CS 4/53201 # Project #3, Question #2 **Operating Systems**

Project #3 portion due via email by 11:59pm on Friday 11 December 1998

The Problems

2. Project #3 — due via email by 11:59pm on Friday 11 December 1998.
(This part counts as 2/3 of your Project #3 grade — 7.5% of your course grade)

The Nachos file system, as currently implemented, does not protect itself against concurrent access by multiple threads. If you think back to Question 4 on Project 2, allowing unsynchronized access to common data structures can lead to unacceptable results (in that case, two callers assigned to the same tech support person). Similar problems can occur in the Nachos file system when there is no synchronization between threads; for example, when creating files, two files could be assigned the same free sector on the disk.

Your job is to fix the Nachos file system so that two threads can use the file system concurrently. More specifically, you must allow multiple threads to have the same file open. In this situation, your file system should implement UNIX file system semantics as follows:

- atomic create — if two threads attempt to create a file of the same name at the same time, one should succeed, and the other should fail
- atomic writes — file writes should be atomic, meaning that if one thread has a file open for writing, and a second has the same file open for reading, the second thread will always be presented with a consistent view of the file. On each read, the returned data should either be the contents of the file before the write started, or after the write ended, but not what it looked like during the write.
- removal — if a thread has a file open, a Remove of that file must be postponed until the file is closed by all threads that have it open

This list is not comprehensive, as there may also be other items that should be fixed to have a file system that truly supports synchronized access, but the three items listed above are the only ones you have to handle for this project. (One hint — implementing these semantics will be much easier if you implement an open file table.)

After writing this code, in a file called **proj3.problem2**, do the following:

a. Include a description of what you modified and why, and how you modified it. Do a good job on this writeup, as it constitutes half of your grade on this question. I'm looking for several pages here, not several paragraphs.

b. Include a sample run to demonstrate that everything you implemented works. The code and this sample run constitute the other half of your grade on this question.

c. If you did not complete this problem, clearly describe what you have done, what is not working, and how you would go about finishing the problem if you had more time.

Submitting Your Project

When you finish, submit all files that you modified (including your **p3.problem2** file) to the TA for grading, in the same manner that you submitted your files for Project 2.