CS 350 File Systems Project II

The second part of this project requires that you implement a simple file system. In particular, you are going to write the software which which will handle dynamic file management. This part of the project will require you to implement the class **Filesys** along with member functions. In the description below, FAT refers to the *File Allocation Table* and ROOT refers to the *Root Directory*. The interface for the class should include :

```
Class Filesys: public Sdisk
{
Public :
Filesys(string filename);
int fsclose();
int fssynch();
int newfile(string file);
int rmfile(string file);
int getfirstblock(string file);
int addblock(string file, string block);
int delblock(string file, int blocknumber);
int getblock(string file, int blocknumber, string& buffer);
int putblock(string file, int blocknumber, string buffer);
int nextblock(string file, int blocknumber);
Private :
int rootsize;           // maximum number of entries in ROOT
int fatsize;            // number of blocks occupied by FAT
vector<string> filename;   // filenames in ROOT
vector<int> firstblock; // firstblocks in ROOT
vector<int> fat;             // FAT
};
```

-possibly add in private.

An explanation of the member functions follows :

- **Filesys(string filename)**

  This constructor reads from the sdisk created with "filename" and either opens the existing file system on the disk ~~or creates one for an empty disk~~. Recall the sdisk is a file of characters which we will manipulate as a raw hard disk drive. This file is logically divided up into number_of_blocks many blocks where each block has block_size many characters. Information is first read from block 1 to determine if an existing file system is on the disk. If a filesystem exists, it is opened and made available. Otherwise, the file system is created.

  The module creates a file system on the sdisk by creating an intial FAT and ROOT. A file system on the disk will have the following segments:

Block 0	Reserved
Block 1	ROOT
Blocks 2 -> n+1	FAT
Blocks (n+2) -> (number_of_blocks-1)	Data Blocks

The file system consists of two primary data objects. The directory is a file that consists of information about files and sub-directories. The root directory contains a list of file (and directory) names along with a block number of the first block in the file (or directory). (Of course, other information about the file such as creation date, ownership, permissions, etc. may also be maintained.) ROOT (root directory) for the above example may look something like

file1	3
file2	7
file3	10

The FAT is an array of block numbers indexed one entry for every block. Every file in the file system is made up of blocks, and the component blocks are maintained as linked lists within the FAT. FAT[0], the entry for the first block of the FAT, is used as a pointer to the first free (unused) block in the file system. Consider the following FAT for a file system with 16 blocks.

Block 0	Reserved 6
Block 1	ROOT 0
Block 2	FAT 0
Block 3	4
Block 4	5
Block 5	0
Block 6	8
Block 7	9
Block 8	13
Block 9	0
Block 10	11

Block 11	12
Block 12	0
Block 13	14
Block 14	15
Block 15	0

In the example above, the FAT has 3 files. The free list of blocks begins at entry 0 and consists of blocks 6, 8, 13, 14, 15. Block 1 on the disk contains the FAT. Block 2 on the disk contains the root directory and is unused in the FAT. File 1 contains blocks 3, 4 and 5; File 2 contains blocks 7 and 9; File 3 contains blocks 10, 11, and 12. Note that a "0" denotes the end-of-file or "last block".

PROBLEM : What should the value of *FAT_size* be in terms of blocks if a file system is to be created on the disk? Assume that we use a decimal numbering system where every digit requires one byte of information and is in the set [0..9].

Both FAT and ROOT are stored in memory AND on the disk. Any changes made to either structure in memory must also be immediately written to the disk.

- **fssynch**

  This module writes FAT and ROOT to the sdisk. It should be used every time FAT and ROOT are modified.

- **fsclose**

  This module writes FAT and ROOT to the sdisk (closing the sdisk).

- **newfile(file)**

  This function adds an entry for the string *file* in ROOT with an initial first block of 0 (empty). It returns error codes of 1 if successful and 0 otherwise (no room or file already exists).

- **rmfile(file)** only remove empty.

  This function removes the entry *file* from ROOT if the file is empty (first block is 0). It returns error codes of 1 if successful and 0 otherwise (not empty or file does not exist).

- **getfirstblock(file)**

  This function returns the block number of the first block in file. It returns the error code of 0 if the file does not exist.

- **addblock(file,buffer)**

  This function adds a block of data stored in the string *buffer* to the end of file F and returns the block number. It returns error code of 1 if successful, 0 if the file does not exist, and returns -1 if there are no available blocks (file system is full!).

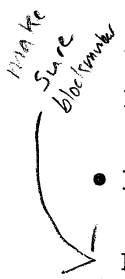
- **delblock(file,blocknumber)**

  The function removes block numbered blocknumber from file and returns an error code of 1 if successful and 0 otherwise.

- **getblock(file,blocknumber,buffer)**

  gets block numbered blocknumber from file and stores the data in the string buffer. It returns an error code of 1 if successful and 0 otherwise.

- **putblock(file,blocknumber,buffer)**



  writes the buffer to the block numbered blocknumber in file. It returns an appropriate error code.

- **nextblock(file,blocknumber)**

  returns the number of the block that follows blocknumber in file. It will return 0 if blocknumber is the last block and -1 if some other error has occurred (such as file is not in the root directory, or blocknumber is not a block in file.)

IMPLEMENTATION GUIDELINES : It is essential that your software satisfies the specifications. These will be the only functions (in your system) which physically access the sdisk.

1. You must add the *accessor functions* **getblocksize** and **getnumberofblocks** to the class **Sdisk**.
2. A test main program will be provided on the class website.