Project 4 Specification
Thumb Drive Forensics Utility

Assigned: April 17, 2019
Final Submission Deadline: May 1, 2019 at 11:59:59pm

Language Restrictions: C/C++, Java, or Python. Starter code for C and Python will be provided.

Additional Restrictions: system() and exec*() system calls may not be used.

Purpose

The purpose of this writing this utility is to familiarize you with three concepts: basic file-system design and implementation and file-system image testing. You will need to understand various aspects of the FAT32 file system such as cluster-based storage, FAT tables, sectors, and byte-ordering (endianness). You will also be introduced to data serialization (i.e., converting data structures into raw bytes for storage or network transmissions), and de-serialization (i.e., converting serialized bytes into data structures). Familiarity with these concepts is necessary for advanced file-system programming and computer storage forensics.

Problem Statement

You will design and implement a simple, user-space, shell-like diagnostic utility that is capable of interpreting a FAT32 file system image. The program must understand basic commands to manipulate the given file system image. The utility must not corrupt the file system image and should be robust. You may NOT reuse kernel file system code, and you may not copy code from other file system utilities. You may work in a group of two or by yourself.

Reminder: The code must be generated by you (not copied from another source or team). I will use a plagiarism detector when grading. You (or your team) must work alone. All code found to be plagiarized will receive a zero grade.

Project Tasks

You are tasked with writing a program that supports file system commands. For good modular coding design, please implement each command in a separate function. Please implement the following functionality:

- `info`

  Description: Some of this function is already made to give you a starting point. You should finish this function. It should out information about the following fields in both hex and base 10:
  - `BPB_BytesPerSec` (already done)
  - `BPB_SecPerClus` (already done)
  - `BPB_RsvdSecCnt` (already done)
  - `BPB_NumFATS` (already done)
  - `BPB_FATSz32` (already done)

  You should also compute and print out the following (need to do!) :
  - The byte (not sector) location of the root directory in hex (This is something need to compute, not look up.)

The following functionality has not been implemented and needs to be made by you.

- `stat <FILE_NAME/DIR_NAME>`

  Description: prints the size of the file or directory name, the attributes of the file or directory name, and the first cluster number of the file or directory name if it is in the present working directory. Return an error if `FILE_NAME/DIR_NAME` does not exist. (Note: The size of a directory will always be zero.)

- `cd <DIR_NAME>`

  Description: changes the present working directory to `DIR_NAME`. In other words, you need to seek to the directory block.

- `ls`

  Description: lists the contents of the current directory, including the "." (here) and ".." (up one directory) directories (if they are there). It **should not** list deleted files or system volume names.

- `read FILE_NAME POSITION NUM_BYTES`

  Description: reads from a file named `FILE_NAME`, starting at `POSITION`, and prints `NUM_BYTES`. Return an error when trying to read a directory.

- `volume`

  Description: prints the volume name of the file system image. If there is a volume name, it will be found in the root directory. If there is no volume name, it print the message "Error: volume name not found."

- `deleted`

  Description: Finds and prints all the deleted files in the current directory. (It should act just like the ls command, but only print the names of deleted files instead.)

- `quit`

  Description: Quits the utility.

Allowed Assumptions

- File and directory names will not contain spaces.
- Use the short directory names (you do not have to support long names)

Full Project Submission Procedure

You will need to submit the following files:

- fat32_reader.c or fat32_reader.py (source code)
- The filled-in Project4-README.docx. (If you are a Mac user, please keep the format as Word or else convert to a pdf. I have problems grading Pages documents on a non-Mac.)