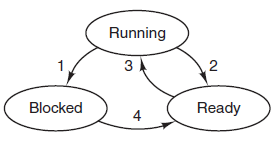
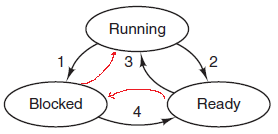
**1.** **In Fig. 2-2, three process states are shown. In theory, with three states, there could be six transitions, two out of each state. However, only four transitions are shown. Are there any circumstances in which either or both of the missing transitions might occur?**

**1） 当系统只有一个进程时，从阻塞到运行状态的转换是可能会发生的，例如。某个进程因为发生了I/O请求产生阻塞，而且当前I/O请求已经结束，如果此时CPU空闲，该进程就可以从阻塞态直接转到运行态。**

**2） 另外一种转换，是从就绪态到阻塞状态，这是不可能的，也是不必要的。一个就绪进程如果不做任何事情，是不可能，也没必要直接转到阻塞状态，必须要经过运行态后才能进入阻塞。**

**2.  Suppose that you were to design an advanced computer architecture that did process switching in hardware, instead of having interrupts. What information would the CPU need? Describe how the hardware process switching might work.**

**CPU应该有一个寄存器存放当前进程表项的指针。当I/O结束之后，CPU将把当前的机器状态存入到当前进程表项中。然后，将转到中断设备的中断向量，读取另一个过程表项的指针（服务例程），然后，就可以启动这个进程了。**

**3. On all current computers, at least part of the interrupt handlers are written in assembly language. Why?**

**通常，高级语言不允许访问CPU硬件，而对于中断处理程序而言，访问硬件是必需的，例如，中断处理程序可能需要禁用和启用某个特定设备的中断服务，或者处理进程堆栈区的数据。另外，中断服务程序需要尽快地执行。**

**4. When an interrupt or a system call transfers control to the operating system, a kernel stack area separate from the stack of the interrupted process is generally used. Why?**

中断或系统调用把控制转给操作系统时，通常将内核堆栈和被中断进程的运行堆栈分离。为什么？

**内核使用单独的堆栈，主要原因是：**

**1）内核堆栈与用户程序堆栈分离，可以使得用户程序具有足够的堆栈空间，有利于用户任务的顺利完成。**

**2）如果内核将数据保留在用户空间，系统调用之后，留在用户空间中的信息，可能就会泄露，出现安全问题。**

**7. If a multithreaded process forks, a problem occurs if the child gets copies of all the parent’s threads. Suppose that one of the original threads was waiting for keyboard input. Now two threads are waiting for keyboard input, one in each process. Does this problem ever occur in single-threaded processes?**

**不会。如果单线程进程中，原始线程正在等待键盘输入，就不能创建子进程，必须等待上一个线程完成后才能创建子进程。**