LOUISIANA STATE UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

CSC 4103: Operating Systems
Fall 2017
Programming Assignment # 3: A Simple Filesystem
Prof. Golden G. Richard III
Due Date: 11/29/2017 @ class time
** TEAMS OF 1 or 2 STUDENTS ARE ALLOWED **
NO LATE SUBMISSIONS

The Mission

In this assignment you will implement a simple filesystem from scratch. Your filesystem will provide a flat namespace (e.g., there is only a single root directory and no subdirectories) and a set of file operations. You may implement either a FAT or *inode*-based block allocation strategy for files, but you must document which strategy is used. I will provide the implementation of a persistent "raw" software disk (available via `softwaredisk.h / softwaredisk.c`), which supports reads/writes of fixed-sized blocks (numbered `0..software_disk_size() - 1`). Your goal is to wrap a higher-level filesystem interface around my software disk. It is your responsibility to allocate blocks for file allocation, the directory structure, and file data. You may use any sensible technique for tracking free disk blocks.

Further requirements:

- You **must** provide a file system initialization program called `formatfs.c` which initializes your file system. Part of this initialization will including initializing the software disk via `init_software_disk()`, which destroys all existing data.
- Your filesystem **must** handle out of space errors (e.g., operations which overflow the capacity of the software disk) and set appropriate error conditions.
- Your filesystem **must** allow filenames of at least 255 characters in length, composed of printable ASCII characters.
- Your filesystem **must** provide the following filesystem interface. This is the interface expected by the sample applications—if your code deviates from this interface, it it is **wrong** and the application programs will break:

`filesystem.h:`

```
// main private file type
typedef FileInternals {
            // private—you implement this
} FileInternals;

// file type used by user code
typedef FileInternals* File;
```

What Do I Get?

An implementation of the software disk is provided. Do not implement this yourself and do not make modifications. You can obtain `softwaredisk.h`, `softwaredisk.c`, and `filesystem.h` from Moodle. The program `exercisesoftwaredisk.c` (available in the same place) tests the functionality of the software disk.

Submission/Grading

In addition to your implementation in (`filesystem.h`, `filesystem.c`, `formatfs.c`), you must submit a concise, typed design document that describes the physical layout of your filesystem on the software disk. This should include a description of which blocks are used for file allocation, how free blocks are tracked, how the directory structure is maintained, etc. You should also document any implementation-specific limits (such as limits on the size of filenames, maximum number of files, etc.). Please name your design document `filesystem_design.pdf` and include it with your submission to `classes.csc.lsu.edu`.

A good grade on this assignment depends on a proper design for your filesystem, your filesystem working flawlessly, and on high-quality, well-designed code.