



**ID1FS**  
File System

# ID1FS RAPPORT DE PROJET

Réalisés par:

Aymane Maghouthi  
Abdelghafor Elgharbaoui  
Ossama Outmani  
Younes Souabeddine

Encadré par :

Pr.Mohamed Cherradi

---

# Sommaire

## 0 – Introduction

## 1 – L'étude de l'existant

## 2 – Outils & Méthodes

## 3 – Conception

### 3.1 Architecture de ID1FS

### 3.2 Fonctionnement de ID1FS

### 3.3 Statuts du système

### 3.4 Démarrage de l'ID1FS

## 4 – Composants Clés

### 4.1 Client

#### 4.1.1 Gestion Des Utilisateurs

#### 4.1.2 Permission

#### 4.1.3 Gestion De Fichiers

### 4.2 Server

#### 4.2.1 Master

##### 4.2.1.1 Méta-données

##### 4.2.1.2 Fichier Log

#### 4.2.2 La section Principale (Main Section)

#### 4.2.3 La section Secondaire (Backup Section)

## 5 – Étude Benchmarking

### 5.1 Bénéfices attendus

### 5.2 Points à Développer

### 5.3 Comparaison Entre ID1FS & HDFS

## 6 – Tableau récapitulatif des commandes

## 0- Introduction

Un système de fichiers est un moyen de stocker et d'organiser les fichiers sur un ordinateur ou un autre périphérique de stockage de données. Il définit la manière dont les fichiers sont nommés, organisés et stockés sur le périphérique.

Il existe de nombreux systèmes de fichiers différents, chacun avec ses propres avantages et inconvénients. Par exemple, le système de fichiers NTFS est utilisé sur les ordinateurs Windows, tandis que le système de fichiers ext4 est couramment utilisé sur les ordinateurs Linux et le système de fichier GFS qui est un système distribué développé par Google pour ses propres besoins de stockage de données à grande échelle.

Le système de fichiers d'un périphérique est généralement défini lors de la formation de celui-ci. Cela implique l'effacement de toutes les données précédemment stockées sur le périphérique et la création d'une structure de fichiers vide.

Les systèmes de fichiers permettent également de définir des autorisations d'accès aux fichiers, ce qui permet de contrôler qui peut lire, écrire ou exécuter les fichiers sur le périphérique. Cela peut être utile pour protéger les données sensibles ou empêcher les utilisateurs non autorisés d'accéder à certains fichiers.

Dans le cadre de construction d'un système de fichier, on a essayé de construire un système de fichier sous nom '**ID1FS**' similaire à GFS et aussi inspiré de HDFS (Hadoop Distributed File System), avec l'ajout de nouvelles fonctionnalités qui son propre a notre système.

# 1- Étude De L'existant

Il existe de nombreux systèmes de fichiers différents, chacun étant optimisé pour différents types de supports de stockage et d'utilisations, dans notre projet on est inspirés de HDFS et GFS.

GFS :(Google File System) est un système de fichiers distribué conçu par Google pour stocker et gérer de grandes quantités de données sur un ensemble de serveurs. Il a été conçu pour être utilisé dans un environnement de calcul distribué et est optimisé pour les lectures et les écritures de gros fichiers de manière séquentielle.

GFS est conçu pour être tolérant aux pannes et pour permettre l'analyse de données volumineuses. Il utilise une architecture de type "maître-esclave" avec un serveur de nommage qui gère les métadonnées et un ensemble de serveurs de stockage qui stockent les données. GFS est utilisé principalement pour le traitement de données en batch et l'analyse de données à grande échelle.

GFS a été l'un des premiers systèmes de fichiers distribués à être utilisé en production à grande échelle et a été largement influent dans le développement de nombreux autres systèmes de fichiers distribués.

HDFS (Hadoop Distributed File System) est un système de fichiers distribué conçu pour stocker et gérer de grandes quantités de données sur un ensemble de serveurs dans un environnement de calcul distribué. Il est utilisé principalement dans le cadre du framework Apache Hadoop et est conçu pour être tolérant aux pannes et pour permettre l'analyse de données volumineuses. HDFS est souvent utilisé pour le traitement de données en batch et l'analyse de données à grande échelle. HDFS utilise une architecture de type "maître-esclave" avec un serveur de nommage (ou "NameNode") qui gère les métadonnées et un ensemble de serveurs de stockage (ou "DataNodes") qui stockent les données. Les fichiers stockés sur HDFS sont divisés en blocs de taille fixe (généralement de 128 Mo à 256 Mo) et répliqués sur plusieurs serveurs de stockage pour assurer la tolérance

aux pannes. HDFS est conçu pour être utilisé avec des fichiers de grande taille et est optimisé pour les lectures et les écritures de manière séquentielle.



## 2- Outils et Méthodes

Le langage de programmation Python, c'est un langage de programmation populaire et polyvalent qui peut être utilisé dans de nombreux domaines, y compris le développement de logiciels, la data science et l'automatisation de tâches. Il est connu pour sa syntaxe claire et concise, ce qui le rend facile à apprendre et à utiliser.

Un autre outil utilisé est l'environnement Pycharm, c'est un environnement de développement intégré (IDE) pour le langage de programmation Python, il est utilisé pour développer des scripts.



Linux (distribution Ubuntu) est un système d'exploitation open source qui est souvent utilisé pour les serveurs et les ordinateurs de bureau. Il est connu pour sa stabilité, sa sécurité et sa flexibilité, et est largement utilisé dans les entreprises et les institutions académiques.



En utilisant Python et Linux pour notre projet, nous avons accès à un large éventail d'outils et de ressources qui peuvent nous aider à réaliser notre projet de manière efficace. Python nous permet de développer rapidement des scripts, tandis que Linux nous offre un environnement stable et sécurisé pour exécuter notre code.

Pour Python on utilise plusieurs bibliothèques :

Argparse Library : Cette bibliothèque nous permet de créer facilement des interfaces en ligne de commande pour votre programme Python. Elle nous permet de définir les arguments et les options que notre programme peut prendre en entrée, et de gérer facilement les erreurs de syntaxe et de validation des arguments.

Shutil library : Cette bibliothèque fournit des fonctions pour effectuer des opérations de gestion de fichiers et de dossiers de haut niveau, comme copier, déplacer, compresser et décompresser des fichiers et des dossiers.

Sys library : Cette bibliothèque fournit des variables et des fonctions qui permettent d'interagir avec notre interpréteur Python et notre système d'exploitation. Par exemple, nous avons utilisé cette bibliothèque pour accéder aux arguments de la ligne de commande, aux variables d'environnement, aux informations sur la plateforme (système d'exploitation, architecture, etc...)

Json library : Cette bibliothèque nous permet de travailler avec des données au format JSON (JavaScript Object Notation). Nous avons utilisé cette bibliothèque pour lire et écrire des fichiers JSON (par exemple les métadonnées seront enregistrées dans des fichiers Json). Puisque les fichiers Json sont basés sur la

logique des dictionnaires (clé : valeur), alors la manipulation des fichiers va être simple.

Logging library : Cette bibliothèque nous permet de créer des journaux de notre programme. Nous avons utilisé cette bibliothèque pour enregistrer des messages à différents niveaux de gravité (debug, info,error.), et configurer la façon dont ces messages sont enregistrés (fichier, console). Cela peut être très utile pour suivre l'exécution de notre programme et déboguer des erreurs.

Os library : cette bibliothèque nous permet de créer des chemins d'accès, les manipuler, extraire le nom du fichier ,son extension ....

Subprocess library : cette bibliothèque nous permet de créer de nouveaux processus, d'accéder a leur entrée/sortie/erreur .elle a été très utiliser pour exécuter des commandes linux à travers python

Multiprocessing library : cette bibliothèque nous permet de lancer et exécuter plusieurs programme/script en parallèle.

Time library : cette bibliothèque fournit des fonctionnalités relatives au temps. Elle a été utilisée surtout pour suspendre l'exécution d'un script pour une durée déterminée.

Datetime library : cette bibliothèque permet de manipuler et exploiter le variable de type datetime.

Ntplib library : cette bibliothèque permet de se connecter à un serveur NTP pour obtenir la date et l'heure actuel afin de les comparer avec celles du système.

Configparser library: cette bibliothèque permet de créer et de manipuler les fichiers config de manière aisée.

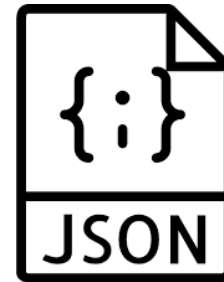
Pour Linux (Ubuntu), on utilise plusieurs outils :

Exiftool est un utilitaire en ligne de commande qui permet d'extraire, lire et écrire les métadonnées de différents types de fichiers.

On a utilisé Exiftool pour afficher les métadonnées des fichiers, pour en extraire certaines et les stocker sous forme de dictionnaire (clé : valeur), pour modifier les métadonnées d'un fichier et par fois pour les supprimer complètement, et on a lie

cette commande avec le code python en utilisant la bibliothèque json pour que à la fin fait construire des fichiers de type json qui contient les métadonnées.

On choisit les fichiers json car ce type est facile à manipuler et facile à considère comme un dictionnaire.



## 3-Conception :

### 3.1 Architecture de L'ID1FS :

Avant tous chose, il faut faire la conception ou bien l'architecture de projet, c'est l'étape la plus importante, alors pour cela on a élaboré notre propre conception:

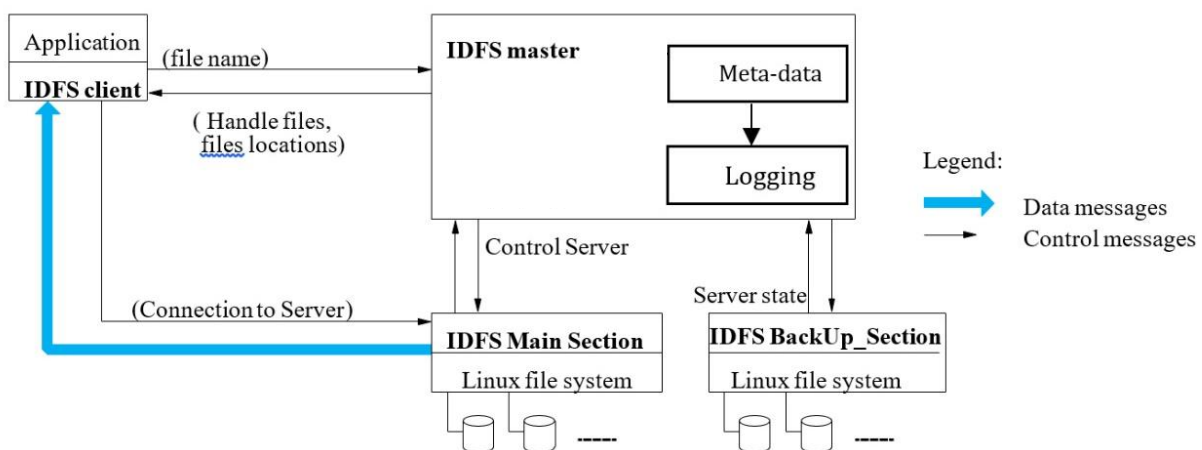


Figure: Architecture de L'ID1FS



**ID1FS** est un système de fichiers distribué (inspiré de HDFS), permet à l'utilisateur de faire plusieurs fonctions pour manipuler les données hétérogènes (des données qui proviennent de différentes sources et qui peuvent être de différents types, formats et structures), notre système contient plusieurs composantes qui sont principalement Client et Server.

La partie Client contient les fonctions principales (Commandes) qui permet à l'utilisateur de gérer et manipuler les fichiers, aussi des algorithmes pour gérer les utilisateurs, contrôler les permissions....

La partie Server est divisée entre quatre parties principales : Master, la section principale et la section secondaire.

## 3.2 Fonctionnement :

ID1FS est un système de fichier basé sur des commandes. Cette propriété permet d'interagir avec ce dernier dans le terminale, autrement dit l'utilisateur peut naviguer à travers sa machine et se connecter au ID1FS en même temps

Les fichiers sont distribués sur des Nodes de taille maximale bien déterminée (500 bytes pour des raisons de test) et qui sont déterminés automatiquement lors de la création ou le transfert du fichier, de telle sorte à minimiser le nombre de Node utilise .Ce processus s'exécute au niveau de la section principale comme pour la secondaire ,de manière indépendantes .

Chaque fichier (créé ou transféré) est automatiquement sauvegardé dans la partie secondaire pour garantir la restauration au cas de panne.

## 3.3 Statuts du système :

Le statut du système de fichiers reflète son état de fonctionnement, qui se représente dans les trois propriétés suivantes:

Status : **on/off** détermine l'état du système **actif/inactif**

Connected user : le nom de l'utilisateur actuellement connectée, **None** si aucun utilisateur n'est connecté.

Time : **configured/not configured** , indique si l'horloge de la machine est configurée correctement.

Pour afficher le statuts d'ID1FS, on exécute la commande suivante :

```
root@localhost $ idfs -s
```

```
ossama@ossama-virtual-machine:~$ id1fs -s
status : off
connected user : None
time : configured
ossama@ossama-virtual-machine:~$
```

### 3.4 Démarrage de L'ID1FS :

Le démarrage de système est la première étape pour utiliser le ID1-file-system, car avec cette action, trois scripts s'exécutent à l'arrière-plan, le premier et responsable sur les nœuds, en effet il scanne la section principale chaque 3 seconde et chèque si un nœud est tombé, si le cas il le récupérer. Le deuxième script est responsable sur le Master, il fait le même fonctionnement que le premier mais cette fois pour récupérer le Master. Le dernier script s'assure que l'horloge de la machine de l'utilisateur est configurée correctement. De plus le statut du système devient **on**.

Pour démarrer l'ID1FS il faut exécuter la commande id1fs avec l'option -o:

```
root@localhost $ id1fs -o
```

```
ossama@ossama-virtual-machine:~$ id1fs -o

-----
ID1FS
-----

Welcome to ID1FS

ossama@ossama-virtual-machine:~$ id1fs -s
status : on
connected user : None
time : configured
ossama@ossama-virtual-machine:~$
```

Et pour l'arrêter, il faut utiliser l'option `-q` :

```
root@localhost $ id1fs -q
```

```
ossama@ossama-virtual-machine:~$ id1fs -q
ID1FS Turned off successfully
ossama@ossama-virtual-machine:~$ id1fs -s
status : off
connected user : None
time : configured
ossama@ossama-virtual-machine:~$
```

### Remarque

Les trois scripts s'arrêtent, et la propriété **status** devient **off** automatiquement après l'exécution de cette commande.

## 4- Composantes clés :

### 4.1 Client :

#### 4.1.1 Gestion Des Utilisateurs :

La gestion des utilisateurs fait référence aux tâches et aux processus liés à la gestion des comptes et des autorisations d'accès aux ressources d'un système informatique. Cela peut inclure la création, la suppression et la modification de comptes d'utilisateurs et des autorisations d'accès aux différentes ressources du système.

Voici quelques exemples de tâches courantes de gestion des utilisateurs :

Création de comptes d'utilisateurs : cela implique de définir un nom d'utilisateur et un mot de passe pour chaque utilisateur.

La commande `signup -u` permet de créer un compte utilisateur ordinaire.

```
root@localhost $ signup -u <username>
```

```
ossama@ossama-virtual-machine:~$ signup -u usr
Enter your password :
Confirm your password :
the user : usr is created successfully
ossama@ossama-virtual-machine:~$
```

La suppression de comptes utilisateurs : cela implique de supprimer les comptes utilisateurs simples

La commande `signup -d` permet de supprimer un compte utilisateur ordinaire.

```
root@localhost $ signup -d
```

```
ossama@ossama-virtual-machine:~$ id1fs -s
status : on
connected user : usr
time : configured
ossama@ossama-virtual-machine:~$ signup -d
The user < usr > is disconnected successfully
The user < usr > is deleted successfully
ossama@ossama-virtual-machine:~$ id1fs -s
status : on
connected user : None
time : configured
ossama@ossama-virtual-machine:~$ connect -u usr
username not existing, please check for spelling mistakes or create a new account
ossama@ossama-virtual-machine:~$
```

Après avoir créé un compte utilisateur, il faut s'y connecter pour pouvoir exécuter des commandes, d'où l'utilisation de la commande `connect` :

```
root@localhost $ connect -u <username>
```

```
ossama@ossama-virtual-machine:~$ connect -u usr
Enter your password :
The user < usr > is connected successfully
ossama@ossama-virtual-machine:~$ id1fs -s
status : on
connected user : usr
time : configured
ossama@ossama-virtual-machine:~$
```

En revanche pour déconnecter d'un compte utilisateur on utilise la même commande avec l'option `-d` :

```
root@localhost $ connect -d
```

```
ossama@ossama-virtual-machine:~$ connect -d
The user < usr > is disconnected successfully
ossama@ossama-virtual-machine:~$ id1fs -s
status : on
connected user : None
time : configured
ossama@ossama-virtual-machine:~$
```

## Le stockage des mots de passe par hachage :

Le hachage de mots de passe est une technique utilisée pour stocker les mots de passe de manière sécurisée. Au lieu de stocker le mot de passe en clair, qui pourrait être facilement compromise en cas de fuite de données, le mot de passe est "haché" en utilisant une fonction de hachage. Le résultat du hachage, appelé "empreinte de hachage", est stocké à la place du mot de passe en clair.

Lorsqu'un utilisateur entre son mot de passe, il est également haché et comparé à l'empreinte de hachage stockée. Si les deux empreintes correspondent, cela signifie que le mot de passe entré est correct et l'utilisateur est autorisé à accéder aux ressources protégées.

Il y a plusieurs avantages au hachage des mots de passe :

**Sécurité :** comme les mots de passe sont stockés sous forme d'empreintes de hachage plutôt que de manière non chiffrée, ils sont moins vulnérables aux attaques de piratage.

**Facilité d'utilisation :** lorsque les mots de passe sont hachés, les utilisateurs n'ont pas besoin de se souvenir de leur mot de passe en clair. Ils n'ont qu'à entrer leur mot de passe et le système s'occupe de le hacher et de le comparer à l'empreinte de hachage stockée.

**Vitesse :** le hachage des mots de passe est généralement plus rapide que le chiffrement des mots de passe, ce qui est un avantage pour les systèmes qui doivent gérer un grand nombre de connexions simultanées.

Ci-dessous, on trouve le fichier **credentials.txt** qui stocke les mots de passes hachés:

```
1 nani : 6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b
2 usr : 6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b
```

Nom de l'utilisateur

mot de passe haché

### 4.1.2 Permission :

La gestion des permissions fait référence à la définition des autorisations d'accès aux différentes ressources d'un système informatique. Cela peut inclure l'accès aux fichiers, et a les données d'une manière générale

Il existe différents types de permissions qui peuvent être accordés aux utilisateurs dans le système de fichiers ID1FS :

Lire : autorise l'utilisateur à lire le contenu d'une ressource, mais pas à la modifier.

Écrire : autorise l'utilisateur à modifier le contenu d'une ressource, mais pas à la supprimer.

Supprimer : autorise l'utilisateur à supprimer une ressource.

ID1FS donne le droit en lecture, écriture et suppression d'un fichiers (données) juste pour le propriétaire par exemple si l'utilisateur X a créé un fichier alors c'est impossible de l'accéder par un autre utilisateur sauf l'utilisateur X.

```
ossama@ossama-virtual-machine:~$ connect -u nani
Enter your password :
The user < nani > is connected successfully
ossama@ossama-virtual-machine:~$ id1fs -f
nani2.csv
1 files found
ossama@ossama-virtual-machine:~$ connect -u usr
Enter your password :
The user < usr > is connected successfully
ossama@ossama-virtual-machine:~$ fopen nani2.csv
Permission denied
ossama@ossama-virtual-machine:~$
```

### 4.1.3 La gestion de fichiers :

La gestion des fichiers fait référence aux tâches et aux processus liés à la création, l'organisation et l'utilisation des fichiers sur un système informatique. Voici les tâches courantes de gestion des fichiers dans ID1FS:



Création de fichiers : cela implique de créer de nouveaux fichiers en utilisant la commande create :

```
root@localhost $ create <file_name>
```

```
ossama@ossama-virtual-machine:~$ create textfile.txt
The file < textfile.txt > created successfully
The file < textfile.txt > successfully backed up
ossama@ossama-virtual-machine:~$ create csvfile.csv
The file < csvfile.csv > created successfully
The file < csvfile.csv > successfully backed up
ossama@ossama-virtual-machine:~$
```

Suppression des fichiers : cela implique de supprimer des fichiers de la section principale (Main section) en utilisant la commande delete :

```
root@localhost $ delete <filename>
```

```
ossama@ossama-virtual-machine:~$ id1fs -f
textfile.txt
csvfile.csv
2 files found
ossama@ossama-virtual-machine:~$ delete textfile.txt
The file textfile.txt removed from main section successfully
ossama@ossama-virtual-machine:~$ id1fs -f
csvfile.csv
1 files found
ossama@ossama-virtual-machine:~$
```

Pour supprimer un fichier de la section principale et secondaire, il faut utiliser la même commande avec l'option -f.

```
root@localhost $ delete <filename> -f
```

```
ossama@ossama-virtual-machine:~$ delete csvfile.csv -f
The file csvfile.csv removed from main section successfully
The file csvfile.csv removed from backup section successfully
ossama@ossama-virtual-machine:~$
```

Restauration des fichiers : cela implique de restaurer des fichiers dans le cas où il est supprimé de la section principale, en utilisant la commande restore.

```
root@localhost $ restore <filename>
```

```
ossama@ossama-virtual-machine:~$ id1fs -f
0 files found
ossama@ossama-virtual-machine:~$ restore textfile.txt
The file textfile.txt restored from backup section successfully
ossama@ossama-virtual-machine:~$ id1fs -f
textfile.txt
1 files found
ossama@ossama-virtual-machine:~$
```

Transfer des fichiers : cela implique d'envoyer des fichiers locaux, en utilisant la commande upload

```
root@localhost $ upload <file_path> <new_file_path>
```

```
ossama@ossama-virtual-machine:~$ upload /home/ossama/Downloads/ossama2.txt usr1.txt
The file < usr1.txt > is uploaded successfully
The file < usr1.txt > successfully backed up
ossama@ossama-virtual-machine:~$
ossama@ossama-virtual-machine:~$
ossama@ossama-virtual-machine:~$ id1fs -f
textfile.txt
usr1.txt
2 files found
ossama@ossama-virtual-machine:~$
```

Téléchargement des fichiers : cela consiste à télécharger des fichiers, en effet transférer des fichiers de L'ID1FS vers un emplacement hors L'ID1FS (le répertoire Téléchargements par défaut), en utilisant la commande download :

```
root@localhost $ download <file_name>
```

```
ossama@ossama-virtual-machine:~$ download usr1.txt
File < usr1.txt > is downloaded successfully
ossama@ossama-virtual-machine:~$
```

Traitement des fichiers : cela consiste à lire des fichiers par plusieurs manières et aussi éditer des fichiers , en utilisant la commande fopen avec différentes options (fopen sans option permet de lire le fichier). Pour afficher le contenu d'un fichier, vous pouvez utiliser la commande suivante :



```
root@localhost $ fopen <file_name>
```

```
ossama@ossama-virtual-machine:~$ fopen usr1.txt
hello1

hello6

hello10

hello16

hello20
ossama@ossama-virtual-machine:~$
```

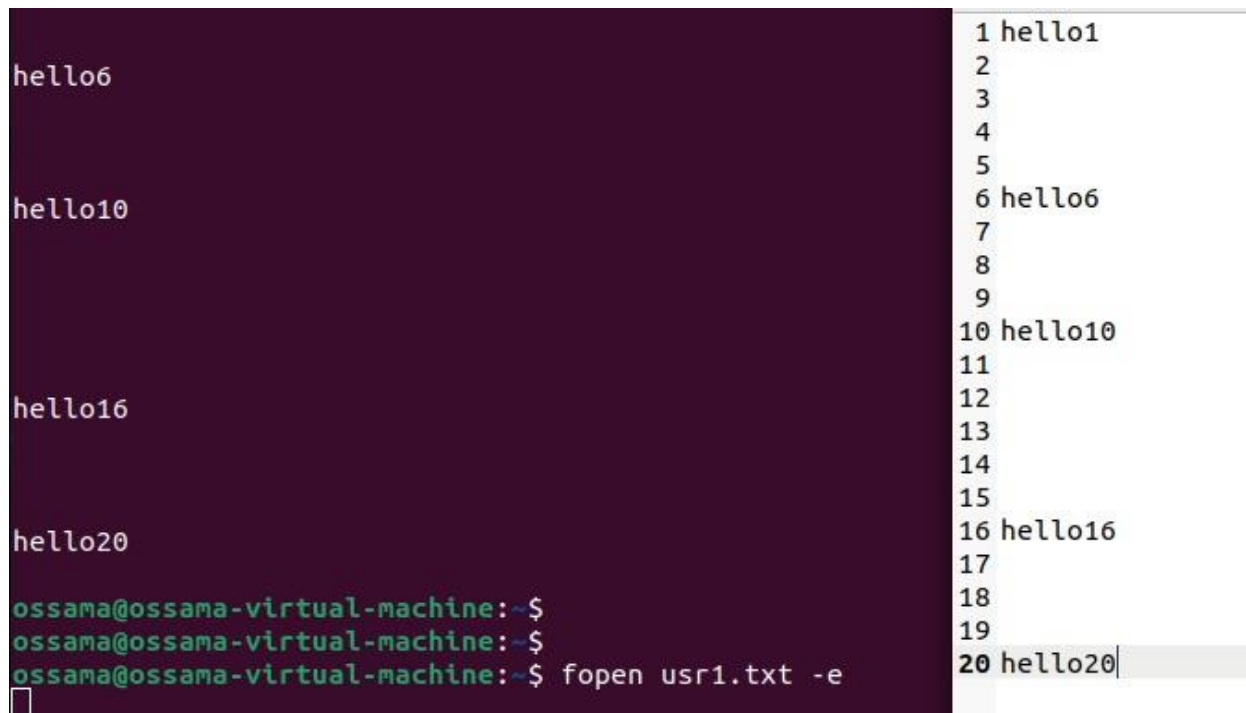
Option	Description
-e	Editer le fichier
-H	Afficher les 10 premières lignes
-t	Afficher les 10 dernières lignes
-m	Affiche le fichier spécifié en argument en remplissant exactement un écran
-l	Similaire à -m, mais qui permet aussi bien de faire des mouvements en arrière qu'en avant dans les fichiers

**Tableau 1 : Les options de la commande fopen**

```
root@localhost $ fopen <file_name> [option]
```

L'option -e :

`fopen -e` lance l'éditeur de texte graphique **gedit**. Il est similaire à des éditeurs de texte tels que Notepad sous Windows ou TextEdit sous Mac. On peut utiliser cette option pour éditer des fichiers.

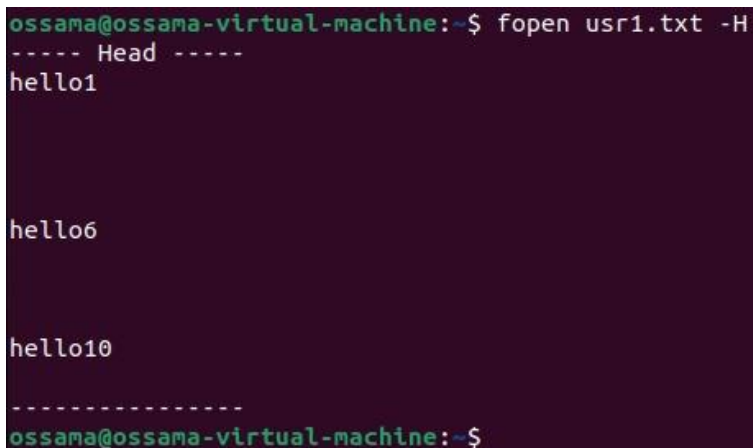


```

ossama@ossama-virtual-machine:~$
ossama@ossama-virtual-machine:~$
ossama@ossama-virtual-machine:~$ fopen usr1.txt -e
1 hello1
2
3
4
5
6 hello6
7
8
9
10 hello10
11
12
13
14
15
16 hello16
17
18
19
20 hello20
  
```

L'option -H :

L'option -H lance la commande **head**, permet de lire les 10 premières lignes d'un fichier à partir de son début. Elle est utilisée pour afficher rapidement le contenu d'un fichier sans avoir à ouvrir un éditeur de texte.



```

ossama@ossama-virtual-machine:~$ fopen usr1.txt -H
----- Head -----
hello1

hello6

hello10
-----
ossama@ossama-virtual-machine:~$
  
```

L'option -t :

L'option -t lance la commande **tail**, permet de lire les 10 dernières lignes d'un fichier. Elle est souvent utilisée pour afficher rapidement le contenu d'un fichier sans avoir à ouvrir un éditeur de texte.

```
ossama@ossama-virtual-machine:~$ fopen usr1.txt -t
----- Tail -----

hello16

hello20

-----
ossama@ossama-virtual-machine:~$
```

L'option -l lance la commande **less**, permet de lire le contenu d'un fichier de manière paginée. Elle est similaire à l'option -m qui lance la commande **more**, et qui permet de lire le contenu du fichier en remplissant exactement un écran, mais offre plus de fonctionnalités et est plus flexible.

```
root@localhost $ fopen usr1.txt -m
```

```
hello1

hello6

hello10

--More-- (29%)
```

```
root@localhost $ fopen usr.txt -l
```

```
hello1

hello6

hello10

:
```

filter : c'est une commande qui permet de trouver tous les fichiers recherchés en se basant sur l'extension si on veut faire une recherche par le symbole de l'extension, respectivement permet aussi de lister les fichiers qui sont inclus dans l'intervalle de taille en octet (byte) donné par l'utilisateur, les options sont respectivement -x, -s.

```
ossama@ossama-virtual-machine:~$ filter -s 100 1000
File Name:  usr1.txt : 104 bytes
1 file found
ossama@ossama-virtual-machine:~$
```

```
ossama@ossama-virtual-machine:~$ id1fs -f
textfile.txt
csvfile.csv
htmlfile.html
usr1.txt
4 files found
ossama@ossama-virtual-machine:~$ filter -x txt html
----- txt -----
textfile.txt
usr1.txt
2 files found
----- html -----
htmlfile.html
1 file found
ossama@ossama-virtual-machine:~$
```

### L'affichage des fichiers d'un utilisateur :

La commande `id1fs -f` permet de lister juste les fichiers de l'utilisateur actuellement connecté.

```
ossama@ossama-virtual-machine:~$ id1fs -f
textfile.txt
csvfile.csv
htmlfile.html
usr1.txt
4 files found
ossama@ossama-virtual-machine:~$
ossama@ossama-virtual-machine:~$
```

N.B :

Chaque commande a une option supplémentaire c'est `-h` (`-- help`), qui donne la description détaillée de la commande et ses autres options .

```
ossama@ossama-virtual-machine:~$ fopen --help
usage: fopen [-h] [-e] [-m] [-H] [-t] [-l] name

positional arguments:
  name                display the content of a text file

options:
  -h, --help          show this help message and exit
  -e, --edit          to edit the given file
  -m, --more          to display (not to edit) the content page by page
  -H, --head          to display the first lines
  -t, --tail          to display the last lines
  -l, --less          to display the content faster
ossama@ossama-virtual-machine:~$
```

## 4.2 Server

### 4.2.1 MASTER

#### 4.2.1.1 Meta données :

Les métadonnées sont des informations qui décrivent ou qui s'appliquent à d'autres données. Elles permettent de fournir un contexte aux données et de les rendre plus faciles à comprendre et à utiliser. Par exemple, les métadonnées d'une image peuvent inclure son titre, son auteur, sa date de création, sa taille, etc. Les métadonnées sont souvent utilisées dans les bases de données, les

fichiers multimédias, les bibliothèques et les archives pour organiser et gérer les informations de manière efficace.

- Fichier :

On a utilisé Exiftool pour afficher les métadonnées des fichiers, pour en extraire certaines et les stocker sous forme de dictionnaire (clé : valeur), pour modifier les métadonnées d'un fichier et par fois pour les supprimer complètement, et on a lié cette commande avec le code python en utilisant la bibliothèque json afin de construire des fichiers de type json qui contiennent les métadonnées.

On choisit les fichiers json car ce type est facile à manipuler et facile à considérer comme un dictionnaire. Pour les fichiers existants dans la section principale, ils sont enregistrés dans le dossier **Master/FILES\_METADATA** et ceux dans la section secondaire dans **Master/BACKUPS\_METADATA**

EX :

```
1 {
2   "ExifTool Version Number": "12.40",
3   "File Name": "usr1.txt",
4   "Directory": "/home/ossama/Desktop/test/idfs/Server/Main/Node2",
5   "File Size": "104 bytes",
6   "File Modification Date/Time": "2023:01:10 09:33:51+01:00",
7   "File Access Date/Time": "2023:01:10 09:33:51+01:00",
8   "File Inode Change Date/Time": "2023:01:10 09:33:51+01:00",
9   "File Permissions": "-rw-rw-r--",
10  "File Type": "TXT",
11  "File Type Extension": "txt",
12  "MIME Type": "text/plain",
13  "MIME Encoding": "us-ascii",
14  "Newlines": "Unix LF",
15  "Line Count": "57",
16  "Word Count": "7",
17  "Path": "/home/ossama/Desktop/test/idfs/Server/Main/Node2/usr1.txt",
18  "Owner": "usr"
19 }
```



Chaque utilisateur peut afficher aux Metadata de ses fichiers en utilisant la commande md :

```
root@localhost $ md <file_name>
```

```
ossama@ossama-virtual-machine:~$ md usr1.txt
File Name : usr1.txt
File Size : 104 bytes
File Modification Date/Time : 2023:01:10 09:33:51+01:00
File Access Date/Time : 2023:01:10 09:33:51+01:00
File Inode Change Date/Time : 2023:01:10 09:33:51+01:00
File Type : TXT
File Type Extension : txt
MIME Type : text/plain
MIME Encoding : us-ascii
Newlines : Unix LF
Line Count : 57
Word Count : 7
Owner : usr
ossama@ossama-virtual-machine:~$
```

L'utilisateur peut également ajouter des Metadata a ses fichiers en utilisant l'option -a:

```
root@localhost $ md <file_name> -a <key>
```

```
ossama@ossama-virtual-machine:~$ md usr1.txt -a Description
['Description'] = ?
This is a description for usr1.txt file
Metadata of < usr1.txt > modified successfully
ossama@ossama-virtual-machine:~$ md usr1.txt
File Name : usr1.txt
File Size : 104 bytes
File Modification Date/Time : 2023:01:10 09:33:51+01:00
File Access Date/Time : 2023:01:10 09:33:51+01:00
File Inode Change Date/Time : 2023:01:10 09:33:51+01:00
File Type : TXT
File Type Extension : txt
MIME Type : text/plain
MIME Encoding : us-ascii
Newlines : Unix LF
Line Count : 57
Word Count : 7
Owner : usr
Added keys : ['Description']
Description : This is a description for usr1.txt file
ossama@ossama-virtual-machine:~$
```

- Node

Pour chaque Node, on enregistre le nom des fichiers qu'il contient sous forme d'un fichier json nommé **nodes\_md.json**, et on stock l'espace disponible pour chaque Node de la section principale dans un fichier texte nommé **main\_nodes.txt** et ceux de la section secondaire dans autre fichier nommé **backup\_nodes.txt**.

```

1 {
2   "Nodes": {
3     "Node1": [
4       "nani.csv",
5       "textfile.txt",
6       "csvfile.csv",
7       "htmlfile.html"
8     ],
9     "Node2": [
10      "usr1.txt"
11    ]
12  }
13 }
```

Remarque l'utilisateur n'a pas droit d'accéder aux Metadata des nodes pour des raisons de sécurité, d'où l'absence d'une commande qui exécute une telle tâche.

#### 4.2.1.2 Logging

Le logging est un processus qui consiste à enregistrer des informations sur l'activité d'un système de fichier. Cela peut être utile pour diverses raisons, telles que le debugging, lorsqu'il y a un problème avec le système, les logs peuvent fournir des informations précieuses pour comprendre ce qui s'est passé et comment le résoudre.

En résumé, le logging est un outil essentiel pour comprendre et gérer l'activité d'un système de fichier. Il peut aider à résoudre les problèmes, à améliorer les performances et à assurer la sécurité du système.



dans notre système de fichiers ID1FS, les logs fournit des informations tels que la date de l'action, le nom de l'utilisateur qui exécute l'action (système si aucune utilisateur connecté) et le message descriptif de l'action, de plus le niveau qui peut prendre trois états :

DEBUG : pour suivre les actions de l'utilisateur.

INFO : pour enregistrer des informations qui sont utiles afin de suivre l'exécution du programme, mais qui ne sont pas nécessairement critiques pour son fonctionnement.

ERROR : pour enregistrer des erreurs dus problèmes internes comme des bugs, ou des problèmes externes comme des erreurs de saisie de l'utilisateur.

Voici quelques exemples de messages qui pourraient être enregistrés au niveau du fichier **actions.log** :

```
1291 DEBUG : system : 2023-01-10 10:08:16,521 : Command used < connect -u usr >
1292 INFO : system : 2023-01-10 10:08:17,602 : The user < usr > is connected successfully
1293 DEBUG : usr : 2023-01-10 10:08:34,993 : Command used < id1fs -s >
1294 DEBUG : usr : 2023-01-10 10:08:48,151 : Command used < signup -d >
1295 INFO : usr : 2023-01-10 10:08:48,153 : The user < usr > is disconnected successfully
1296 INFO : usr : 2023-01-10 10:08:48,154 : The user < usr > is deleted successfully
1297 DEBUG : system : 2023-01-10 10:08:50,870 : Command used < id1fs -s >
1298 DEBUG : system : 2023-01-10 10:09:02,294 : Command used < connect -u usr >
1299 ERROR : system : 2023-01-10 10:09:02,295 : Failed to connect --> Username not found
```

## 4.2.2 La section principales (main section) :

la section principale(Main section) dans notre système est un dossier qui contient tous les fichiers qui sont créé et transfère par l'utilisateur a notre système, ce dossier est un crucial c'est pour cette raison on le sécurise par une autre copie de dossier(Backup section), dans si la section principale a été supprimée notre system de fichier fait la restaurée en dupliquant le Backup section, en revanche si le Backup section a été tomber alors notre file system le restaurée par la section principale.

## 4.2.3 La section secondaire (backup section) :

La sauvegarde (ou "backup" en anglais) est l'opération consistant à enregistrer une copie de tous les fichiers et données d'un système de fichiers. Elle a pour but

de protéger ces données en cas de perte ou de dommage, par exemple en cas de panne matérielle, de virus ou de suppression accidentelle.

Il y a plusieurs raisons pour lesquelles la sauvegarde est importante dans un système de fichiers :

**Sécurité des données:** en cas de problème, les sauvegardes permettent de restaurer le système de fichiers à un état précédent, ce qui peut être crucial pour préserver les données importantes.

**Continuité des activités:** en cas de panne du système, les sauvegardes permettent de reprendre rapidement les activités en restaurant le système à partir de la sauvegarde la plus récente.

## 5- Étude Benchmarking:

### 5.1 Bénéfices attendus :

- Restauration : la sauvegarde des fichiers.
- Consistance : assurer par la vérification périodique de la configuration de la date et l'horloge, en se connectant a un serveur NTP (Network time protocole) externe référence tant que le système est actif, grâce au script **ntptime.py**
- fiabilité : assurer par deux scripts qui s'exécutent simultanément tant que le système est actif, autrement dit, tant que la propriété **status** est **on**, ce sont :
  - **nodes\_reliable.py** : responsable de la restauration des fichiers en cas de suppression brusque pour une raison ou une autre.
  - **master\_backup.py** : responsable de la restauration de master en cas de suppression brusque.

### 5.2 Les Points à Développer :

Les fonctionnalités à développer de notre projet ID1FS sont :

- Versionning
- Setup.py : un seul script d'installation

- Security : l'optimisation des algorithmes de sécurités
- Connexion au serveur distant

## 5.3 Comparaison Entre ID1FS & HDFS :

Voici un tableau comparatif qui décrit les aspects de ressemblance et différences entre **ID1FS** et **HDFS**.

Comparison criteria	ID1FS	HDFS
Platform	Linux	Cross-platform
Developer	ENSAH students	Yahoo. now it is an open-source framework
servers	Maser, Main/Backup section	Name node, data node
Hardware used	Commodities hardware	Commodities hardware
Read and write	It is a multiple read write model	Write once and reading can be done many times
File deletion	Optional delete (from just main section or from main and backup section)	The files deleted are renamed into a specific folder and after that the file will be removed thru garbage
Logging	YES	YES
Data Security	YES	YES
Other operation	Append and remove	Only append is possible
Simplicity	YES	NO
Flexibility	YES	YES
Filtering (search)	YES	YES
fault tolerance	YES	YES
scalable	YES	YES

**Tableau 2: Comparison between ID1FS and HDFS.**

## 6 – Tableau récapitulatif des commandes :

Commande [option]	Description
<b>id1fs -s</b>	Afficher le statu de id1fs
<b>id1fs -o</b>	Démarre le id1fs
<b>idfs -q</b>	Arrête le id1fs
<b>idfs -f</b>	Permet de lister les fichiers de l'utilisateur actuelle
<b>sinup -u</b>	Cree un compte utilisateur
<b>sinup -d</b>	Supprimer un compte utilisateur
<b>connect -u</b>	Se connecter a un compte utilisateur
<b>connect -d</b>	Se déconnecter a un compte utilisateur
<b>create</b>	Cree un nouveau fichier
<b>delete</b>	Supprime un nouveau fichier de la section principale
<b>delete -f</b>	Supprime un nouveau fichier de la section principale et secondaire
<b>restore</b>	Restore un fichier supprimé de la section principale
<b>upload</b>	Transfère un fichier vers le id1fs
<b>download</b>	Télécharge un fichier de l'id1fs
<b>fopen</b>	Permet de visualiser le contenu
<b>fopen -e   -H   -t   -m   -l</b>	-e : Editer un fichier   -H : Afficher les 10 premières lignes   -t : Afficher les 10 dernières lignes   -m : Affiche le fichier spécifié en argument en remplissant exactement un écran   -l : Similaire à -m, mais qui permet aussi bien de faire des mouvements en arrière qu'en avant dans les fichiers
<b>filter -x   -s</b>	-x : trouver des fichiers recherchés en se basant sur extension   -s : lister les fichiers qui sont incluse dans l'intervalle de taille en octet (byte) donné par l'utilisateur
<b>md</b>	Afficher les Meta data des fichiers
<b>md &lt;file_name&gt; -a &lt;key&gt;</b>	Ajouter des métadonnées au fichier

# Fin.