Assignment 7

CS 325

Extra credit, due May 4, 2012 at noon

Each problem is worth eight points. Show all work.

1. Assume a process has a fixed allocation of page frames. When a process references a page that is not in one of its frames, a replacement algorithm is invoked to select a page to replace. Consider the following variant of the LRU replacement strategy:

   ```
   if at least one page has not been modified
      replace the least recently used unmodified page;
   else
      replace the least recently used page /* all have been modified */
   ```

   This algorithm replaces unmodified pages in preference to modified ones.

   Give a reference string which illustrates that this replacement algorithm is not a stack algorithm. Mark modifying references with an asterisk to distinguish them from non-modifying references. Show the memory contents at the end of the reference for two values of m (the number of page frames) to illustrate the absence of memory inclusion. There are quite short reference strings (six references) which will do the trick.

2. Consider a demand-paged computer system which uses a paging drum, global LRU replacement, and an allocation policy which shares frames equally among processes (i.e., if there are m frames and n processes, then each process gets $\lfloor \frac{m}{n} \rfloor$ frames). The degree of multiprogramming is currently fixed at four. The system was recently measured to determine utilization of the CPU and the paging drum. Consider each of the following to be the results of the measurement. For each case, what is happening? Can the degree of multiprogramming be increased to increase the CPU utilization? Is the paging helping?

   (a) CPU utilization 13%; disk utilization 97%.

   (b) CPU utilization 87%; disk utilization 3%.

   (c) CPU utilization 13%; disk utilization 3%.

3. In a tree-structured file system such as Unix, permission bits are usually associated with each directory and file. Consider the following modification to the scheme. Associated with each directory is a *password* and a set of *global permissions*. For a user to access a directory which she does not own, the correct password must first be typed. Once this is performed, the user can manipulate the directory and files within that directory according to the global permissions READ, WRITE, and CREATE. If the global permission associated with the directory is READ, then the user can list the associated file names and read *any* file within the directory. If the permission is WRITE, then the user can list file names as well as read

and write any file. If the permission is CREATE, then the user can list file names and create new files within that directory, but *cannot* read or write any other files. The owner of the directory can perform any action on any file in the directory.

Consider a system where class accounts are organized so that the instructor and each student has a directory. A user logged onto the instructor account can access any files in the student accounts, despite the permissions associated with them. Among other things, this allows the instructor to run a grading program, executing each student's program against a common data file.

Given the protection scheme described above, describe how a class account would be organized. In particular, describe how the instructor can run the grading program without having to know the password associated with each student's directory.

4. An operating system only supports a single directory, but allows that directory to have arbitrarily many files with arbitrarily long file names. Explain how something approximating a hierarchical file system (including relative and absolute pathnames and the concept of a current working directory) can be created.