Project 3 Specification
FAT32 File System Utility

Assigned: March 27, 2018
Due: April 20, 2018

Language Restrictions: C/C++, Python, Java (Note: C/C++ code MUST run on Ubuntu Linux 16.04)
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, file-system image testing, and data serialization/de-serialization. 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 mounting and un-mounting of file system images onto a running Linux system, 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.

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. 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 sizeof the fileor 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 ot exist. (Note: The size of a directory will always be zero.)

- `size <FILE_NAME>`

  Description: prints the size of file `FILE_NAME` in the present working directory. Log an error if `FILE_NAME` does not exist.

- `cd <DIR_NAME>`

  Description: changes the present working directory to `DIR_NAME`. Log an error if the directory does not exist. DIR_NAME may be "." (here) and ".." (up one directory). You don't have to handle a path longer than one directory.

- `ls <DIR_NAME>`

  Description: lists the contents of `DIR_NAME`, including "." and "..".

- `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 an unopened file.

- `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, print "Error: volume name not found."

- `quit`

  Quit the utility.

Allowed Assumptions

- File and directory names will not contain spaces.
- Because you are not writing any data, you can treat the FAT32 free list superficially.
- Your C/C++ program ***must*** work on Ubuntu Linux 16.04. If it does not, you can only earn a maximum of 50%.
- Java and Python programs must work with the latest Java 8/9 and Python 3.

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
- ls
- stat

Grading

Please refer to the grading sheet.

Submission Procedure

You must zip up your README file, your source code, and anything that is needed to compile it (such as a Makefile), and submit the entire zip to Canvas by the due date.