Programming Assignment 3: FAT32 File System

Due: April 3 2019 5:30PM

Description

This assignment will familiarize you with the FAT32 file system. You will become familiar with file allocation tables, endieness, as well as file access. You will implement a user space shell application that is capable of interpreting a FAT32 file system image. The utility must not corrupt the file system image and should be robust. No existing kernel code or any other FAT 32 utility code may be used in your program.

You may complete this assignment in groups of two or by yourself. If you wish to be in a group of two the group leader must email me your group member's names **by March 27, 2019**. Your email must have the subject line "3320 [Section #] Project 3 Group" where section number is 001 or 003. (003 is the 5:30pm class, 001 is 7:00pm)

The code you submit for this assignment will be verified against a database consisting of kernel source, github code, stackoverflow, previous student's submissions and other internet resources. Code that is not 100% your own code will result in a grade of 0 and referral to the Office of Student Conduct.

You can find a FAT32 file image, fat32.img, on blackboard under Assignment 3. Also under Assignment 3 you can find the fat 32 specification, fatspec.pdf. You will need this specification to interpret the file system image correctly.

Program Requirements

Your program shall be named mfs.c and shall be implemented in C or C++. You shall not use the system calls `system()` or any of the `exec` family of system calls.

Your program shall print out a prompt of mfs> when it is ready to accept input.

The following commands shall be supported:

`open <filename>`

This command shall open a fat32 image. Filenames of fat32 images shall not contain spaces and shall be limited to 100 characters.

If the file is not found your program shall output: “Error: File system image not found.”. If a file system is already opened then your program shall output: “Error: File system image already open.”.

`close`

This command shall close the fat32 image. If the file system is not currently open your program shall output: “Error: File system not open.” Any command issued after a `close`, except for `open`, shall result in “Error: File system image must be opened first.”

`info`

This command shall print out information about the file system in both hexadecimal and base 10:

- BPB_BytesPerSec
- BPB_SecPerClus
- BPB_RsvdSecCnt
- BPB_NumFATS
- BPB_FATSz32

`stat <filename> or <directory name>`

This command shall print the attributes and starting cluster number of the file or directory name. If the parameter is a directory name then the size shall be 0. If the file or directory does not exist then your program shall output “Error: File not found”.

`get <filename>`

This command shall retrieve the file from the FAT 32 image and place it in your current working directory. If the file or directory does not exist then your program shall output “Error: File not found”.

`put <filename>`

This command shall retrieve the file from the current working directory and place it in your FAT 32 image. If the file or directory does not exist then your program shall output “Error: File not found”.

`cd <directory>`

This command shall change the current working directory to the given directory. Your program shall support relative paths, e.g `cd ../name` and absolute paths.

`ls`

Lists the directory contents. Your program shall support listing "." and ".." . Your program shall not list deleted files or system volume names.

`read <filename> <position> <number of bytes>`

Reads from the given file at the position, in bytes, specified by the position parameter and output the number of bytes specified.

Grading

The assignment will be graded out of 100 points. Compiler warnings are there to tell you something is not correct. Pay attention to them and you will save yourself a lot of late nights debugging code. Code that does not compile will earn 0.

Your code will be compiled and graded on omega.uta.edu . Your code must compile as:

g++ mfs.c -o mfs

Category	Points
open / close	5
info	5
stat	10
cd	15
ls	15
get	15
put	15
read	15
No extra debug or extraneous output	5

How to submit homework

1. Submit your file mfs.c on blackboard.