TAGXFS

PREAMBLE

Tagxfs is a semantic file system. It extends the user space file system to a tag based hierarchy.

SCENARIO

Imagine that you want to add a new folder to your existing folder structure. But semantically it belongs into more locations, e.g. you have the trailer for the movie 'Indiana Jones 4' and existing a folder structure (ref.1.1). It belongs in the folder ***ADVENTURE*** and also in ***2000S*** (production year 2008).

ref.1.1

- **MOVIE**
  - **GENRE**
    - **SCIFI**
    - **ADVENTURE**
    - **HORROR**
  - **YEAR**
    - **2010S**
    - **2000S**
    - **1990S**
- **MUSIC**

WHAT YOU CAN DO ?

A. hold two copies, first in the ***ADVENTURE*** folder and second in ***2000S*** ... no way !!!
B. hold only one copy in the ***ADVENTURE*** folder, and symbolic link in ***2000S*** ... that's a better solution, but maintaining symbolic links in case you add/remove folder/category, will be a pain.
C. use a file system extension which will solve this problem for you.

WHY IS IT GOOD TO CATEGORIZE ITEMS ?

;) ... That's not exactly a soup question, is it? Try wiki categories.

CONCEPT

Extension consists of links, tags and node hierarchy.

TAGS

Tags represent unique categories and can be of the following types:

- filter tag (f.e. ***SCIFI***, ***ADVETURE***, ***MOVIE,*** ...)
- group/void tag (f.e. ***GENRE***, ***YEAR***)

Location ***/MOVIES/GENRE/SCIFI*** filters content by two categories: ***MOVIES*** and ***SCIFI***. ***GENRE*** is used as a helper tag, which groups genre tags into one folder.

LINKS

Link is a symbolic representation of the folder which we want to categorize. It's represented by a unique short name and full path on base file system. We can mark links with a set of tags, later used as filters at node hierarchy. For our concept we stick only with folders. Files are too small an entity and mostly in big quantities, to manage them by tags. They can be organized by the default file system.

NODE HIERARCHY

- **NODE**
  - **NODE**
    - **LEAF**
    - **LEAF**
  - **NODE**
    - **NODE**
      - **LEAF**
      - **LEAF**
    - **NODE**
      - **LEAF**
      - **LEAF**

- **TAG**
  - **TAG**
    - **LINK**
    - **LINK**
  - **TAG**
    - **TAG**
      - **LINK**
      - **LINK**
    - **TAG**
      - **LINK**
      - **LINK**

Node hierarchy is our tag representation in a tree folder structure. It consists of nodes (Hierarchy nodes), and leaves (Links). For each node we can setup one tag, set of sub-nodes, mode, and unique position in the existing node hierarchy. The mode of hierarchy node defines which links will be listed.

Hierarchy node modes:

- **All** - subnodes + collection of all links, filter tags aren't applied
- **Links** - subnodes + collection of specific(filtered) links, filter tags are applied
- **None** - subnodes

The first mode, **All** lists all links available in our hierarchy under the targeted node. There are no filter criteria applied. The second mode, **Links** lists links filtered by tags from path corresponding to the targeted node. In other words - only those links are listed which contain all tags assigned to tag nodes on the path from root node to targeted node. And the last, third mode **None** doesn't list any links at targeted node.

Following example shows the node hierarchy, links with associated tags and hierarchy nodes with assigned mode.

- **TAG1**
  - **TAG2**
    - **LINK1**
    - **LINK2**
  - **TAG3**
    - **TAG2**
      - **LINK1**

LINK1: TAG1, TAG2, TAG3
LINK2: TAG1, TAG2
LINK3: TAG1, TAG2, TAG3, TAG4
LINK4: TAG1, TAG3, TAG4, TAG5

/TAG1 - (NONE)

- **LINK3** **/TAG1/TAG2 - (LINKS)**
- **TAG4** **/TAG1/TAG3/TAG2 - (LINKS)**
  - **LINK3** **/TAG1/TAG3/TAG4 - (LINKS)**
  - **LINK4**

HOW TO USE IT:

HELP

Tagxfs supports help for all commands and subcommands through the **help** command.

```
$ tagxfs help [<command> [<subcommand> ...]
```

REPOSITORY LOCATION

Node hierarchy is stored in sql3lite repository. We have two ways how to define it's location.

- **Environment variable:** TAGXFS_REPOSITORY
- **Executable option:** --repository=<repository>

```
$ export TAGXFS_REPOSITORY=./test_repository.tagfs
$ tagxfs --repository=./test_repository.tagfs <commands>
```

CREATING REPOSITORY

After we setup a repository path, we need to create an instance. We can do that using the **database create** command.

```
$ tagxfs database create
```

ADDING TAGS

Before we can create a node hierarchy, it's necessary to define tags. For operations with tags we have these commands available: **tag add|list|remove**

```
$ tagxfs tag add movie genre scifi adventure horror year 2010s
2000s 1990s music
```

CHOOSING VOID/FILTER TAGS

We need to separate void tags from filter tags. Newly added tags are filter tags by default. We can change that using the **tag set filter** command.

```
$ tagxfs tag set filter 0 genre year
```

CREATING NODE HIERARCHY

After we have added the tags we can create a node hierarchy. We will create hierarchy nodes by assigning tags to specific locations. For operations with hierarchy nodes we have following commands available: **hierarchy add|remove|copy|move|list|tree**

```
$ tagxfs hierarchy add / movie
$ tagxfs hierarchy add / music
$ tagxfs hierarchy add /movie genre
$ tagxfs hierarchy add /movie year
$ tagxfs hierarchy add /movie/genre scifi
$ tagxfs hierarchy add /movie/genre adventure
$ tagxfs hierarchy add /movie/genre horror
$ tagxfs hierarchy add /movie/year 2010s
$ tagxfs hierarchy add /movie/year 2000s
$ tagxfs hierarchy add /movie/year 1990s
```

SETTING UP MODE FOR HIERARCHY NODES

To list links we need to define a collection mode for each hierarchy node it applies to. By default the hierarchy node doesn't list any. We setup a collection mode using the **hierarchy mode** command.

```
$ tagxfs hierarchy mode /movie/genre/scifi links
$ tagxfs hierarchy mode /movie/genre/adventure links
$ tagxfs hierarchy mode /movie/genre/horror links
$ tagxfs hierarchy mode /movie/year/2010s links
$ tagxfs hierarchy mode /movie/year/2000s links
$ tagxfs hierarchy mode /movie/year/1990s links
```

ADDING LINKS

Now we can put links into our repository. By entering short name, we will override a default link name extraction from the original path. We can add links using the **link add** command.

```
$ tagxfs link add <your_media_folder>/movie/the.5th.element
$ tagxfs link add <your_media_folder>/movie/aliens
$ tagxfs link add <your_media_folder>/movie/event.horizon
$ tagxfs link add <your_media_folder>/movie/Indiana.Jones.4
indiana.jones.4
```

ADDING TAGS TO LINKS

Finally we'll categorize newly added links. We'll use the **link tag add** command.

```
$ tagxfs link tag add the.5th.element movie scifi 2000s
$ tagxfs link tag add aliens scifi movie horror 1990s
$ tagxfs link tag add event.horizon movie scifi horror 2000s
$ tagxfs link tag add indian.jones.4 movie adventure 2010s
```

MOUNTING FILESYSTEM

We created repository and filled with the data. Let's mount it.

```
$ tagxfs mount <mount_destination>
```

LINKS

- Download

- Source Forge