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

Due in class on Friday 13 September 2002

1. **(Exercise 1.7 from OSC 6th edition) We have stressed the need for an operating system to make efficient use of the computing hardware. When is it appropriate for the operating system to forsake this principle and to "waste" resources? Why is such a system not really wasteful?**

   If certain resources are not needed by other users or processes at a particular time, there may be no need to conserve those resources. For example, if there is more memory than what is needed by all currently executing process, it really is not necessary to try to reduce the memory needed for each process. Also, on single-user systems such as a personal computer, it may be desirable to give that user fast response time on interactive processes, "wasting" CPU time and possibly slowing down other tasks.

   Other reasonable answers were also accepted.

2. **(Exercise 2.2 from OSC 6th edition) How does the distinction between monitor mode and user mode function as a rudimentary form of protection (security) system?**

   This distinction limits the use of more dangerous privileged instructions to monitor mode, and allows only the OS to use monitor mode. In this way, user programs are prevented from dangerous and malicious acts such as controlling the memory management, halting the machine, etc. However, those same programs can make requests to the OS (through system calls), and if the OS chooses to do so, it can perform those same actions itself on behalf of the user program.

   Note that this question asked how the distinction provided protection, not for a definition of the two modes. Read the question carefully, and then clearly answer the question that was asked!

3. **Of the various OS structures described in class and in Chapter 3, which provides the best separation of OS policies versus OS mechanisms, and why?**

   The OS policies decide what must be done, whereas the OS mechanisms decide how it is to be done. The microkernel approach probably fits this model best, possibly with the policy (e.g., the CPU scheduler that decides how the length of the CPU time slice and what must be saved during a context switch) written as a user-mode program, with the low-level implementation details (i.e., the dispatching mechanism) handled by the microkernel.

   You could also make a similar argument for the layered approach, but since that approach is generally less efficient the microkernel approach would probably be better.

4. **(Exercise 4.2 from OSC 6th edition) Describe the difference among short-term, medium-term, and long-term scheduling.**

   The short-term scheduler moves processes between ready / waiting lists and the CPU, whereas the medium-term scheduler moves processes between main memory and disk, and the long-term scheduler loads programs from disk into memory (in effect, running a program).

   The short-term scheduler runs whenever a process reaches the end of its time slice, blocks, or terminates, whereas the medium-term scheduler runs whenever a process on disk needs to be

brought into memory to execute. The long-term scheduler really isn't part of most modern operating systems.

*Note that this question asked for the differences between the two, not a definition of each where *I* was expected to notice the difference between your two definitions. Definitions that are simply copied from the book are even less acceptable, as you are supposed to answer the questions in your own words. Read the question carefully, and then clearly answer the question that was asked!*

5. Compare Remote Procedure Calls to Remote Method Invocation, describing the similarities and differences of the two techniques.

Similarities: both execute code on a remote machine, packing data into messages and sending the data to that code, and receiving results from that code.

Differences: RPC passes simple data structures to a remote procedure (procedural programming), whereas RMI passes objects to a method on a remote object (object-oriented programming).