Mini File System

A COMP 304 Project by Alp Özaslan and Onur Eren Arpacı

In this project we implemented a file system API library which operates on virtual disks. The file system is based on File Allocation Table (FAT), and it does not support directories, i.e., all files are stored flat in the virtual disk. The virtual disk itself is actually a binary file.

First we implemented disk manipulation functions:

> **mini_fat_create(filename, block size, block count)** to create new filesystem
> The file system itself is a binary file. So when creating a new filesystem, it creates a binary file and it saves the metadata of the filesystem. It uses the helper function mini_fat_create_internal(filename, block_size, block_count) to manage the metadata in a more compact way.
>
> **mini_fat_save(fs)** to save the filesystem to real file on the disk.
> It saves only the metadata of the filesystem. First, the metadata of the file system is saved. Then, with the help of a loop, the metadata of the files inside is written to the real file.
>
> **mini_fat_load(filename)** to load the fs that is previously saved.
> Similar to saving, first the metadata of the file system is read. Metadata of the files in the file system is also read after they are converted to FAT_FILESYSTEM object for easier management. Returns the created FAT_FILESYSTEM object.

After we wrote the basic functions for the file system, we added the file manipulation functions using them:

> **mini_file_open(fs, filename, is_write)** to open the file
> If a file with the same name has been saved to the file system before, it finds its place in the file system and returns it, thanks to the helper function mini_file_find(fs, filename). If there is no previously saved file, it creates a new one. Returns the FAT_OPEN_FILE object, which holds the location of the file, where it was last left in the file, and the opening mode of the file.
>
> **mini_file_close(fs, open_file)** to close the file
> Closes an open file. Returns false if the file is not already open, otherwise returns true.
>
> **mini_file_delete(fs, filename)** to delete the file
> Deletes the given file. The deletion process is simply to convert the metadata of all the blocks the file is in to EMPTY_BLOCK. Returns false if the file does not exist or is open, otherwise returns true.
>
> **mini_file_seek(fs, open_file, offset, from start)** to change the position of the cursor in an opened file. In each file, information about where the cursor is at that moment is kept. This function moves the cursor forward from the last location of the cursor or from the beginning of the file, depending on the given parameter.
>
> **mini_file_write(fs, open_file, size, buffer)** to write buffer to the file
> Writes the buffer contents to the given file. If it reaches the end of the file, it stops, even if the buffer hasn't finished writing its contents. It returns the number of bytes written.

mini_file_read(fs, open_file, size, buffer) to read the file to the buffer
It reads the desired byte of data from the given file to the buffer. Even if it does not reach the desired size, it stops when it reaches the end of the file. Returns how many bytes have been read.

We used C file functions like fopen, fseek, fread etc. to manipulate the file. When opening a file we used the "rb+" option to open it as a binary file.

All parts of the code are working well.