TagFS - A Tag Based Filesystem

Cătălina Macaleţ
Computer Science Department
Politehnica University of Bucharest
Email: catalina.macalet@cti.pub.ro

Eugen Hristev
Computer Science Department
Politehnica University of Bucharest
Email: eugen.hristev@cti.pub.ro

Mihai Dinu
Computer Science Department
Politehnica University of Bucharest
Email: mihai.dinu@cti.pub.ro

Sorin Dumitru
Computer Science Department
Politehnica University of Bucharest
Email: sorin.dumitru@cti.pub.ro

***Abstract*—File systems are an integral part of every operating system. Because of the high capacity of modern storing devices file systems need a better way of organizing and accessing data in order to be easier for one to retrieve exactly the files he/she is looking for. Also, the need of users to personalize the content stored and to find specific data, pushes manufacturers to employ alternatives for the curent design.**

TagFS implements a tag based file system in Linux, using a user space application which offers support for tagging files and browsing files by tags, and a kernel module that hooks into the VFS to keep metadata about files. We present a simple way to implement such a system and how the regular user can benefit from file tagging.

***Index Terms*—File systems, Tags, VFS, metadata**

I. Introduction

In most operating systems the files are hierarchically organized. This means that there usually is a starting point, or parent directory. In Windows based systems there are multiple starting points based on the physical hard drive partitions. In UNIX-like systems, there is a single root drive with different mount points available for users to add or remove subtrees from different drives, partitions, etc. In these filesystems a user organizes related data by storing it in the same folder but say that a user, Bob, has two separated folders one for storing photos taken in the mountains (Mountain-pics) and one for storing photos in which a certain person appears (Alice-pics). Two questions arise, one, where should Bob store a picture taken in the mountains in which Alice appears and two, how could Bob find the pictures taken in the mountains in which Alice appears. For current filesystems the answer to the first question might be storing the photo in either folder and in the second one creating a link to this photo or, store it in both folders. The answer to the second one could be naming the photo in such a way that retrieving them based on the previously stated criteria would work. TagFS file system aims to bring a different approach, based on tags rather than hierarchical system that is rooted for a long time in modern operating system. For the above example, for a TagFS filesystem the answer to both questions would be adding tags to photos (<mountains><Alice>) and then search for files that contain these tags. The question of where to store a specific photo would not be that important anymore. A pure tag file system is difficult to implement starting from zero, so we tried to adapt the current file system in Linux to support tags and see how the two systems can coexist on an end-user machine. A tag file system should be able to organize files, data on the disk regardless of hierarchical logical approach. The position of the files on the disk is irrelevant and completely transparent to the user. The file system should be able to put files on disks and simply recover them on demand based on tags requests. In our approach, logical directory based organization and file tags coexist, in order to see how the two systems can fit and how the user can use alternatives for searching and clustering the information it has. We implemented a tag layer in the Linux Virtual File System and tested how this impacts the regular user. We have added posibilities for the user to manipulate the tags (add, delete, search) in order to increase the flexibility of the filesystem and the way it interacts with the user. TagFS is expected to make it easier to work with files, especially personal ones.

II. State of the art

The idea of tagging files in order to access them in an easier fashion is not a new one and various attempts to implement solutions have been made. Some of these are specialized solutions for special kind of data, such as Calibre[1] which makes ebook management easier, implemented in userspace. The vast majority of these applications rely on a database where mappings between files and associated metadata are stored and expose a set of commands which translate to specific queries for the database.

A. *Nepomuk-KDE*[2]

Nepomuk-KDENepomuk-KDE is an implementation of Nepomuk which has been integrated with KDE and that allows adding metadata to items stored on a computer and making queries based on that metadata. Based on the Nepomuk specification, Nepomuk- KDE is able to store in a RDF (Resurse Description Frame- work) semantic data from desktop applications. For example the Dolphin desktop manager is able to add simple tags to les or more complicated comments.

[1]http://calibre-ebook.com/
[2]http://nepomuk.kde.org/

This solution is not limited to files metadata. Almost every application can use the RDF store to add semantic metadata to theire objects. For example KMail can do it for emails, Amarok can do it for music les, but it is used mostly for tagging les.

B. TaggedFrog

TaggedFrog[1] is a Windows application based on the convenint drag'n'drop technique. It allows you to organize your files, documents and Web links just by adding objects to the library and tagging them with any keywords. Moreover, you are able tagging files directly from Windows File Explorer because the application is integrated with Explorer's context menu.

C. pyTAGSfs

pyTAGSfs[2] is a FUSE filesystem, written in python for Linux and Mac OS X systems, that arranges media files in a virtual directory structure based on the file tags. File tags can be changed by moving and renaming virtual files and directories. The virtual files can also be modified directly, and, of course, can be opened and played just like regular files.

D. TagFS:Bringing Semantic Metadata to the Filesystem

TagFS: Bringing Semantic Metadata to the Filesystem[3] is a research projected started at the University of Koblenz which, as Nepomuk, relies on RDF for defining semantics and SPARQL. Metadata is stored in a repository having an associated graph, and various opperations can be performed on it(additions, updates, etc).

III. TAGFS

TagFS is a software application that implements a tag-based filesystem in Linux, more specifically, TagFS allows tagging a file at creation time or at a later time, adding and removing tags, listing the tags associated to a file at a given time and, the most important characteristic, TagFS allows browsing for files having specific tag(s).

What is different from the other implementations is that the filesystem hierarchy whill remain unchanged but files will have associated tags (an example is presented in Figure 1).

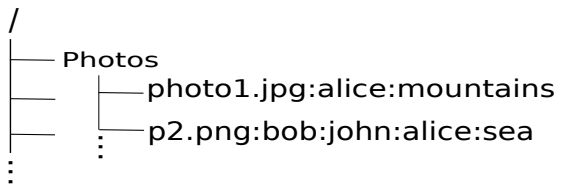

Fig. 1. TagFS hierarchy

The TagFS application architecture is presented in Figure 2. The *CLI* is used to issue commands for tag manipulation. In order for the transition to this new file system to be as user-friendly as possible, we have implemented a different way of manipulating the tags. There are two types of commands. The first type consists of file manipulation commands available on every UNIX-like operating system, such as *ls, touch, mv, cp* whose behaviour and implementation was slightly changed in TagFS implementation in order to support tags. The second type of commands reffers to new TagFS commands implemented in order to provide more tag-related operations - list, remove, add new tags. The implementation changes are related to hooks created in *VFS* and will be detailed in subsection *Tag handling*. No implementation changes at filesystem level were required.

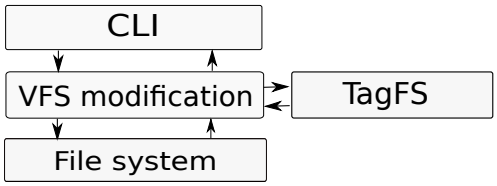

Fig. 2. TagFS architecture

A. Architectural decisions

TagFS started as an idea to create a more user-friendly file system; remembering tags is easier than remembering the name of a file or the place where it is stored but, at the same time a pure tag filesystem might not offer a simpler way of organizing data in a hierarchical manner, a choice to implement TagFS as a new filesystem, from scratch, thus would have been time consuming and would have required a lot of changes into the kernel and user interface. The other choice was to implement TagFS as hooks in VFS in order to store and retrieve needed metadata. Since changes are made at VFS level there will be an overhead for filesystems that subsequently are to be used without tag support. A main concern in implementation was to reduce this overhead to as little as possible.

From the beginning the focus was on the changes needed at VFS and file system level and not on storage possibilities of the mappings between files and associated tags and so these mappings are stored in a file which is always in RAM memory.

The keyword of entry point was defined in order to designate a point in the file system hierarchy starting from which a distinct TagFS begins meaning that only for that part of the file system tags apply; for a file system multiple entry points can be declared. This allowed to keep the changes necessary for tagfs isolated so that the performance for the normal case is not affected. This way the only difference from a vanilla kernel is that we do a string comparison for each entry point dened.

B. Storage

The mapping between a file and its associated tags is presented in Figure 3. Each file contains a bitmask in which

[1] http://lunarfrog.com/
[2] http://www.pytagsfs.org/
[3] http://www.eswc2006.org/poster-papers/FP31-Schenk.pdf/

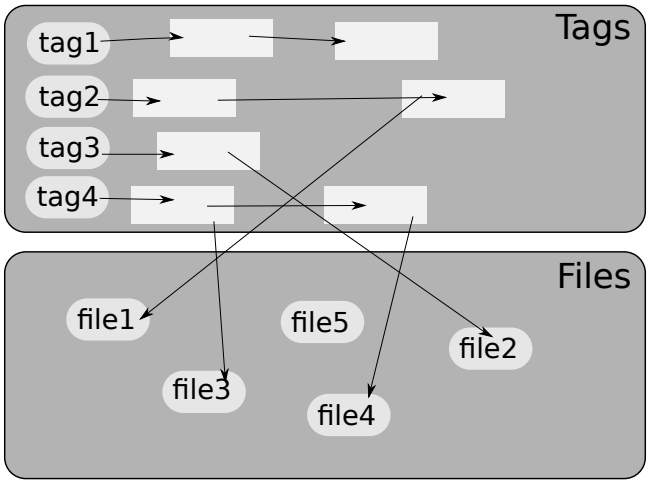

Fig. 3. Tag Storage

each bit represents a tag. If the bit is set to 1 than the file has the tag with that id and if not it does not. The bitmask is static so we will only be able to have only a limited number of tags. This defaults to 8192 tags but it is configurable through the kernel config file with impact on the memory and disk space used.

Each tag has a list of file pointers to files which have that tag. This is done so we do not have duplicated information and so that we will need to do updates in only one place.

C. Tag commands

The idea of tagging files is to be able to add a number of tags to a file but since the number of tags that will be associated to a file is not known beforehand we establish a convention that a filename will be separated from its associated tags by ":" whenever a command that envolves tags is issued. Also, one tag is separated by another tag by ":".

The behaviour of *ls* command was changed so that when issued with an argument beginning with ":" it lists all the files that are tagged with the given words.

An existent file can be assigned tags by issueing the *tag* command with *-a* parameter followed by the filename and the tags that one wants to attach to that file. There are two constraints that one has to take into account when wanting to add tags to a file. One is that the implementation of TagFS limits the maximum number of tags that can be assigned to a file to 256 and the second one is a limitation imposed by the kernel implementation and it reffers to the fact that the total length of filename, tags and separator must be less or equal to the value of MAX_PATH_LENGTH(256).

Tags can be removed from a file with *tag -d filename:tag[:tag*]* command taking into consideration the second constraint stated above.

The output of *tag -l filename* command is the list of tags associated to a file at a given time.

TagFS permits the creation of entry points in the file system which indicate that starting from that point down the hierarchy tags may be used. This was introduced in order to reduce the overhead for file systems where the user does not require to use tags, limiting it to a couple of comparisons. An entry point can be created using *tag -c* command meaning that the current working directory is a new TagFS entry point.

The available commands as well as a short description is listed in Table1.

Command	Params	Args	Description
ls	-	:tag[:tag]*	List files having the specified tags
tag	-c	-	Create a new entry point for TagFS
tag	-a	filename:tag[:tag]*	Add tags to file
tag	-d	filename:tag[:tag]*	Remove tags from filename
tag	-l	filename	List all the tags for filename

TABLE I
TAGFS IMPLEMENTED COMMANDS

D. Implementation details

1) Metadata structure:

We hold two hashtables in the memory. One of them keeps the tags from the system and the other hold the files. Structures are added in this hashtables when we add tags to a file. If the file is not tracked, we create a new entry for it in the hashtable. The same is done for each tag. Each tag from the tag hashtable holds a list of file pointers, each pointing to a specific file. This way we dont have to duplicate the information about each file. To associate a tag to a file, each file will have a bitvector with enough space for each tag.

2) VFS Hooks:

The VFS hooks allow TagFS to break out of the normal flow of the kernel and performs certain verifications in order to determine if a particular file should or not be treated as a tagable file and afterwards, if necessary, performing the desired changes. These will usually strip the tags from the filename so that normal operations will work as expected(E.g executing ls dir:tag would try to open the file dir:tag but the file does not exist so it would fail. Because of this, it is necessary to strip the tags). After this they will continue to do operation specic things.

To determine where we would need to insert our hooks, we did a strace on a command and checked what syscalls it makes. For example for a "ls dir" command:

3) Userspace application:

The application in userspace adds the tag layer to the normal file operations using the tag command. This command is a normal user space application that can be called by the user. This way the user can add and remove tags from a file, and also list the current tags. The application allows creating an entry point in the file system in a similar way. Tagging application works in user space and calls the specic api that further sends the requests to the kernel. The adding and removing of the tags keep the specified convention, using : as delimitator. The tag listing keeps the same format as well. The tag user application

Syscall	Action
fstat64(path)	Remove tags from path
open(path)	Remove tags from path and make association between <process, fd> and tags
getdents64(fd)	Get the tags associated with the fd and search the storage. Store the results in the dents structure passed from userspace.
close(fd)	Remove the association between fd and tags

TABLE II
LS SYSCALLS

uses fcntl system call in order to send the operation type and required arguments. New command types have been defined for fcntl for TagFS operations. From fcntl syscall other kernel-level tag specific functions are called for adding tags, deleting tags, searching and creating entry points.

IV. Experimental results

Because of the way TagFS is implemented there will be two major testing scenarios. The first scenario will consist of testing the UNIX-based commands, while the second one will test the tag manipulation commands. There may be an extra scenario for performance testing, but this is not the scope of this article as the main goal of TagFS is to prove that a tag based file system is implementable and it is more intuitive than the hierarchical one. The testing process will be concluded by executing a series of commands, noting the output of the commands and comparing it with the expected results.
The test platform is represented by Fedora 14 OS with a 2.6.35.10 kernel compiled with the changes for TagFS. At the moment the *ls* command is the only UNIX-based command hacked to allow the use of tags and three TagFS commands (tag -c tag -a, tag -d) are fully implemented and tested. We started with *ls* because it is the command with the most importance, the main use of tags being to search by them, and we did not have enough time to implement the other commands.

A first test scenario was to add tags to a file and browse the file based on one or more of its associated tags

```
$ cd ~/tagFS
$ ls
tag_test  untagged
$ tag -c
$ tag -a tag_test:soa:tag:fs
$ ls :soa
tag_test
$ ls :fs:soa:tag
tag_test
$ tag -d tag_test:soa
$ ls :fs:soa:tag
$
```

A second test scenario is having more tagged files and to browse by various combinations of these tags.

```
$ ls
tag_test  untagged
$ tag -a untagged:soa:fs:test
$ tag -a tag_test:second:soa:test
$ ls :soa:test
tag_test  untagged
```

V. Conclusion

There are still a couple of common UNIX commands whose behaviour we want to modify in order to issue them with tags. Also we would like to add a tag-specific command for listing the tags associated with a file.

Command	Description	VFS function
touch	Add tags to a file at creation time	do_sys_open
mv	The new file keeps tag information	rename
cp	The new file copies tag information from the old one	unlink
tag -l	TagFS new command for listing tags associated to a file	-

TABLE III
LS SYSCALLS

Possible future work
In a pure tag file system, the disk mechanism could be improved in the following way: We know that tags can be added to some files, we have no hierarchical structure of the files. This way we can find blocks of files based on tags which could reduce disk fragmentation. Clustering tag data can give insight on how much space there is required of a certain tag type files and how accessible this should be to the user. This could lower the external fragmentation of the disk if properly used. However more tests should be done regarding this problem.

Given the fact that Linux implements Extended Attributes(also called xattrs) which are name/value pairs associated with files as an extension to normal inode-based attributes, tags could be inserted in xattrs and could be easily displayed of graphical file browsers.

The current implementation, where we add hooks in the kernel syscall so that existing applications can use TagFS, can be augmented by adding new syscalls or fcntl options so that new application can use the system better.

REFERENCES

[1] Stephan Bloehdorn and Max Volkel,Tagfs-tag semantics for hierarchical file systems,

[2] Yuan-Liang Tai1, Shang-Rong Tsai, Guang-Hung Huang, Chia-Ming Lee, Lian-Jou Tsai, Kuo-Feng Ssu and Shou-Jen Wey, A Label-Based File System

[3] Nepomuk-KDE http://nepomuk.kde.org/

[4] TaggedFron http://lunarfrog.com/

[5] pyTAGSfs http://www.pytagsfs.org/

[6] TagFS: Bringing Semantic Metadata to the Filesystem http://www.eswc2006.org/poster-papers/FP31-Schenk.pdf/