Operating Systems (CS-323) Spring 2013
Assignment #3 File System

May 1, 2013

1 Objectives

1. Learn basics of VFS APIs, especially about FUSE, Filesystem in USErspace
2. Learn the structure of FAT (File Allocation Table), the most widely used file system to date

2 Introduction

In this assignment, you will develop a read-only file system driver for the FAT32 file system. FAT is very widely used these days and is present in most consumer electronics devices which offer storage. For example, USB keys or SD cards come with FAT32 format, and most MP3 players also use the FAT32 file system. During the exercise, your are encouraged to try your file system driver on your own FAT32 media.

There exist as many file system APIs (the so-called VFS API) as there are different operating system kernels. The FUSE (Filesystem in USErspace) API is another file system API, however it allows the development and execution of file system drivers in userland instead. The advantage of this approach is that less code is run in the kernel, which reduces the risk of crashes or exploits. Since FUSE presents a simple, abstracted file system interface, it is easily possible to use FUSE file system applications on different operating systems, such as Linux, FreeBSD, Mac OS, OpenSolaris or even Windows. Running in userland also enables the use of various language bindings, allowing the implementation of FUSE file systems in different progamming languages such as C, C++, Java, C#, Python, Perl, OCaml, etc.

3 Assignment

Your goal is to write a read-only file system driver for FAT32. Your implementation needs to be able to deal with long file names and has to support the `getattr`, `readdir`, and `read` FUSE file system operations. You need to return information as precise as possible. Your submission does not have to be able to run as multi-threaded application. You are *required* to use the skeleton implementation provided on the Moodle page. You must make sure that we can compile and run your submission.

4 The FAT file system

As primary source, please refer to the WikiPedia page on FAT, since it contains all necessary information to understand and implement a read-only FAT file system driver.

You should also have a look at existing FAT32 file systems with a hex editor (of course not limited to our sample file system), e.g.:

```
% hd testfs.fat | less
```

You can create new, empty FAT file systems using `dd` and `mkdosfs`:

```
% dd if=/dev/zero of=newfs.fat bs=1M count=5
5+0 records in
5+0 records out
5242880 bytes (5.2 MB) copied, 0.470387 s, 11.1 MB/s
% mkdosfs -F 32 newfs.fat
mkdosfs 3.0.7 (24 Dec 2009)
WARNING: Not enough clusters for a 32 bit FAT!
```

5 Running and testing

Development environment

To build the code, you will need the FUSE library in userland. On a Linux distribution that supports *deb* packages, such as Debian and Ubuntu, you can install it by the following command:

```
% sudo apt-get install libfuse2 libfuse-dev
```

Mounting

The supplied skeleton already performs basic FUSE argument parsing. To mount the file system, run the application with file system and destination mount point as arguments:

```
% mkdir dest
% ./vfat -s -f ./testfs.fat dest
% ls -l dest
```

The argument `-s` specifies single-threaded operation, and `-f` instructs FUSE to stay in the foreground and not to place itself in the background. These settings are useful for debugging.

Unmounting

In case your file system does not shut down properly, you might experience an error message, such as `ls: cannot access dest: Transport endpoint is not connected`. In this case you will have to unmount the file system manually:

```
% fusermount -u -z dest
```

Testing

We provide a compressed sample file system to test against: `testfs.fat`. Bear in mind that this file will expand to 500MB when extracted. You can test your implementation with standard `mount` by comparing the structure of the mounted sample file system.

You are also encouraged to use your own FAT32 devices as test file system. For example, you could try to list the contents of your digital camera, or listen to music stored on your MP3 player. As long as you open the device read-only, your program will not be able to destroy your data.

6 References

FUSE file system operations `http://fuse.sourceforge.net/doxygen/structfuse__operations.html`

FAT at WikiPedia `http://en.wikipedia.org/wiki/File_Allocation_Table`

FAT for microcontrollers `http://www.pjrc.com/tech/8051/ide/fat32.html`

7 Deliverables

Deliverables for this project are:

- Full source code for the assignment
- Short project report about your experiences on implementation and testing (no more than two pages)

The deadline for this assignment is 23:55 CET, May 24, 2013.