CS 4/53201 # Homework #1 **Operating Systems**

Due in class on Monday 18 September 2000

1. **(Exercise 1.4(a) from OSC) In a multiprogramming and time-sharing environment, several users share the system simultaneously. This situation can result in various security problems. What are two such problems?**

   If multiple user programs can share memory simultaneously, there is the possibility that one program might access or modify memory belonging to the other, either accidentally or maliciously. Since the two programs should "share" the CPU, there is the possibility of one program hogging the CPU, perhaps while stuck in an infinite loop. Finally, the OS must also control access to various shared system resources, such as the file system, network, or other I/O devices, to prevent similar problems.

2. **(Exercise 2.9 from OSC) When are caches useful? What problems do they solve? What problems do they cause? If a cache can be made as large as the device for which it is caching (for instance, a cache as large as a disk), why not make it that large and eliminate the device?**

   Caches provide fast access to a small set of frequently accessed data that would otherwise be stored in a larger, but slower, storage area. They solve the problem of slow access by providing fast access to at least some of the data — that data that has been accessed most recently or is most likely to be needed next. They cause problems due to the cost of the cache, and by causing consistency problems if the same data is stored in multiple caches or if multiple processes are filling the cache as the CPU switches between those processes. If a cache could be made as large as the device it was caching from, it would be either too slow to be useful, or too expensive to be practical.

   *Note: on a question like this, I want to see **clear, separate** answers to each of the four questions asked. A long paragraph describing how caches work does **not** clearly present independent answers to even the first three questions.*

3. **(Exercise 3.11 from OSC) Why is the separation of mechanism and policy a desirable property?**

   The policy decides what is to be done, while the mechanism specifies how it is to be done. Separating these two provides flexibility in a variety of ways. First, the same mechanism can be used to implement a variety of policies, so changing the policy might not require the development of a new mechanism, but just a change in parameters for that mechanism, or the choice of a new mechanism from a library of mechanisms. Second, the mechanism can be changed, for example to increase its efficiency or to move to a new platform, without changing the overall policy.

   Note: on a question like this, be careful to avoid copying material directly from the book.

4. **(Exercise 4.3 from OSC) A DECSYSTEM-20 computer has multiple register sets. Describe the action of a context switch if the new context is already loaded into one of**

the register sets. What else must happen if the new context is in main memory rather than a register set, and all the register sets are in use?

If the context is already in a register set, all the OS has to do is to change a pointer that indicates the "active" register set to the register set of the new process. If the context of the new process is in main memory instead, then it must replace the contents of one of the register sets, and the contents of that register set saved to main memory (in the process' process control block).

5. **(Exercise 4.10 from OSC) Consider an operating system that supports both the IPC and RPC schemes. Give examples of problems that could be solved with each type of scheme. Explain why each problem is best solved by the method that you specify.**

   Both packages are used for communicating between processes. Both can be used for client / server communication, although RPC tries to hide more of the packaging of parameters into messages, conversion between different data formats, error handling, etc. and may be a better choice for such problems. IPC, instead, does not necessarily require a response to a message, so may be more suitable for one-way communication, or asynchronous communication (if non-blocking sends are used). Finally, RPC is generally more oriented toward communication between processes on separate machines in a distributed system.