Rapport de Projet

Simulateur Simplifié d'un Système de Gestion de Fichiers (SGF)

NOM	PRENOM	MATRICULE	SECTION	GROUPE
Chennit	Zineb	232331517819	C	3
Chabour	Cerine	232331425808	C	1
Laboubi	Hadjer	232331549614	C	1
Bentebbal	Sabrine	232331782512	C	3
Daghoudj	Hiba	232331692020	C	3

-Rédiger pour: Abdelmadjid Lahreche

PROJET FINALE, MOUDULE SFSD

10 JANVIER 2025

Table des matières

1. Introduction

1.1 Contexte du Projet

Le projet consiste en l'implémentation d'un simulateur simplifié de Système de Gestion de Fichiers (SGF) en langage C. Ce système permet de gérer le stockage et la manipulation de fichiers sur une mémoire secondaire virtuelle, reproduisant les mécanismes fondamentaux d'un SGF réel.

1.2 Objectifs

- **Simuler une mémoire secondaire avec une gestion de blocs**
- **Implémenter différentes organisations de fichiers (contiguë et chaînée)**
- **Gérer les métadonnées des fichiers**
- **Permettre les opérations CRUD (Create, Read, Update, Delete) sur les fichiers**
- **Fournir une interface utilisateur interactive**

2. Analyse et Conception

2.1 Architecture Globale

Le simulateur est structuré en plusieurs composants principaux :

- **Gestionnaire de mémoire secondaire**
- **Gestionnaire de fichiers**
- **Gestionnaire de métadonnées**
- **Interface utilisateur**

2.2 Choix de Conception

- **Organisation de la mémoire en blocs de taille fixe**
- **Table d'allocation pour suivre l'état des blocs**
- **Support de deux types d'organisation des fichiers**
- **Gestion des métadonnées dans des blocs séparés**

3. Structures de Données

3.1 Structure d'Enregistrement (Record)

c
Copy

```c
typedef struct {
    int id;
    char isDeleted;
} Record;
```

Cette structure représente un enregistrement individuel dans un fichier, avec :

- **Un identifiant unique**
- **Un indicateur de suppression logique**

3.2 Structure de Métadonnées (Metadata)

c
Copy

```c
typedef struct {
    char filename[MAX_FILENAME];
    int numBlocks;
    int numRecords;
    int firstBlockAddress;
    char globalOrganization;
    char internalOrganization;
} Metadata;
```

Cette structure maintient les informations essentielles sur chaque fichier.

3.3 Structure de Bloc (Block)

c
Copy

```c
typedef struct {
    Record* records;
    int numRecords;
    int nextBlock;
    char filename[MAX_FILENAME];
} Block;
```

Représente un bloc de la mémoire secondaire.

3.4 Structure de Mémoire Secondaire (SecondaryMemory)

c
Copy

```c
typedef struct {
    Block* blocks;
    int totalBlocks;
    int blockSize;
    char* allocationTable;
    Metadata* fileMetadata;
    int numFiles;
} SecondaryMemory;
```

Structure principale gérant l'ensemble du système.

4. Algorithmes Implémentés

4.1 Allocation de Fichiers

Organisation Contiguë

1. **Recherche d'un espace contigu suffisant**
2. **Allocation des blocs adjacents**
3. **Mise à jour de la table d'allocation**
4. **Création des métadonnées**

Organisation Chaînée

1. **Recherche de blocs libres**
2. **Création des liens entre les blocs**
3. **Mise à jour des pointeurs de chaînage**
4. **Création des métadonnées**

4.2 Recherche d'Enregistrements

- **Pour l'organisation contiguë : Parcours séquentiel des blocs**
- **Pour l'organisation chaînée : Suivi des pointeurs de chaînage**
- **Optimisation pour les fichiers triés : Recherche dichotomique**

4.3 Gestion de l'Espace

- **Compactage : Réorganisation des blocs pour éliminer la fragmentation**
- **Défragmentation : Réorganisation des enregistrements dans un fichier**
- **Gestion des suppressions logiques et physiques**

5. Tests et Validation

5.1 Scénarios de Test

1. **Création et manipulation de fichiers**
    - **Création de fichiers de différentes tailles**
    - **Test des deux modes d'organisation**
    - **Vérification des métadonnées**
2. **Opérations sur les enregistrements**
    - **Insertion dans différentes positions**
    - **Recherche d'enregistrements existants et non existants**
    - **Suppressions logiques et physiques**
3. **Gestion de l'espace**
    - **Test de fragmentation**
    - **Validation du compactage**
    - **Vérification de la défragmentation**

5.2 Résultats

- Initialisation de la secondaire et Menu Principale:

```
PS C:\Users\samsung> cd 'c:\Users\samsung\Desktop\SFSD\output'
PS C:\Users\samsung\Desktop\SFSD\output> & .\'sgf.exe'
Initialisation de la memoire secondaire
Nombre de blocs: 2
Taille des blocs (facteur de blocage): 3

Menu Principal:
1. Creer un fichier
2. Afficher l'etat de la memoire
3. Afficher les metadonnees
4. Rechercher un enregistrement
5. Inserer un enregistrement
6. Supprimer un enregistrement
7. Defragmenter un fichier
8. Supprimer un fichier
9. Renommer un fichier
10. Compacter la memoire
11. Vider la memoire
0. Quitter
Choix:
```

- Etat de la mémoire secondaire au debut:

```
Etat de la memoire secondaire:
[Libre] [Libre] [Libre]
```

- Apres avoir créer 2 fichiers SFSD et projet:

```
Choix: 1
Nom du fichier: SFSD
Nombre d'enregistrements: 2
Organisation globale (C/L): 3
Organisation interne (T/N): 1
```

```
Choix: 1
Nom du fichier: projet
Nombre d'enregistrements: 2
Organisation globale (C/L): 2
Organisation interne (T/N): 2
```

- L'Etat de la mémoire secondaire devient:

```
Etat de la memoire secondaire:
[SFSD-0] [projet-0] [Libre]
```

- En Affichant les metadonnees :

```
Choix: 3

Metadonn├®es des fichiers:
Nom                 Blocs      Enregistrements Premier bloc    Org. Globale        Org. Interne
-------------------------------------------------------------------------------
SFSD                1          2               0               3                   1
projet              1          2               1               2                   2
```

- On Insére un Enregistrement dans notre fichier :

```
Choix: 5
Nom du fichier: SFSD
ID du nouvel enregistrement: 2
```

- Si On fait la Recherche d'un Enregistrement:

```
Choix: 4
Nom du fichier: SFSD
ID de l'enregistrement: 2
Enregistrement trouve dans le bloc 0, position 0
```

- Si On fait supprimer un Enregistrement:

```
Choix: 6
Nom du fichier: SFSD
 ID de l'enregistrement: 2
 Type de suppression (1: Logique, 2: Physique): 1
Enregistrement marque comme supprime
```

- Defragmentation du fichier projet:

```
Choix: 7
 Nom du fichier: projet
Defragmentation terminee
```

- Supprimer le fichier SFSD:

```
Choix: 8
Nom du fichier: SFSD
Fichier supprime avec succes
```

➢ Renommer un fichier:

```
Choix: 9
 Ancien nom: projet
 Nouveau nom: PROJET
Fichier renomme avec succes
```

```
Etat de la memoire secondaire:
[Libre] [PROJET-0] [Libre]
```

➢ Compacter la mémoire :

```
Etat de la memoire secondaire:
[PROJET-0] [Libre] [Libre]
```

➢ Vider la mémoire:

```
Choix: 11
Memoire secondaire videe
```

```
Etat de la memoire secondaire:
[Libre] [Libre] [Libre]
```

6. Conclusion

6.1 Objectifs Atteints

- Implémentation réussie des fonctionnalités principales
- Interface utilisateur fonctionnelle
- Gestion efficace de la mémoire secondaire

6.2 Difficultés Rencontrées

- Gestion de la fragmentation
- Maintient de la cohérence des métadonnées
- Implémentation du chaînage des blocs

6.3 Perspectives d'Amélioration

- Ajout de nouveaux modes d'organisation
- Optimisation des algorithmes de recherche
- Amélioration de l'interface utilisateur