Part1 死锁

1. 在操作系统中，死锁出现是指(A)

A.多个进程竞争资源出现了循环等待　　B. 一个进程进入死循环

C. 进程释放资源　　D. 多个进程竞争使用共享型的设备

2. 死锁的四个必要条件中，无法破坏的是(C)

A. 非抢夺式分配　　B. 占有且等待资源

C. 互斥使用资源　　D. 环路等待资源

3. 若系统 S1 采用死锁避免方法，S2 采用死锁检测方法。下列叙述中，正确的是(B)

I. S1 会限制用户申请资源的顺序，而 S2 不会

II. S1 需要进程运行所需的资源总量信息，而 S2 不需要

III. S1 不会给可能导致死锁的进程分配资源，而 S2 会

A. 仅 I、II　　B. 仅 II、III　　C. 仅 I、III　　D. I、II、III

解析：

1. 死锁预防采用破坏产生死锁的 4 个必要条件中的一个或几个来防止发生死锁。其中之一的“破坏循环等待条件”，一般采用资源有序分配法，限制用户申请资源的顺序，因此 **I 的前半句属于死锁预防的范畴**。因此 I 错误。

2. 银行家算法是最著名的死锁避免算法，需要资源总量信息，若检测到不安全，则本次分配作废。而在死锁检测中，系统为进程分配资源时不采取任何措施，但提供死锁的检测手段。因此 II 和 III 正确。

4、某系统有 n 台互斥使用的同类设备，三个并发进程分别需要 3，4，5 台设备，可确保系统不发生死锁的设备数 n 最小为(B)

A. 9　　B. 10　　C. 11　　D. 12

解析：考虑极端情况，3 个进程分别申请到 2/3/4 台设备，此时产生死锁。只要再增加一台设备就能避免以上情况，因此 n 最小为 2+3+4+1=10。

5. 下面是一个并发进程的程序代码，正确的是(B)

```
Semaphore x1=x2=y=1
Int c1=c2=0;
P1()                                P2()
{                                   {
 while(1){                           while(1){
    P(x1);                              P(x2);
    If(++c1==1) P(y);                   If(++c2==1) P(y);
    V(x1);                              V(x2);
    computer(A);                        computer(B);
    P(x1);                              P(x2);
    If(--c1==0) V(y);                   If(--c2==0) V(y);
    V(x1);                              V(x2);
    }                                   }
}                                   }
```

A. 进程不会死锁，也不会“饥饿”　　B. 进程不会死锁，但是会“饥饿”

C. 进程会死锁，但是不会“饥饿”　　D. 进程会死锁，也会“饥饿”

解析：若进程执行过程如下：①假设 P1 进程稍快，P2 进程稍慢，同时运行。②P1 进程首先进入 if 条件语 句，因此获得了 y 的互斥访问权，P2 被阻塞；③在第一个 P1 进程未释放 y 之前，又有另一个 P1 进入，c1 的值变成 2，当第一个 P1 离开时，P2 仍然被 阻塞，这种情形不断发生；④在这种情况下，P1 顺利执行，P2 长期被阻塞。

6. 死锁检测方法可以获得最大的并发性。并发性排序从小到大排序：资源 预分配、银行家算法、死锁检测方法。

7.

$$Need = Max - Allocation = \begin{bmatrix} 0 & 0 & 4 \\ 1 & 7 & 5 \\ 2 & 3 & 5 \\ 0 & 6 & 4 \\ 0 & 6 & 5 \end{bmatrix} - \begin{bmatrix} 0 & 0 & 3 \\ 1 & 0 & 0 \\ 1 & 3 & 5 \\ 0 & 0 & 2 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 7 & 5 \\ 1 & 0 & 0 \\ 0 & 6 & 2 \\ 0 & 6 & 4 \end{bmatrix}$$

I:根据 Need 矩阵可知，当 Available 为(1,4,0)时，可以满足 P2 的需求；P2 结束释放资源之后，Available 为(2,7,5)可以满足 P0、P1、P3、P4 中任一进程的需求，所以系统不会出现死锁，处于安全状态。 II: 当 Available 为(0,6,2)时，系统可以满足进程 P0、P3 的需求，二者释放资源 后，Available 为(0,6,7)，仅可以满足 P4 的需求。P4 结束释放资源后，Available 为(0,6,8)，此时不能满足余下任一进程的需求，系统出现死锁，当前处

于非安全状态。 III: 当 Available 为(1,1,1)时，系统可以满足 P0、P2 的需求；这两个进程结束后释放资源，Available 为(2,4,9)，此时不能满足余下任一进程的需求，系统出现死锁，当前处于非安全状态。 IV: 当 Available 为(0,4,7)时，系统可以满足 P0 的需求；进程结束后释放资源， Available 为(0,4,10)，此时不能满足余下任一进程的需求，系统出现死锁，当前处于非安全状态。

Part2 CPU 调度

1. 若某单处理器多进程系统中有多个就绪态进程，则下列关于处理机调度的叙述中，错误的是(D)。

A. 创建新进程后能进行处理机调度

B. 在进程结束时能进行处理机调度

C. 在系统调用完成并返回用户态时能进行处理机调度

D. 在进程处于临界区时不能进行处理机调度

解析：当进程处于临界区时，说明进程正在占用处理机，只要不破坏临界资源的使用规则，是不会影响处理机调度的。

2. P1、P2 和 P3 是在某个系统中正在执行的三个进程，各进程的计算(CPU)时间和 I/O 时间比例如下表所示：

进程	计算时间	I/O 时间
P1	50%	50%
P2	20%	80%
P3	30%	70%

为提高系统资源利用率，合理的进程优先级设置是(C)。

A. P1>P2>P3　　B. P3>P2>P1　　C. P2>P3>P1　　D. P1>P3>P2

解析:计算进程会占用大量的 cpu 时间,而 i/o 大的会占用较少的 cpu 资源,相当于短作业,所以应该优先权更高。

3. 下列选项中，降低进程优先级的合理时机是(B)。

A.进程刚完成 I/O，进入就绪队列　　B. 进程的时间片用完

C.进程从就绪状态转为运行态　　D. 进程长期处于就绪队列中

解析：对于 A 选项，该进程已经进入就绪态；对于 C 选项，该进程刚运行就降低优先级，可能被抢断；对于 D 选项，该进程长期未能运行，再降低优先级可能饿死。

4. 陷阱指令（trap）可以使执行流程从用户态陷入内核，在用户进程使用陷阱（trap）执行调用内核函数的过程中，以下哪些步骤是由操作系统内核完成的(A)

① 执行用户进程的 main 函数

② 切换至内核模式

③ 跳转至陷阱处理器（trap handler）

④ 处理陷阱（handle trap）

⑤ 执行系统调用

A. ④⑤　　B. ③④⑤　　C. ②③④⑤　　D. ①⑤

解析：题目参考了《Operating Systems-Three Easy Pieces》 第 6 章第 5 页的 Figure 6.2。系统调用(system call)是提供给用户程序用于执行更高权限操作的途径。为了执行系统调用，程序必须先执行特殊的 trap 指令，硬件将特权等级切换至内核模式；系统调用结束后，操作系统会调用特殊的 return-from-trap 指令，返回回用户程序。但为了保证能正确地返回，硬件需要在执行 trap 指令时保存调用者（用户程序）的一些寄存器，并在操作系统处理 return-from-trap 时取出。因此，只有处理 trap 和执行系统调用是由操作系统完成的。

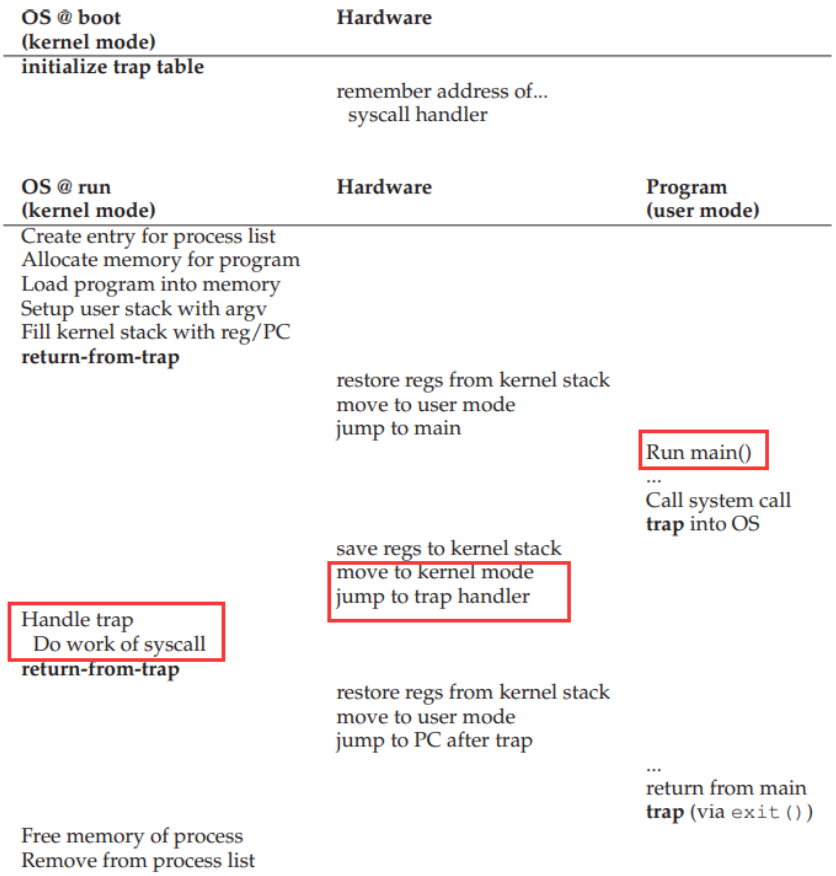


Figure 6.2: **Limited Direct Execution Protocol**

5. 时钟中断（timer interrupt）是整个操作系统的脉搏，系统利用时钟中断维持系统时间、**保证进程共享 CPU**、确定调度优先级。考虑如下场景：系统开机后，先执行进程 A，接着遇到时钟中断，转为执行进程 B，过程中需要执行以下操作

① 从陷阱中返回（return from trap）
② 初始化陷阱表（initialize trap table）
③ 处理时钟中断（timer interrupt）
④ 开启中断计时器（start interrupt timer）
⑤ 执行进程 A
⑥ 执行进程 B

操作的正确顺序是（ D ）

A. ②④⑤③⑥①　　B. ⑤②④③⑥①　　C. ④②⑤③①⑥　　D. ②④⑤③①⑥

解析：题目参考了《Operating Systems-Three Easy Pieces》第 6 章第 10 页的 Figure 6.3。初始化陷阱表（initialize trap table），让操作系统告诉硬件 trap table 放在内存中的什么位置，一般在系统启动时就会执行这条指令，因此初始化陷阱表是最先执行的。为了使得不同的进程能在 CPU 上切换，CPU 需要隔一段时间就暂停执行当前进程，转去执行别的进程，这就需要时钟中断（timer interrupt）。因此在系统启动后，需要开启中断计时器，记录系统执行当前进程的时间，这是第二个操作。接着系统执行进程 A，这是第三个操作。在执行 A 的过程中，中断计时器时间结束，触发了时钟中断（timer interrupt），这是第四个操作。此时系统 trap 到内核模式，中断结束后通过 return-from-trap 从内核模式返回，这是第五个操作。返回后，CPU 开始执行进程 B。因此正确的执行顺序是②④⑤③⑥①。

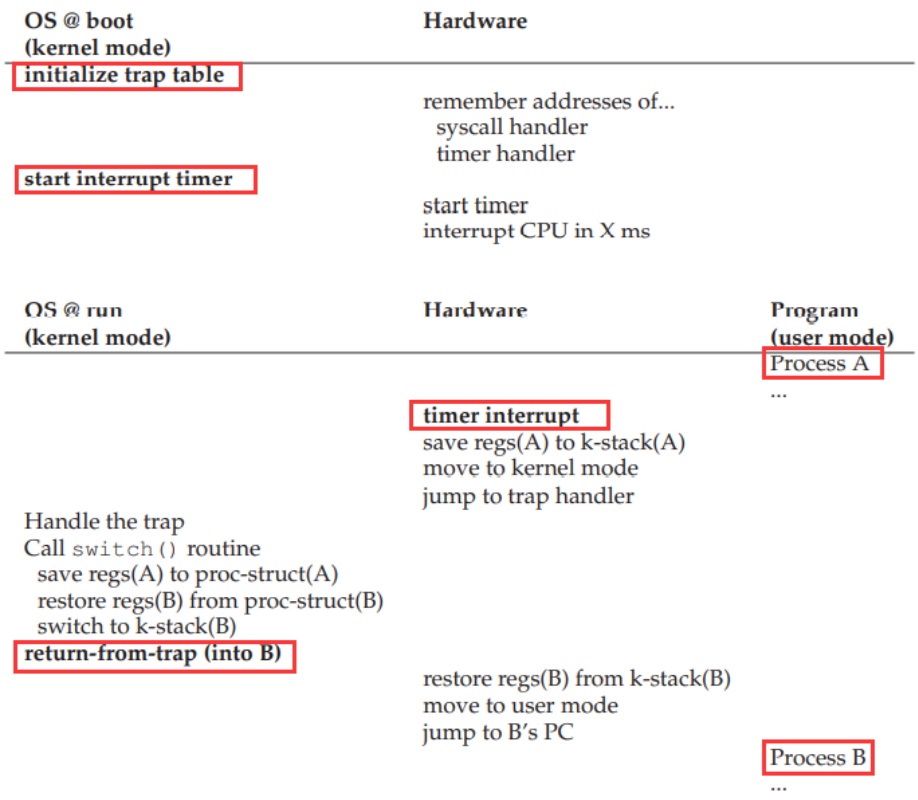
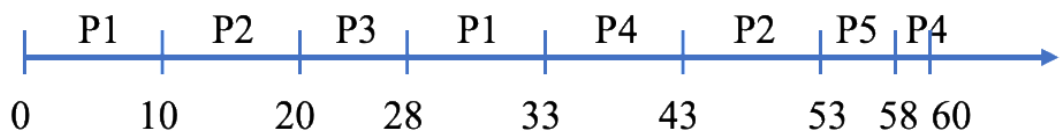


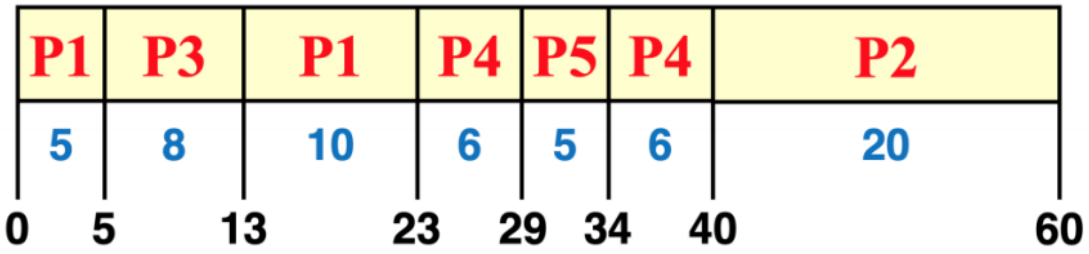
Figure 6.3: **Limited Direct Execution Protocol (Timer Interrupt)**

6.

1）



2）



3）

平均响应时间:

RR：（ 0+10+(20-5)+(33-13)+(53-29) ） / 5 = 13.8 ms

SJF：（ 0+40+(5-5)+(23-13)+(29-29) ） / 5 = 10 ms

平均周转时间

RR：（ (33-0)+(53-0)+(28-5)+(60-13)+(58-29) ） / 5 = 37 ms

SJF：（ (23-0)+(60-0)+(13-5)+(40-13)+(34-29) ） / 5 = 24.6 ms