ECE329 HW #6

Part 1: Write a class called `BufferCache` that simulates a file system buffer cache. The interface to the class should be as follows:

```
int FindFreeBlock();            // return the index of a free block; if there
                                //     are no free blocks, then create one and
                                //     return its index
char* GetBlockData(int index);  // return a pointer to the data of a
                                //     specified block
void ReleaseBlock(int index);   // free block for use by someone else
```

The buffer cache should be implemented as an array or linked list of fixed-sized blocks (size: 1 kB each). The class should be written so that it is thread-safe.

Part 2: Write a simulated UNIX shell with the following command:

- *cp file1 file2*
  duplicates an actual file on the hard disk using either a relative or absolute path. Example: *cp a.txt b.txt* copies an actual file named *a.txt* on the Windows hard disk to an actual file named *b.txt*.

The copy command must exhibit the following behavior:

1. Files should not be copied directly. Rather, the entire source file should first be copied from the hard disk to the buffer cache; then the file should be copied from the buffer cache to the destination file on the hard disk.
2. If the user types the ampersand (&) at the end of the line (with a space before it), then the copy should proceed in the background. I.e., multiple copies should spawn multiple threads which run concurrently. Similar to UNIX, when the copy begins it should print a unique number identifying the job, and when the copy completes it should print 'Done' along with the identifying number. (For simplicity, the number can be a global counter that increments with each job.)

Note that this assignment does not depend at all on your simulated file system of assignments 2 and 3, and it only minimally builds upon assignment 1. Only the functionality mentioned above will be tested; no other UNIX commands need work in your shell.

Hint: When a thread is copying data to the buffer cache, it should store an array of the indices (or handles) of the blocks, as well as the total number of bytes in the file (in case the final block is not filled). Be sure to wrap all your calls to the BufferCache with mutexes to prevent race conditions from the threads.

Separately, provide a question (with solution) that you would include on the final exam if you were teaching the course. Your question should cover material that you believe is fundamental, important, and relevant.