EFTAS

Enhanced File Tagging System

Anna Pietzak
anna.pietzak@unibas.com

Timothy Roberts
t.roberts@stud.unibas.ch

Opareting Systems Course
University of Basel
Spring Semester 2024
12.06.2024

Contents

1 Introduction

This project was created in the context of the Operating Systems Course at the University of Basel. Our project takes up the problem of file organisation when one only has one device / one filesystem to store different kind of data. A person might want to save many differnet things: private holiday photographs, pictures of experiments or university assignments and evidence of a scratch on the company's car for their insurance. On common OSs all these files would be mixed together and placed into an "images" folder by default. This is where our project can shine. It allows users to further differentiate the pictures from one another by tagging them with their purpose. Such tags could be "private", "education" or "work".

Our tagging system EFTAS (Enhanced File Tagging System) provides the user with further options of sorting and categorizing files and keeping multiple areas of work separate, an interface to add and remove tags from files, to list all files with a certain tag and controls the names of the tags to ensure consistency within the used tags.

2 Background

As file sorting and tagging is not a new desire, there are already multiple approaches and solutions to this problem. For example MacOS provides natively colored tags for files and folders since the release of system 7 [1]. There are also third party applications like JabRef, which provide functionality to sort bibliographic references after groups and tags [2] or Microsoft outlook, which allows users to sort E-Mails into colorful categories [3]. To our advantage, such functionality is also partially available in C and Linux.

2.1 Base of EFTAS

From the start we decided to implement our project in C and use the existing file system of Ubuntu 22.04.03/04 as a reference as to what a "default" file system can and cannot do and how we can add to it. We have considered multiple ways of implementing our solution. One idea was to use the source code of the existing open source ext4 filesystem [4], as it is the default for most Linux distributions [5] (and already used by ubuntu), and modify it to accommodate our needs. However, the source code seemed rather large and we worried that it would take a big part of our time to get accustomed to the source code before we can even start prototyping a solution. So, we decided against it. We also looked into writing our own simple filesystem, which could then be mounted to the existing filesystem on Ubuntu. However, as we were only aiming for a filesystem extension, this solution seemed also more time consuming and not really necessary.

We discovered that Linux and C respectively already provide methods to set custom tags for files and get the tag of a file, when given the filepath, called setfattr and getfattr [6] and the respective system calls `setxattr` and `getxattr` [7].

3 Methodology

3.1 Functionality

EFTAS provides the user with an easy way to manage file tags. It ensures that the tags are of valid size (255 characters) and content (as in that the string only contains alphanumeric characters). EFTAS keeps track of all tags, that are permitted to be added to files. This way, users cannot set misspelled tags or use words with similar meaning to refer to the same category of files. In addition, users can display all available tags and add new tags to that list and of course they can add and remove tags to and from files.

3.2 Development

We used a GitHub repository to work on our code, which provides version control functionality [8]. This helps in developing a project in a team and made it a lot easier to test our code in a virtual machine, since we did not need to set up shared folders or copy our code to the virtual machine but

could just fork the project onto the virtual machine instead. This allowed development on the VM and the host machine simultaneously. EFTAS was developed, run and tested on Ubuntu version 22.04.03 and 22.04.04.

4 Results

4.1 Architecture

The figure 1 shows our code segments and how they work together. The labelled rectangles represent our different C files and bash scripts, and the blue labels show which console commands calls which modules. We utilize one central bash script (`eftas.sh`) to call various C scripts that performs the desired action. For example the command call "help": With `eftas help` we first call the central `eftas.sh` script, which in turn calls the `open_help.sh` script. This then displays the `help.md` file on the console for the user to read.

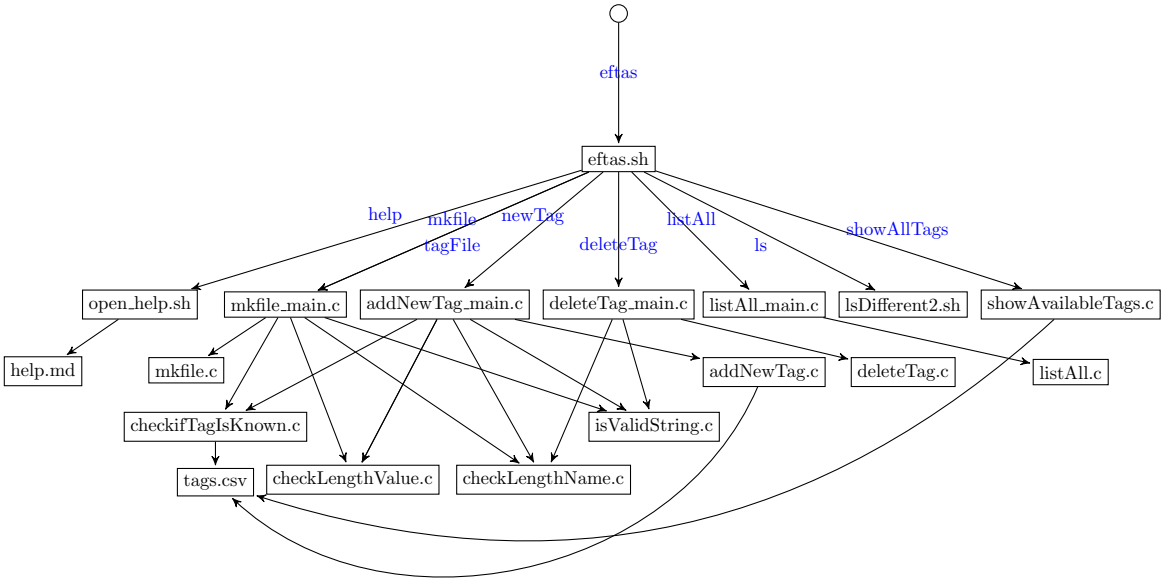

Figure 1: EFTAS architecture

To allow users to call EFTAS like any other terminal command, our central script needs to be in `/usr/local/bin/` [9]. Thus all our executable scripts have to have a clearly determined path. Both of those problems are solved by our setup script (`setup_eftas.sh`). This needs to be executed with root rights so that everything gets set up properly: First we need to download the `attr` package to call setfattr and getfattr. Then we make all other bash scripts executable, create a new hidden folder `/home/$USER/.eftas`, compile and link all C files according to the above dependencies and copy them into the new folder.

To compile and link our files we use a Makefile [1]. The resulting executables are copied into the new `.eftas` folder. This provides two advantages: Firstly, we can hardcode the path to the executable files into the EFTAS script. Secondly, they can only be used by the user that set up EFTAS, since the folder was created under their home directory. Other users cannot access the needed executables. So even if they were able to call eftas, it would not alter their files in any way.

If the user wishes to remove the executables and the EFTAS command again, they are provided with the `remove_eftas.sh` script. This deletes all copied files and returns the download folder to the origin. But it doesn't touch the initial download folder, thus can EFTAS always be set up again by running `setup_eftas.sh`.

[1] We utilized ChatGPT (`https://openai.com/chatgpt/`) to generate the basic structure of the Makefile we needed and then built off of it, making sure all dependencies are considered.

4.2 User Interface

The final version of EFTAS provides the following interface to the user:

- `help`
  Functionality for user
  This command displays a manual page in the console, which greets users with an EFTAS banner [2] and explains all possible arguments that can be used.
  Implementation
  This is accomplished by opening a markdown file through the `open_help.sh` script with the `less` command.

- `mkfile <tag_name> <file_path>`
  Functionality for user
  A file is created with the specified tag_name added. If the file already exists, then it will just add the tag to that file. Users can only add tags that are known to the filesystem.
  Implementation
  The command calls the `mkfile_main` executable where all arguments are checked for correct length and content. This means, tags may only contain letters and numbers. Then it is checked whether this tag is already known to the system (as in saved in our csv file) or not. If everything passes, then a file is created with `open()` and the tag is set.

- `tagFile <tag_name> <file_path>`
  Functionality for user
  The specified file is tagged with the specified name.
  Implementation
  This is an alias for mkfile, making it more intuitive for the user, if they want to tag an existing file. It can be used in the same way, since the `open()` command is used. Thus will it either create a new file if it doesn't exist, or open the existing file.

- `ls <ls,-a,tag_name>`
  Functionality for user
  Lists files in different ways, depending on the argument given.
  - `ls` shows all files in the current directory
  - `help` shows all possible arguments and an explanation
  - `-a` shows all the files, including the ones that are hidden in the current directory
  - `tag_name` will show the files with that specific tag_name from the current directory and all subdirectories

  Implementation
  A bash script with a switch case is used to differentiate the arguments. `ls` and `-a` call `ls` and `ls -a` respectively. The `help` command prints out the usage. And if a tagname is given all files are checked, if they possess the desired tag and are then displayed to the user if they have it.

- `deleteTag <tag_name> <file_path>`
  Functionality for user
  Deletes the specified tag from the specified file.
  Implementation
  First it is checked if the specified tag is assigned to the file and if yes, it is removed with `removexattr` [10].

[2] The banner and the cat in the `help.md` file were created by a large language Model via LMSYS (https://chat.lmsys.org/) so that users have something nice to look at.

- `listAll <file_path>`
  Functionality for user
  Lists all tags that are assigned to the specified file.
  Implementation
  First it is checked with `listxattr` [11] how many tags are assigned to the file (if any). Then we loop [3] over all tags and print them out one after another.

- `newTag <tag_name>`
  Functionality for user
  Add a new tag to the known tags in our tags.csv file, so it can be used for a file.
  Implementation
  First the new tag is checked for allowed content, length and if it is not already known. If all checks pass, the new tag is appended to the end of the csv file.

- `showAllTags`
  Functionality for user
  Shows all tags, that are allowed to be added to files.
  Implementation
  The csv with all known files is opened and printed onto the console line by line, until the end is reached.

5 Discussion

EFTAS provides a reliable way to add tags to files. By limiting the allowed tag-names, names first have to be specified before they can be used, avoiding the risk of accidental misspelling and synonyms being used for files that should have the same tag. Files can be created with tags or tags can be added to existing files. And EFTAS can list all files that have a certain tag in the current directory and all subdirectories, giving the user two independent ways of grouping files within a folder.

If one were to develop EFTAS further, it could grow into a bigger application by adding more features such as deleting tags from the EFTAS known tags. Deleting a tag would require being deleted from the systems `tags.csv` file and from all files it was assigned to, so that we can ensure consistency between the "known" tags and the used ones. Other additions could be automated file modifications based on the given tags, for example files with the private tag could autmatically be moved to hidden folders, or have their file permission changed to prohibit read-access. Or a graphical user interface, where tagged files would be visually marked with a color or a representative symbol. That way users, that are not familiar with the command line, could identify tagged files very easy by their visuals.

6 Lessons Learned

We are very happy with our final project and our teamwork. We made sure to inform our group partner of any major changes / additions to the project, we would regularly check in on the other member and discuss what is left to do and in what time frame we want it done.

The parts that we could each improve on lie more within the workflow itself. Meaning when we are effectively writing the code. When writing the report, planning the presentation, trying to explain a solution to our team member, or simply trying to remember what we coded the day before, we often had to retrace our steps and re-search solutions that we already found once. Changes, that would improve these points, would be documenting our research whilst programming. So when we were able to solve an issue that we were stuck on for minutes or hours, we should note down the steps to the solution and any links or information that solved the initial issue. That way we are way faster in re-accessing and sharing that knowledge later on.

[3] the basic structure of the while loop was generated with the use of ChatGPT https://openai.com/chatgpt/

References

[1] Poole, Lon. *Macworld-Handbuch zu System 7* St. Gallen: Midas Verl., 1992. Print.

[2] MLEP, *Organizing your database with JabRef*, https://docs.jabref.org/finding-sorting-and-cleaning-entries accessed on 11.06.2024

[3] Microsoft, *Set categories, flags, or reminders - Microsoft Support*, https://support.microsoft.com/en-au/office/set-categories-flags-or-reminders-a894348d-b308-4185-840f-aff63063d076, accessed on 11.06.2024

[4] Linus Torvalds, *linux/fs/ext4* https://github.com/torvalds/linux/tree/master/fs/ext4, accessed on 11.06.2024

[5] yuvrajjoshi31, *Linux File System*, https://www.geeksforgeeks.org/linux-file-system/, accessed on 11.06.2024

[6] Michael Kerrisk, *xattr(7) — Linux manual page*, https://man7.org/linux/man-pages/man7/xattr.7.html, accessed on 09.06.2024

[7] Michael Kerrisk, *setxattr(2) — Linux manual page*, https://man7.org/linux/man-pages/man2/setxattr.2.html, accessed on 11.06.2024

[8] GitHub Docs *About Git*, https://docs.github.com/en/get-started/using-git/about-git, accessed on 11.06.2024

[9] Martin Ueding, *Making sh script into program*, https://unix.stackexchange.com/questions/313610/making-sh-script-into-program accessed on 11.06.2024

[10] Michael Kerrisk, *removexattr(2) — Linux manual page*, https://man7.org/linux/man-pages/man2/removexattr.2.html, accessed on 11.06.2024

[11] Michael Kerrisk, *listxattr(2) — Linux manual page*, https://man7.org/linux/man-pages/man2/listxattr.2.html, accessed on 11.06.2024

[12] *CUnit - A Unit Testing Framework for C*, https://cunit.sourceforge.net/, accessed on 11.06.2024

A Additional Results

A.1 Example usage

The figures 2 through 7 show an example of how EFTAS can be used.

```
anna@Ubuntu-22:~/Desktop/OS-Project_File-System/download_eftas$ sh setup_eftas.sh
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
attr is already the newest version (1:2.5.1-1build1).
The following packages were automatically installed and are no longer required:
  libwpe-1.0-1 libwpebackend-fdo-1.0-1
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 90 not upgraded.
gcc -Wall -Wextra -std=c99 -c mkfile_main.c
gcc -Wall -Wextra -std=c99 -c mkfile.c
gcc -Wall -Wextra -std=c99 -c isValidString.c
gcc -Wall -Wextra -std=c99 -c checkLengthName.c
gcc -Wall -Wextra -std=c99 -c checkLengthValue.c
gcc -Wall -Wextra -std=c99 -c checkFilePath.c
gcc -Wall -Wextra -std=c99 -c checkIfTagIsKnown.c
gcc mkfile_main.o mkfile.o isValidString.o checkLengthName.o checkLengthValue.o checkFilePath.o checkIfTagIsKnown.o -o m
kfile_main
gcc -Wall -Wextra -std=c99 -c deleteTag_main.c
gcc -Wall -Wextra -std=c99 -c deleteTag.c
gcc deleteTag_main.o deleteTag.o isValidString.o checkLengthName.o -o deleteTag_main
gcc -Wall -Wextra -std=c99 -c listAll_main.c
gcc -Wall -Wextra -std=c99 -c listAll.c
gcc listAll_main.o listAll.o isValidString.o -o listAll_main
gcc -Wall -Wextra -std=c99 -c addNewTag_main.c
gcc -Wall -Wextra -std=c99 -c addNewTag.c
gcc addNewTag_main.o addNewTag.o isValidString.o checkLengthName.o checkLengthValue.o checkIfTagIsKnown.o -o newTag_main
gcc -Wall -Wextra -std=c99 -c showAvailableTags.c
gcc showAvailableTags.o -o showAvailableTags
Successfully compiled all files, eftas is ready for use now.
Call our extension with 'eftas' to get started.
anna@Ubuntu-22:~/Desktop/OS-Project_File-System/download_eftas$
```

Figure 2: A successful run of `setup_eftas.sh`

```
anna@Ubuntu-22:~/Desktop/OS-Project_File-System/download_eftas$ sh remove_eftas.sh
[sudo] password for anna:
rm -f *.o mkfile_main deleteTag_main listAll_main newTag_main showAvailableTags
Removed all compiled files, restored download folder to origin.
anna@Ubuntu-22:~/Desktop/OS-Project_File-System/download_eftas$
```

Figure 3: A successful run of `remove_eftas.sh`

A.2 Testing

To speed up our development and to ensure that we always got the desired behaviour, we added some test programmes for our main parts using CUnit [12]. We wrote tests dedicated to the following functionalities: `mkfile, listAll, is_valid_string, deleteTag, check_tag_value & check_tag_name` [4]. All functions are tested with valid and invalid values. The tests can be run by first running the Makefile and then calling the `test_suite`. This creates a suite for each test-file and runs the corresponding tests in that suite, see figure 8.

[4] the basic structure test_mkfile.c and test_suite.c was generated with the use of ChatGPT https://openai.com/chatgpt/

```
anna@Ubuntu-22:~/Desktop$ eftas
Usage: /usr/local/bin/eftas {help|mkfile|tagFile|ls|deleteTag|listAll|newTag|showAllTags}
anna@Ubuntu-22:~/Desktop$ eftas showAllTags
Available tags:
test
work
education
private
anna@Ubuntu-22:~/Desktop$ eftas mkfile education report.pdf
education
Tag education set successfully for file report.pdf with name education
anna@Ubuntu-22:~/Desktop$ eftas mkfile work application.txt
work
Tag work set successfully for file application.txt with name work
anna@Ubuntu-22:~/Desktop$ eftas tagFile education application.txt
education
Tag education set successfully for file application.txt with name education
anna@Ubuntu-22:~/Desktop$ eftas ls work
# file: application.txt
user.work="work"

anna@Ubuntu-22:~/Desktop$ eftas ls education
# file: report.pdf
user.education="education"

# file: application.txt
user.education="education"

anna@Ubuntu-22:~/Desktop$ eftas listAll application.txt
Extended attributes for application.txt:
user.work = "work"
user.education = "education"
anna@Ubuntu-22:~/Desktop$
```

Figure 4: Two files are created, one of them has two different tags. When searching all "work" related files, only one shows up, when searching for "education" related files, both show up.

```
anna@Ubuntu-22:~/Desktop$ eftas showAllTags
Available tags:
test
work
education
private
anna@Ubuntu-22:~/Desktop$ eftas newTag example
example
Add tag example to the system
Tag example is added to system.
Tag example added successfully
anna@Ubuntu-22:~/Desktop$ eftas showAllTags
Available tags:
test
work
education
private
example
anna@Ubuntu-22:~/Desktop$ eftas mkfile example example.txt
example
Tag example set successfully for file example.txt with name example
anna@Ubuntu-22:~/Desktop$ eftas ls example
# file: example.txt
user.example="example"

anna@Ubuntu-22:~/Desktop$
```

Figure 5: A new tag is added to the known files and applied to a file.

```
anna@Ubuntu-22:~/Desktop$ eftas listAll application.txt
Extended attributes for application.txt:
user.work = "work"
user.education = "education"
anna@Ubuntu-22:~/Desktop$ eftas deleteTag education application.txt
Tag education removed successfully from file application.txt
anna@Ubuntu-22:~/Desktop$ eftas listAll application.txt
Extended attributes for application.txt:
user.work = "work"
anna@Ubuntu-22:~/Desktop$ eftas deleteTag work application.txt
Tag work removed successfully from file application.txt
anna@Ubuntu-22:~/Desktop$ eftas listAll application.txt
File application.txt doesn't have any tags attached
anna@Ubuntu-22:~/Desktop$
```

Figure 6: All tags are deleted from a file until there are no more.

```
anna@Ubuntu-22:~/Desktop$ mkdir folder
anna@Ubuntu-22:~/Desktop$ mkdir folder/folder2
anna@Ubuntu-22:~/Desktop$ mkdir folder/folder2/folder3
anna@Ubuntu-22:~/Desktop$ eftas mkfile test topLayer.txt
test
Tag test set successfully for file topLayer.txt with name test
anna@Ubuntu-22:~/Desktop$ eftas kfile test folder/secondLayer.txt
Usage: /usr/local/bin/eftas {help|mkfile|tagFile|ls|deleteTag|listAll|newTag|showAllTags}
anna@Ubuntu-22:~/Desktop$ eftas mkfile test folder/secondLayer.txt
test
Tag test set successfully for file folder/secondLayer.txt with name test
anna@Ubuntu-22:~/Desktop$ eftas mkfile test folder/folder2/thridLayer.txt
test
Tag test set successfully for file folder/folder2/thridLayer.txt with name test
anna@Ubuntu-22:~/Desktop$ eftas mkfile test folder/folder2/folder3/finalLayer.txt
test
Tag test set successfully for file folder/folder2/folder3/finalLayer.txt with name test
anna@Ubuntu-22:~/Desktop$ eftas ls test
# file: topLayer.txt
user.test="test"

# file: folder/secondLayer.txt
user.test="test"

# file: folder/folder2/folder3/finalLayer.txt
user.test="test"

# file: folder/folder2/thridLayer.txt
user.test="test"

anna@Ubuntu-22:~/Desktop$
```

Figure 7: All subdirectories are searched by `eftas ls test` and all files with the test tag are displayed.

```
anna@Ubuntu-22:~/Desktop/OS-Project_File-System/src/tests$ ./test_suite
Initializing the CUnit test registry
Adding suite for is_valid_string
Adding suite for check_tag
Adding suite for test_mkfile
Adding suite for test_deleteTag
Adding suite for test_listAllRunning all tests


     CUnit - A unit testing framework for C - Version 2.1-3
     http://cunit.sourceforge.net/

Setting up the environment

Suite: Suite for is_valid_string
  Test: test_is_valid_string_valid ...passed
  Test: test_is_valid_string_invalid ...passed
  Test: test_is_valid_string_empty ...passed
  Test: test_is_valid_string_edge_cases ...passed
Removing everything that was created
Setting up the environment

Suite: Suite for check_tag
  Test: test_check_tag_name_valid ...passed
  Test: test_check_tag_name_too_long ...passed
  Test: test_check_tag_value_valid ...passed
  Test: test_check_tag_value_too_long ...passed
Removing everything that was created
Setting up the environment

Suite: Suite for test_mkfile
  Test: test_create_file_with_valid_attributes ...passed
  Test: test_missing_tagName ...Error: One of the three attributes is missing. Please check that all of them are there
passed
  Test: test_missing_tagValue ...Error: One of the three attributes is missing. Please check that all of them are there
passed
  Test: test_missing_filePath ...Error: One of the three attributes is missing. Please check that all of them are there
passed
Removing everything that was created
Setting up the environment

Suite: Suite for test_deleteTag
  Test: test_delete_tag_success ...passed
  Test: test_delete_tag_missing_tagName ...Error: One of the two attributes is missing. Please check that all of them ar
e there
passed
  Test: test_delete_tag_missing_filePath ...Error: One of the two attributes is missing. Please check that all of them a
re there
passed
  Test: test_delete_tag_non_existing_attribute ...This tag doesn't exist on testfile2.txt. Please choose another one to
delete
passed
Removing everything that was created
Setting up the environment

Suite: Suite for test_listAll
  Test: test_list_all_tags_success ...Extended attributes for testfile_with_attrs.txt:
user.validTagName = "validTagValue"
passed
  Test: test_list_all_tags_no_attributes ...File testfile_no_attrs.txt doesn't have any tags attached
passed
  Test: test_list_all_tags_non_existent_file ...Something went wrong when accessing the file: No such file or directory
passed
Removing everything that was created


Run Summary:    Type  Total    Ran Passed Failed Inactive
              suites      5      5    n/a      0        0
               tests     19     19     19      0        0
             asserts     26     26     26      0      n/a

Elapsed time =    0.001 seconds
Tests completed
anna@Ubuntu-22:~/Desktop/OS-Project_File-System/src/tests$
```

Figure 8: all tests run by test suit

B Declaration of Independent Authorship

We attest with our individual signatures that we have written this report independently and without outside help. We also attest that the information concerning the sources used in this work is true and complete in every respect. All sources that have been quoted or paraphrased have been marked accordingly.

Additionally, we affirm that any text passages written with the help of AI-supported technology are as such, including a refrence to the AI-supported programm used.

This report may be checked for plagiarism and use of AI-supported technology using the appropriate software. We understnad that unethical conduct may lead to a grade of 1 or "fail" or explusion from the study program.

Basel, 11.06.2024, Anna Pietzak

Basel, 11.06.2024, Timothy Roberts