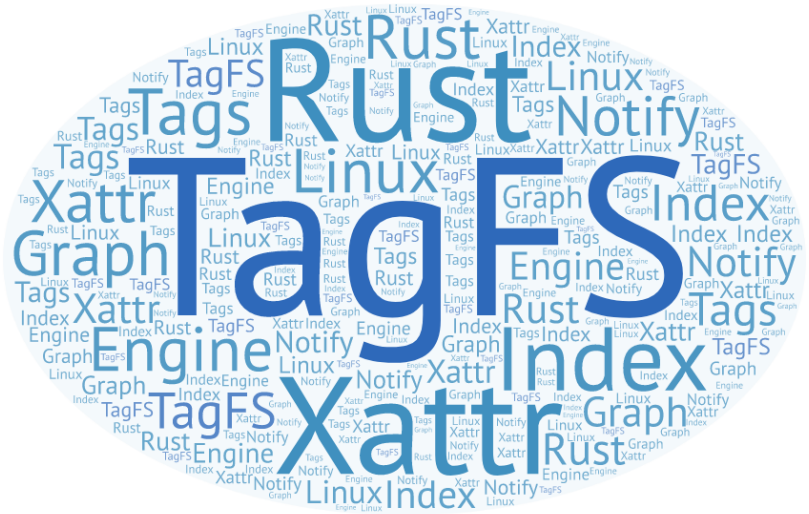
TagFS - Système d'étiquetage des fichiers avec Rust

Steven Liatti

Projet de bachelor - Prof. Florent Glück - Hepia ITI 3ème année

4 septembre 2018



h e p i a
Haute école du paysage, d'ingénierie
et d'architecture de Genève

Hes·SO GENÈVE
Haute Ecole Spécialisée
de Suisse occidentale

Plan

1. Introduction
2. Solutions existantes
3. Architecture
4. Technologies
5. Réalisation
6. Discussion
7. Conclusion

Problématique

- Nombre de fichiers énorme.

Problématique

- Nombre de fichiers énorme.
- Difficulté à retrouver des fichiers.

Problématique

- Nombre de fichiers énorme.
- Difficulté à retrouver des fichiers.
- Plusieurs emplacements logiques pour un seul fichier.

Problématique

- Nombre de fichiers énorme.
- Difficulté à retrouver des fichiers.
- Plusieurs emplacements logiques pour un seul fichier.

Système de tags de fichiers et répertoires avec possibilité de recherche par tags.

Objectifs

- Étudier et s'approprier le langage Rust.

Objectifs

- Étudier et s'approprier le langage Rust.
- Répertorier les applications existantes permettant d'étiqueter les fichiers.

Objectifs

- Étudier et s'approprier le langage Rust.
- Répertorier les applications existantes permettant d'étiqueter les fichiers.
- Étudier les XATTR lors des manipulation courantes sur les fichiers.

Objectifs

- Étudier et s'approprier le langage Rust.
- Répertorier les applications existantes permettant d'étiqueter les fichiers.
- Étudier les XATTR lors des manipulation courantes sur les fichiers.
- Explorer les méthodes de surveillance du système de fichiers.

Objectifs

- Étudier et s'approprier le langage Rust.
- Répertorier les applications existantes permettant d'étiqueter les fichiers.
- Étudier les XATTR lors des manipulation courantes sur les fichiers.
- Explorer les méthodes de surveillance du système de fichiers.
- Analyser les moyens d'indexer une arborescence de fichiers.

Objectifs

- Étudier et s'approprier le langage Rust.
- Répertorier les applications existantes permettant d'étiqueter les fichiers.
- Étudier les XATTR lors des manipulation courantes sur les fichiers.
- Explorer les méthodes de surveillance du système de fichiers.
- Analyser les moyens d'indexer une arborescence de fichiers.
- Concevoir et implémenter un système performant.

Objectifs

- Étudier et s'approprier le langage Rust.
- Répertorier les applications existantes permettant d'étiqueter les fichiers.
- Étudier les XATTR lors des manipulation courantes sur les fichiers.
- Explorer les méthodes de surveillance du système de fichiers.
- Analyser les moyens d'indexer une arborescence de fichiers.
- Concevoir et implémenter un système performant.
- Mesurer les performances de ce système.

TMSU et Tagsistant

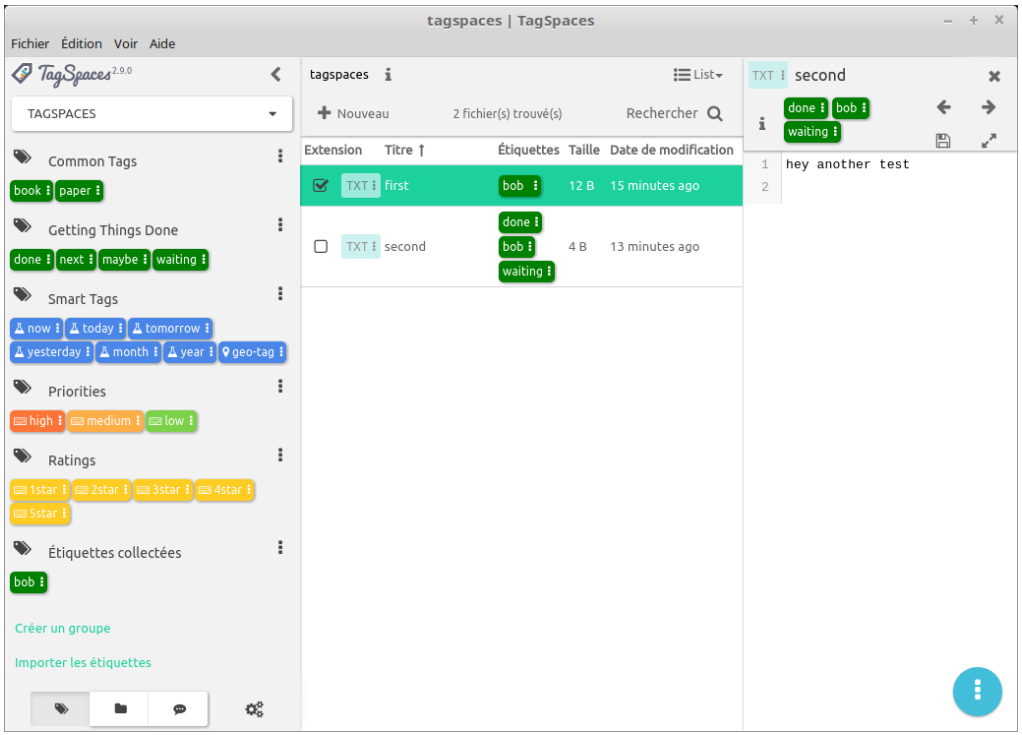
- CLI.
- Gestion des tags.
- Liste des fichiers associés à des tags.
- Usage de commandes usuelles (`cp`, `ls`, `mkdir`) pour manipuler les tags (Tagsistant).

TMSU et Tagsistant

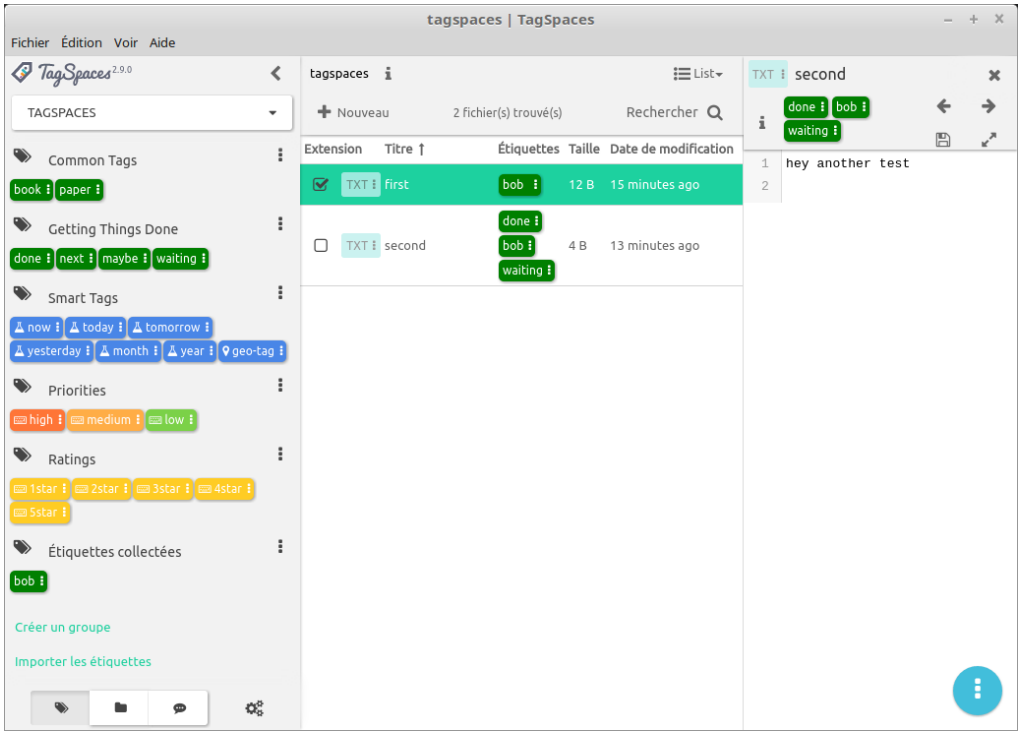
- CLI.
- Gestion des tags.
- Liste des fichiers associés à des tags.
- Usage de commandes usuelles (`cp`, `ls`, `mkdir`) pour manipuler les tags (Tagsistant).

Points positifs	**Points négatifs**
Simple (CLI)	Dépendance à FUSE
Rapide et efficace	Dépendance à une BDD externe
Open source	Modification et accès uniquement par l'application (Tagsistant)

TagSpaces

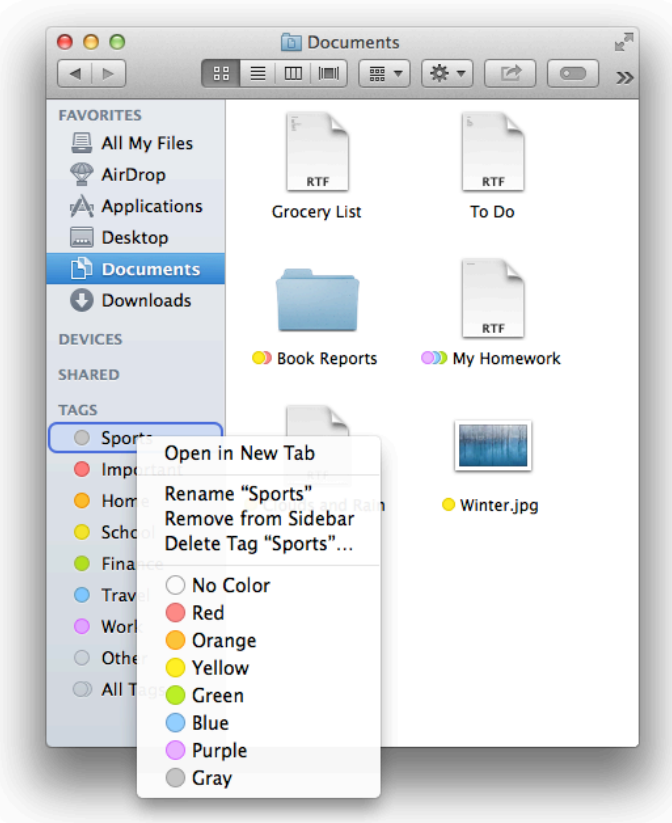

tagspaces | TagSpaces
Fichier Édition Voir Aide
TagSpaces 2.9.0
TAGSPACES
Common Tags
book paper
Getting Things Done
done next maybe waiting
Smart Tags
now today tomorrow
yesterday month year geo-tag
Priorities
high medium low
Ratings
1star 2star 3star 4star 5star
Étiquettes collectées
bob
Créer un groupe
Importer les étiquettes
tagspaces
List
Nouveau
2 fichier(s) trouvé(s)
Rechercher
Extension Titre Étiquettes Taille Date de modification
TXT first bob 12 B 15 minutes ago
TXT second done bob waiting 4 B 13 minutes ago
TXT second
done bob waiting
1 hey another test
2

TagSpaces

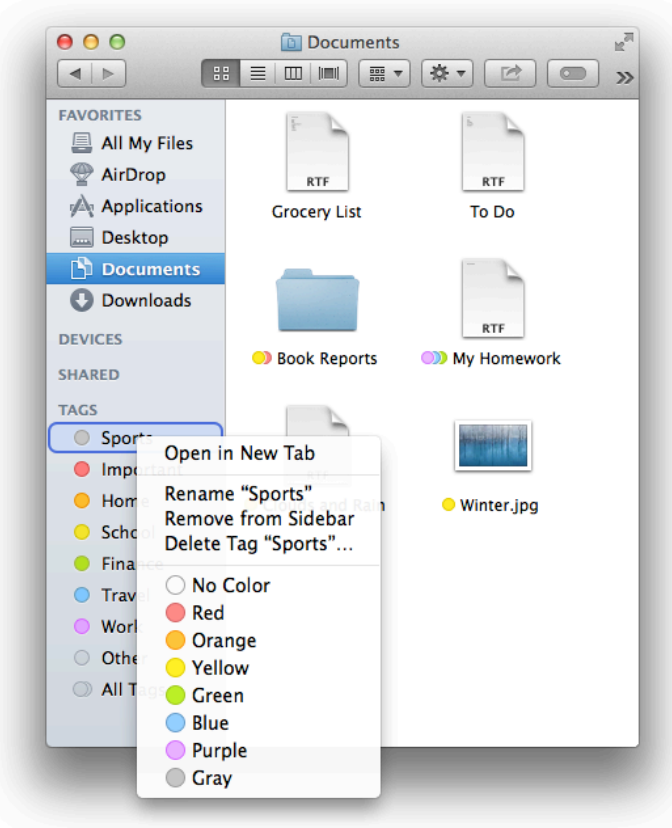


Points positifs	Points négatifs
Rapide et efficace	Dépendance à une BDD externe
Open source	Modification et accès uniquement par l'application
GUI	

macOS


Documents
FAVORITES
All My Files
AirDrop
Applications
Desktop
Documents
Downloads
DEVICES
SHARED
TAGS
RTF
Grocery List
RTF
To Do
Book Reports
RTF
My Homework
Winter.jpg
Open in New Tab
Rename "Sports"
Remove from Sidebar
Delete Tag "Sports"...
No Color
Red
Orange
Yellow
Green
Blue
Purple
Gray

macOS

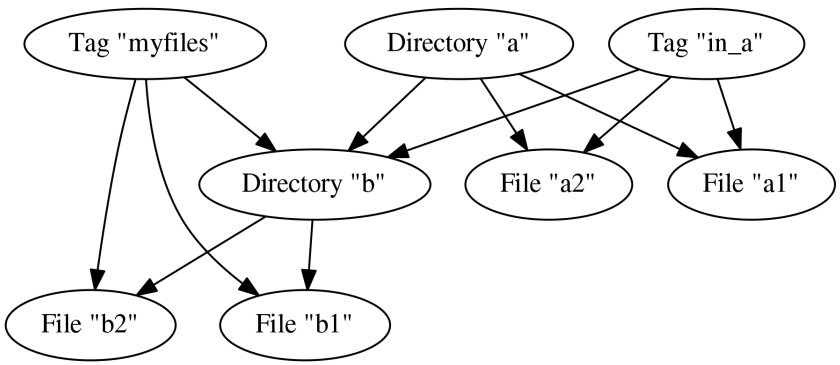


Points positifs	Points négatifs
Système de tags intégré à l'explorateur de fichiers	Pas open source
Stocke les tags dans les attributs des fichiers	Seulement pour macOS
Performant	Non écrit en Rust

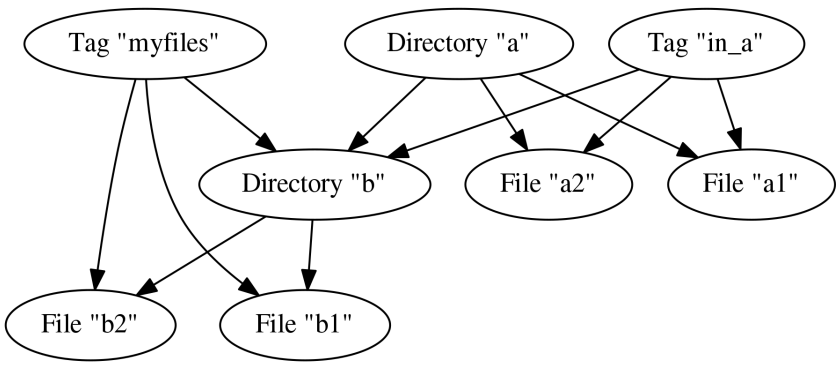
Gestion des tags

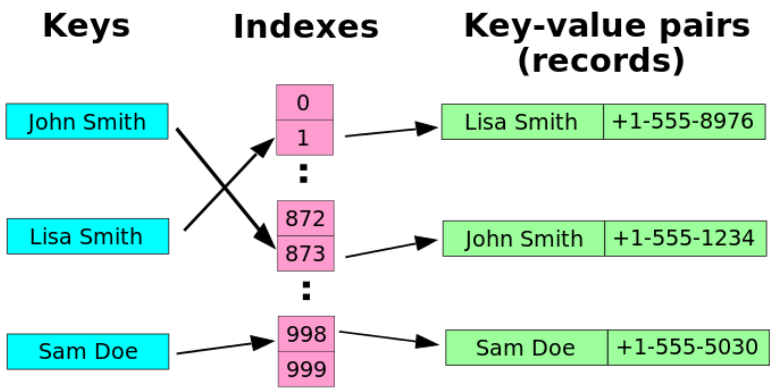
- Stockage des tags dans les attributs étendus (XATTR).
- Outil dédié plutôt que reprendre les commandes existantes.
- Confort d'utilisation.

Indexation des fichiers et des tags



Indexation des fichiers et des tags





Surveillance du système de fichiers

- Mise à jour du graphe lors des événements suivants :
  - Changement sur les tags.
  - Création de fichiers/répertoires.
  - Suppression de fichiers/répertoires.
  - Déplacement/renommage de fichiers/répertoires.
- Pattern producer-consumer.

Requêtes de tags et fichiers

- Lister les fichiers et répertoires associés à des tags (expressions logiques).
- Lister les tags existants.
- Renommer un tag.

Rust
Généralités (1)

- Langage moderne, performant, fiable, compilé, et fortement typé.
- Disponible sur Linux, Windows et macOS.
- Cargo : système de compilation et d'exécution et gestionnaire de paquets intégré à Rust.
- Structures, collections, énumérations et pattern matching.

```
1 enum Direction { North, South, East, West }
2 match direction {
3   Direction::North => println!("Go North"),
4   Direction::South => println!("Go South"),
5   _ => println!("Go East or West") //clause par défaut
6 }
```

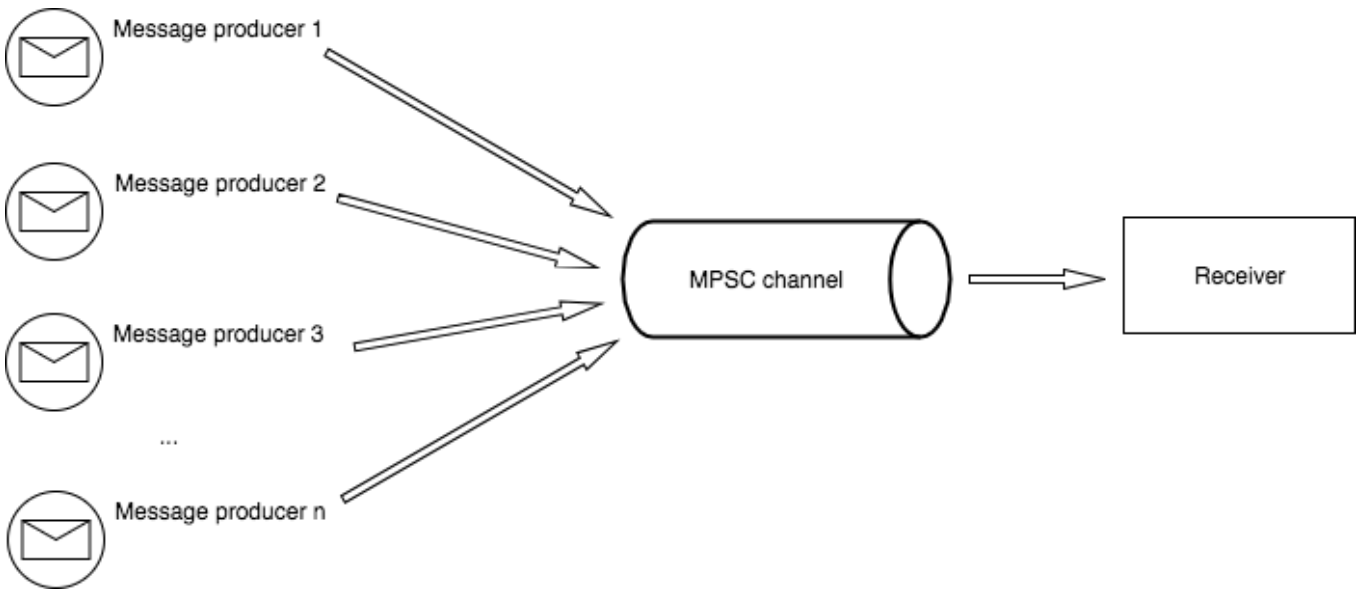
Rust
Généralités (2)

- Tests.
- Gestion des erreurs.

```
1 enum Result<T, E> { Ok(T), Err(E), }
2 match value {
3   Ok(data) => println!("u32 value : {}", data),
4   Err(error) => println!("Error : {}", error)
5 }
```

Rust
Généralités (3)

- Unsafe Rust.
- Concurrence et Threads.



Rust
Ownership et Borrowing (1)

Ownership

- Chaque variable est dite le "possesseur" (*owner*) d'une valeur.
- Il ne peut y avoir qu'un seul *owner* pour une valeur.
- Lorsque l'*owner* est détruit ou change de portée, la valeur est détruite.

Borrowing

1. À tout moment, il ne peut y avoir soit une seule référence mutable, soit plusieurs références immutables, mais pas les deux en même temps.
2. Les références doivent toujours être valides.

Rust

Ownership et Borrowing (2)

```rust
let a = 10;
let b = a;
let mut my_vec = vec![3, 2, 1];
let other_vec = my_vec;
my_vec.push(42); // Erreur, la valeur a été déplacée
```

```rust
fn main() {
    let mut my_vec = vec![3, 2, 1];
    ref_immutable(&my_vec);
    ref_mutable(&mut my_vec);
}
fn ref_immutable(v : &Vec<i32>) { println!("{:?}", v); }
fn ref_mutable(v : &mut Vec<i32>) { v.push(42); }
```

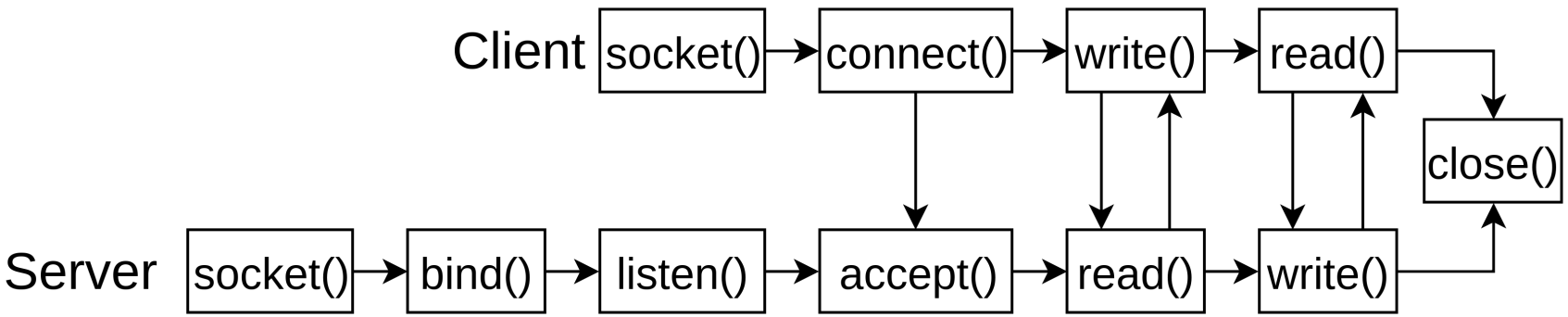
Attributs étendus (XATTR)

- Métadonnées sous forme de paire `espace.nom:valeur`.
- Nom = chaine de caractères, valeur = chaine de caractères ou données binaires.
- Existent sous ext2-3-4, XFS, Btrfs, UFS1-2, NTFS, HFS+, ZFS.

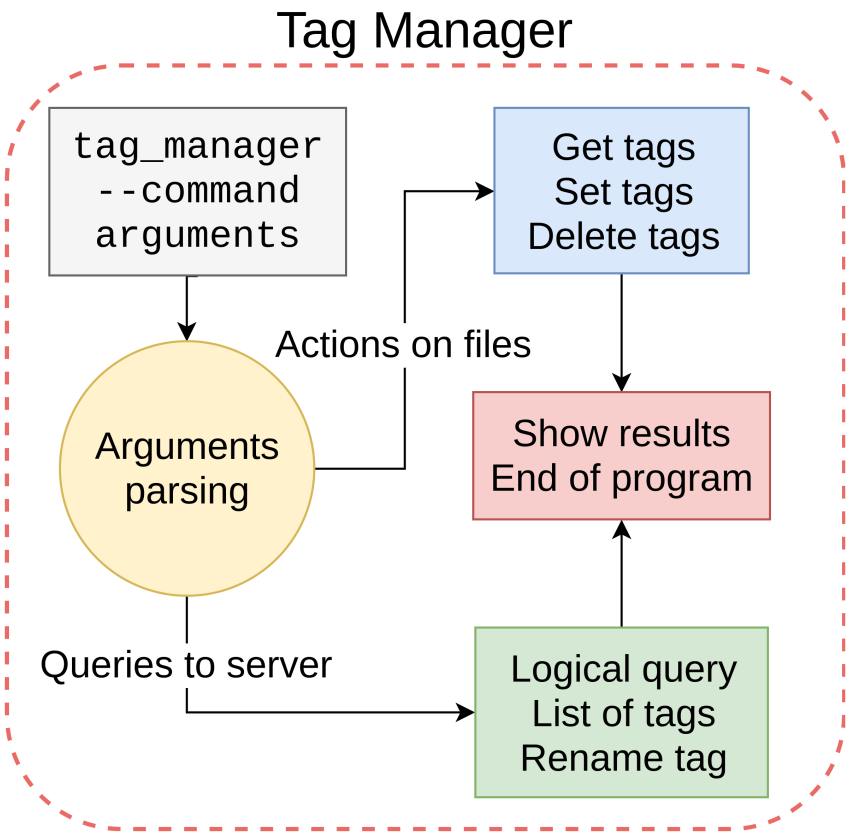
Inotify

- API de notifications d'événements sur le système de fichiers.
- Trois appels système : initialisation, ajout de surveillance sur un chemin de fichiers donné et suppression de cette surveillance.
- Lecture d'un événement avec read().

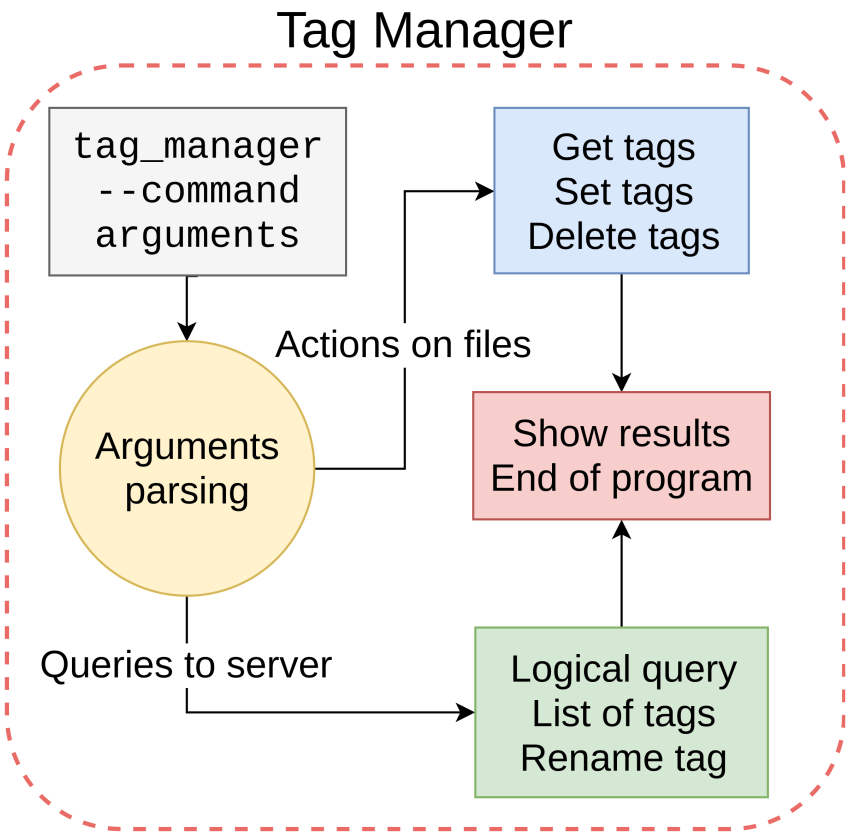
Sockets


Client
socket()
connect()
write()
read()
close()
Server
socket()
bind()
listen()
accept()
read()
write()

Tag Manager

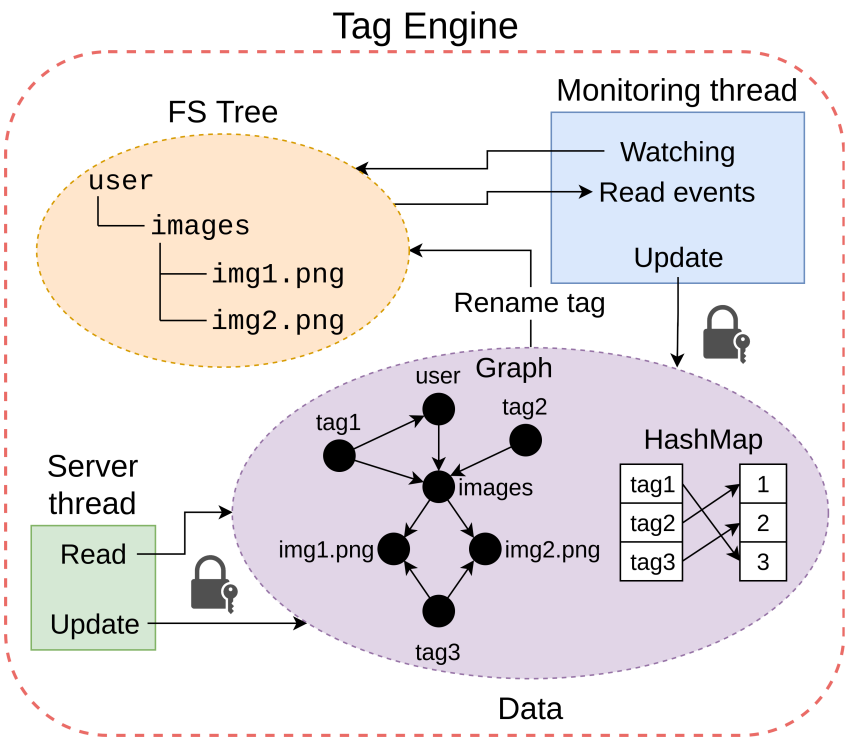

Tag Manager
tag_manager
--command
arguments
Get tags
Set tags
Delete tags
Actions on files
Arguments
parsing
Show results
End of program
Queries to server
Logical query
List of tags
Rename tag

Tag Manager

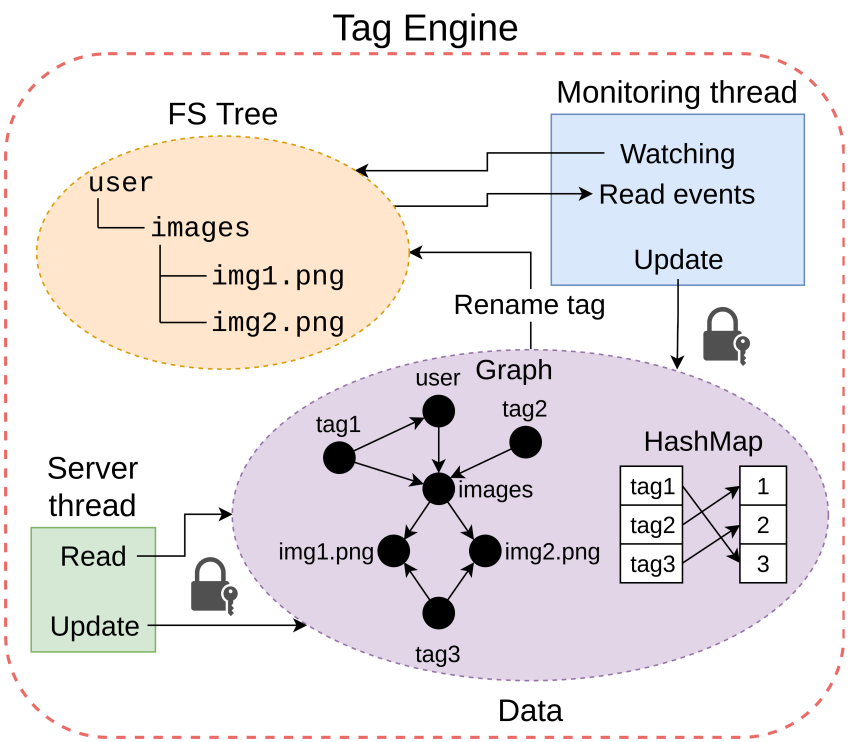


- CLI.
- Gestion des tags.
- Requêtes sur les tags et fichiers.
- Manipule les XATTR des fichiers.
- Programme "client".

Tag Engine

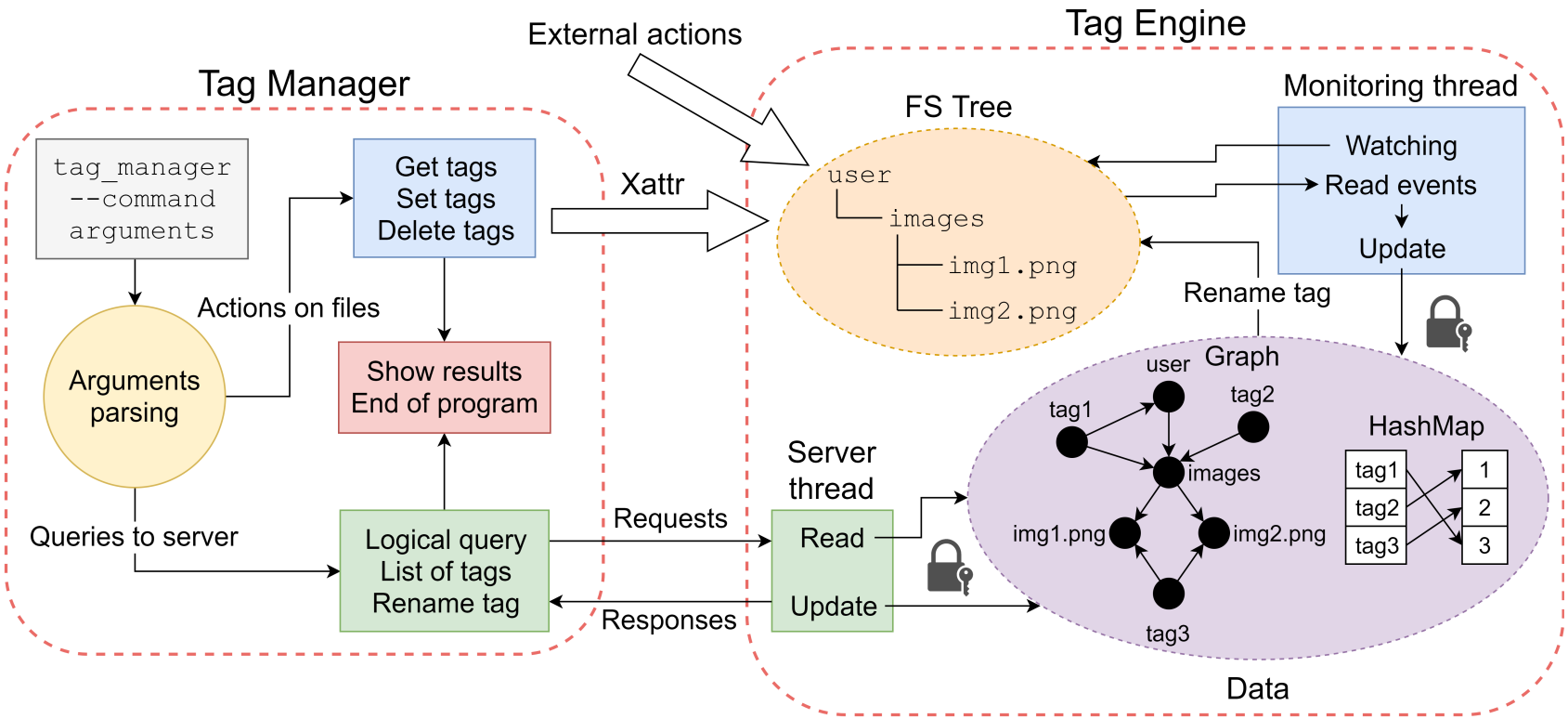

Tag Engine
FS Tree
user
images
img1.png
img2.png
Monitoring thread
Watching
Read events
Update
Rename tag
Graph
user
tag1
tag2
images
img1.png
img2.png
tag3
HashMap
tag1
tag2
tag3
1
2
3
Server thread
Read
Update
Data

Tag Engine



- Surveille l'arborescence des fichiers.
- Écoute sur une socket les requêtes provenant de Tag Manager.
- Maintient la relation entre tags, fichiers et répertoires à l'aide d'un graphe orienté et d'une hashmap.
- Les changements sur le FS sont répercutés sur le graphe.
- Multithread.

TagFS


Tag Engine
External actions
Tag Manager
FS Tree
Monitoring thread
tag_manager
--command
arguments
Get tags
Set tags
Delete tags
Xattr
user
images
img1.png
img2.png
Watching
Read events
Update
Actions on files
Rename tag
Arguments
parsing
Show results
End of program
user
Graph
tag1
tag2
images
HashMap
Server
thread
Queries to server
Logical query
List of tags
Rename tag
Requests
Read
Update
Responses
img1.png
img2.png
tag3
tag1
tag2
tag3
1
2
3
Data

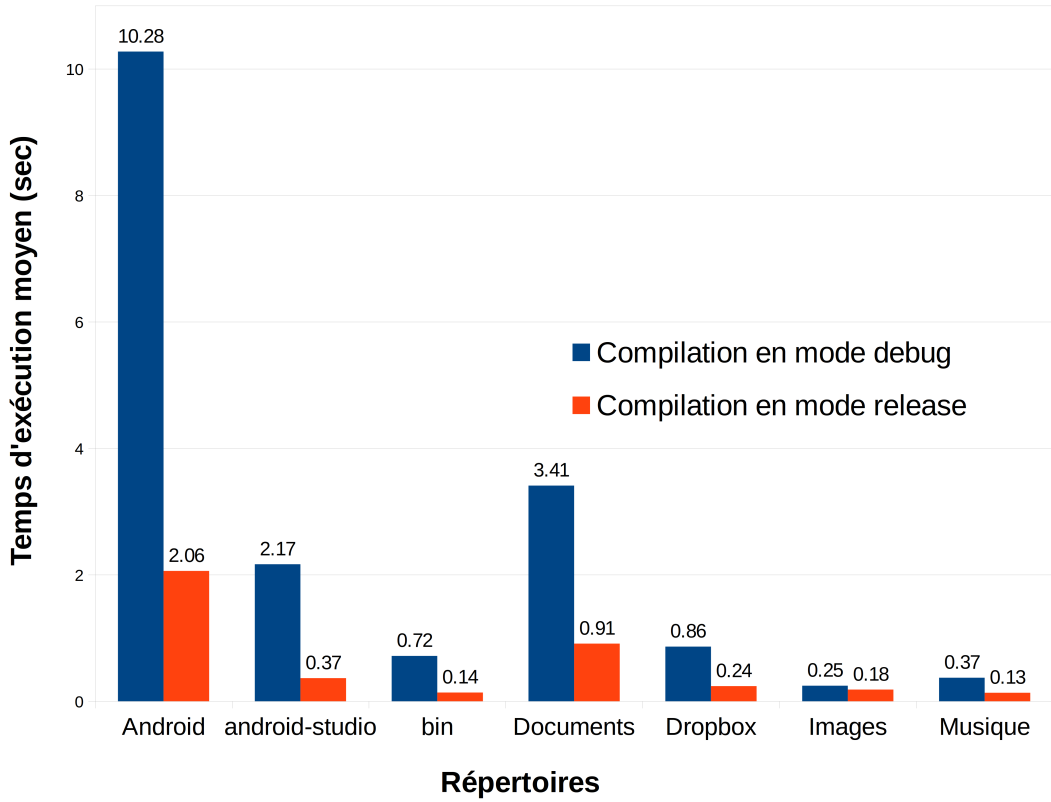
Démo

Mesures de performances

Répertoire	Sous-répertoires	Fichiers
Android	15'172	112'046
android-studio	3'331	13'287
bin	553	9'306
Documents	15'442	64'486
Dropbox	2'377	8'659
Images	5	863
Musique	135	1'352

Mesures de performances

Répertoire	Sous-répertoires	Fichiers
Android	15'172	112'046
android-studio	3'331	13'287
bin	553	9'306
Documents	15'442	64'486
Dropbox	2'377	8'659
Images	5	863
Musique	135	1'352



Rust vs C

Avantages par rapport à C	**Inconvénients par rapport à C**
Garanties sécurité mémoire	Courbe d'apprentissage plus longue
Détection des erreurs à la compilation	Contraintes du langage parfois handicapantes
Compilateur verbeux	Moins répandu
Performances égales ou très proches	Manque de soutien global
Gestion des erreurs	
Absence de NULL	
Cargo et Crates.io	
Librairie standard	
Généricité	

Conclusion

Remerciements

Références