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?

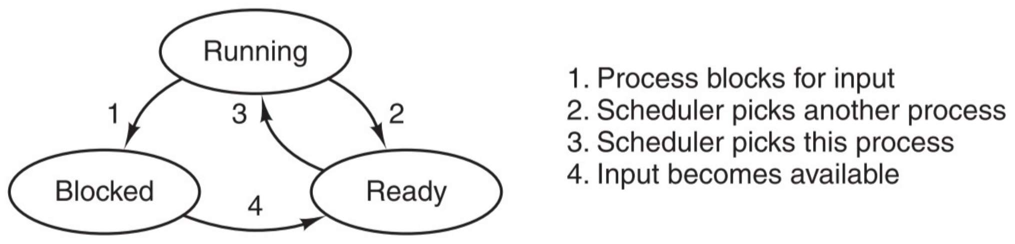

Running
Blocked
Ready
1
2
3
4
1. Process blocks for input
2. Scheduler picks another process
3. Scheduler picks this process
4. Input becomes available

Figure 2-2. A process can be in running, blocked, or ready state. Transitions between these states are as shown.

阻塞->运行：可能。
就绪->阻塞：不可能。

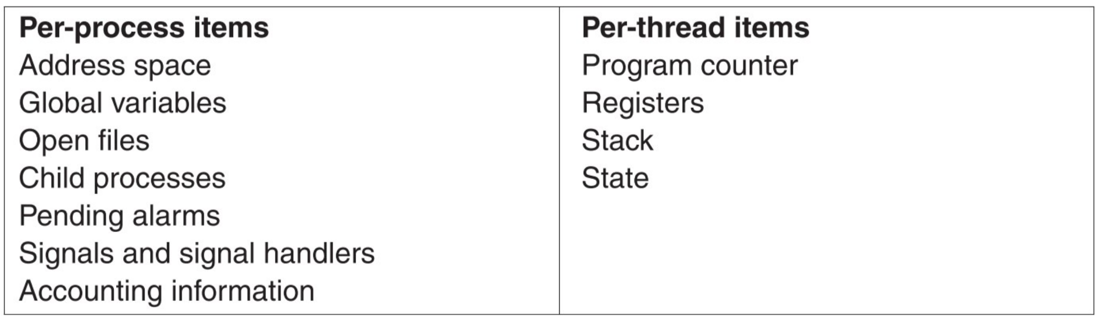
从阻塞到运行的过渡是可能的。假设某一进程，它由于等待接收数据而处于阻塞状态，那么当其接收到数据时，若 CPU 恰好空闲、且就绪队列为空,它便可能直接进入 running 状态。
从就绪到阻塞是不可能的。只有正在运行的进程才会被阻塞。

2. On all current computers, at least part of the interrupt handlers are written in assembly language. Why?
中断处理程序可能需要来启用和禁用某个特定设备的中断服务, 或处理进程堆栈区域内的数据，但通常高级语言不允许访问 CPU 硬件。
另外，中断服务程序必须尽快执行，高级语言的执行速度不如直接的汇编语言，汇编语言效率更高。

3. 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?
保护内核堆栈数据。若不这样做，则系统调用返回后内核堆栈数据仍留在用户程序的内存空间中，将可被其它用户程序访问，造成信息的不安全。

4. A computer has 4 GB of RAM of which the operating system occupies 512 MB. The processes are all 256 MB (for simplicity) and have the same characteristics. If the goal is 99% CPU utilization, what is the maximum I/O wait that can be tolerated?
可以容纳 (4 * 1024 - 512) / 256 = 14 个进程
假设每个进程处于 I/O 等待的概率为 p
CPU 的利用率为 $1 - p^{14} >= 99\%$
算得 $p <= 71.97\%$

5. In Fig. 2-12 the register set is listed as a per-thread rather than a per-process item. Why? After all, the machine has only one set of registers.



Per-process items	Per-thread items
Address space	Program counter
Global variables	Registers
Open files	Stack
Child processes	State
Pending alarms	
Signals and signal handlers	
Accounting information	

Figure 2-12. The first column lists some items shared by all threads in a process. The second one lists some items private to each thread.

当线程作为调度的基本单位时，当前线程的暂停执行，需要保存线程当前执行位置(Program Counter)，还需要保存 CPU 上当前的各类寄存器值(Registers)，在将来继续运行时，恢复到正确的位置和值；每个线程运行时创建的临时变量需要保存在栈(Stack)中，以免互相干扰，另外，每个线程需要记录其状态(State)，便于调度。

6. Why would a thread ever voluntarily give up the CPU by calling thread yield? After all, since there is no periodic clock interrupt, it may never get the CPU back.
进程中的线程是合作状态，它们彼此不敌对。如果应用程序需要阻塞以运行得更好，那么一个线程可以调用 thread yield 自愿放弃 CPU。

7. What is the biggest advantage of implementing threads in user space? What is the biggest disadvantage?
最大的优势就是效率。不需要陷入内核来切换线程。最大的缺点是，如果一个线程阻塞，整个进程都会阻塞。

8. Can a measure of whether a process is likely to be CPU bound or I/O bound be determined by analyzing source code? How can this be determined at run time?
I/O 密集指的是系统的 CPU 效能相对硬盘/内存的效能要好很多，此时，系统运行大部分的时间是 CPU 在等待 I/O 的读/写，CPU 密集则相反。
阅读源码，对不同文件进行增量地读写的程序通常是 I/O 密集的；而大部份程序都是用来做计算、逻辑判断等 CPU 动作，而读写的命令很少的程序通常是 CPU 密集。
在运行时可以通过任务管理器查看该程序的的 CPU 占用率，CPU 占用多的是 CPU 密集程序，否则是 I/O 密集程序。

9. Consider a real-time system with two voice calls of periodicity 5 msec each with CPU time per call of 1 msec, and one video stream of periodicity 33 ms with CPU time per call of 11 msec. Is this system schedulable?

$\sum_{i=1}^{3}\frac{c_i}{p_i}=\frac{1}{5}+\frac{1}{5}+\frac{11}{33}=\frac{11}{15}<1$ ，所以是可调度的。

10. Consider the following piece of C code: void main( ) { fork( ); fork( ); exit( ); } How many child processes are created upon execution of this program?
3 个
main 进程执行两次 fork，创建 2 个子进程，main 进程第一次 fork 创建的子进程会执行第二个 fork，再创建一个子进程（main 的孙进程），则共创建 3 个进程。

11. Five batch jobs. A through E, arrive at a computer center at almost the same time. They have estimated running times of 10, 6, 2, 4, and 8 minutes. Their (externally determined) priorities are 3, 5, 2, 1, and 4, respectively, with 5 being the highest priority. For each of the following scheduling algorithms, determine the mean process turnaround time. Ignore process switching overhead. (a) Round robin. (b) Priority scheduling. (c) First-come, first-served (run in order 10, 6, 2, 4, 8). (d) Shortest job first. For (a), assume that the system is multiprogrammed, and that each job gets its fair share of the CPU. For (b) through (d), assume that only one job at a time runs, until it finishes. All jobs are completely CPU bound.
A:时间片轮转
刚开始，有 5 个作业。在前 10 分钟里，每个作业获得 1/5 的 CPU 时间。在第 10 分钟时，C 作业完成。
此时剩余 4 个作业。在接下来的 8 分钟里，每个作业获得 1/4 的 CPU 时间。在第 18 分钟，D 作业完成。
此时剩余 3 个作业。在接下来的 6 分钟里，每个作业获得 1/3 的 CPU 时间。在第 24 分钟，B 作业完成。
此时剩余 2 个作业。在接下来的 4 分钟里，每个作业获得 1/2 的 CPU 时间。在第 28 分钟，E 作业完成。
此时剩余 1 个作业。在第 30 分钟，A 作业完成。
因此，平均进程周转时间为(10+18+24+28+30)/5=22min

B:优先级调度
由题意，任务执行顺序为 B-E-A-C-D，它们完成时的时间依次为 6min、14min、 24min、26min、30min，因此平均进程周转时间为(6+14+24+26+30)/5=20min

C:先来先服。
由题意，任务执行顺序为 A-B-C-D-E，它们完成时的时间依次为 10min、16min、 18min、22min、30min，因此平均进程周转时间为(10+16+18+22+30)/5=19.2min

D:最短作业优先
由题意，任务执行顺序为 C-D-B-E-A，它们完成时的时间依次为 2min、6min、12min、20min、30min，因此平均进程周转时间为(2+6+12+20+30)/5=14min。

12. The aging algorithm with a = 1/2 is being used to predict run times. The previous four runs, from oldest to most recent, are 40, 20, 40, and 15 msec. What is the prediction of the next time?

预测时间计算公式为$T_{n+1} = at_n + (1-a)T_n$，其中T为预测时间，t为实际的观测时间

因此有$t_1 = T_1 = 40ms$，$t_2 = 20ms$，$t_3 = 40ms$，$t_4 = 15ms$，所以

$$T_5 = \frac{1}{2}t_4 + \frac{1}{2}T_4 = \frac{1}{2}t_4 + \frac{1}{2}\left(\frac{1}{2}t_3 + \frac{1}{2}T_3\right) = \frac{1}{2}t_4 + \frac{1}{4}t_3 + \frac{1}{4}T_3 = \frac{1}{2}t_4 + \frac{1}{4}t_3 + \frac{1}{4}\left(\frac{1}{2}t_2 + \frac{1}{2}T_2\right)$$
$$= \frac{1}{2}t_4 + \frac{1}{4}t_3 + \frac{1}{8}t_2 + \frac{1}{8}T_2 = \frac{1}{2}t_4 + \frac{1}{4}t_3 + \frac{1}{8}t_2 + \frac{1}{8}\left(\frac{1}{2}t_1 + \frac{1}{2}T_1\right) = 25ms$$