Assignment 1

CS 325

50 points, due Feb. 25, 2009

1. (15 pts.) Consider a single processor timesharing system which supports a large number of interactive users. Each time a process gets the processor, the interrupt timer is set to interrupt after the quantum expires. Assume that each process has the same quantum.
   (a) What would be the effect of setting the quantum at a very large value, say 10 minutes? What sort of process behavior(s) would preclude this effect?
   (b) What would be the effect if the quantum were set to a very small value, say a few processor cycles (machine instructions)?
   (c) Obviously, an appropriate quantum must be between the values previously mentioned. If you could vary the quantum, how would you determine when you had chosen the "right" value? What factors make this value right from the user's standpoint? What factors make it right from the system's standpoint?
2. (10 pts.) In Linux, a signal is an abstraction of an interrupt. Non-privileged users are permitted to install their own signal handlers. Recall that interrupt handlers are run in the CPU's kernel mode.
   (a) Explain why it would be a *very bad idea* to permit a user's signal handler to run in kernel mode.
   (b) Assuming that the mechanism used to invoke a signal handler places the CPU in kernel mode, suggest a way (by writing pseudo-code) to safely run user's signal handlers. (Hint: assume that the CPU has a `UserMode` machine instruction which places the CPU in user mode.)
3. (5 pts.) An I/O bound process is one that spends more time waiting for I/O than using the processor. A processor-bound process is the opposite. Suppose a short-term scheduling algorithm favors those programs that have used little processor time in the recent past. Explain why this algorithm favors I/O bound processes and yet does not permanently deny processor time to processor-bound processes
4. (15 pts.) Figure 3.6 suggests that a process can only be in one device queue at a time.
   (a) Is it possible that you would want to allow a process to wait on more than one device at the same time? If so, provide a reasonable example.
   (b) If we extend the concept of a device queue to that of an event queue, where an event is defined rather broadly and encompasses such things as receiving a TCP packet over a socket connection and receiving a return value from a child process, would you want to allow a process to wait on more than one event at the same time? Again, if so, provide a reasonable example.

(c) How would you modify the queuing data structure of Figure 3.6 to support processes waiting within more than one queue?

5. (5 pts.) In a number of early computers, an interrupt caused the register values to be stored in fixed memory locations associated with the particular interrupt that occurred. Under what circumstances is this a practical technique? Explain why it is inconvenient in general.