CSC 415-02

File System Design 1 (Milestone 0)

Team Name: Diligence

Team members: Zeel Hareshbhai Diyora, Mark Kim, Peter Truong, Chengkai Yang

-What your Volume Control Block looks like (what fields and information it might contain)

Here is what our Volume Control Block looks like:

```
struct VCB
{
    int num_blocks          // number of blocks in fs (4 bytes)
    int size_block          // size of each block in fs (4 bytes) – 512 bytes
    int free_space          // block number of the free_space (4 bytes)
    int root                // block number for root (4 bytes)
    long magic              // unique identifier for this volume (8 bytes)
    char name[40]           // name of volume (40 bytes)
} VCB;
```

-How you will track the free space

In order to track the free space, we are going to use FAT. The free_space entry of the VCB will reference the block number of the first free space block. The free space will be initialized by writing a reference to each subsequent block into the previous block and terminating the last block to 0xFFFFFFFE. Space will be allocated to a file by calculating the number of blocks necessary to contain the file, then assigning the block number of the first block of the free space

to the file. The free_space entry in the VCB will then be re-referenced to the block following the previously allocated space. If the file is deleted, the last block of the free space will be referenced to the first block of the file and the last block of the file will be referenced to 0xFFFFFFFE. If the file shrinks in size, the (n - <number of blocks removed>) block of the file will be referenced to 0xFFFFFFFE and the final block of the free space will be referenced to the freed space from the file shrink. If the file increases in size, the final block of the file will be referenced to the first block of the free space (the final block of the newly allocated space will be referenced 0xFFFFFFFE), and the free space reference in the VCB will be re-referenced to whatever block follows the increased file blocks.

-What your directory entry will look like (include a structure definition).

Our directory entry struct is outlined below:

```
struct d_entry
{
    unsigned int size;          // size of file/directory (4 bytes)
    unsigned int block;         // block number/location of file/directory (4 bytes)
    time_t created;             // time of file creation (4 bytes)
    time_t modified;            // time that file was last modified (4 bytes)
    time_t accessed;            // time that file was last accessed (4 bytes)
    char attr[1];               // file attribute flag: directory, file, etc. (1 byte)
    char name[43];              // name of file or directory (43 bytes)
} d_entry;
```

We will be using FAT to manage files as well. Like in the VCB, the block variable will hold the block number/location of the file. Each subsequent block will contain a reference to the next block until the final block, which will be referenced to 0xFFFFFFFE. Management of files and/or free space is outlined above.

-What metadata do you want to have in the file system

As shown above, the metadata that we will be including in our file system is as follows:

1. Size of each file
2. Block number/location of the file
3. Creation/modification/access information of the file
4. A flag of the file attributes (e.g. whether it is a directory or a file and possibly more flags depending on what we decide to implement in our file system)
5. The name of the file/directory.