**Rapport du Projet : Simulateur Simplifié de Gestion de Fichiers (SGF)**

**Introduction**

Ce projet implémente un simulateur simplifié d’un système de gestion de fichiers (SGF). Il permet de gérer une mémoire secondaire organisée en blocs, avec des fichiers organisés en mode **contigu** ou **chaîné**, tout en assurant une gestion efficace des métadonnées associées.

Le SGF offre des fonctionnalités telles que :

1. L'initialisation de la mémoire secondaire.
2. La création, la suppression, l'insertion, et la recherche d'enregistrements dans des fichiers.
3. Les opérations avancées comme le compactage, la défragmentation, et le vidage de la mémoire.
4. Une gestion intuitive via un menu interactif.

**1. Fonctionnalités principales et exemples d'exécution**

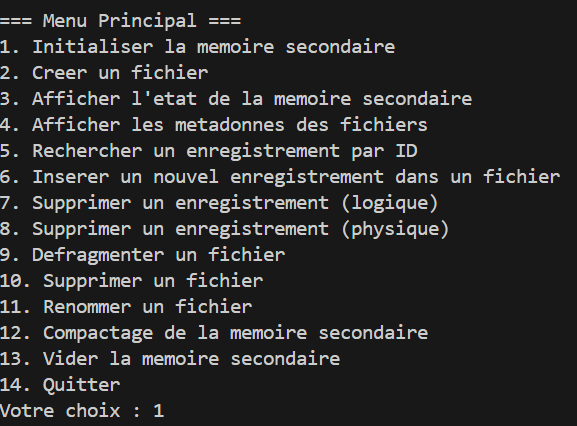
**1.1 Menu Principal**

Le menu principal est au cœur de l’interaction avec le simulateur SGF. Il permet aux utilisateurs de naviguer facilement entre les fonctionnalités du programme et d’exécuter les différentes opérations. Ce menu est structuré pour offrir une vue claire des fonctionnalités, tout en guidant l’utilisateur avec des instructions précises.

Chaque option correspond à une opération spécifique, organisée de manière intuitive :

1. Initialisation de la mémoire secondaire : Configure la mémoire pour accueillir des blocs.
2. Gestion des fichiers : Inclut la création, la suppression, l'insertion, et la recherche d'enregistrements.
3. Opérations avancées : Compactage, défragmentation, et vidage complet de la mémoire.

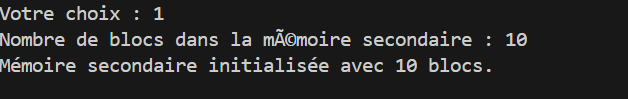
**Exemple d'interaction avec le menu principal :**



**1.2 Initialisation de la mémoire secondaire**

L'initialisation configure la mémoire avec un nombre donné de blocs. Chaque bloc est marqué comme libre au départ.

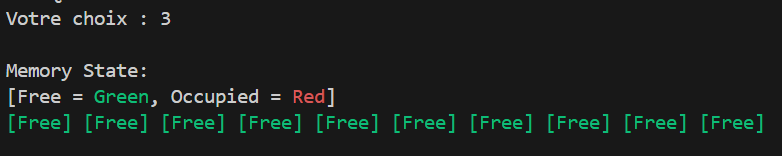
**Exemple d’exécution :**



**1.3 Affichage de l’état de la mémoire secondaire**

Affiche graphiquement les blocs libres et occupés.

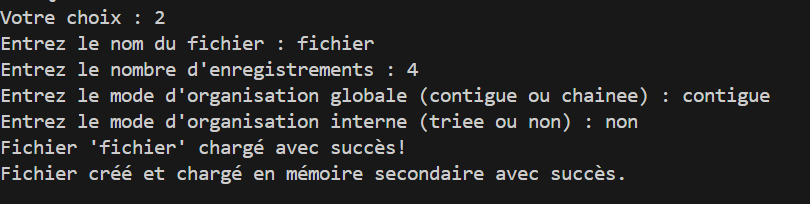
**Exemple d’exécution :**



**1.4 Création d’un fichier**

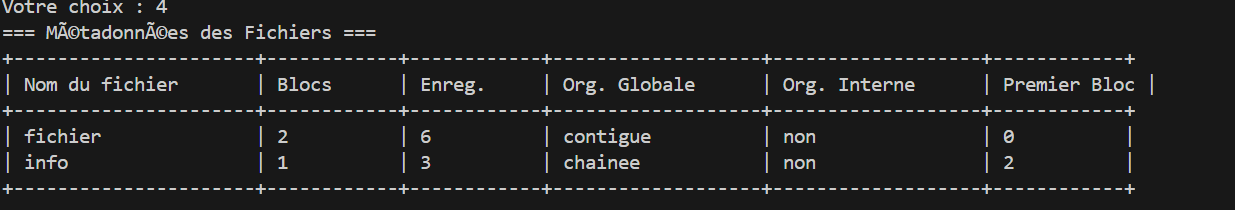
Permet de créer un fichier avec un nom, un nombre d’enregistrements, et un mode d’organisation.

**Exemple d’exécution :**



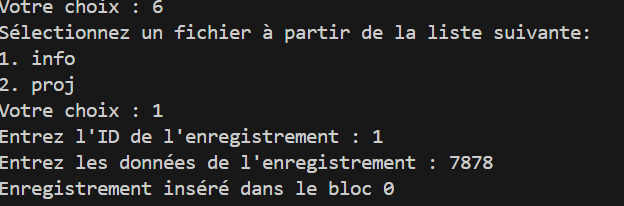
**1.5 Gestion des Métadonnées**

Les métadonnées permettent de suivre les informations essentielles sur chaque fichier, telles que la taille, le mode d'organisation, et la localisation du premier bloc.



### 1.6 Insertion d’Enregistrements

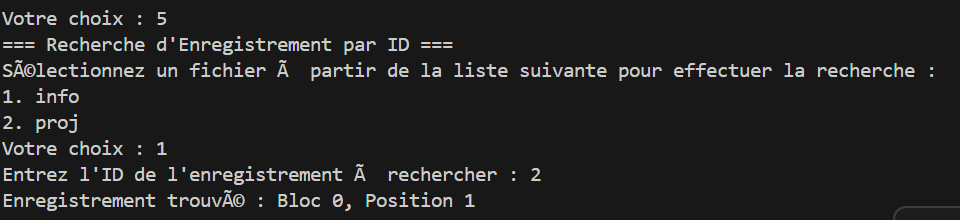
Permet d’ajouter des enregistrements à un fichier, en fonction de son organisation (trié ou non).



### 1.7 Recherche d’Enregistrements

Permet de rechercher un enregistrement par ID, avec des algorithmes adaptés (linéaire ou binaire ).

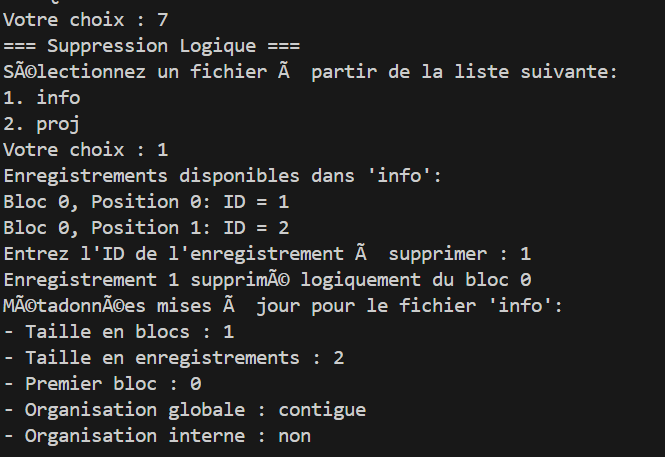
**Exemple d’exécution :**



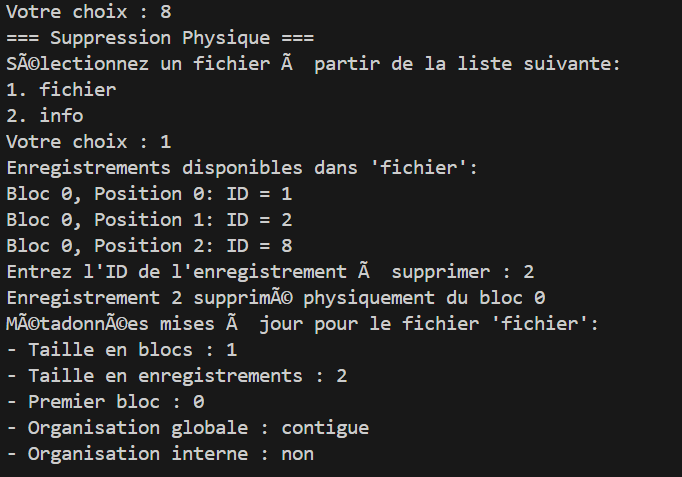
### 1.8 Suppression d’Enregistrements

* **Suppression Logique :** L’enregistrement est marqué comme supprimé sans déplacer les autres.

**Une mise à jour de métadonnées est effectuée après une suppression logique.**



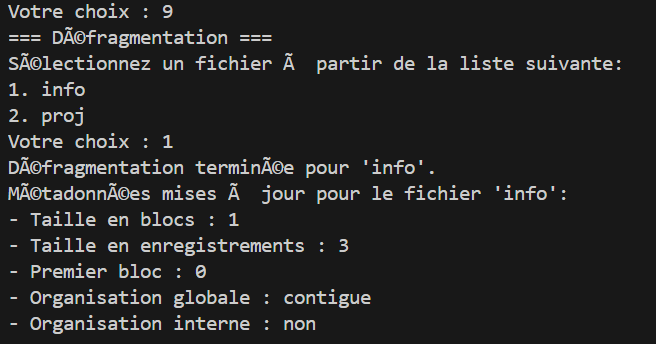
* **Suppression Physique :** L’enregistrement est éliminé et les blocs sont réorganisés.



### ****1.9 Défragmentation des Fichiers****

L'utilisateur lance une défragmentation pour réorganiser les blocs d'un fichier.

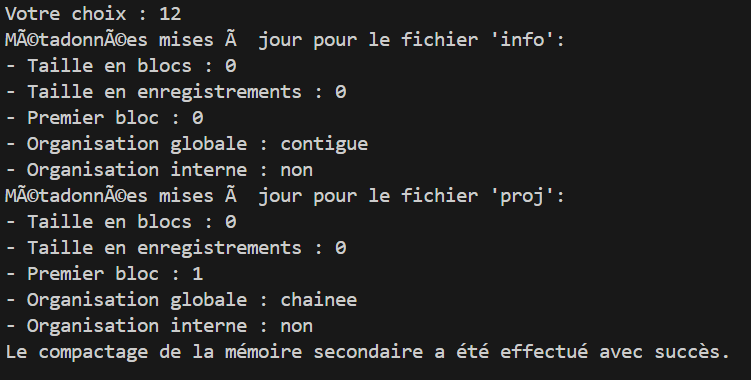
**Exemple d’exécution :**



### ****1.10 Compactage de la Mémoire Secondaire****

L'utilisateur effectue un compactage global pour regrouper les blocs occupés.

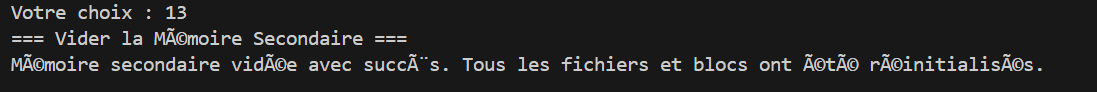
**Exemple d'exécution :**



### ****1.11 Vidage de la Mémoire Secondaire****

L'utilisateur réinitialise complètement la mémoire.

**Exemple d'exécution :**



### ****1.12 Quitter le Programme****

L'utilisateur termine l'exécution du programme.

**Exemple d'exécution :**



1. **Conclusion**

Le simulateur simplifié de gestion de fichiers (SGF) est une implémentation robuste et pédagogique des concepts fondamentaux de la gestion des fichiers dans un système de gestion de mémoire secondaire. Il offre une interface utilisateur claire et interactive via un menu principal, permettant d’effectuer un large éventail d’opérations. Ces opérations incluent la création, l'insertion, la recherche, et la suppression de fichiers, ainsi que des fonctionnalités avancées comme le compactage, la défragmentation et le vidage complet de la mémoire.

Le projet a permis de démontrer plusieurs concepts importants, notamment :

* Organisation des fichiers : Gestion des modes contigu et chaîné pour stocker les enregistrements de manière optimisée.
* Gestion des métadonnées : Suivi précis des propriétés de chaque fichier, telles que le nombre de blocs utilisés et l'adresse du premier bloc.
* Optimisation de l’espace : Réduction de la fragmentation grâce au compactage et à la défragmentation.
* Interopérabilité : Une structure modulaire qui facilite l’ajout ou la modification de fonctionnalités.

Les exemples d’exécution fournis montrent que le simulateur fonctionne correctement et gère efficacement les opérations courantes ainsi que les scénarios avancés. Les fonctionnalités sont conçues pour être extensibles et s’adapter à des cas d’utilisation plus complexes, tels que des volumes de données plus importants ou des environnements multi-utilisateurs.

**Avantages du simulateur :**

* Clarté et simplicité d’utilisation : Le menu principal est intuitif, et les options sont bien documentées.
* Fonctionnalités complètes : Tous les aspects du SGF, du stockage de base aux optimisations avancées, sont pris en charge.
* Robustesse : Gestion appropriée des erreurs et des scénarios limites (par exemple, mémoire pleine, suppression répétée).
* Portabilité : Le code est conçu pour être exécuté sur n’importe quelle plateforme prenant en charge le langage C.

**Améliorations et perspectives :**

Bien que le simulateur réponde aux exigences initiales, certaines améliorations peuvent être envisagées :

1. **Interface utilisateur graphique (GUI)** : Permettre une visualisation plus claire des blocs et des fichiers.
2. **Algorithmes optimisés** : Utiliser des structures de données avancées pour gérer de grands volumes de données plus efficacement.
3. **Tests automatisés** : Intégrer des scripts pour valider les résultats des opérations automatiquement.
4. **Persistance avancée** : Ajouter des mécanismes pour sauvegarder automatiquement l’état de la mémoire secondaire et des métadonnées.