



---

# RAPPORT DE PROJET

---

## Conception personnalisée d'un système de fichiers et d'une distribution Linux.

ENCADRE PAR

*PR. MOHAMED CERRADI*

RÉALISÉ PAR

-OUAQUIF DIKRA -ERAHOUTEN ILIASS  
-ATMANI OUAMAIMA -BOUHSSAR WIAME  
-WAKRIM YOUSSEFA -ANOUK ZAKARIAE

## ***Remerciements au Professeur Cherradi***

Au terme de cette aventure captivante à travers les méandres du développement d'un système de fichiers et d'une distribution Linux personnalisée, nous souhaitons exprimer notre profonde gratitude pour votre encadrement éclairé. Votre expertise, vos conseils judicieux et votre soutien indéfectible ont été les catalyseurs de notre réussite. Ce projet a été une formidable immersion dans le monde complexe de la programmation système, et chaque ligne de code a été façonnée par votre inspiration.

Merci d'avoir été notre boussole dans ce voyage technologique, transformant les défis en opportunités d'apprentissage enrichissantes. Nous sommes reconnaissants pour votre dévouement, qui a rendu cette expérience formatrice et mémorable.



# **Sommaire**

---

## **1. Système de fichier:**

**01. Introduction**

**02. CONCEPTION ET DEVELOPPEMENT DE IDFS**

**03. L'ARCHITECTURE DE L'IDFS**

**04. Diagramme de Classes UML**

**05. Hiérarchie du IDFS**

**06. Cadre de Développement et Choix Technologiques**

**07. Réalisation du Projet**

**08. LES COMMANDES**

**09. JOURNALISATION**

**10. Les métadonnées:**

# **Sommaire**

---

## **2.Distribution:**

**01. Crédit d'une Distribution Linux personnalisée**

**02. Préparation Essentielle**

**03 Personnalisation Avancée et Génération de l'Image ISO**

**04. Benchmark**

**05. CONCLUISION**

# 1. Introduction

---

## 1. Contexte

Le paysage actuel des systèmes de fichiers et des distributions Linux révèle une diversité de solutions, chacune conçue pour répondre à des besoins spécifiques. Les systèmes de fichiers existants, tels que ext4, Btrfs et XFS, offrent des performances variées et des fonctionnalités spécifiques, mais ils peuvent ne pas toujours répondre de manière optimale aux exigences particulières d'un projet spécifique.

De même, les distributions Linux, qu'elles soient populaires comme Ubuntu, Debian, ou spécialisées comme Arch Linux, sont élaborées pour satisfaire un large éventail d'utilisateurs. Cependant, chaque projet informatique possède des besoins uniques, qu'ils soient liés à la sécurité, à la performance, à la taille de l'empreinte mémoire, ou à d'autres caractéristiques spécifiques.

La nécessité de développer une solution personnalisée découle de l'identification de ces lacunes dans les systèmes et distributions existants, ainsi que des besoins spécifiques du projet en question. Ces besoins peuvent être liés à des contraintes de performances, à des exigences de sécurité strictes, à des configurations matérielles spécifiques, ou à d'autres critères particuliers qui ne sont pas entièrement adressés par les solutions disponibles sur le marché.

En réalisant une étude approfondie des besoins spécifiques, nous avons identifié des opportunités d'optimisation et de personnalisation qui justifient la création d'une solution sur mesure. Cela permettra non seulement de répondre de manière précise aux exigences du projet, mais également de garantir une performance, une sécurité et une efficacité maximales dans le contexte particulier dans lequel cette solution sera déployée.

En somme, le contexte de ce projet émerge de la reconnaissance des limites inhérentes aux solutions existantes et de la volonté de concevoir une alternative sur mesure, taillée sur mesure pour les impératifs spécifiques et les ambitions de ce projet de développement de système de fichier et de distribution Linux personnalisée.

## 2.Objective

Le projet a pour but principal de répondre à la demande croissante des utilisateurs en proposant une distribution personnalisée. Cette personnalisation avancée permettra aux utilisateurs de configurer leur environnement informatique de manière détaillée, améliorant ainsi la satisfaction et l'efficacité de l'utilisateur final.

Dans le même ordre d'idées, la création d'une distribution personnalisée mettra l'accent sur la flexibilité pour répondre aux divers besoins des utilisateurs, qu'ils soient professionnels, académiques ou personnels. L'objectif est d'assurer une adaptation transparente aux différents contextes d'utilisation, garantissant une expérience utilisateur optimale.

Un autre objectif crucial du projet sera l'optimisation des performances du stockage et de la gestion des données en développant un système de fichiers personnalisé. Cela implique une conception attentive du système de fichiers pour maximiser l'efficacité des opérations de lecture, d'écriture et de recherche.

En outre, le projet ne se limitera pas à répondre aux besoins actuels mais visera également à anticiper les évolutions technologiques et les demandes émergentes. En adoptant une approche modulaire et évolutive, la solution sera prête à évoluer avec les progrès de l'informatique, assurant ainsi une pertinence à long terme.

Enfin, les métadonnées joueront un rôle crucial en permettant une classification avancée des fichiers. En exploitant les métadonnées de manière stratégique, le projet cherche à offrir aux utilisateurs un contrôle accru sur leurs données, facilitant la recherche, l'organisation et la compréhension du contenu. Cette approche renforcera la convivialité et la pertinence du système dans son ensemble.

## **2. STRUCTURE DU RAPPORT**

Ce rapport constitue une synthèse des travaux accomplis au cours du projet et est structuré en deux chapitres :

- **Premier chapitre :**

Dans cette première section, on va aborder la dimension théorique du projet, en exposant l'architecture de notre système de fichiers et en explorerant le diagramme de classes UML, détaillant la hiérarchie des classes dans notre conception.

Dans la deuxième section , on va explorer les diverses technologies mises en œuvre ainsi que la mise en pratique de notre système de fichiers.

- **Deuxième chapitre :**

Ce chapitre se consacrera à la concrétisation de notre distribution personnalisée. On détaillera les choix technologiques, la configuration du système de fichiers personnalisé, le développement de la distribution, les tests réalisés, et enfin, le déploiement pratique de la solution, offrant ainsi une perspective complète du projet.

# I.CONCEPTION ET DEVELOPPEMENT DU SYSTEME DE FICHIER

## **1.C'est quoi un système de fichier**

Un système de fichier constitue l'infrastructure fondamentale qui régit l'organisation, le stockage et la gestion des données sur un support de stockage au sein d'un système d'exploitation. Il agit comme une méthode structurée pour classer et accéder à des fichiers, tels que des documents, des programmes ou des médias. La hiérarchie est établie à travers des répertoires, qui servent de conteneurs organisés, et des fichiers, qui représentent les données individuelles. Les noms de fichiers attribuent des identifiants uniques à chaque entité, tandis que les permissions définissent les droits d'accès. Les structures de stockage dictent la manière dont les données sont physiquement stockées sur le support. Chaque système d'exploitation a son propre système de fichiers, tel que NTFS pour Windows, ext4 pour Linux, ou HFS+ pour macOS, optimisé en fonction de ses exigences spécifiques et offrant des fonctionnalités distinctes.

## **2.L'ARCHITECTURE DE L'IDFS**

---

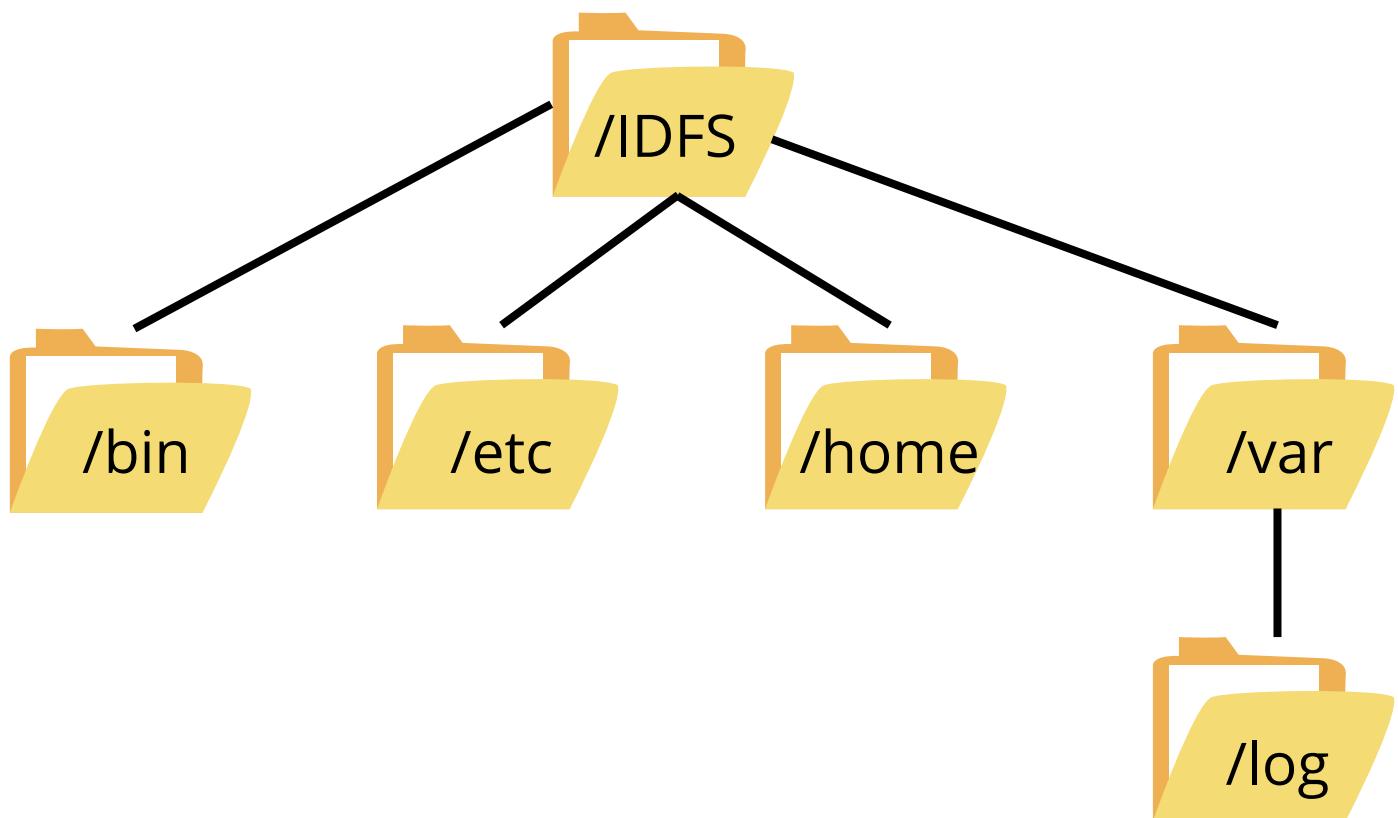
Au sein de notre IDFS , chaque répertoire incarne un rôle unique, créant une symphonie cosmique de fonctionnalités informatiques. Le "**SunMaster**" se tient en tant que noyau lumineux, abritant les configurations cruciales qui guident notre système. La "**Planet\_BIN**" évoque une planète d'exécutables puissants, générant une énergie informatique essentielle. Quant à la "**Planet\_ETC**", elle forme une orbite délicate, offrant des configurations exquises qui sculptent les contours de notre univers informatique. Naviguons vers la "**Planet\_LOG**", un ciel étoilé de journaux électroniques où chaque fichier lumineux raconte l'histoire des événements informatiques. Enfin, la "**Planet\_VAR**" danse avec grâce, abritant des données variables en mouvement constant, créant une dynamique essentielle à notre système. Ainsi, chaque répertoire de IDFS joue un rôle distinct dans cette galaxie d'éléments informatiques, orchestrant une danse cosmique où données et idées fusionnent harmonieusement.

## 1. Diagramme de Classes UML



## 2.Hiérarchie du IDFS

La disposition du système de fichiers IDFS suit la structure suivante



- **/bin** : Contient les commandes système essentielles.
- **/etc** : Inclut les fichiers de configuration, **passwd**, **group**, **shadow**, et un fichier de **métadonnées** pour chaque utilisateur.
- **/home** : Dédié aux répertoires personnels des utilisateurs.
- **/var** : Contient le sous-répertoire **/log** qui héberge le fichier de journalisation **logfile.log**.

### 3.Cadre de Développement et Choix Technologiques

---

#### 1-Intégration de Linux

##### Pourquoi linux?

Le choix de Linux comme plateforme fondamentale pour la distribution repose sur plusieurs considérations cruciales qui démontrent son aptitude exceptionnelle à servir de base au système de fichiers. Nous avons opté pour l'intégration de Linux dans la conception de notre système de fichiers distribué en raison de divers facteurs :

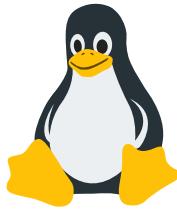


Figure: Logo Linux

**-Stabilité et Fiabilité** :choix de Linux pour sa réputation de stabilité, assurant un fonctionnement continu et fiable.

**- Personnalisation Modulaire** : La nature modulaire de Linux permet une personnalisation étendue, adaptée aux spécificités du projet.

**-Interopérabilité avec Divers Matériels** :Linux offre une interopérabilité exceptionnelle avec une variété de matériels, assurant une compatibilité maximale.

**- Adaptabilité et Flexibilité** :La capacité d'adaptation de Linux le rend flexible, capable de répondre aux besoins changeants des utilisateurs.

**- Optimisation des Ressources** :Linux optimise efficacement l'utilisation des ressources matérielles, garantissant des performances optimales du système de fichiers.

En outre, Linux est également connu pour être un système d'exploitation open source, ce qui signifie que son code source est accessible à tous et peut être modifié pour répondre aux besoins spécifiques de chaque utilisateur. Cette caractéristique permet une grande flexibilité et une personnalisation encore plus poussée. De plus, la communauté open source qui soutient Linux est très active et offre un support technique en temps réel ainsi que des mises à jour régulières pour améliorer la sécurité et les performances du système. En fin de compte, le choix de Linux pour notre système de fichiers distribué est donc un choix évident qui garantit une base solide, fiable et adaptable pour notre projet.

## 2-Intégration Python

### 1- Rôle de Python dans le Système de Fichiers:

La contribution de Python au développement du système de fichiers est multifacette. Tout d'abord, Python agit en tant que catalyseur dans la réalisation de scripts système, facilitant ainsi la mise en œuvre de fonctionnalités spécifiques. Ses atouts résident notamment dans sa flexibilité, sa simplicité syntaxique et sa capacité à être intégré harmonieusement dans divers aspects du système.



### 2- Utilisation de Bibliothèques Python:

**-Subprocess:** la bibliothèque subprocess est intégrée pour l'exécution transparente de commandes système, jouant un rôle essentiel dans la gestion des processus et des commandes shell.

**-Logging :** l'utilisation de la bibliothèque logging est cruciale pour assurer le suivi des opérations du système, offrant ainsi une visibilité et une traçabilité indispensables.

**-Argparse :** une bibliothèque dédiée à l'analyse des arguments en ligne de commande, simplifie significativement la personnalisation des scripts dans le contexte du système de fichiers.

**-Shutil:** La bibliothèque shutil est exploitée pour des opérations de haut niveau sur les fichiers et les répertoires, apportant une couche d'abstraction puissante au système de fichiers.

**-Sys, Systat, Getpass:** les bibliothèques sys, systat et getpass jouent des rôles spécifiques dans la gestion des informations système, avec des applications particulières au sein du système de fichiers.

**-Datetime:** Python, à travers la bibliothèque datetime, offre des fonctionnalités de manipulation des dates et heures, contribuant ainsi à la gestion temporelle du système de fichiers.

**-Pwd, Grp, Stat :** les bibliothèques pwd, grp et stat sont intégrées pour la gestion des utilisateurs, groupes et statistiques de fichiers, jouant un rôle crucial dans la configuration et la gestion des droits d'accès.

**-Signal:** la gestion des signaux, facilitée par la bibliothèque signal, contribue au contrôle efficace des processus dans le contexte du système de fichiers.

**-Path:** la manipulation des chemins de fichiers, effectuée à l'aide de la bibliothèque path, revêt une importance primordiale dans la gestion des répertoires et des fichiers.

**-Json :** la bibliothèque json est exploitée pour le traitement des données au format JSON, apportant une méthodologie efficace pour la configuration du système de fichiers et le stockage de paramètres.

**-Bcrypt:** la sécurisation des données sensibles est assurée par la bibliothèque bcrypt, qui intègre des mécanismes robustes de hachage dans le contexte de la sécurité des fichiers.

**-Tarfile:** la manipulation des archives tar, grâce à la bibliothèque tarfile, est intégrée pour des opérations de sauvegarde et de distribution de fichiers au sein du système.

**-Re (Expressions Régulières):** l'utilisation d'expressions régulières, à travers la bibliothèque re, offre des capacités avancées de recherche et de filtrage de fichiers, contribuant ainsi au traitement efficace des noms de fichiers.

## 4. Réalisation du Projet

### 1. Explication de IDFS.py

Le script Python, débutant par la déclaration de l'interpréteur Python et des encodages utilisés, importe plusieurs modules essentiels tels que `pathlib`, `os`, `subprocess`, `logging`, `bcrypt`, `argparse`, et `shutil` pour la gestion des opérations liées au système de fichiers. La récupération du répertoire du script est effectuée à l'aide de la fonction `Path(\_\_file\_\_).resolve().parent`.

En suivant, le script crée le répertoire racine 'IDFS' s'il n'existe pas déjà, utilisant le module `pathlib`. Il établit ensuite les sous-répertoires 'bin', 'etc', et 'var' au sein de 'IDFS', garantissant leur création uniquement si ces répertoires n'existent pas déjà.

En ce qui concerne le répertoire 'etc', le script crée des fichiers spécifiques ('shadow', 'group', 'passwd') simulant la configuration des utilisateurs et des groupes. Le fichier 'passwd' est particulièrement initialisé avec des informations spécifiques à l'utilisateur root, telles que le nom d'utilisateur, l'UID, le GID, le répertoire personnel, et le shell.

Les fichiers 'group' et 'shadow' sont également initialisés avec des données simulées pour l'utilisateur root, tout en utilisant le module `bcrypt` pour crypter le mot de passe de manière sécurisée. Le répertoire 'var' est créé, et à l'intérieur, le sous-répertoire 'log' est établi, incluant le fichier 'logfile.log'.

Le script prend également en charge la création d'un répertoire 'home' au sein de 'IDFS'. Enfin, il définit les permissions du répertoire 'bin' à 755 (rwxr-xr-x), assurant ainsi un certain niveau de sécurité et d'accès aux fichiers d'exécution.

Il est important de noter que le script est bien commenté, permettant une compréhension claire des sections, et inclut des indications sur les endroits où des valeurs spécifiques, telles que les informations de l'utilisateur root et le mot de passe, peuvent être ajustées en fonction des besoins spécifiques du projet. En somme, ce script fournit une base robuste pour simuler une structure de système de fichiers dans un contexte Linux.

## 2.Les commandes:

Dans cette section, nous dévoilons la concrétisation et l'intégration de notre projet. Dans un premier temps, nous exposons l'environnement de travail ainsi que les outils de développement employés. Par la suite, nous procédons à une présentation détaillée du projet, mettant en lumière notre avancée significative dans le développement de notre système de fichiers. Au cours de cette étape, nous dévoilons l'essence de notre expertise en mettant en avant un arsenal de plus de 40 commandes que nous avons minutieusement élaborées. Notamment, notre système de fichiers se distingue par sa capacité à gérer les utilisateurs, les groupes, ainsi que les options avancées sur les fichiers.

### 1-Gestion des Utilisateurs:

La gestion des utilisateurs dans un système de fichiers est cruciale pour assurer la sécurité des données en définissant des autorisations spécifiques. Cela permet un contrôle précis sur l'accès aux fichiers, favorisant un environnement informatique sécurisé et organisé.

#### 1.galaxyuadd:

Cette commande requérant des droits d'administration (être le root), permet de créer un nouvel utilisateur avec un mot de passe haché, un répertoire personnel et une affiliation à un groupe défini. Elle effectue également une vérification pour éviter la création d'un utilisateur déjà existante.

```
root@dikra-VirtualBox:/# galaxyuadd
Entrez le nom d'utilisateur : try
Entrez le mot de passe :
root@dikra-VirtualBox:/#
```



#### 2.nameu:

La commande "nameu" permet d'extraire le nom de l'utilisateur récemment connecté. Elle offre une manière efficace d'obtenir cette information sans avoir besoin de parcourir des répertoires spécifiques. Son utilité réside dans la facilitation de l'identification rapide de l'utilisateur actif, contribuant ainsi à une gestion efficace des sessions et des opérations système.

```
root@dikra-VirtualBox:/IDFS# nameu
User name : root
root@dikra-VirtualBox:/IDFS#
```

## **3.Delus:**

"delus", réservée aux priviléges root, simplifie et sécurise la suppression d'utilisateurs en nécessitant des autorisations élevées. Cette commande offre une solution efficace pour éviter les erreurs potentielles liées à des manipulations manuelles. (delus)

## **4.Utr:**

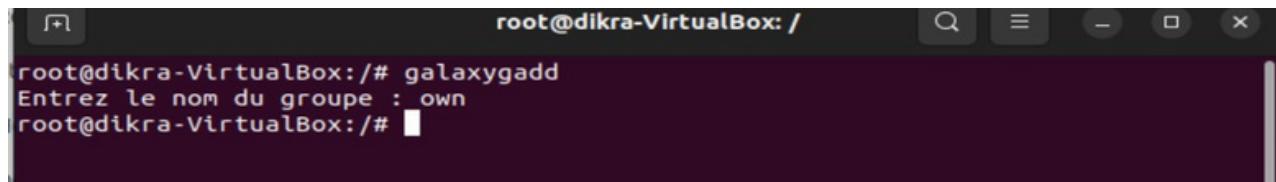
La commande "utr" facilite la transition d'un utilisateur simple vers le statut root. Cette commande offre une solution pratique pour les utilisateurs nécessitant temporairement des droits administratifs sans se déconnecter et se reconnecter en tant que root (utr)

## **2-Gestion des groupes:**

La gestion des groupes dans un système de fichiers est essentielle pour organiser les utilisateurs, simplifier la gestion des autorisations et renforcer la sécurité. Elle permet d'attribuer des autorisations de manière groupée, facilitant la maintenance et limitant les risques liés à des autorisations inappropriées

## **1Galaxygadd**

La commande "galaxygadd" facilite la création d'un nouveau groupe. Son exécution simplifie le processus de gestion des groupes en permettant la création efficace de nouveaux groupes selon les besoins spécifiques du système. (cette commande ne peut être exécuté que par root)



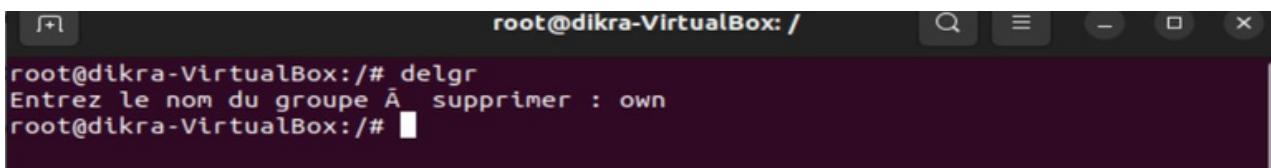
```
root@dikra-VirtualBox: /  
root@dikra-VirtualBox:/# galaxygadd  
Entrez le nom du groupe : own  
root@dikra-VirtualBox:/#
```



## Partie technique:

### 2.Delgr:

La commande "delgr" simplifie la suppression d'un groupe, nécessitant des priviléges élevés pour son exécution.



```
root@dikra-VirtualBox:/# delgr
Entrez le nom du groupe à supprimer : own
root@dikra-VirtualBox:/#
```

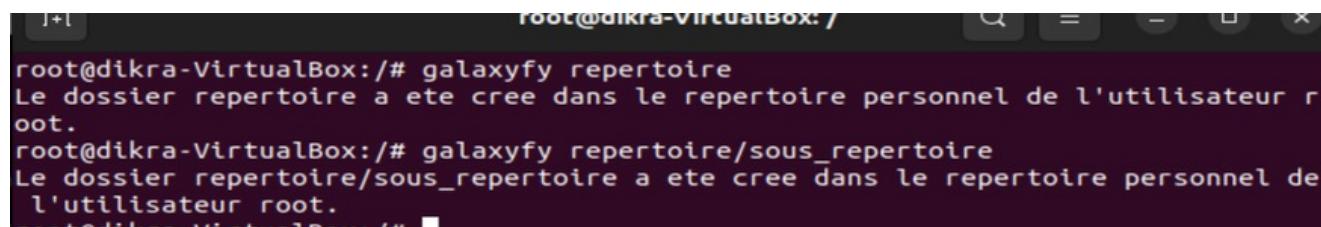


## 3-Fonctionnalités des Fichiers : Contrôle d'Accès, Métadonnées et Gestion Efficace des Données

La création, la suppression, le contrôle d'accès, les métadonnées et la gestion efficace des données sont tous essentiels dans un système de fichier. Ils assurent la sécurité, l'organisation, les performances optimales et la traçabilité des informations, formant ainsi les fondements d'un système de stockage robuste.

### 1.Galaxyfy:

La commande "galaxyfy" a été spécialement conçue pour simplifier la création de répertoires et de sous-répertoires dans un environnement de fichiers. Elle offre une manière rapide et intuitive de structurer des espaces de stockage en générant une hiérarchie de dossiers



```
root@dikra-VirtualBox:/# galaxyfy repertoire
Le dossier repertoire a été créé dans le répertoire personnel de l'utilisateur root.
root@dikra-VirtualBox:/# galaxyfy repertoire/sous_repertoire
Le dossier repertoire/sous_repertoire a été créé dans le répertoire personnel de l'utilisateur root.
```

# Partie technique:

## 2.Ignite:

En utilisant cette commande, les utilisateurs peuvent instantanément générer des fichiers en spécifiant le nom, le format et l'emplacement désirés. Ignite simplifie le processus de création de fichiers, offrant une flexibilité maximale pour répondre aux besoins spécifiques de l'utilisateur.

```
root@dikra-VirtualBox:/# ignite file  
File created successfully: /IDFS/home/root/file  
Metadata file created at /IDFS/etc/metadata_root
```

## 3.copy:

La commande **cp** est utilisée pour copier des fichiers et des répertoires dans le système de fichiers

```
root@wiame-VirtualBox:~# copy fichier1 /IDFS/home/root/dossier1
```

## 4.shifft:

Cette commande déplace ou renomme des fichiers et répertoires, facilitant ainsi la gestion dynamique des données sur les systèmes

```
root@wiame-VirtualBox:~# shiftt fichier2 /IDFS/home/root/dossier1
```

## 5. cwd:

La commande **cwd** (Current Working Directory) est utilisée pour afficher le chemin absolu du répertoire de travail actuel dans le terminal . Lorsqu'elle est exécutée, elle imprime le chemin du répertoire dans lequel vous vous trouvez à un instant donné.

```
root@wiame-VirtualBox:~# cwd  
/root
```

## 6.dcache

La commande "**dcache**" est utilisée pour lister les fichiers et répertoires dans un répertoire, y compris les fichiers cachés

```
root@wiame-VirtualBox:~# dcache
logfile.log
.local
bashrc
.bashrc
.bash_history
snap
.profile
.cache
```

## 7.writ:

Cette commande est un éditeur de texte en mode console . Elle offre une interface conviviale pour créer, éditer et visualiser des fichiers texte directement dans le terminal. Elle simplifie la modification de fichiers en offrant des raccourcis clavier intuitifs pour copier, coller, rechercher et effectuer d'autres actions éditoriales .

```
root@dikra-VirtualBox:/# writ fichier.txt
```



## 8. ddetail:

La commande **ddetail** est utilisée pour lister les fichiers et répertoires d'un répertoire en format détaillé (long)

```
root@wiame-VirtualBox:~# ddetail
-rw-r--r--    1      root    root    410    2023-12-25 12:49:56.468565    l
logfile.log
drwxr-xr-x    3      root    root   4096    2023-12-25 11:07:17.535286    .
local
-rw-r--r--    1      root    root     34    2023-12-25 12:28:23.509583    b
bashrc
-rw-r--r--    1      root    root   3136    2023-12-25 12:34:10.638430    .
bashrc
-rw-------    1      root    root    140    2023-12-25 12:30:07.861434    .
bash_history
```

## Partie technique:

### 9.deld:

Cette commande supprime des répertoires vides , ne fonctionnant que lorsque le répertoire ne contient aucun fichier ni sous-répertoire.

```
root@wiame-VirtualBox:~# deld dossier1  
Le dossier dossier1 n'existe pas.
```

```
root@wiame-VirtualBox:~# deld dossier1  
Le dossier dossier1 a été supprimé du répertoire personnel de l'utilisateur root.  
root@wiame-VirtualBox:~#
```

### 10.fmask:

La commande **fmask** définit les permissions par défaut pour les fichiers et répertoires nouvellement créés par un utilisateur en soustrayant une valeur umask prédéfinie . Elle influence la sécurité en limitant les autorisations par défaut pour renforcer la protection des fichiers.

```
root@wiame-VirtualBox:~# fmask  
Le masque de création de fichiers actuel est : 22
```

### 11. galaxycat:

La commande **cat** dans les systèmes UNIX/Linux est utilisée pour afficher le contenu d'un fichier à l'écran

```
root@wiame-VirtualBox:~# cat fichier.txt  
bonjour  
bonsoir  
salut
```

## Partie technique:

### 12. upper

La commande **upper** est utilisée pour afficher les premières lignes d'un fichier texte ou les premières parties de la sortie d'une commande dans un terminal.

```
ezfkzg  
fezigjz  
f  
greg  
ge  
ge  
  
rh  
h  
h  
t  
r  
h  
rt
```

### 13.galaxyscan

La commande **galaxyscan** est utilisée pour afficher le contenu d'un fichier texte page par page dans le terminal. Elle permet de visualiser le fichier progressivement, en faisant défiler une page à la fois.

```
root@ubuntut0:/# galaxyscan mnt.txt  
ezfkzg  
fezigjz  
f  
greg  
ge  
ge  
  
rh  
h  
h  
Press Enter for more, q to quit:
```

### 14.tat

La commande **tat** est utilisée pour créer, afficher, ajouter, extraire ou manipuler des archives

```
ta@Ubuntu:/IDFS/home/root# ls
essi.py  fich1  fich2  fich3  logfile.log
or
root@Ubuntu:/IDFS/home/root# tat archive.tat fich1 fich2 fich3
root@Ubuntu:/IDFS/home/root#
```



## 15.mochdl/mochdn

Les commandes **mochdl** et **mochdn** sont utilisées pour modifier les permissions d'accès à un fichier ou un répertoire

La commande **mochdn** avec le format numérique est de spécifier explicitement les droits pour le propriétaire, le groupe et les autres utilisateurs. Chaque chiffre dans la notation octale représente une combinaison de permission

```
-rw-r--r--    1      root    root   0      2023-12-25 15:25:11.441387      wiwi.sh
```

```
root@Ubuntu:/IDFS/home/root# mochdn 555 wiwi.sh
root@Ubuntu:/IDFS/home/root# ddetail
drwxr-xr-x    3      root    root   4096    2023-12-25 15:28:21.640199      IDFS
-rwxr-xr-x    1      root    root   0      2023-12-25 15:19:14.568934      oumaima.sh
-rwxr-xr-x    1      root    root   0      2023-12-25 15:25:11.441387      wiwi.sh
```

La commande **mochdl** donne les permissions avec la notation symbolique

```
root@Ubuntu:/IDFS/home/root# ignite oumaima.sh
File created successfully: /IDFS/home/root/oumaima.sh
Metadata added to /IDFS/etc/metadata_root
root@Ubuntu:/IDFS/home/root# ddetail
-rw-r--r--    1      root    root   0      2023-12-25 15:19:14.568934      oumaima.sh
```

```
root@Ubuntu:/IDFS/home/root# mochdl +x oumaima.sh
```

```
root@Ubuntu:/IDFS/home/root# ddetail
-rwxr-xr-x    1      root    root   0      2023-12-25 15:19:14.568934      oumaima.sh
```

## 16.tr

La commande **tr** est utilisée pour rechercher des fichiers et des répertoires dans une hiérarchie de répertoires spécifiée.

## 17.trv

La commande **trv** est utilisée pour rechercher des motifs (patterns) dans des fichiers ou des flux de texte. Elle est extrêmement puissante pour filtrer et extraire des lignes qui correspondent à un motif spécifique

## 18.Galaxyown

La commande **galaxyown** est utilisée pour changer le propriétaire et/ou le groupe d'un fichier ou d'un répertoire

## 4.Gestion système :

### 1.adresse

Cette commande permet d'afficher les détails sur les interfaces réseau, y compris les adresses IP associées.

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:29:94:58 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute enp0s3
        valid_lft 74589sec preferred_lft 74589sec
    inet6 fe80::d971:4415:684d:f12f/64 scope link noprefixroute
```

### 2.ctop

La commande ctop est une commande utilisée dans les systèmes d'exploitation Unix et Linux pour afficher des informations en temps réel sur l'utilisation des ressources système et sur les processus en cours d'exécution

# Partie technique:

```
top - 01:31:42 up 14:38, 3 users, load average: 0.05, 0.07, 0.17
Tasks: 202 total, 1 running, 201 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 12.5 sy, 0.0 ni, 87.5 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 1959.7 total, 73.1 free, 1344.0 used, 542.6 buff/cache
MiB Swap: 2680.0 total, 1647.7 free, 1032.3 used, 419.1 avail Mem

      PID USER      PR  NI    VIRT    RES    SHR   S %CPU %MEM     TIME+ COMMAND
19071 root      20   0  13080  4096  3456 R  12.5  0.2  0:00.02 top
  1 root      20   0 166660  9472  6272 S  0.0  0.5  0:04.98 systemd
  2 root      20   0      0      0      0 S  0.0  0.0  0:00.01 kthreadd
  3 root      0 -20      0      0      0 I  0.0  0.0  0:00.00 rcu_gp
  4 root      0 -20      0      0      0 I  0.0  0.0  0:00.00 rcu_par+
  5 root      0 -20      0      0      0 I  0.0  0.0  0:00.00 slub_fl+
  6 root      0 -20      0      0      0 I  0.0  0.0  0:00.00 netns
  8 root      0 -20      0      0      0 I  0.0  0.0  0:00.00 kworker+
 10 root      0 -20      0      0      0 T  0.0  0.0  0:00.00 mm_perc+
```

## **3.timin**

La commande date est utilisée pour afficher ou définir la date et l'heure du système sous les systèmes d'exploitation Unix et Linux. Voici comment utiliser la commande.

```
root@youssra-VirtualBox:/# timin
2023-12-25 01:39:17
```

La sortie peut varier en fonction de la configuration locale de votre système.

## **4.eliminate**

La commande eliminate est utilisée sous les systèmes Unix et Linux pour envoyer des signaux à des processus. Elle permet de terminer ou de manipuler l'état d'un processus en cours d'exécution

## **5.clean**

la commande clean est principalement utilisée pour effacer le contenu de l'écran du terminal, fournissant ainsi une nouvelle surface de travail propre.

# Partie technique:

## 6.ginp

La commande **ginp** est utilisée pour tester la connectivité réseau entre votre ordinateur (ou tout autre appareil) et un autre appareil sur un réseau, généralement identifié par son adresse IP ou son nom d'hôte.

```
Ping result for youssra-VirtualBox:PING youssra-VirtualBox (127.0.1.1) 56(84) bytes of data.
64 bytes from youssra-VirtualBox (127.0.1.1): icmp_seq=1 ttl=64 time=0.020 ms
64 bytes from youssra-VirtualBox (127.0.1.1): icmp_seq=2 ttl=64 time=0.059 ms
64 bytes from youssra-VirtualBox (127.0.1.1): icmp_seq=3 ttl=64 time=0.035 ms
64 bytes from youssra-VirtualBox (127.0.1.1): icmp_seq=4 ttl=64 time=0.061 ms

--- youssra-VirtualBox ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3077ms
rtt min/avg/max/mdev = 0.020/0.042/0.061/0.017 ms
```

## 7.prinx

La commande **prinxy** est utilisée pour afficher du texte ou des variables dans le terminal sous les systèmes d'exploitation Unix et Linux

```
root@youssra-VirtualBox:/# prinxy "hello world"
hello world
```

## 8.sp

La commande **sp** est utile pour examiner les processus actifs sur un système, analyser leur utilisation des ressources et diagnostiquer d'éventuels problèmes liés aux processus en cours d'exécution.

## 9 voidrmf

La commande **voidrmf** est une commande puissante qui permet de supprimer récursivement des répertoires et leur contenu sous les systèmes Unix et Linux. Voici une explication plus détaillée :

```
root@youssra-VirtualBox:/IDFS/home/root# voidrmf doc1
Le repertoire doc1 et son contenu ont ete supprimes.
root@youssra-VirtualBox:/IDFS/home/root# cd /
root@youssra-VirtualBox:/# voidrmf IDFS
Le repertoire IDFS et son contenu ont ete supprimes.
root@youssra-VirtualBox:/#
```

## 3.Journalisation:

### 1. Approche de journalisation dans notre système.

Dans notre système de fichier, nous avons mis en œuvre une approche rigoureuse de la journalisation (logging) pour suivre les opérations clés telles que la création et la suppression de fichiers. En utilisant le module de journalisation de Python, nous avons configuré un fichier journal dédié situé à '/IDFS/var/log/logfile.log'. Cette stratégie de journalisation enregistre chaque opération avec un horodatage, un niveau de gravité et un message associé. Par exemple, lorsque des fichiers sont créés ou supprimés, les détails pertinents sont consignés dans le fichier journal. Cette méthode garantit une traçabilité précise des activités du système de fichier, facilitant ainsi la détection d'éventuels problèmes, l'analyse des tendances d'utilisation et le suivi des changements opérationnels au fil du temps. La configuration et l'utilisation du module de journalisation contribuent à une gestion transparente et sécurisée de notre système de fichier, renforçant ainsi la robustesse et la fiabilité de l'ensemble du système.

```
5 import json
6 import getpass
7 import logging
8 from datetime import datetime
9 import shutil
10
11 # Configuration du système de logging
12 log_file_path = '/IDFS/var/log/logfile.log' # Correction du chemin du fichier journal
13 logging.basicConfig(filename=log_file_path, level=logging.INFO,
14                     format='%(asctime)s - %(levelname)s - %(message)s')
15
16 def get_current_username():
17     try:
18         return getpass.getuser()
19     except Exception as e:
20         logging.error("Impossible d'obtenir le nom d'utilisateur : %s", e)
21         sys.exit(1)
22
23 def create_directory(directory_name):
24     try:
25         # Obtenir le nom d'utilisateur actuel
26         username = get_current_username()
27         user_home_dir = f"/IDFS/home/{username}"
28
29         # Si l'utilisateur actuel n'est pas root, vérifier les droits de l'utilisateur
30         if username != 'root':
31             if not os.getuid() == 0:
32                 logging.error("L'utilisateur doit être root pour créer le dossier")
33                 sys.exit(1)
34
35         # Créer le dossier
36         os.makedirs(user_home_dir, exist_ok=True)
37         logging.info(f"Dossier '{directory_name}' créé avec succès à l'emplacement {user_home_dir}")
38
39     except Exception as e:
40         logging.error(f"Une erreur s'est produite lors de la création du dossier : {e}")
41         sys.exit(1)
```

### Notre fichier de Logging:

```
1 2023-12-25 01:24:24,887 - INFO - Le groupe helloworld a été ajouté avec succès.
2 2023-12-25 01:25:33,227 - INFO - Le groupe ps a été ajouté avec succès.
3 2023-12-25 01:26:41,523 - WARNING - Le groupe slma n'existe pas.
4 2023-12-25 01:26:47,073 - INFO - Le groupe salma a été supprimé avec succès.
5 2023-12-25 01:27:25,608 - INFO - Le groupe salma a été ajouté avec succès.
6 2023-12-25 01:27:35,877 - INFO - Le groupe imane a été ajouté avec succès.
7 2023-12-25 01:27:42,687 - INFO - Le groupe imane a été supprimé avec succès.
8 2023-12-25 01:41:42,624 - INFO - Le groupe own a été ajouté avec succès.
9 2023-12-25 01:42:13,350 - WARNING - Le groupe salma n'existe pas.
10 2023-12-25 01:42:17,332 - INFO - Le groupe om a été supprimé avec succès.
11 2023-12-25 01:42:23,310 - WARNING - Le groupe own n'existe pas.
12 2023-12-25 01:42:38,789 - WARNING - Le groupe n'existe pas.
13 2023-12-25 01:43:02,723 - INFO - Le groupe Emile a été ajouté avec succès.
14 2023-12-25 01:45:44,336 - INFO - Le groupe own a été ajouté avec succès.
15 2023-12-25 02:11:52,811 - INFO - Le groupe own a été supprimé avec succès.
16 2023-12-25 02:13:03,923 - WARNING - Le groupe own n'existe pas.
17 2023-12-25 02:45:01,524 - INFO - b'File created successfully: /IDFS/home/root/file'
```

## 4- Les métadonnées:

On a développé une fonction appelée **md** , qui prend en argument le chemin d'un fichier. Elle est conçue pour extraire diverses métadonnées du fichier, comme le temps de création, le temps de dernière modification, la taille, les permissions, etc . Les données extraites sont ensuite enregistrées dans un fichier texte nommé **metadata** . On appelle cette fonction à chaque fois qu'un fichier est créé ou modifié, ou lorsqu'on modifie les données du fichier texte **metadata** .

```
2  {
3      "File Name": "/IDFS/home/root/fichier1",
4      "Creation Date": "2023-12-25 14:40:14",
5      "Modification Date": "2023-12-25 14:40:14",
6      "File Size": 0,
7      "Permissions": "644"
8  }
9 1
```

## 5-Le code de la fonction :

```
6
7 def get_current_username():
8     return os.getlogin()
9
10 def lire_metadonnees(nom_fichier):
11     username = get_current_username()
12     # Si l'utilisateur est root, utilisez "/IDFS/etc/metadata_root" au lieu de "/IDFS/etc/
 metadata_{username}"
13     chemin_metadata = f'/IDFS/etc/metadata_root' if username == 'root' else f'/IDFS/etc/
 metadata_{username}'
14
15     if os.path.exists(chemin_metadata):
16         with open(chemin_metadata, 'r') as fichier_metadata:
17             metadonnees_globales = json.load(fichier_metadata)
18             # Chercher les meadonnees specifiques au fichier
19             for metadonnees in metadonnees_globales:
20                 if metadonnees['Nom du fichier'] == nom_fichier:
21                     return metadonnees
22             # Si les mÃ©tadonnÃ©es pour le fichier spÃ©cifiÃ© ne sont pas trouvÃ©es
23             return None
24     else:
25         return None
26
27 # VÃ©rification de la presence du nom de fichier en ligne de commande
28 if len(sys.argv) != 2:
29     print("Utilisation: python script.py <nom_fichier>")
30     sys.exit(1)
31
```

## **II. DISTRIBUTION LINUX PERSONNALISE**

### **1- Personnalisation Optimisée : Création d'une Distribution Linux personnalisée**

La personnalisation d'une distribution Linux repose sur des choix stratégiques clés, tels que la sélection du noyau, du gestionnaire de fenêtres, et des paquets logiciels. Des outils d'optimisation des performances, comme la gestion de l'énergie, sont intégrés pour une utilisation efficace des ressources. L'aspect visuel est personnalisé avec des thèmes et des icônes, tandis que la configuration du terminal offre une flexibilité supplémentaire. L'ensemble de ces éléments permet de créer une distribution Linux sur mesure, adaptée aux besoins spécifiques de chaque utilisateur.

#### **1- Cubic : Personnalisation Poussée de Distributions Linux avec Simplicité"**

**Cubic**, un outil puissant de personnalisation de distributions Linux, offre une approche conviviale pour créer des systèmes sur mesure. Son utilisation est motivée par la n

##### **pourquoi utiliser Cubic :**

**Cubic** est privilégié pour sa facilité d'utilisation et son interface graphique intuitive. Il permet aux utilisateurs de personnaliser divers aspects de leur distribution Linux sans nécessiter une expertise technique approfondie. Cubic simplifie la création de distributions sur mesure, offrant un moyen accessible de façonner un système adapté à des besoins particuliers.

##### **Points Forts de Cubic :**

- 1.Interface Graphique Intuitive : Cubic propose une interface graphique conviviale, réduisant la complexité du processus de personnalisation.
- 2.Gestion Simplifiée du Noyau : Il offre un contrôle aisné sur le choix et la configuration du noyau, permettant une personnalisation précise des performances du système.
- 3.Personnalisation des Paquets : Cubic facilite la sélection et la personnalisation des paquets logiciels, évitant ainsi l'encombrement inutile du système.

##### **Les commandes d'installation:**

L'installation de Cubic est simplifiée et a été réalisée en quelques étapes :

## Partie technique:

on a tapé sur les commandes suivante:

```
$sudo add-apt-repository ppa:cubic-wizard/stable
```

```
[sudo] password for dikra:
[sudo] password for dikra:
repository: 'deb https://ppa.launchpadcontent.net/cubic-wizard/release/ubuntu/ j
immy main'
description:
PPA for Cubic release branch

      //////
     // \ \
    // \  \_ \
   \ \_ \_ \
  \ \_ \_ \
  \ \_ \_ \
Custom Ubuntu ISO Creator

cubic (Custom Ubuntu ISO Creator) is a GUI wizard to create a customized Ubuntu
or Debian Live ISO image.

Website:  https://github.com/PJ-Singh-001/Cubic/wiki
More info: https://launchpad.net/~cubic-wizard/+archive/ubuntu/release
Adding repository.
Press [ENTER] to continue or Ctrl-c to cancel.
found existing deb entry in /etc/apt/sources.list.d/cubic-wizard-ubuntu-release-
```

**\$sudo apt-get update**

```
$sudo apt-get install cubic
```

```
dikra@dikra-VirtualBox:~$ sudo apt-get update
Get:1 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Hit:2 http://ma.archive.ubuntu.com/ubuntu jammy InRelease
Get:3 http://ma.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Hit:4 https://dl.google.com/linux/chrome/deb stable InRelease
Hit:5 https://ppa.launchpadcontent.net/cubic-wizard/release/ubuntu jammy InRelease
Hit:6 https://ppa.launchpadcontent.net/danielrichter2007/grub-customizer/ubuntu
jammy InRelease
Hit:7 http://ma.archive.ubuntu.com/ubuntu jammy-backports InRelease
Fetched 229 kB in 3s (76.3 kB/s)
Reading package lists... Done
dikra@dikra-VirtualBox:~$ sudo apt-get install cubic
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages will be upgraded:
  cubic
1 upgraded, 0 newly installed, 0 to remove and 20 not upgraded.
Need to get 201 kB of archives.
After this operation, 0 B of additional disk space will be used.
```

Après avoir saisi la commande "cubic", nous serons redirigé vers la page d'accueil de Cubic. Une fois sur cette interface conviviale,

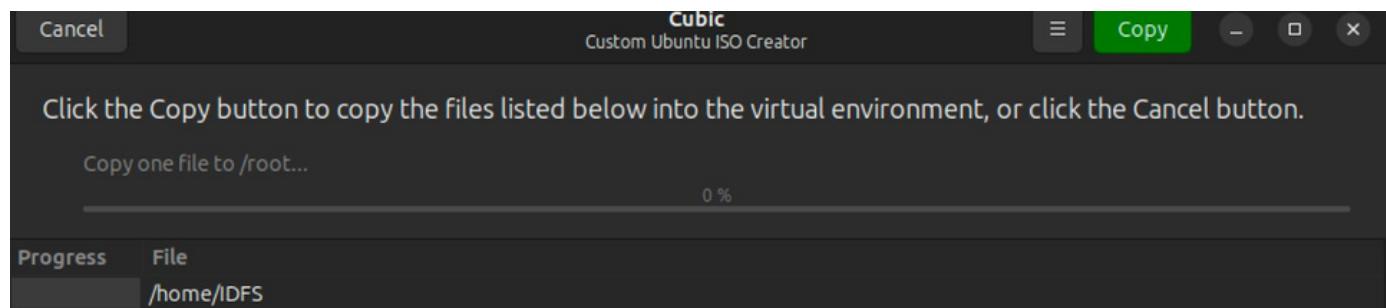


la prochaine étape consiste à créer un répertoire dédié pour déposer notre image ISO. Ce répertoire servira de base pour la personnalisation de notre distribution Linux. En organisant notre espace de travail, nous pourrez facilement naviguer dans les fichiers et configurations nécessaires à la création de notre distribution DIOWIZ



## 2- Préparation Essentielle : Personnalisation Avant l'Environnement Chroot

Absolument, avant d'entamer l'environnement de chroot, la phase de personnalisation exige une préparation minutieuse des fichiers extraits de l'image ISO d'Ubuntu. Cette étape initiale revêt une importance capitale pour assurer une configuration optimale de l'environnement chroot. En effectuant ce prétraitement, on s'assure que les éléments essentiels, tels que le noyau, les fichiers de démarrage, les configurations système, et d'autres composants clés, sont prêts à être intégrés dans le nouvel environnement. Cette approche soigneuse garantit une transition fluide vers l'étape de chroot, où les ajustements et personnalisations spécifiques à la distribution seront effectués.



Une fois la phase de prétraitement des fichiers extraits de l'image ISO d'Ubuntu achevée, l'étape suivante consiste à mettre à jour l'environnement chroot à l'aide des commandes **apt update** et **apt upgrade**.

```
root@DIOWYZ:~# apt update  
root@DIOWYZ:~# apt upgrade
```

Cette étape est cruciale pour garantir que le système dans l'environnement chroot dispose des dernières mises à jour de sécurité et des versions les plus récentes des logiciels installés. En exécutant ces commandes, on s'assure que la distribution personnalisée est à jour, stable et prête à intégrer les modifications spécifiques de l'utilisateur.

## 2. Personnalisation de la Distribution :

### Étape 1 : Installation de l'environnement Xubuntu

Après l'installation de base de la distribution, le processus de personnalisation commence par l'intégration de l'environnement de bureau Xubuntu. Cette étape s'effectue en exécutant la commande suivante;

```
sudo apt-get install xubuntu-desktop
```

```
root@DIOWYZ:/# apt-get install xubuntu-desktop
```

### Pourquoi XUBUNTU?

**Xubuntu** a été choisi pour ses caractéristiques légères et son interface utilisateur épurée, offrant ainsi une expérience similaire à celle de Windows, tout en optimisant les performances.

### Étape 2 : Installation des thèmes

La personnalisation se poursuit avec l'installation de thèmes spécifiques pour améliorer l'esthétique de l'interface. Les thèmes choisis sont :

```
sudo apt-get install arc-theme papirus-icon-theme breeze-cursor-theme
```

```
root@DIOWYZ:/# apt-get install arc-theme papirus-icon-theme breeze-cursor-theme
```

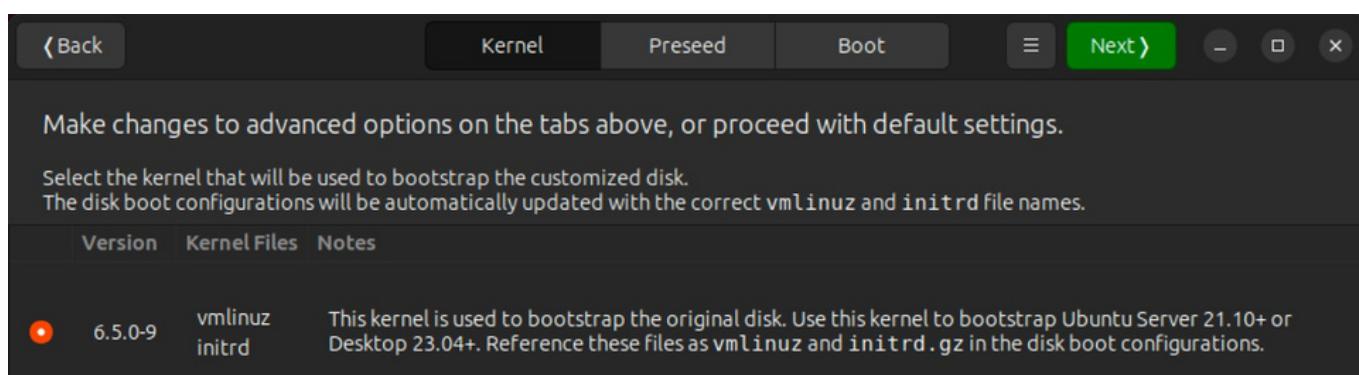
- **Arc Theme** : Choisi pour son design moderne et épuré, Arc offre une apparence visuellement attrayante et unifiée à l'ensemble du système.
- **Papirus Icon Theme** : Sélectionné pour ses icônes claires et cohérentes, Papirus complète l'esthétique globale en offrant une bibliothèque d'icônes bien conçue.
- **Breeze Cursor Theme** : Ce thème de curseur a été privilégié pour sa simplicité et son élégance, ajoutant une touche de fluidité à l'expérience utilisateur.

## Étape 3 : Personnalisation Avancée et Génération de l'Image ISO

La personnalisation de la distribution se poursuit avec des étapes avancées, permettant un contrôle précis sur les composants clés du système. Ces étapes comprennent la sélection du noyau (kernel) et le choix du type d'archivage (gzip).

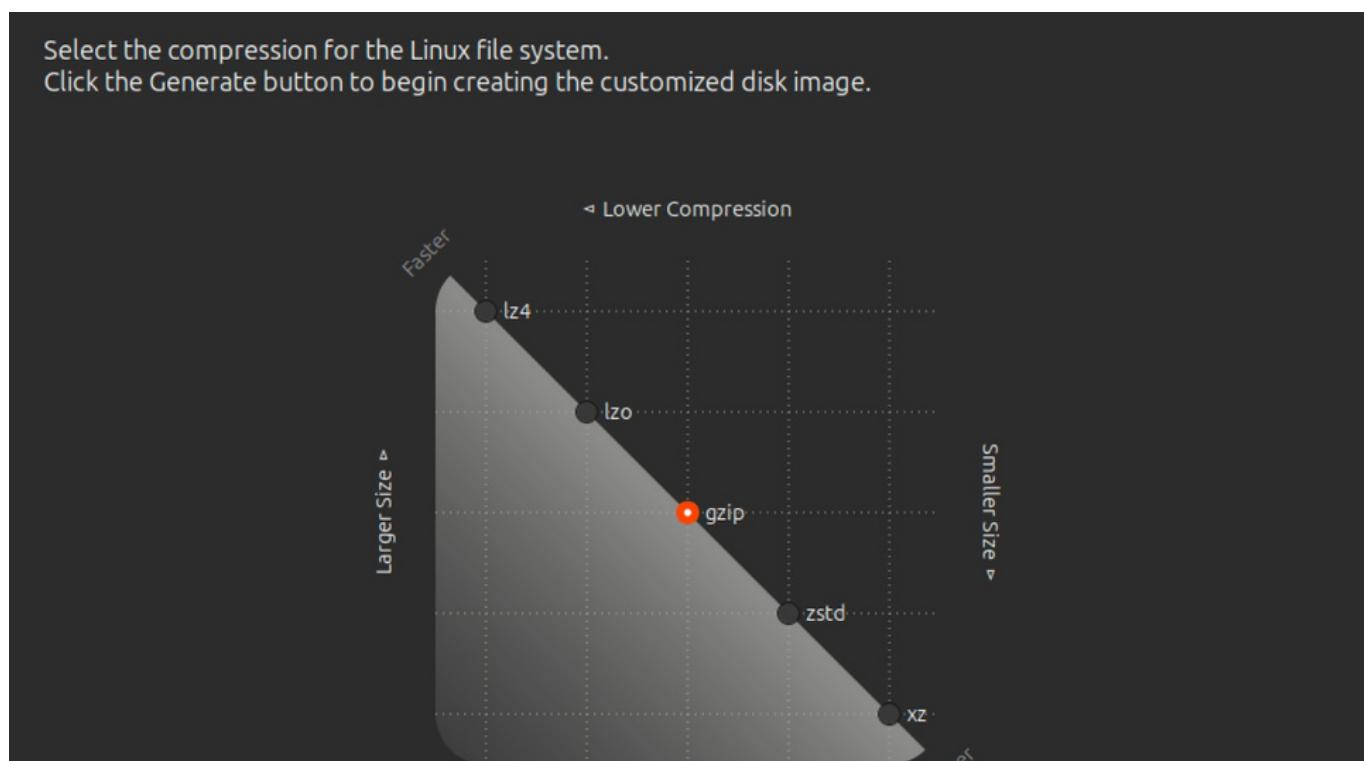
### Choix du Noyau

La possibilité de sélectionner le noyau offre une flexibilité essentielle, permettant d'adapter la distribution aux besoins spécifiques de l'utilisateur. Le noyau choisi est important pour la stabilité et la performance du système.



### Type d'Archivage (gzip)

Le type d'archivage **gzip** a été retenu pour optimiser l'espace de stockage tout en garantissant une compression efficace. Cette décision vise à maintenir l'équilibre entre la taille de l'image ISO générée et la performance du système lors de l'installation.



## Accès au Répertoire de Stockage

Après avoir finalisé les choix de personnalisation, l'accès au répertoire désigné pour stocker l'image ISO est essentiel. Cela permet de garantir que l'image générée est sauvegardée dans l'emplacement prévu, facilitant ainsi sa récupération ultérieure.

```
ziko2003@ziko:~$ cd /home/ziko2003/imiso/
```

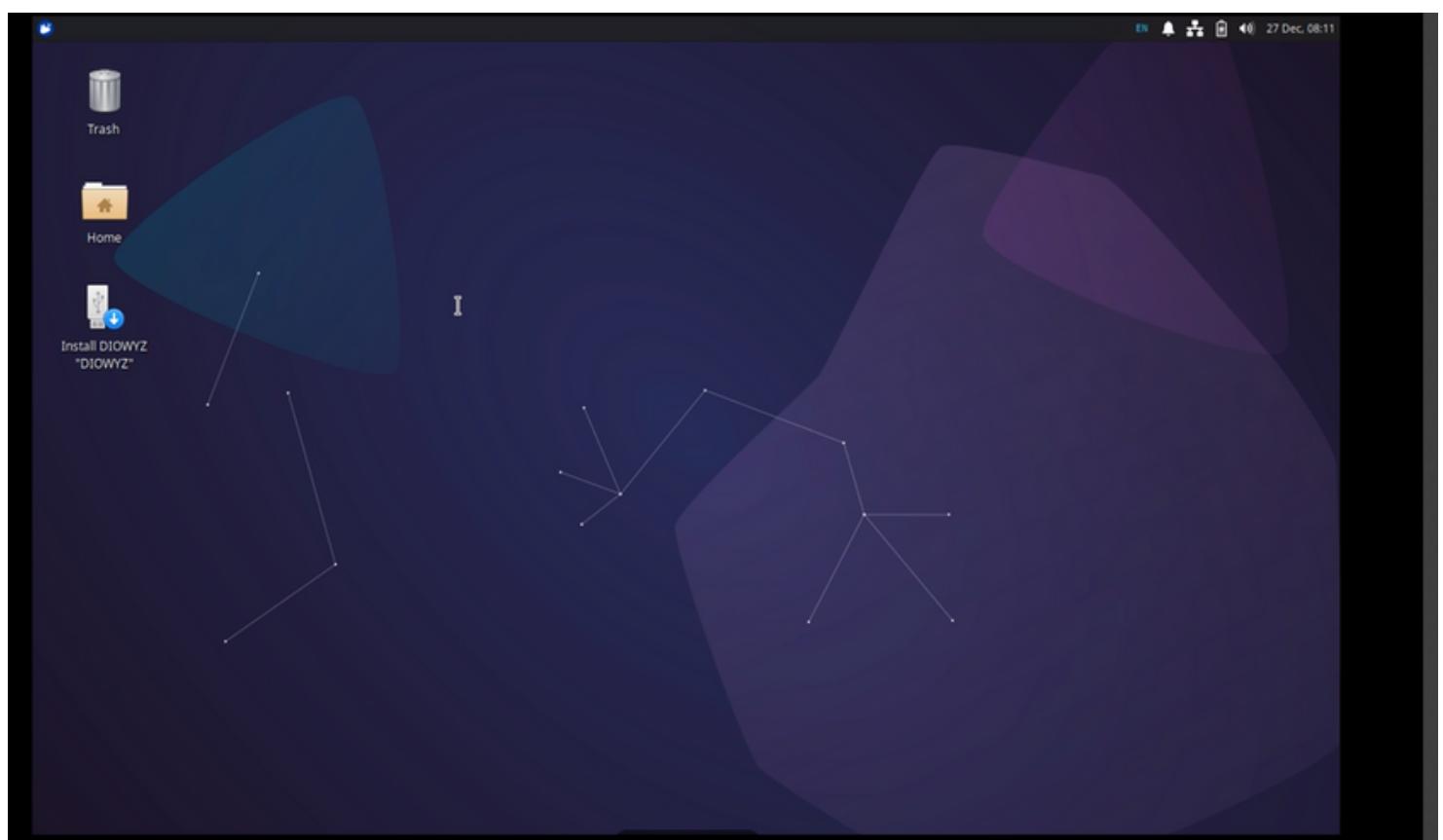
## L'Image ISO Finale

Une fois toutes les étapes de personnalisation accomplies, l'image ISO finale est disponible dans le répertoire déterminé. Cette image reflète la distribution Linux entièrement personnalisée, prête à être déployée sur des systèmes compatibles.

```
cubic.conf  custom-root  DIOWXYZ.md5      partition-2.img  
custom-disk  DIOWXYZ.iso  partition-1.img
```

## Partie technique:

**Consultez les captures d'image ci-dessous pour visualiser les améliorations apportées à l'interface après la personnalisation.**



```
Terminal - diowyz@diowyz: ~
File Edit View Terminal Tabs Help

.0000000000. 00000 .000000. 000000 000000 0000 000000000000
.888. `Y8b `888` d8P` `Y8b `888. `888. .8` `888. .8` d`d888`
888 888 888 888 888. `888. .8888. .8` `888. .8` `888P
888 888 888 888 888 `888 .8` `888. .8` `888.8` d888
888 888 888 888 888 `888.8` `888.8` `888.8` `888P
888 d88` 888 `88b d88` `888` `888` `888 d888
o888bood8P` o888o `Y8bood8P` .8` .8` o888o .8888888888P

Input #0, wav, from '/tmp/tmpby6s2ru3.wav': 0KB sq= 0B f=0/0
Duration: 00:00:02.05, bitrate: 768 kb/s
Stream #0:0: Audio: pcm_s16le ([1][0][0][0] / 0x0001), 48000 Hz, 1 channels, s16, 768 kb/s
 1.95 M-A: -0.000 fd= 0 aq= 0KB vq= 0KB sq= 0B f=0/0
diowyz@diowyz:~$ timin
```

### 3.Un Benchmark avec le Ubuntu standard :

Nous réalisons actuellement un benchmark entre le système de fichiers IDFS de notre distribution personnalisée et celui d'Ubuntu pour évaluer et comparer leurs performances respectives .

- **Sécurité :**

IDFS :Moyenne .

- **Gestion d'utilisateurs :**

IDFS : Excellent.

- **Traitement des Big Data :**

IDFS : Faible.

- **Les permissions :**

IDFS : Excellent.

- **Accès aux métadonnées :**

IDFS : Excellent.

## Partie technique:

### 4. Conclusion:

En résumé, la création d'une distribution Linux personnalisée et d'un système de fichiers sur mesure a été un processus essentiel pour atteindre nos objectifs spécifiques. Cette approche nous a permis de façonner un environnement informatique optimisé, répondant précisément à nos besoins, et renforçant ainsi notre compréhension des systèmes d'exploitation.

La personnalisation de la distribution Linux a offert une flexibilité accrue, améliorant les performances, la sécurité et l'efficacité globale du système. Parallèlement, la conception d'un système de fichiers adapté a contribué à une gestion optimale des données et des ressources, renforçant la stabilité et les performances du système.

En conclusion, ce projet a enrichi notre expertise technique et nous a dotés d'outils personnalisés adaptés à nos besoins spécifiques. Cette expérience servira de base solide pour des projets futurs et souligne l'importance de la personnalisation dans la création de solutions informatiques efficaces et sur mesure.