fluide entre les responsables de projet et les équipes de développement. Cette situation met en lumière la nécessité de consignes claires et définitives dès le début du projet pour éviter des ajustements de dernière minute qui compromettent la qualité et la faisabilité du travail (comme avec la partie réseaux de notre programme).

Conclusion

Ce projet a été une expérience riche en enseignements, bien que marquée par de nombreux défis et obstacles. Dès le départ, notre estimation initiale de 51 heures s'est révélée largement insuffisante, nécessitant finalement 115 heures de travail. Ce dépassement de temps est principalement dû à notre manque d'expérience avec des projets complexes et à une sous-estimation des tâches à accomplir. Nous avons appris l'importance de découper les tâches plus finement et de prévoir une marge pour les imprévus et le temps d'apprentissage.

Sur le plan technique, la compréhension du concept de déduplication a posé des difficultés, entraînant un retard significatif dans le développement du module. Ce retard aurait pu être évité avec une recherche plus approfondie dès le début du projet. De plus, les fuites mémoire, découvertes tardivement en raison de la compilation tardive du travail de chacun, ont nécessité un débogage intensif en fin de projet. Cela nous a montré la nécessité d'intégrer des tests dès le début du développement pour détecter et corriger les problèmes plus tôt.

La communication avec le responsable de projet a également été un facteur limitant. Nous avons souvent dû attendre ses réponses pour avancer, ce qui a ralenti notre progression. De plus, les consignes ont continué à évoluer jusqu'à trois jours avant le rendu final, rendant le projet difficile à finaliser dans les délais impartis. Il est regrettable que le projet n'ait pas été prêt à être donné aux élèves dans ces conditions, soulignant l'importance d'une meilleure préparation et d'une communication plus fluide.

Cette expérience nous a permis d'identifier plusieurs axes d'amélioration cruciaux pour nos futurs projets : adopter une approche plus structurée dans l’estimation des tâches, intégrer des tests dès le début du développement, documenter les décisions techniques importantes, prototyper les fonctionnalités complexes, et améliorer la communication avec les responsables de projet. En fin de compte, ce projet nous a montré l'importance d'une approche plus professionnelle et méthodique dans le développement logiciel, compétences essentielles pour notre future carrière d'ingénieur.