Projet FAT32

Tom Favereau Élise Suspène Alicia Shao

Table des matières

Remerciement

Nous tenons à exprimer notre gratitude anticipée envers les correcteurs qui prendront le temps d'examiner ce projet. nous apprécions par avance leur indulgence, car tout cela est loin d'être parfait. Nous leur souhaitons une agréable correction et les remercions d'avance pour leur précieux retours.

1 Implémentation

N.B.: les durées *approximatives* passées sur les classes sont données entre parenthèses.

1.1 Lecture et écriture dans le disque

Nos premiers pas dans le projet ont était faits avec la tables d'allocation. Nous avons entrepris de comprendre le fonctionement des disques en regardant les informations stockées dans la FAT. Afin de lire ces informations nous avons dû lire les « reserved sectors » et stocker les informations lues.

Le fait que nous ayons commencé par l'accès à la table d'allocation explique pourquoi les données des reserved sectors sont stockées dans notre classe **FatAccess (10 h)**.

Afin d'accéder aux données de la FAT qui sont stockées en bytes nous avons dû d'une part comprendre le fonctionement de la classe Device et d'autre part apprendre à écrire et lire des données sur plusieurs bytes. En effet la classe byte est signée ce qui voulait dire transformer le premier bit des bytes.

Nous avons choisi d'implémenter nous-mêmes la lecture et l'écriture sur plusieurs bytes afin de nous familiariser avec cette notion d'informatique et également d'être sûrs de respecter la convention.

Cela mis à part, notre classe FatAccess est relativement simple et se contente de faire des lectures et écritures dans la FAT à partir des informations lues dans les secteurs réservés.

La gestion de la zone de données a été plus laborieuse, nous avons choisi d'implémenter une classe **DataAccess (Tom: 14h, Alicia: 4h)** afin de gérer les lectures et écritures dans la zone de données.

1.2 Huffmann

Le package **huffmam (Tom : 5h)** implémente un algorithme de compression sans perte mais n'as pas été racordé au reste du projet.

1.3 Gestion des commandes de l'utilisateur

La classe **Path (10h)** implémentée est inspirée à l'origine de l'interface Path de Java. Sa principale fonction est de retrouver dans les données les métadonnées des fichiers composant un chemin dans un disque. Elle a été développée en parallèle de la classe DataAccess et de la gestion des commandes et a été mise à jour tout au long du projet, également par Tom.

Cette classe interagit avec la classe DataFile et DataAccess, et est utilisée par FileSystem.

La gestion des chemins relatifs contenant . et .. est gérée indépendamment dans la classe FileSystem.

1.4 frontend et logiciel pour l'utilisateur

1.4.1 Shell

Le package **shell (5h)** implémente un shell basique reconaissant les fonctions

- ls
- less
- mkdir
- touch
- echo « foo » >> foo.h et echo « foo » > foo.h

La coloration donne les propriétés des fichiers :
- jaune : fichier système
- bleu : dossier
- rouge fichier compressé

1.4.2 Graphics

Le package **graphics (14h)** implémente une interface graphique pour l'utilisateur. On s'est proposé d'implémenter un explorateur de fichiers permettant de naviguer dans les fichiers, d'écrire dans les fichiers. Nous implémentons également un logiciel de shell disponible dans l'explorateur de fichiers. Enfin nous proposons un logiciel de gestion du disque permetant de visualiser la fragmentation du disque ainsi que de formater le disque.

Pour lancer l'explorateur de fichier il faut executer la classe MainFileManager.

2 Répartition du travail

2.1 Élise Suspène

J'ai codé la classe **Shell** avec Tom. Pour chaque fonction classique d'un explorateur de fichier, nous avons fait le lien entre l'input utilisateur (la ligne de commande) et les classes FileStream et FileSystem par exemple, pour répondre à la demande de l'utilisateur.

2.2 Alicia Shao

La classe **Path** a été gérée tout au long du projet par mes soins. J'ai également implementé **FileStream (2h)** et **FileSystem (Alicia: 3h, Tom: 2h)**. Tom a également contribué à l'élaboration de cette dernière classe.

Je me suis également concentrée sur l'écriture des **Java docs et des commentaires (6h)** des classes principales et le compactage de l'implémentation de la classe DataAccess.

2.3 Tom Favereau

Je me suis concentré sur la partie backend et j'ai notemament fait FatAccess et DataAccess. J'ai également passé beaucoup de temps sur la résolution des bugs. Je me suis aussi aucupé du package Graphics et de la compression.

3 Piste d'améliorations du travail

3.1 Extention

- Finir la compression de fichiers.
- Stocker des photographies et implémenter une visionneuse d'images.
- Améliorer la partie graphique avec de nouvelles fonctionalités.
- Compatibilité avec un disque physique.
- Gérer le cas des secteurs défectueux.
- Faire un outil de récupération de données sur un disque formaté.

3.2 Bug

Les principaux bug se trouve dans le package graphics sur lequel nous n'avons pas pu nous consacrer à 100%. Ainsi qu'un problème d'optimisation du code dans l'écriture du fichier qui crée des *timemout* (ce problème est nous l'éspéron résolue). Les tests du package tests sont également problématique, il ne passe plus car ils étaient basé sur un fichier qui à malheuresement été formaté, nous n'avons pas eu le temps de les refaires.