CS 4414 Machine Problem 3 (MP3)

Due November 6, 2014

Purpose

The primary objective of this homework is to familiarize you with file system organization. Specifically you will learn how file allocation tables (FATs) are used to organize block clusters and free space, and how directories are implemented in order to provide a human-readable, hierarchical organizational structure.

Problem

You are to implement a command line tool that can inspect and manipulate FAT16 file system data. Your program will act like a very primitive shell/file-manager that bridges two file system namespaces: (1) the unified file system exposed by the shell that launched your program, and (2) the FAT16 file system that your file-manager will load.

Your program will have one command-line parameter: the path to a file containing raw device block data. Through the power of abstraction, the POSIX file API allows you to treat any number of raw data sources in the same manner. For example, you can simply `open()` a file descriptor to a raw USB flash drive (they are abstracted as SCSI block-storage devices, e.g., `/dev/sda`) in the same manner that you would a binary image file of a raw disk (e.g., `~/sampledisk.raw`).

Your program will load the specified file system and then repeatedly prompt the user for commands, presenting them with the current working directory as a prompt (initially the root of the file system), e.g.,

```
: / >
: /pictures >
: /pictures/sprbrk >
```

Commands are read from the prompt, one line at a time. Your program will implement the following five commands:

- `cd <directory>` Changes the current working directory of your loaded file system to the

relative path specified by <directory>

- ls [directory] Displays the files within <directory>, or the current working directory if no argument is given
- cpin <src> <dst> Copies the contents of a file specified by the path <src> from the "real" unified file system (exposed by the shell that launched your program) into the file system loaded by your program, linking it there as the path specified by <dst>
- cpout <src> <dst> Copies the contents of a file specified by the path <src> from the file system loaded by your program into the "real" unified file system (exposed by the shell that launched your program), linking it there as the path specified by <dst>
- exit Exits your program.

We will provide you with two resources:

- Microsoft Extensible Firmware Initiative Fat32 File System Specification. This whitepaper describes everything you need to know about the organization of FAT12/16/32 media. It provides code snippets that you should find very useful. Please read this document as soon as possible to begin familiarizing yourself with the details of FAT. You will not need to implement code for processing FAT12 (original FAT) data.
- A FAT binary disk image sampledisk.raw that you may use as a test subject.

Miscellaneous

- While not required, you should find that the additional work to support FAT32 is negligible; the benefit of doing so would be that your tool could interact with the majority of USB/flash drives, allowing you to experiment with your own media.
- You must use Unix/Linux for this homework.
- Your implementation must be in C/C++ and use the POSIX APIs for manipulating file data. You must use the GNU C/C++ development tools (e.g., g++).
- You must submit your solution by uploading an archive (i.e., .tar) of your source file(s) to the course's Collab portal. Along with your sources (.c/.cpp/.h files), you must submit a Makefile that will allow us to

compile your solution, specifically to an executable named “fat”

- You must submit in class a printed writeup regarding your solution in accordance with the requirements set in the Beginning of Course Memo (BOCM).
- If you have any other questions about the homework please ask.