CSCI 144
Homework 1
Bee Cha

(4)

a. An operating system should allow time sharing with the processor. It shouldn't give the entire processor to each application/user because this can cause processor idle time when the I/O is being fetched or when accessing memory. In order to avoid CPU idle time, I believe the best method to perform multiple task is the round robin method. Each task is given the same amount of time to perform until it's finish. That way when a long task is given the 5 seconds to perform, the processor will jump to the next task and give it 5 seconds to perform because the next task may take less than 5 seconds to finish.

b. The operating system should allocate physical memory in blocks to avoid memory fragmentation. When there isn't enough physical memory to fit an application, the operating system give out some virtual memory. Or perhaps it can give it some temporary hard disk memory and then clear up the hard disk memory when the application is exited.

c. Disk allocation should be done by having an index that contains pointers that points to the physical disk address of each file. The first usuer should not get all the free space because chances are the user may not even the space that's given to him/her. There will be unused disk space and this is inefficient.

(6)

a. Communication between appplications through the file system is ok as long as the file system is set to read-only permission. Even then it is still a potential security risk.

b. Applications should NOT communicate between applications. The application may potentially access the physical memory location of the other application and modifying the memory values thus posing a security flaw.

c. First of all, memory shouldn't be shared between two applications. Second, if the memory is shared, you run into lethal memory access between the two applications. It's as if sharing the money in your bank with another person.

(9)

Designing a system will have *reliability.* The system will perform the way that it was designed. Next, it will have *availability.* The system will be bug free, doesn't crash, and be able to fend off malicious attacks. Which brings me to *security.* The system will only allow access to certain files and memory with authorization. It will prohibit access from malicious attacks. Computer hardware changes every year, so my design has to be *portable* for computer upgrades. And lastly, I will make my system an *open system* so that many different users can change the code to adapt to their liking or to their hardware issues.