Operating Systems Sample Questions

# Events versus Threads

1. In the lecture titled "Why threads are a bad idea" ,
   a) what is the key reason for discouraging the use of threads for most applications?
   b) What are the drawbacks of thread-based programming that event-driven programming doesn't have?
   c) When should you use threads, then?

2. Traditional operating systems support the fundamental abstractions of Processes/Threads which are stateful, blocking, and (typically) long-running. Most OS services, such as the CPU scheduler, have evolved to manage process transitions across the process lifecycle. Even event-driven programming is currently implemented using a process that monitors different events in a while loop.

   Consider an alternative OS design in which there is no process or thread abstraction. Instead the fundamental OS abstraction is “event-driven” tasks which are stateless, non-blocking, and (typically) short-lived. Describe how this abstraction could be supported by the OS and what would be the key changes in the OS, particularly for CPU scheduler, I/O processing, and concurrency primitives.

3. Consider the “events vs threads” argument in the context of monilithic operating system kernels (like Linux or Windows). (a) Which model do these operating systems primarily use -- events or threads? Why? (b) Let’s say you that have to design an operating system that uses the opposite model to what you just answered in (a). What would be the major design changes you would make to the kernel in terms of CPU scheduling, memory management, and I/O processing subsystems?