Project 3 Specification
FAT32 File System Utility

Assigned: April 8, 2016
Halfway Submission Deadline: April 18, 2016 at 11:59:59pm
Final Submission Deadline: April 27, 2016 at 11:59:59pm

Language Restrictions: C/C++, Python, Java (Note: If your code is not in one of the languages listed, you must clear it with me within a week of receiving this specification.)

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

Purpose

The purpose of this project 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.

Problem Statement

You will design and implement a simple, user-space, shell-like 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.

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: prints out information about the following fields in both hex and base 10:
  - `BPB_BytesPerSec`
  - `BPB_SecPerClus`
  - `BPB_RsvdSecCnt`
  - `BPB_NumFATS`
  - `BPB_FATSz32`

- `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`.

- `ls <DIR_NAME>`

  Description: lists the contents of `DIR_NAME`, including the "." (here) and ".." (up one directory) directories. 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."

- `quit`

  Quit the utility.

Allowed Assumptions

- File and directory names will not contain spaces.
- You may assume `STRING` always begins with `OPEN_QUOTE` and ends with `CLOSE_QUOTE`.

Create a README file

Please create a README text file that contains the following:

- The names of all the members in your group
- A listing of all files/directories in your submission and a brief description of each
- Instructions for compiling your programs
- Instructions for running your programs/scripts
- Any challenges you encountered along the way
- Any sources you used to help you write your programs/scripts

Halfway Submission

Proper endian-ness functions for numbers must be used. Also, the following commands should be completed and work with the fat32.img provided to you:

- `info`

The following command(s) should be started or working:

- `ls`
- `stat`

Full Submission

All commands should work.

Grading

Please refer to the grading sheet.

Submission Procedure for both Submissions

You must zip up your README file and your source code, and submit the entire zip archive to eLearning by the due date. Code must compile to receive a grade.