Rapport de soutenance 1

File Explorer

Dimanche, 25 Février 2024

Arthur Garraud

Tristan Faure

Iliane Formet

GroupAléa2

Sommaire

1 Introduction

Il existe une multitude d'explorateurs de fichiers, notamment ceux du système d'exploitation de votre ordinateur comme Finder pour MacOS ou File Explorer pour Windows (pour ne citer qu'eux).
Cependant, nous avons remarqué que l'algorithme de recherche de ceux-ci peut s'avérer être extrêmement lent et que le design pouvait être amélioré pour une meilleure expérience utilisateur.
Nous avons donc décidé de créer notre propre explorateur de fichiers, avec une interface plus agréable et de meilleurs algorithmes.
Cet explorateur de fichiers utilisera une interface graphique.
Il nous permettra d'apprendre à maîtriser Rust, à comprendre et implémenter la manipulation de fichiers et dossiers et de savoir concevoir une interface utilisateur simple et efficace.
Notre objectif est de réussir à créer un explorateur de fichiers qui sera non seulement plus esthétique (design simple sans superflu), mais surtout plus performant.
Cet explorateur sera réalisé en Rust et disponible uniquement sur Linux.

2 Présentation du groupe

2.1 Arthur Garraud

Passionné par l'informatique depuis des années, j'ai toujours été curieux et autodidacte dans de nombreux domaines. J'ai réalisé de nombreux projets qui m'ont permis d'apprendre à maîtriser des outils puissants, que ce soit des langages de programmation ou bien des applications de 3D ou de graphisme.
J'apprécie également le cinéma et j'ai l'habitude de me passionner pour un sujet précis pendant quelques semaines.
Maintenant j'ai décidé de finir ingénieur en informatique afin de pouvoir avoir la possibilité de travailler sur des projets et des missions très différentes de l'une à l'autre.

2.2 Iliane Formet

Depuis mon plus jeune âge, j'ai toujours été passionné par la technologie, la logique et les casse-tête. Ingénieur était donc ce à quoi je me destinais.
Lors d'un projet de labyrinthe pour robot en terminale, j'ai découvert les algorithmes et la programmation. Je me suis alors intéressé et ai commencé à toucher un peu à tout ce qui touche à ce domaine (jeux vidéo, sites internet, algorithmes de calcul). J'ai alors décidé de me tourner vers EPITA pour mes études. Je souhaite dans le futur utiliser mes compétences acquises en école à travers les cours, les conférences et les projets pour entreprendre.

Mes autres loisirs sont le sport, les jeux vidéo, la musique, le cinéma, la nourriture et les sorties entre amis.
Avec ce projet, j'espère perfectionner mes fondamentaux en Rust, mais aussi mieux comprendre la relation entre une interface et des algorithmes.

2.3 Tristan Faure

Plongé dans l'informatique depuis mes 10 ans par mon oncle. Toujours passionné par de nouveaux langages informatiques, j'ai longtemps cherché le domaine qui me passionne le plus en informatique.
J'ai premièrement commencé en autodidacte à apprendre le codage web, commençant par des sites avec HTML, CSS et JavaScript puis je me suis vite tourné vers la programmation de jeux vidéo en faisant de nombreux jeux sur navigateur, certains plutôt complet, comportait plusieurs mondes à la manière d'un Mario.
Je suis également passé par la programmation en python à l'école et seul afin de créer des bots discords et en LUA lorsque j'ai créé mon serveur GTA 5 RP en début d'année 2020. Toutes ces expériences et les longues heures de conférence et de tuto que j'ai regardé m'ont finalement orienté vers la data science et l'Intelligence Artificiel. J'ai donc décidé de rejoindre Epita pour atteindre mes objectifs. Je pense que ce choix m'aidera à progresser rapidement et efficacement.

3 Avancement du projet

3.1 Avancement prévu

Voici le tableau de l'avancement des tâches prévues pour la première soutenance :

Tâche	**Avancement**
UI	Prise en main de l'outil de création d'interface et première interface basique
Commandes bash	Implémentation des commandes bash de base
Algorithmes	Création d'un algorithme de recherche non optimisé

3.2 Rappel de la répartition par membre

Resp = Responsable et Supp = Suppléant

Tâche / Membre	**Arthur**	**Iliane**	**Tristan**
Gestion du projet	-	Resp	Supp
UI	-	Supp	Resp
Commandes bash	Supp	Resp	-
Algorithmes	Resp	-	-
Site internet	Supp	-	Resp

3.3 UI

Dans le cadre de ce projet, j'ai consacré un effort considérable au développement d'une application en Rust en utilisant la bibliothèque GTK4 via gtk4-rs. L'objectif était de créer une interface utilisateur (UI) robuste et intuitive, mettant en avant trois éléments clés, chacun offrant une fonctionnalité unique et enrichissante.

La première composante majeure que j'ai développée est une barre de recherche dynamique. Conçue pour être élégante et fonctionnelle par la suite, cette barre de recherche se dévoile gracieusement lorsqu'un utilisateur clique sur une icône de loupe discrète intégrée dans le header de l'application sur la droite de l'écran. Cette transition fluide attire l'attention de l'utilisateur sur la fonction de recherche sans compromettre l'esthétique de l'interface. Une fois activée, la barre de recherche permet à l'utilisateur de saisir des termes de recherche et de voir instantanément les résultats correspondants s'afficher dans un champ de texte dédié. Cette fonctionnalité de mise à jour en temps réel garantit une expérience de recherche efficace et réactive.
En parallèle, j'ai intégré un bouton interactif dans l'application, permettant de lancer un jeu de pile ou face basique. Inspiré par le plaisir simple des jeux de hasard, cette fonctionnalité offre aux utilisateurs une parenthèse divertissante au sein de l'application et m'a permis de prendre en main différentes fonctionnalités de gtk4-rs. Grâce à l'utilisation d'un générateur de nombres aléatoires, chaque clic sur le bouton affiche le résultat de ce pile ou face.

Enfin, j'ai inclus un affichage de l'heure et de la date dans le header de l'application mais sur la gauche cette fois, offrant ainsi aux utilisateurs une référence temporelle précise et fiable. Cette fonctionnalité, bien que discrète, est essentielle pour de nombreuses applications, notamment celles où le timing est critique. En se mettant à jour automatiquement pour refléter l'heure et la date actuelles du système, cet élément de l'interface assure une synchronisation précise avec le contexte temporel de l'utilisateur, améliorant ainsi la fonctionnalité globale de l'application.

L'ensemble de ces composants ont été soigneusement conçus et intégrés. En démontrant les capacités puissantes de Rust et de GTK4, cette application m'a permis de prendre en main et de découvrir correctement toute l'empleure des possibilités offerte par gtk4-rs et représente un exemple de ce à quoi va ressembler notre application dans le futur avec l'union entre la robustesse du développement logiciel et la convivialité de l'interface utilisateur moderne.

En plus des fonctionnalités mentionnées précédemment, j'ai également implémenté un Column View Datagrid pour offrir une visualisation avancée des données. Ce Datagrid est composé de deux colonnes, chacune contenant 10 000 éléments, offrant ainsi une vue exhaustive des informations. Pour assurer une navigation fluide, j'ai intégré une barre de défilement permettant aux utilisateurs de parcourir rapidement et efficacement les données affichées.

Ce Column View Datagrid constitue un premier exemple de ce à quoi ressemblera l'application une fois que nous aurons intégré et organisé les informations à afficher. Il offre un aperçu convaincant de la capacité de l'application à gérer et à présenter de grandes quantités de données de manière claire et organisée. En intégrant ces éléments de visualisation avancée, nous visons à fournir aux utilisateurs une expérience immersive et informative, tout en démontrant les capacités étendues de l'application.

3.4 Commandes Bash

Pour ce qui est des commandes bash, nous devions réaliser l'implémentation des fonctionnalités de base.

Pour rappel, ces fonctionnalités étaient :
- Directory Listing : Afficher en permanence le chemin du dossier dans lequel on travaille
- Navigation : Navigation entre les dossiers depuis l'interface graphique
- Directory creation : Créer un dossier
- Directory deletion : Supprimer un dossier
- File creation : Créer un fichier
- File deletion : Supprimer un fichier

Pour ce faire, nous avons utilisé deux modules de la bibliothèque standard : `std::fs` et `std::env`.
En Rust, `std::fs` permet d'effectuer des opérations sur le système de fichiers. Il fournit des fonctions et des types permettant d'interagir avec le système de fichiers, comme la lecture et l'écriture de fichiers, la création et la suppression de répertoires et la manipulation de métadonnées de fichiers.
`std::env`, lui, fournit des fonctions et des types pour interagir avec l'environnement, y compris l'accès aux arguments de la ligne de commande, aux variables d'environnement et au répertoire de travail actuel.

Nous avons donc implémenté nos fonctions en utilisant ce module. Le

problème était que nous devions faire attention aux erreurs. Pour cela, il a fallu implémenter plusieurs tests en amont de l'utilisation de chaque fonction pour éviter tout problème lors de l'utilisation du file explorer.

Pour la prochaine soutenance, il faudra finir d'implémenter les commandes bash, mais surtout faire en sorte de les relier avec l'interface utilisateur afin qu'elle puisse fonctionner.

3.5 Algorithmes

Pour cette partie, les attendus pour la première soutenance était d'implémenter un premier algorithme de recherche non optimisé, qui permet de parcourir la liste des fichiers de l'ordinateur et de trouver ceux dont le nom correspond. Malgré l'apparente facilité de cette tâche, le choix et l'utilisation des types de donnés à manipuler dans Rust pour pouvoir parcourir récursivement les dossiers de l'utilisateur était un véritable défi. En effet, les fonctions de recherche et de parcours manipulant des millions de fichiers, il était nécessaire de ne pas faire de parcours de listes inutiles et d'éviter d'utiliser des types de donnés très gourmand en mémoire.

C'est pourquoi l'on a implémenté une fonction qui parcourt récursivement les dossiers et fichiers de l'utilisateur et qui pour chaque, va stocker dans un vecteur, un tuple contenant le nom du fichier/dossier et son chemin absolu.
Ainsi, cette fonction permet de construire un vecteur qui est ensuite trié en utilisant les noms de fichiers. Ce vecteur est sauvegardé tant que le programme n'est pas quitté, et il est la seule source de données qui va être utilisée par les différents algorithmes de recherche.

Pour cette première soutenance, nous avons développé un simple algorithme de recherche dichotomique qui permet d'extraire tous les tuples qui ont un nom de fichier qui correspond au fichier recherché. Malgré sa simplicité, cet algorithme est pour l'instant suffisant puisqu'il dispose d'une très bonne complexité log(n).

D'ici la prochaine soutenance il sera nécessaire de rajouter une recherche plus globale et peut-être plus optimisée si besoin et de trier les résultats en fonction de leur pertinence.

4 Conclusion

4.1 Conclusion

Nous avons donc bien implémenter l'ensemble des fonctionnalités prévues pour la première soutenance :

- un début d'interface est créé
- l'implémentation des premières fonctions d'opérations sur les fichiers
- un algorithme de recherche et de tri des fichiers

4.2 Pour la prochaine soutenance

Voici le tableau de l'avancement des tâches prévu pour la prochaine soutenance :

Tâche	**Avancement**
UI	Mise en place de l'interface
Commandes bash	Implémentations des commandes bash restantes
Algorithmes	Implémentation de l'algorithme optimisé et début de l'implémentations du "zip"
Site Internet	Création du site internet