Conception d'un système de "tagging" des fichiers avec Rust

Steven Liatti

Projet de bachelor - Prof. Florent Glück

Hepia ITI 3ème année

21 juin 2018

Résumé Le but de ce projet est de concevoir et développer un "moteur de gestion de tags" pouvant gérer des dizaines de milliers de fichiers et tags associés de manière efficace, en Rust. Le stockage des tags utilisera le mécanisme des "extended attributes" disponibles dans la plupart des systèmes de fichiers modernes. Le moteur d'indexation devra surveiller les fichiers modifiés, créés, ou supprimés afin d'indexer les tags avec un minimum de latence (temps réel). Si le temps le permet, le système développé sera intégré à un environnement desktop choisi (Gnome, KDE, etc.).

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

Table des matières

Table des figures

Table des listings de code source

Conventions typographiques

Lors de la rédaction de ce document, les conventions typographique ci-dessous ont été adoptées.

- Tous les mots empruntés à la langue anglaise ont été écrits en *italique*
- Toute référence à un nom de fichier (ou dossier), un chemin d'accès, une utilisation de paramètre, variable, ou commande utilisable par l'utilisateur, est écrite avec la police d'écriture `Courier New`.
- Tout extrait de fichier ou de code est écrit selon le format suivant :

```
fn main() {
    println!("Hello, world!");
}
```

Acronymes

XATTR *Extended Attributes*. 7, 14

1 Introduction

XATTR

1.1 Motivations

1.2 Buts

[1]

2 Analyse de l'existant

Dans cette section, nous allons analyser les principales solutions existantes, qu'elles soient sous la forme d'applications utilisateur ou intégrées directement dans un système d'exploitation. Jean-Francois Dockes en dresse également une liste avec avantages et inconvénients sur son site [1].

2.1 Applications utilisateur

2.1.1 TMSU

TMSU [2] est un outil en ligne de commande (CLI) qui permet d'attribuer des tags à des fichiers et d'exécuter des recherches par tags. On commence par initialiser TMSU dans le dossier choisi. Une commande liste les tags associés à un ou plusieurs fichiers et une autre liste les fichiers qui possèdent le ou les tags donnés. TMSU offre la possibilité à l'utilisateur de "monter" un système de fichier virtuel avec FUSE (Filesystem in UserSpacE). L'outil est rapide et efficace, mais il comporte quelques défauts :

— Pas d'interface graphique

— Dépendance à FUSE pour monter le système de fichiers virtuel

— Stockage des tags dans une base de données SQLite

2.1.2 Tagsistant

Tagsistant [3] est autre outil CLI de gestion de tags. Il dépend de FUSE et d'une base de données (SQLite ou MySQL) pour fonctionner. Comme pour TMSU, il faut donner un dossier à Tagsistant pour son usage interne. À l'intérieur de ce dernier, se trouvent différents dossiers :

```
/
├── alias -- Dossier contenant les requêtes les plus courantes
├── archive -- Dossier listant les fichiers
├── relations -- Dossier contenant les relations entre les tags et fichiers
├── stats -- Dossier contenant des infos sur l'utilisation de Tagsistant
├── store -- Dossier où sont taggés les fichiers
└── tags -- Dossier de gestion des tags
```

Chaque dossier a un rôle bien précis. Tout se fait avec le terminal et des commandes usuelles (`cp`, `ls`, `mkdir`, etc.). Dans Tagsistant, un dossier créé dans le dossier `tags` correspond à un tag. On se retrouve finalement avec une arborescence de tags et de fichiers [4]. Bien que cet outil soit performant d'un point de vue de la rapidité d'exécution, il comporte les défauts de TMSU ainsi que des nouveaux :

— Pas d'interface graphique
— Dépendance à FUSE pour monter le système de fichiers virtuel
— Stockage des tags dans une base de données
— Utilisation des différents dossiers peu intuitive

2.1.3 TaggedFrog

TaggedFrog [5] est un programme disponible sur Windows uniquement et ne partage pas ses sources. Son fonctionnement interne n'est pas documenté. L'interface est agréable, on peut ajouter des fichiers par Drag & Drop. L'interface créé au fur et à mesure un "nuage" de tags, comme on peut le retrouver sur certains sites web. On peut exécuter des recherches sur les tags et les fichiers. On peut supposer que TaggedFrog maintient une base de données des tags associés aux fichiers, ce qui ne correspond à nouveau pas à nos besoins.

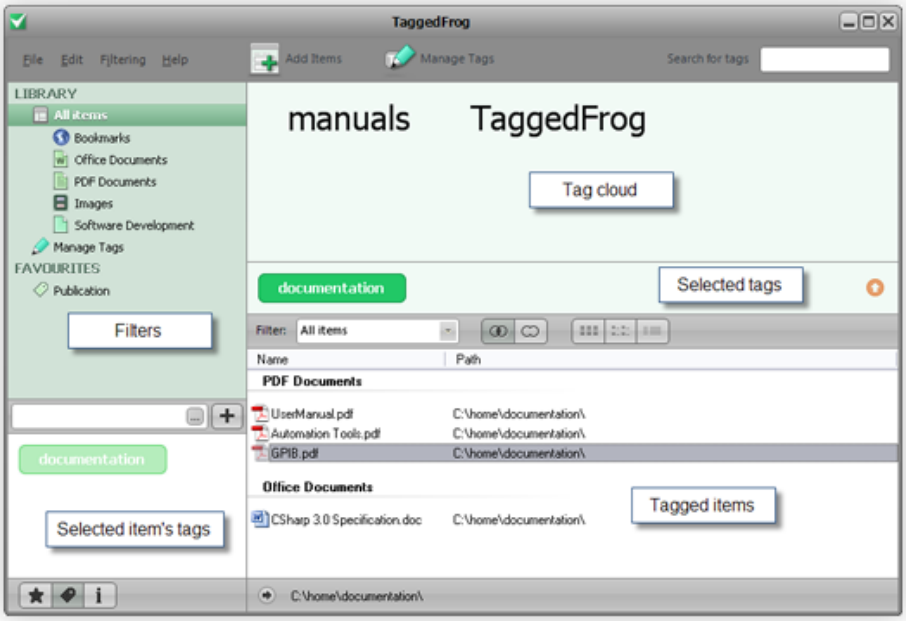

FIGURE 1 – TaggedFrog en utilisation [5]

2.1.4 TagSpaces

TagSpaces [6] est un programme avec une GUI permettant d'étiqueter ses fichiers avec des tags. L'application est agréable à utiliser, on commence par connecter un emplacement qui fera office de dossier de destination aux fichiers. On peut ajouter ou créer des fichiers depuis l'application. Les fichiers existants ajoutés depuis l'application sont copiés dans le dossier (cela créé donc un doublon). Sur le panneau de gauche se situe la zone de gestion des tags. TagSpaces ajoute automatiquement certains tags dits "intelligents" aux fichiers nouvellement créés avec l'application (par exemple un tag avec la date de création).

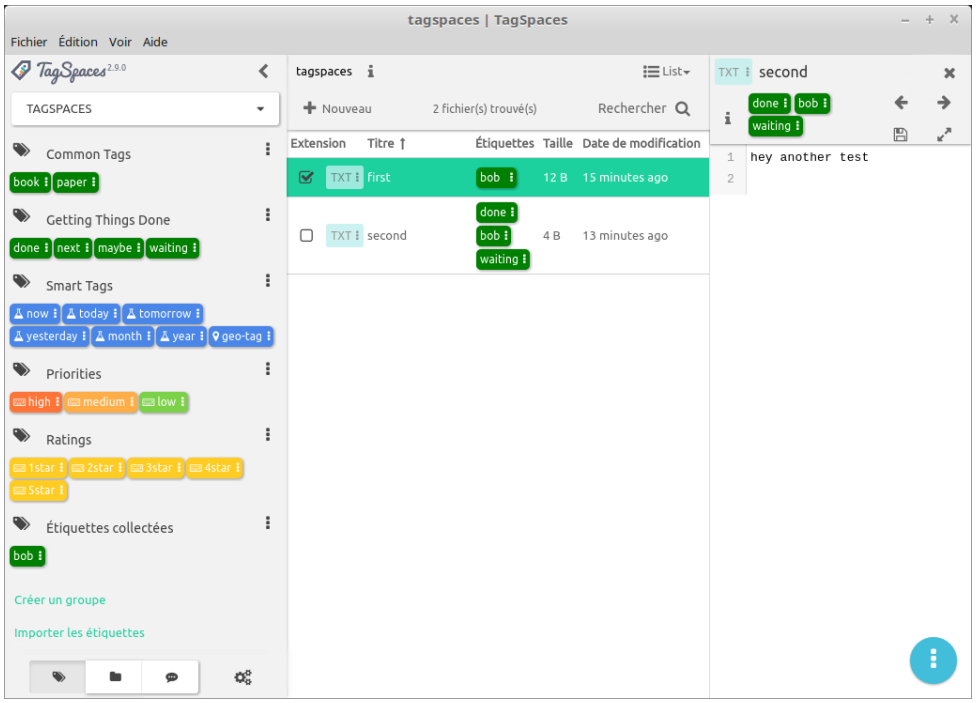

FIGURE 2 – TagSpaces en utilisation

Globalement, l'application est fonctionnelle et *user friendly*. Cependant, 2 points noirs sont à déplorer :

1. L'application copie les fichiers déjà existants sélectionnés par l'utilisateur, ce qui créé une contrainte supplémentaire dans la gestion de ses fichiers personnels.
2. TagSpaces stocke les tags directement dans le nom du fichier, modifiant ainsi son nom [7]. Bien que pratique dans le cas d'une synchronisation à l'aide d'un service cloud, le fichier devient dépendant de TagSpaces, si l'utilisateur décide de changer son nom sans respecter la nomenclature interne, il risque de perdre les tags associés au fichier.

2.2 Fonctionnalités incluses dans le système d'exploitation

2.2.1 Windows

À partir de Windows Vista, Microsoft a donné la possibilité aux utilisateurs d'ajouter des méta-données aux fichiers ; parmi ces méta-données se trouvent les tags. Il existe une fonctionnalité appelée *Search Folder* qui permet de créer un dossier virtuel contenant le résultat d'une recherche sur les noms de fichiers ou d'autres critères [8]. Depuis Windows 8, l'utilisateur a la possibilité d'ajouter des méta-données à certains types de fichiers (ceux de la suite office par exemple), dont des tags. Il peut par la suite exécuter des recherches ciblées via recherche de l'explorateur de fichiers Windows du type `meta:value` [9]. C'est dommage que Windows ne prenne pas en compte davantage de types de fichiers, comme les PDFs ou les fichiers `.txt`.

2.2.2 macOS

macOS possède son propre système pour étiqueter des fichiers. Il est intégré depuis la version OS X 10.9 Mavericks. Depuis l'explorateur de fichiers, l'utilisateur a la possibilité d'ajouter, modifier, supprimer et rechercher des tags. Les fichiers peuvent avoir plusieurs tags

associés. Un code couleur permet de plus facilement se souvenir et visualiser les tags attribués. Dans l'explorateur de fichiers, les tags se retrouvent sur le bas côté, pour y accéder plus rapidement.

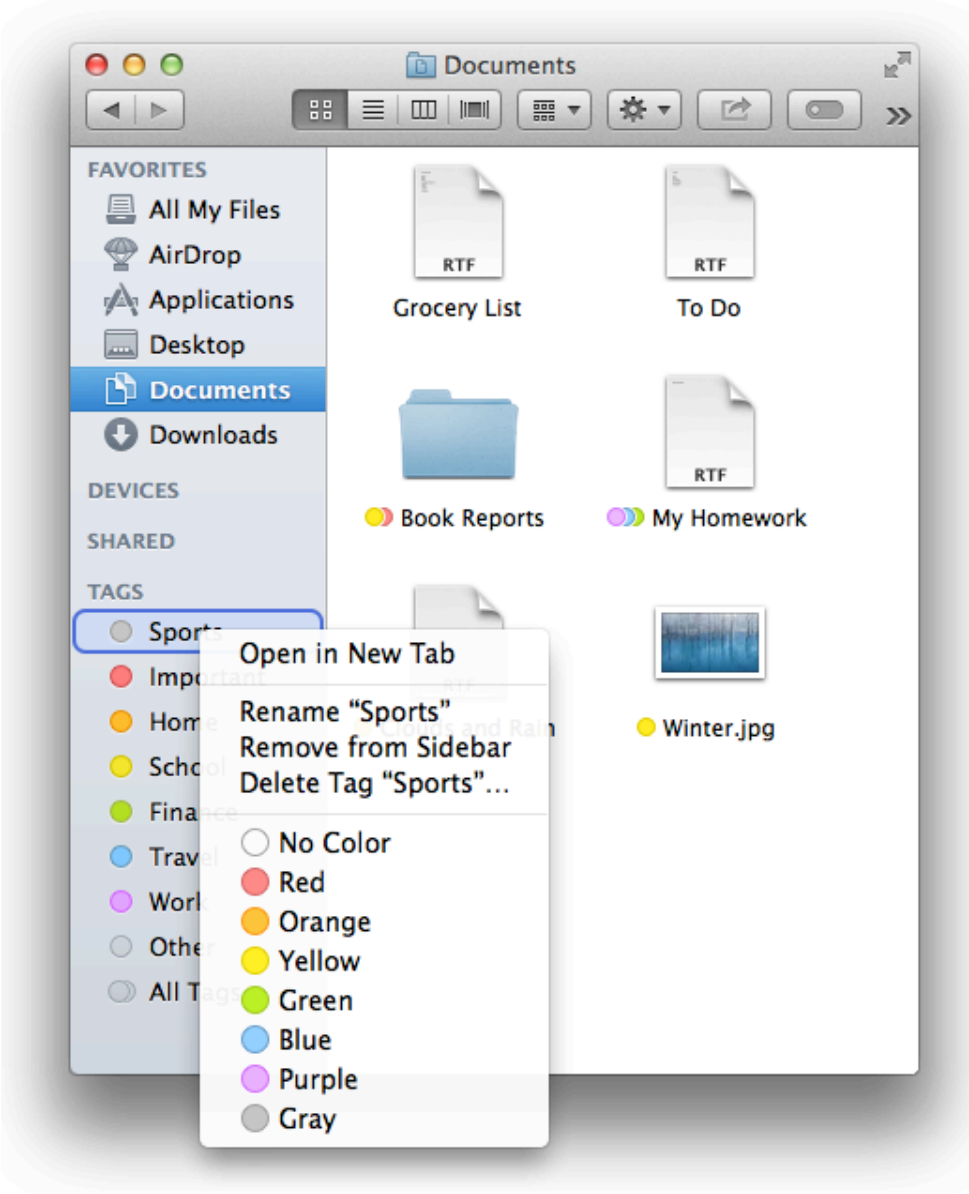

FIGURE 3 – Vue et gestion d'un tag dans le Finder macOS [10]

Lorsque l'on clique sur un tag, une recherche Spotlight est effectuée. Spotlight est le moteur de recherche interne à macOS. Spotlight garde un index des tags, fournissant un accès rapide aux fichiers correspondants [11]. Tous ces tags peuvent se synchroniser sur les différents "iDevices" via iCloud. Finalement, un menu de réglages permet la gestion des tags (affichage, suppression, etc.) [10], [12]. L'implémentation de ce système utilise les extended attributes (voir section 4.2) pour stocker les tags. Les différents tags se trouvent dans l'attribut `kMDItemUserTags`, listés les uns à la suite des autres. Via le Terminal, à l'aide de la commande `mdls`, nous pouvons afficher la liste des tags associés à un fichier, nommé "Hello" pour l'exemple :

```
% mdls -name kMDItemUserTags Hello
kMDItemUserTags = (
    Green,
    Red,
    Essential
)
```

Listing 1 – `mdls` listant les tags d'un fichier sous macOS [13]

Ici, ce fichier "Hello" est étiqueté avec trois tags, "Green", "Red" et "Essential". Le fait que l'indexation est réalisée avec Spotlight implique une réindexation des fichiers dans le cas d'un changement de nom pour un tag donné sous macOS. Le framework système `FSEvents` donne une solution partielle : c'est une API (utilisée également par Spotlight) qui offre aux applications la possibilité d'être notifiées si un changement a eu lieu sur un dossier (un événement toutes les 30 secondes). `FSEvents` maintient des logs de ces changements dans des fichiers, les applications peuvent ainsi retrouver l'historique des changements quand elles le veulent [14].

3 Analyse des besoins

[1]

4 Technologies

4.1 Rust

[15] [16] [17]

4.2 Les extended attributes

4.2.1 Théorie

Les extended attributes, ou "attributs étendus" en français, sont un moyen d'attacher des méta-données aux fichiers et dossiers sous forme de paires `espace:nom:valeur`. L'espace de nom, ou *namespace* en anglais, définit les différentes classes d'attributs. Dans le cadre de ce projet, l'accent est mis sur ext4 sous Linux, il actuellement 4 espaces de noms ou classes : `user`, `trusted`, `security` et `system`. L'espace qui nous intéresse est `user`. C'est là que l'utilisateur ou l'application, pour autant qu'il ait les droits usuels UNIX sur les fichiers, peut manipuler les extended attributes. Les 3 autres espaces de noms sont utilisés entre autres pour les listes d'accès ACL (`system`), les modules de sécurité du kernel (`security`) ou par `root` (`trusted`) [18] [19]. Le nom est une chaine de caractères et la valeur peut être une chaine de caractères ou des données binaires. Les extended attributes sont stockés dans les fichiers. De nombreux systèmes de fichiers gèrent leur usage : ext2-3-4, XFS, Btrfs, UFS1-2, NTFS, HFS+, ZFS. Ces systèmes de fichiers sont utilisés par les 4 systèmes d'exploitation les plus répandus : Windows, macOS, Linux et FreeBSD. Windows utilise les extended attributes notamment dans sa gestion des permissions Unix dans le shell Linux intégré à Windows 10 [20]. macOS, comme vu à la section 2.2.2, les utilise entre autres dans son système de gestion des tags.

La commande `xattr` permet de les manipuler. Sous Linux, il en existe 3 : `attr`, `getfattr` et `setfattr`. Sous Linux avec ext2-3-4, chaque attribut dispose d'un bloc de données (1024, 2048 ou 4096 bytes) [19]. Apple et freedesktop.org préconisent la notation DNS inversée pour nommer les attributs [21], [22] car n'importe quel processus peut modifier les attributs dans l'espace utilisateur. En préfixant du nom du programme le nom de l'attribut, par exemple `user.myprogram.myattribute`, on diminue le risque qu'une autre application utilise le même nom d'attribut. Malheureusement, la plupart des outils CLI Linux pour manipuler les fichiers comme `cp`, `tar`, etc. ne prennent pas en compte les attributs avec leur syntaxe par défaut [23].

4.2.2 Petites manipulations

Pour vérifier la portabilité des extended attributes, quelques tests ont été réalisés entre un SSD faisant office de disque système à Linux Mint avec 2 clés USB (de 8 et 64 Go) et un emplacement réseau monté en NFS. Le listing suivant montre la sortie de la commande `df`, qui renvoie l'utilisation des différents emplacements de stockage, dans l'ordre : le disque système, en ext4, la clé de 8 Go formatée une fois en FAT32, puis une autre fois en NTFS, la clé de 64 Go formatée en ext4 et finalement une machine virtuelle sous Debian 9 montée en NFS.

```
1 Sys. de fichiers        Type   Taille Utilisé Dispo Uti% Monté sur
2 /dev/sda2               ext4     451G    334G   96G  78% /
3 /dev/sdg1               vfat     7.7G    4.0K  7.7G   1% /media/pc/cle1
4 /dev/sdg1               fuseblk  7.7G     41M  7.7G   1% /media/pc/cle1
5 /dev/sdg1               ext4      59G     33G   23G  59% /media/pc/cle2
6 192.168.1.21:/home/user nfs4     916G    198G  673G  23% /mnt/debian
```

Listing 2 – Output de `df -Th` : le disque système, les clés USB et le NFS

La démarche est la suivante : un extended attribute dans l'espace `user` avec comme nom `author` et comme valeur `steven` est ajouté au fichier `file.txt` avec `attr`. Ce fichier est copié avec `cp` en prenant garde à préserver l'attribut (option `--preserve=xattr`). Une fois copié, on tente de lire le même attribut, toujours avec `attr`. Les résultats sont les suivants :

```
45 ~ $ attr -s author -V steven file.txt
46 L'attribut "author" positionné à une valeur de 6 octets pour file.txt :
47 steven
48 ~ $ cp --preserve=xattr file.txt /media/pc/cle1
49 cp: setting attributes for '/media/pc/cle1/file.txt': Opération non
   supportée
```

Listing 3 – Copie sur clé USB 8 Go, FAT32

```
~ $ attr -s author -V steven file.txt
L'attribut "author" positionné à une valeur de 6 octets pour file.txtă:
steven
~ $ cp --preserve=xattr file.txt /media/pc/cle1
~ $ cd /media/pc/cle1
/media/pc/cle1 $ attr -g author file.txt
L'attribut "author" avait une valeur de 6 octets pour file.txtă:
steven
```

Listing 4 – Copie sur clé USB 8 Go, NTFS

```
~ $ attr -s author -V steven file.txt
L'attribut "author" positionné à une valeur de 6 octets pour file.txtă:
steven
~ $ cp --preserve=xattr file.txt /media/pc/cle2
~ $ cd /media/pc/cle2
/media/pc/cle2 $ attr -g author file.txt
L'attribut "author" avait une valeur de 6 octets pour file.txtă:
steven
```

Listing 5 – Copie sur clé USB 64 Go, ext4

```
~ $ cp --preserve=xattr file.txt /mnt/debian
cp: setting attributes for '/mnt/debian/file.txt': Opération non
supportée
```

Listing 6 – Copie sur l'emplacement réseau distant, NFS

On constate que l'opération est infructueuse sur la clé en FAT32 et sur l'emplacement réseau monté en NFS alors qu'elle réussit sur les clés USB en NTFS et ext4.

4.3 Notifications

5 Indexation

[24] 2 index (clé -> valeur) :

1. tag -> fichiers : Index dense (Index inversé) -> hashmap : clé = tag, valeur = ensemble des path des fichiers -> intersection des ensembles
2. fichier -> tags -> 2 possibilités :
   2.1. Index dense (Index inversé) -> hashmap : clé = path du fichier, valeur = ensemble des tags du fichier
   2.2. B-arbre -> noeud = dossier/fichier, feuille = fichier -> chaque élément contient en ensemble de tags (ensemble vide si pas de tags)

Cas de figure/utilisation des fichiers par l'utilisateur :

— Un tag est ajouté à un fichier :

    - Index 1 : obtenir l'ensemble des fichiers associés au tag et ajouter le fichier. Si tag non présent, ajouter la nouvelle clé => O(1)
    - Index 2.1 : obtenir l'ensemble des tags d'un fichier et ajouter le tag => O(1)
    - Index 2.2 : parcourir l'arbre à la recherche du fichier et ajouter le nouveau tag => O(log(n))
- Un tag est supprimé d'un fichier :
    - Index 1 : obtenir l'ensemble des fichiers associés au tag et supprimer le fichier => O(1)
    - Index 2.1 : obtenir l'ensemble des tags associés au fichier et supprimer le tag => O(1)
    - Index 2.2 : parcourir l'arbre à la recherche du fichier et supprimer le tag => O(log(n))
- Un tag change de nom (opération lourde) : 2 possibilités :
    1. Dans les 2 index, les tags pointent vers une "entité" tag
        - Index 1 : "rien de spécial"
        - Index 2.1 et 2.2 : "rien de spécial"
    2. Dans les 2 index, les tags sont des valeurs (strings) inscrits en dur
        - Index 1 : renommer la clé / retirer l'ensemble et le rajouter selon le nouveau nom
        - Index 2.1 et 2.2 : récupérer les fichiers grâce à index 1 et boucler pour changer le nom

    Dans les deux cas, parcourir les fichiers sur le disque pour changer la valeur des tags dans les XATTR
- Un fichier est supprimé :
    - Index 1 : supprimer toutes les clés correspondant aux tags
    - Index 2.1 : supprimer la clé fichier
    - Index 2.2 : parcourir et supprimer la clé fichier
- Un fichier est copié :
    - Index 1 : parcourir tous les tags du fichier et ajouter le nouveau fichier créé aux ensembles
    - Index 2.1 et 2.2 : créer une nouvelle clé et copier l'ensemble de tags vers cette clé
- Un fichier est déplacé ou change de nom :
    - Index 1 : parcourir tous les tags du fichier et changer le nom du fichier
    - Index 2.1 et 2.2 : renommer la clé / retirer l'ensemble et le rajouter selon le nouveau nom

6 Architecture

7 Graphes

8 Réalisation

8.1 Tag Manager

La première réalisation de ce projet est un outil en ligne de commande, écrit en Rust, permettant de facilement lister, ajouter et supprimer des tags à des fichiers et dossiers. Il fait usage de deux *crates* disponibles sur crates.io : clap [25] et xattr [26]. Clap (Command Line Argument Parser for Rust) est une librairie pour parser les arguments d'un programme en ligne de commande. Xattr est une API en Rust pour récupérer, lister, ajouter/modifier et supprimer des extended attributes avec Rust. Elle fait appel aux appels systèmes en C.

9 Discussion/résultats

10 Conclusion

10.1 Remerciements

11 Références

[1] Jean-Francois Dockes. Extended attributes and tag file systems. https://www.lesbonscomptes.com/pages/tagfs.html, juillet 2015. Consulté le 04.05.2018.

[2] Paul Ruane alias oniony. Tmsu. https://tmsu.org/. Consulté le 18.05.2018.

[3] Tagsistant. Tagsistant : semantic filesystem for linux. http://www.tagsistant.net/. Consulté le 18.05.2018.

[4] Tagsistant. Tagsistant 0.8.1 howto. http://www.tagsistant.net/documents-about-tagsistant/0-8-1-howto, mars 2017. Consulté le 18.05.2018.

[5] Andrei Marukovich. Taggedfrog - quick start manual. http://lunarfrog.com/projects/taggedfrog/quickstart. Consulté le 18.05.2018.

[6] TagSpaces. Your offline data manager. https://www.tagspaces.org/. Consulté le 18.05.2018.

[7] TagSpaces. Organize your data with tags. https://docs.tagspaces.org/tagging. Consulté le 18.05.2018.

[8] Greg Shultz. An in-depth look at windows vista's virtual folders technology. https://www.techrepublic.com/article/an-in-depth-look-at-windows-vistas-virtual-folders-technology/, octobre 2005. Consulté le 21.05.2018.

[9] Russell Smith. Manage documents with windows explorer using tags and file properties. https://www.petri.com/manage-documents-with-windows-explorer-using-tags-and-file-properties, avril 2015. Consulté le 21.05.2018.

[10] Apple team. Os x : Tags help you organize your files. https://support.apple.com/en-us/HT202754, février 2015. Consulté le 08.05.2018.

[11] John Siracusa. Mac os x 10.4 tiger - spotlight. https://arstechnica.com/gadgets/2005/04/macosx-10-4/9/, avril 2005. Consulté le 08.05.2018.

[12] John Siracusa. Os x 10.9 mavericks : The ars technica review - tags. https://arstechnica.com/gadgets/2013/10/os-x-10-9/8/, octobre 2013. Consulté le 08.05.2018.

[13] John Siracusa. Os x 10.9 mavericks : The ars technica review - tags implementation. https://arstechnica.com/gadgets/2013/10/os-x-10-9/9/, octobre 2013. Consulté le 08.05.2018.

[14] John Siracusa. Mac os x 10.5 leopard : the ars technica review - fsevents. https://arstechnica.com/gadgets/2007/10/mac-os-x-10-5/7/, octobre 2007. Consulté le 08.05.2018.

[15] Rust Team. The rust programming language, 2nd edition. https://doc.rust-lang.org/stable/book/second-edition/. Consulté le 25.04.2018.

[16] Manuel Hoffmann. Are we (i)de yet ? https://areweideyet.com/. Consulté le 25.04.2018.

[17] Computational Geometry Lab. Learning rust with entirely too many linked lists. http://cglab.ca/~abeinges/blah/too-many-lists/book/. Consulté le 28.04.2018.

[18] Jeffrey B. Layton. Extended file attributes rock ! http://www.linux-mag.com/id/8741/, juin 2011. Consulté le 09.05.2018.

[19] Andreas Gruenbacher. attr(5) - linux man page. https://linux.die.net/man/5/attr. Consulté le 09.05.2018.

[20] Jack Hammons. Wsl file system support. https://blogs.msdn.microsoft.com/wsl/2016/06/15/wsl-file-system-support/, juin 2016. Consulté le 21.05.2018.

[21] John Siracusa. Mac os x 10.4 tiger - extended attributes. https://arstechnica.com/gadgets/2005/04/macosx-10-4/7/, avril 2005. Consulté le 08.05.2018.

[22] freedesktop.org. Guidelines for extended attributes. https://www.freedesktop.org/wiki/CommonExtendedAttributes/, mai 2018. Consulté le 04.05.2018.

[23] Jean-Francois Dockes. Extended attributes : the good, the not so good, the bad. https://www.lesbonscomptes.com/pages/extattrs.html, juillet 2014. Consulté le 04.05.2018.

[24] Philippe Rigaux. 4. indexation : l'arbre b. http://sys.bdpedia.fr/arbreb.html. Consulté le 28.05.2018.

[25] Kevin Knapp. clap. https://crates.io/crates/clap, mars 2018. Consulté le 14.05.2018.

[26] Steven Allen. xattr. https://crates.io/crates/xattr, juillet 2017. Consulté le 14.05.2018.