Sheet2
Threads

1) What are the two separate and potentially independent characteristics embodied in the concept of process?

   Resource ownership and scheduling/execution.

2) Give four general examples of the use of threads in a single-user multiprocessing system.

   Foreground/background work
   Asynchronous processing
   Speedup of execution by parallel processing of data
   Modular program structure.

3) What resources are typically shared by all of the threads of a process?

   Address space, file resources, execution privileges are examples.

4) List three advantages of ULTs over KLTs.

   - Thread switching does not require kernel mode privileges because all of the thread management data structures are within the user address space of a single process. Therefore, the process does not switch to the kernel mode to do thread management. This saves the overhead of two mode switches (user to kernel; kernel back to user).
   - Scheduling can be application specific. One application may benefit most from a simple round-robin scheduling algorithm, while another might benefit from a priority-based scheduling algorithm. The scheduling algorithm can be tailored to the application without disturbing the underlying OS scheduler.
   - ULTs can run on any operating system. No changes are required to the underlying kernel to support ULTs. The threads library is a set of application-level utilities shared by all applications.

5) List two disadvantages of ULTs compared to KLTs.

   - In a typical operating system, many system calls are blocking. Thus, when a ULT executes a system call, not only is that thread blocked, but also all of the threads within the process are blocked.
   - In a pure ULT strategy, a multithreaded application cannot take advantage of multiprocessing. A kernel assigns one process to only one processor at a time. Therefore, only a single thread within a process can execute at a time.

6) Define jacketing.

   Jacketing converts a blocking system call into a nonblocking system call by using an application-level I/O routine which checks the status of the I/O device.

7) In the discussion of ULTs versus KLTs, it was pointed out that a disadvantage of ULTs is that when a ULT executes a system call, not only is that thread blocked, but also all of the threads within the process are blocked. Why is that so?

   Because, with ULTs, the thread structure of a process is not visible to the operating system, which only schedules on the basis of processes.

8) OS/2 is an obsolete OS for PCs from IBM. In OS/2, what is commonly embodied in the concept of process in other operating systems is split into three separate types of entities: session, processes, and threads. A session is a collection of one or more processes associated with a user interface (keyboard, display, mouse). The session represents an interactive user application, such as a word processing program or a spreadsheet. This concept allows the personal computer user to open more than one application, giving each one or more windows on the screen. The OS must keep track of which window, and therefore which session, is active, so that keyboard and mouse input are routed to the appropriate session. At any time, one session is in foreground mode, with other sessions in background mode. All keyboard and mouse input is directed to one of the processes of the foreground session, as dictated by the applications. When a session is in foreground mode, a process performing video output sends it directly to the hardware video buffer and thence to the user's screen. When the session is moved to the background, the hardware video buffer is saved to a logical video buffer for that session. While a session is in background, if any of the threads of any of the processes of that session executes and produces screen output, that output is directed to the logical video buffer. When the session returns to foreground, the screen is updated to reflect the current contents of the logical video buffer for the new foreground session.
   There is a way to reduce the number of process-related concepts in OS/2 from three to two. Eliminate sessions, and associate the user interface (keyboard, mouse, screen) with processes. Thus one process at a time is in foreground mode. For further structuring, processes can be broken up into threads.
   a) What benefits are lost with this approach?
   b) If you go ahead with this modification, where do you assign resources (memory, files, etc.): at the process or thread level?
      a) The use of sessions is well suited to the needs of an interactive graphics interface for personal computer and workstation use. It provides a uniform mechanism for keeping track of where graphics output and keyboard/mouse input should be directed, easing the task of the operating system.
      b) The split would be the same as any other process/thread scheme, with address space and files assigned at the process level.
9) Consider an environment in which there is a one-to-one mapping between user-level threads and kernel-level threads that allows one or more threads within a process to issue blocking system calls while other threads continue to run. Explain why this model can make multithreaded programs run faster than their single-threaded counterparts on a uniprocessor computer.
   The issue here is that a machine spends a considerable amount of its waking hours waiting for I/O to complete. In a multithreaded program, one KLT can make the blocking system call, while the other KLTs can continue to run. On uniprocessors, a process that would otherwise have to block for all these calls can continue to run its other threads.
