ECE329 HW #3

In this assignment you will write the basics of a simple file system. The file system will be flat (all files in a single directory), and files will not (yet) contain any data. Your entire simulated file system will be stored in a single file on the hard disk. Augment your UNIX shell with the following commands:

- *ls*
  lists the files in the directory (single column is okay). If *–l* is given as an argument, then files are listed in long format (permissions, owner, size, last modified date, file name). For this assignment, the size of all files will be zero, and permissions will include two categories of people ("user" and "other") and two types of access ("read" and "write").
- *touch file*
  sets the modification time of file, or creates an empty file if none exists. For simplicity, file names can be limited to 14 characters (as in UNIX System V). The default permissions should be *rw*-- (i.e., read-write for "user", nothing for "other".
- *rm file*
  removes a file. Must have permission to do this (i.e., must be *root* or owner of file). If file is read-only, then should prompt user before removing.
- *chown file user*
  changes the owner of a file. Must be root to do this.
- *chmod [uo][+-][rw] file*
  changes permissions of a file. Must have permission to do this (i.e., must be *root* or owner of file).

As indicated above, the commands to modify a file should first check the owner and read/write permissions of the file to ensure that the current user has the appropriate access. If not, an error message should be printed.

Instead of using your own simulated hard disk, your code should interface with the SimulatedHardDisk provided by the instructor. Note that in this class the read/write lines do *not* go low automatically. As a result, it is your responsibility to set these lines to low yourself in the interrupt handler to prevent the hard disk from initiating another read/write.

In addition, you will need to modify *mkfs* to format the simulated hard disk. Your file system will be divided into blocks. For simplicity, set the block size to be the same as the sector size. The file system should consist, in order, of

- an unused boot block
- a super block that specifies the number of i-nodes in the list (i.e., the maximum number of files that the file system can handle). For simplicity, limit the number of i-nodes to the number that can fit into a single block.
- an i-node list (i.e., an array of i-nodes, one per file). Each i-node should contain a valid bit, the file name, owner, permissions, last modified date, and size, along

with an indicator of the location of the actual file data. If the valid bit is 0, then the other fields are ignored.
- file data (empty for now)

Note: To add a file to the project in VC++ 6.0, click on the FileView tab in the Workspace window, then right-click on the name of the project and select "Add Files to Project."

Separately, answer the following problems in Chapter 2 of the textbook (Tanenbaum, *Modern Operating Systems*, 3rd ed.): 23, 24, 25, 26, 28, 29, 33.