CSE 121 Winter 2006
Home Work 1

Due: Thursday, January 26 2006

1. **Operating System Review**

   (a) We know that the OS provides various abstractions for applications (processes, file system, virtual memory). What is the major advantage of providing these abstractions to application developers? What is the major disadvantage of implementing one generalized abstraction for all the available hardware (we have one interface to access various types of storage devices)?

   (b) There are a number of popular computing products that are currently used to run just one application (for example set-top-boxes like TiVo, cell phones, Xbox, etc.). Why might these applications' designers still want to implement them on top of an operating system rather than the bare hardware?

   (c) One of the original motivations for virtual memory was to provide a large address space to processes even though the underlying hardware did not have much physical memory (back in 80s PCs used to have few 100 KB of memory). Why do today's operating systems still provide the virtual memory abstraction even though the current hardware has huge amount of memory (now, it is very common to have several GB of main memory)?

2. **Fast File System**

   (a) The FFS paper describes a set of rules (on page 186 of the paper) to assign a block to new data added to a file. What are these rules are trying to achieve?

   (b) FFS has an entry size field in the directory file entry; why it is needed? Can it be eliminated?

   (c) In FFS, the directory file contains the inode number for each file and the inode data is stored separately. If inode data is included in the directory file, the file system performance might improve by reducing the number of disk accesses to access a file. Why did FFS designers separate them? (Think about converting names to inodes and implementing links.)

3. **Log Structured File System**

   (a) Why does the checkpoint region in LFS have to be in a fixed location?

   (b) Why did the greedy algorithm that cleans least utilized segments *not* perform as expected for the hot-and-cold data set?

   (c) LFS makes metadata updates asynchronous so that it buffers up data to be written to take advantage of sequential write bandwidth. Could we get similar performance from FFS if we made metadata updates asynchronous? Why or why not?