**COMP 304 Project 3**

**Yakup Enes GÜVEN – Alp Doğukan Halitoğulları**

**Disk Manipulation**

**1. mini fat create(filename, block size, block count)** creates a new virtual disk

and stores it inside the filename.

I created virtual disk file which is exactly block count\*block size bytes and

reside in project directory immediately after creation.

The function returns a FAT FILESYSTEM object with its fields set properly.

**2. mini fat save(fs)** which completely saves a filesystem to the real file on the disk.

After the save operation, the virtual filesystem completely stored inside the

single file passed to create and can be loaded later.

**3. mini fat load(filename)** load a previously saved virtual disk filesystem from

a real file in project directory. The loaded virtual filesystem is identical

to the one that was saved.

**4. mini fat dump(fs)** which dump the metadata of the virtual filesystem and the

virtual files within it, sans the actual file data.

**File System Manipulation**

**1. mini file dump(fs, file)** provides a human-readable presentation of the file’s metadata,

including its filename, size, blocks used to store its data, etc.

**2. mini file open(fs, filename, is write)** will open a file for reading or writing.

Once a file is opened, a FAT OPEN FILE structure is returned as a file handle that can be

passed to other APIs that work with open files. This structure should at least include

whether the file was opened in read or write mode, and what the current cursor position

is inside the opened handle.

Note that a file can be opened multiple times for reading, but can only be opened once

for writing. The API should return NULL if it cannot open the file.

Also note that opening a non-existing file for writing will automatically create it.

**3. mini file size(fs, filename)** will return the size of a file as an integer (in bytes).

If the file does not exist, returns 0.

**4. mini file close(fs, open file)** will close an open file handle.

**5. mini file delete(fs, filename)** will delete a file from the virtual disk. If the file is

already open, it cannot be deleted. Returns true on success.

**6. mini file seek(fs, open file, offset, from start)** should change the position of

the cursor in an opened file. Note that each open file handle has its own separate cursor.

In relative mode (i.e., when from start is false), the cursor should be moved by offset

bytes, which can be positive or negative.

In absolute mode (when from start is true) the cursor should be set to the provided

offset.

The cursor cannot go beyond the beginning or end of the file. As such, the function

should return true on success, and false on failure.

**7. mini file write(fs, open file, size, buffer)** should write size bytes taken from

buffer to the open file.

The write API should automatically add new blocks to the file as it is writing into it.

It should also automatically update the file size as it writes data into it.

The API returns the number of bytes successfully written. This should always match

the size argument unless the virtual disk has no more blocks and is thus full.

**8. mini file read(fs, open file, size, buffer)** should read size bytes from open file

and put it into buffer.

The function should return the number of bytes successfully read, which only differs

from size if the end of file is reached.