# Table des matières

[Table des matières 1](#_Toc60923592)

[Evolution 2](#_Toc60923593)

[Le file system Ext2 2](#_Toc60923594)

[Le file system Ext3 2](#_Toc60923595)

[Le file system Ext4 2](#_Toc60923596)

[Le file system Xfs 2](#_Toc60923597)

[Le file system Zfs et OpenZfs 2](#_Toc60923598)

[Le FS Ex2 2](#_Toc60923599)

[Structure 2](#_Toc60923600)

[Calcul de capacité d’adressage 3](#_Toc60923601)

[Calcul de nombre de blocs 3](#_Toc60923602)

[Structure d’un répertoire 4](#_Toc60923603)

[Processus de recherche 4](#_Toc60923604)

[Sync 5](#_Toc60923605)

[Le FS Ext 3 5](#_Toc60923606)

[Le FS Ext 4 6](#_Toc60923607)

[Compatibilités 6](#_Toc60923608)

[Migration 6](#_Toc60923609)

[Montage 6](#_Toc60923610)

[Manipulation 7](#_Toc60923611)

[Le FS XFS 7](#_Toc60923612)

[Le FS Zfs vs OpenZfs 7](#_Toc60923613)

[Comparatif 8](#_Toc60923614)

[Sauvegarde 9](#_Toc60923615)

[Incrémentielle 9](#_Toc60923616)

[Différentielle 9](#_Toc60923617)

[Avantages Inconvénient 9](#_Toc60923618)

# Evolution

## Le file system Ext2

C'est l'ancien standard Linux.

Il est basé sur la structure SYSTEM V d'un file system Unix.

## Le file system Ext3

C'est une évolution de l'Ext2.

La gestion d'un journal a été ajoutée.

## Le file system Ext4

C'est l'actuel standard des systèmes Linux (RHEL6).

C'est un système de transition.

## Le file system Xfs

C'est l'actuel standard des systèmes Linux (RHEL7).

Il offre plus de fonctionnalités que l'Ext4.

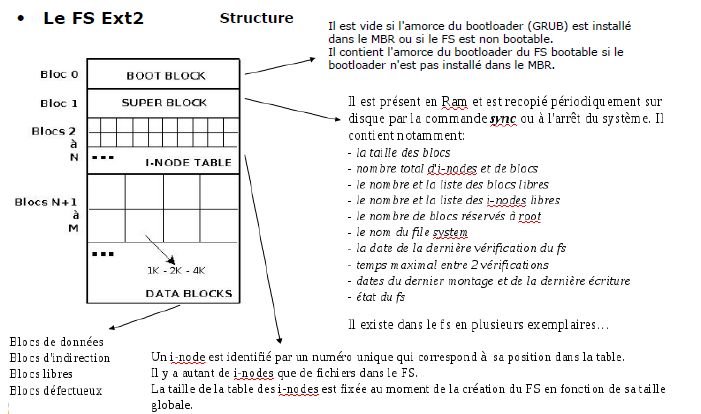
## Le file system Zfs et OpenZfs

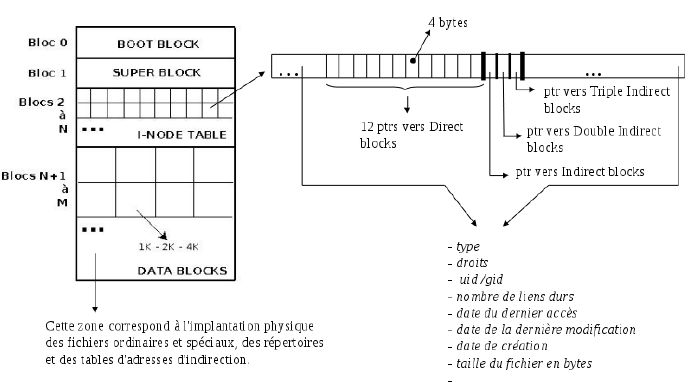
ZFS (fs propriétaire pour l'OS Solaris d'Oracle)

OpenZfs: Alternative OpenSource (Linux)

# Le FS Ex2

## Structure





## Calcul de capacité d’adressage

Calcul pour des blocs de 1Ko

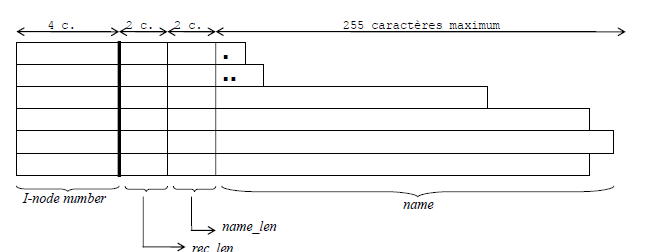
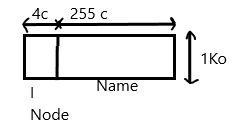
1. Nombre de blocs de données :  
    12 \* 1Ko = 12 Ko
2. Indirect blocks :  
    12 Ko + 254 \* 1 Ko = 268 Ko
3. Double IB :  
    256²
4. Direct blocks :   
    268Ko + DIBlocks \* 1Ko = 65536 Ko -> 65804Ko
5. Triple Indirect Blocs :  
    2563 \* 1 Ko + 65804Ko = 16843020 Ko = 16 Go

## Calcul de nombre de blocs

On demande de calculer le nombre de blocs total (données + indirection) monopolisés par un fichier dont la taille réelle s'élève à 10.456.373 caractères. (Block de 1Ko)

1. Nombre de blocs de données :  
    10.456.373 / 1024 = 10 212 blocs
2. Direct blocks : 12 blocs qui sont référencés  
    Reste 10 200 blocs a référencés
3. Indirect Blocks : (+1) (256 blocs)  
    Reste = 10200 – 256 = 9944 blocs
4. Double IB :   
    Nb de bloc de 1er niveau : +1  
    Nb de bloc de 2eme niveau : 9944 / 256 = +39
5. Total :  
    1 + 3 + 4 (1) + 4(2)  
    10212 + 1 + 1 +39 = 10253 blocs

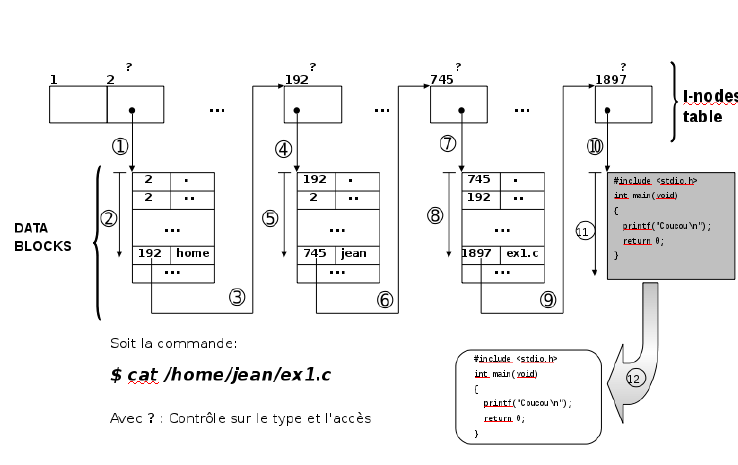
## Structure d’un répertoire



Remarques :  
Le i-node n°2 références toujours le répertoire racine.  
Le i-node n°1 référence les blocs défectueux.

Un dossier aura la même structure d’inode qu’un fichier.  
Name ne peut dépasser 255 caractères car en C c’est impossible  
  
4C : ne peut dépasser en hexa : FF | FF | FF | FF (16) = ~ 4 milliards (10)

## Processus de recherche



- Il va pointer l’inode 2 en vue de rechercher une entrée home

- Il prend l’inode de home (192)

- Il pointe vers l’inode 192, regarde si c’est bien un dossier et si on a la permission d’exécuter le dossier.

- Il cherche et trouve jean. Il pointe l’inode 745

Etc.

Une fois le fichier trouver, il regarde si on a les permissions, si oui il se met sur le 1er bloc du fichier et le lit

## Sync

La table des i-nodes et le super-bloc sont recopiés en Ram lors du montage du fs et sont recopiés périodiquement (~1x/10 sec.) sur le support via la commande **sync** (ou lors de la fermeture du système).

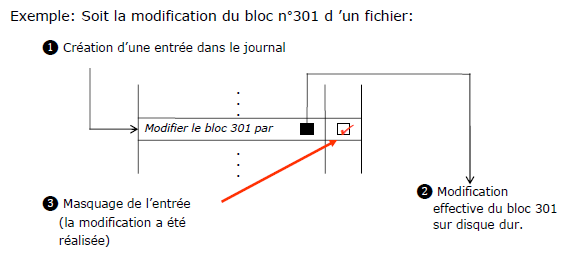
Taille d'un bloc par défaut sur disque dur : 4 Ko

Il est non-journalisé

☹ Risque de perte d’intégrité si coupure de courant ou extinction à chaud. fsck au rebootage (très lent si taille importante)

# Le FS Ext 3

* Pour remédier aux faiblesses du fs Ext2.
* Principe : fs non-journalisé Ext2 + gestion d’un fichier journal



☹Ecriture en double des données de modification

* Une fois dans le journal
* Une fois dans le bloc correspondant

… mais le temps de mise à jour est minimisé en utilisant le temps libre du système pour parcourir le journal et ainsi procéder aux modifications.

😊Si rupture d’alimentation :

* Pendant l’écriture dans le journal :

Seule l’entrée dans le journal est incorrecte, le fichier reste intact.

Il n’est simplement pas mis à jour et risque donc pas d’être perdu.

* Pendant la mise à jour du fichier :

Au rebootage, le système repère les entrées journaux non réalisés et réalise alors les mises à jour qui s’imposent. Il n’y a donc plus de fsck global à réaliser ...

# Le FS Ext 4

Ext 4 = ext 2 +journal + extent

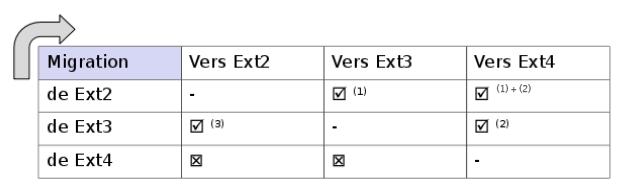
En remplacement des 'indirects blocks', il prend en charge la notion de zone étendue (extent).

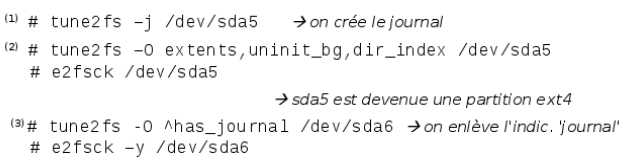
Un 'extent' est une zone du disque contigüe éventuellement de très grande capacité (Go).

L'utilisation d'extent réduit la fragmentation et améliore les performances lors de l'utilisation de gros fichiers.

# Compatibilités

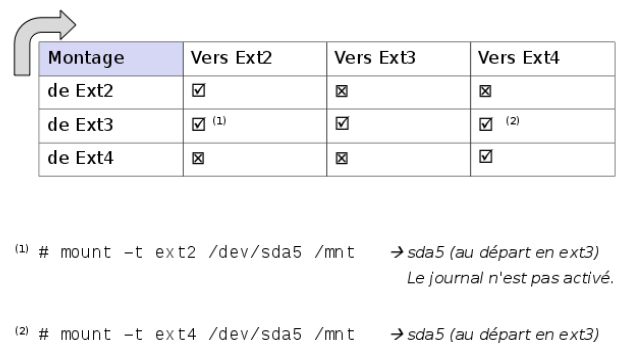
## Migration





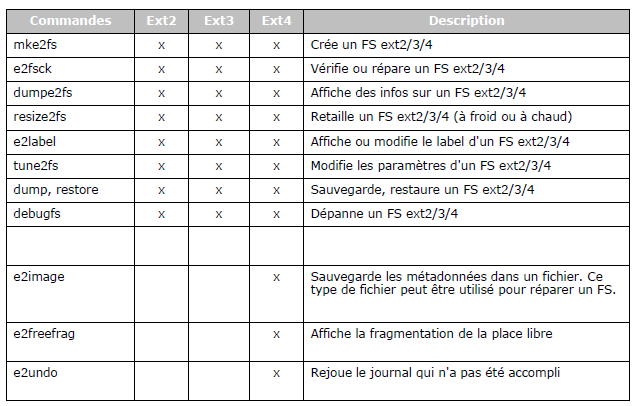
-j = journal  
^has\_journal = not has journal

## Montage



-t = type

# Manipulation



# Le FS XFS

Xfs est un système de fichiers développé par Silicon graphic, Inc.

Comme Ext4, il est également basé sur les "extents" mais apporte des fonctionnalités supplémentaires.

**Fonctionnalités supplémentaires**

* Journalisation de métadonnées récupération après incident plus rapide
* Défragmentation et élargissement possibles alors qu'il est monté et actif.
* Meilleures performances lors des E/S.

## Le FS Zfs vs OpenZfs

**Zfs:**

* File system conçu par SunMicrosystem et embarqué sur ses OS Solaris.
* SunMicrosystem est racheté par Oracle en 2009.
* Décision d'Oracle de réduire les dépenses en R&D pour Solaris
  + Menace sur la pérennité de ZFS
  + Tentative de transformer la licence CDDL en licence plus ouverte (ex. GPL)  
     … pour pouvoir l'intégrer à la distribution Oracle Linux et peut-être à la RHEL qui a abandonné le Fs BTRFS (concurrent libre du Zfs).

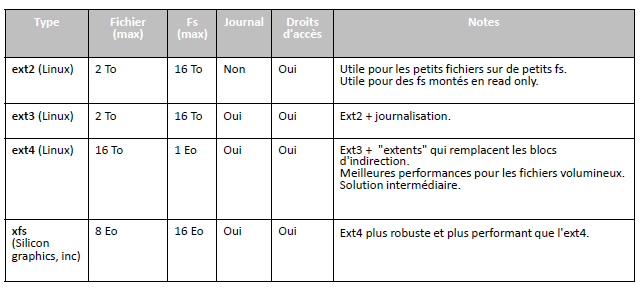
**OpenZfs:**

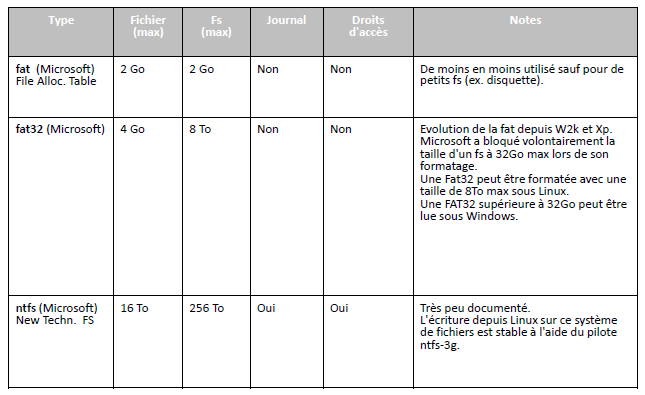
Projet visant à continuer le développement de ZFS d'une manière réellement open source.

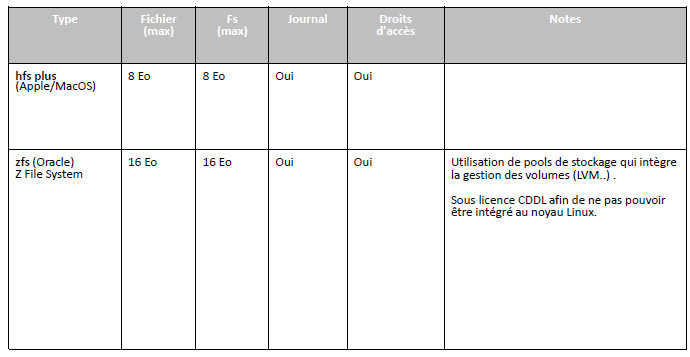
Fonctionnalités principales

* Très haute capacité de stockage et très léger.
* Meilleure gestion de l'espace occupé par les petits fichiers.
* Possibilité de créer des snapshots et des sous-volumes.
* Intégrité des données plus robuste via un mécanisme de checksum de chaque bloc.
* Compression possible et à la volée des données lors de leur écriture.
* Sauvegarde incrémentale intégrée au système de fichiers.
* Défragmentation à chaud.
* Utilisation de pools de stockage facilitant grandement la gestion des systèmes de fichiers …

# Comparatif







# Sauvegarde

## Incrémentielle

Lundi : Full Mardi : Incrément 1 Mercredi Incrément2 Jeudi : crash donc Restore : Full + I1 +I2

Incrément 1 = différence entre mardi et lundi

Incrément 2 = différence entre mercredi et mardi

## Différentielle

Lundi = Full Mardi = D1 Mercredi = D2 Jeudi crash donc : Restore FULL + D2

D1 = différence entre mardi et lundi

D2 = différence entre mercredi et lundi

## Avantages / Inconvénient

|  |  |  |
| --- | --- | --- |
|  | + | - |
| Incrémentielle | * Taille d’incrément petit * Temps de fabrication bas | * Restore inefficace si perte d’un I * Temps de restore haut |
| Différentielle | * Temps de restore bas | * Taille de backup volumineux * Temps de fabrication de D haut |