**CS 4414 Operating Systems – Fall 2014**

**Machine Problem 3**

Homework #3  
Duevyn Cooke

I did not complete 100% of the assignment, but my code implements majority of the required functionality. The cpin function is the only one the does not work.

**Problem Description**

The goal of this assignment is to implement a command line tool that can inspect and manipulate Fat16 file system data. The program will act as a shell that bridges two file system namespaces: the unified file system exposed by the shell that launched the program, and the FAT16 file system loaded by the program in question. The latter is given to the program by the user as the single command-line parameter. After the program loads the given FAT16 file-system, it presents the current working directory and repeatedly prompts the user for commands. The following commands are available to the user:

* 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” unifed file system (exposed by the shell that launched the 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 field 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 the program.

**Approach**

I decided to implement an object oriented solution to this problem but creating structures in C. The first step in doing so was creating a FATFileSystem structure tracks important information about the loaded file system such as BIOS Parameter Block (boot sector) values. This structure also maintains pointers to specific locations in the loaded file-system to manage navigation for reading and writing data.

The heart of implemented functions mentioned in the previous section is the listdir(FATFileSystem \*fat, unsigned int clnum, char \*namev[], unsigned char \*bytev[]) function. The four parameters serve in the following ways: fat points to the FATFileSystem structure in order to grant listdir access to necessary information; clnum is the cluster number; namev is an array of strings that stores the name of each entry read in the current cluster; bytev is an array of pointers to the 32-bit entry the corresponds to the strings in namev. The two vector parameters, namev and bytev are related through their index. The listdir function scans the contents of a directory and stores the information of each entry in the two arrays of pointers. My implementation of each command calls this function immediately, before proceeding in its own unique way. After completion, listdir returns an integer valuse representing the number of values stored in the given vectors.

The cd function is almost solved after calling listdir. If the desired directory name exists in namev, the function refers to the same index in bytev in order to determine the location to switch too. Once this is found, the cd function changes the current cluster pointer in \*fat. The uniqueness of the cd function comes from the requirement of updating the current path of the fat. The function concatenates the new directory name onto the current path except when changing to the parent directory. In this case, cd sets a pointer to the end of path and changes each letter to the value NULL until it reaches a “/” symbol.vLikewise, the ls function does little beyond the diry work performed by the listdir function. After calling listdir, ls simply loops through the namev vector and prints each value to the screen after making sure the corresponding value in bytev does not have a value in the hidden attribute slot.

In contrast, the file relocation functions, cpout and cpin, both require a significant amount of instruction beyond what is done by listdir; still, both begin by making a call to the essential listdir function. In particluar, cpout continues by initiating an array of unsigned short integers to store a list of cluster's occupied by the file sought to be moved. This array is then filled by setting the file offset to the the correct location in the file allocation table, and then looping through until it finds the value corresponding the last cluster. Next, each cluster value is read into a data array. A file is created on the host machine, and the contents of the data array is written to this file.

**Analysis & Conclusion**

The homework was designed for students to better understand file system organization. For me, this assignment did just that, and then some. I faced a few difficulties while working on this assignment. As this is my second time working on this assignment, the most notable hurdle arrose from a bit of carelessness. Instead of clearing my working directory of all images with the “rm \*.jpg” command, I deleted 100% of the work I'd completed by, instead, giving a command of “rm \* .jpg”; a single space after the start is the sole difference in the two commands. To prevent this from happening a second time, I periodically copied my work to a backup folder. Another major difficulty was found in the precision necessary to gather and manipulate data found in the FAT16 file. A single-byte error in addressing the file data can cause major problems in the program's execution while also proving extremely difficult to find. The last problem I encountered came from the similarity of the functionality required of the program. Each function needed to similarly loop through the directory in question and obtain essential information. While the similar steps could be accomplished individually by each command, the redundancy created by such a solution does not conform to the luck of redundancy practiced by good programmers. After considerable thought, this problem was solved by created the important listdir function.

**Pledge:** On my honor as a student, I have neither given nor received unauthorized aid on this assignment—Duevyn Cooke.