1.
对资源进行管理，在相互竞争的应用程序之间有序地控制软硬件资源的分配、使用和回收，使资源能够在多个程序之间共享
提供进程抽象、虚存抽象和文件抽象，防止硬件资源被失控应用程序滥用以及屏蔽复杂的硬件操作细节，为应用程序提供使用硬件资源的简单且一致的方法

2.
多道程序设计的主要原因是想在等待 I/O 操作完成时，CPU 仍然可以处理其他任务，I/O 操作的速度比较慢而 CPU 运算速度快，如果没有 DMA，每个字节数据都由 CPU 读写，那么 CPU 一直忙于 I/O 操作，导致 CPU 利用率很低
多道程序的主要原因是当等候 I/O 完成时 CPU 有事可做。如果没有 DMA，I/O 操作时 CPU 被完全占有，因此，多道程序无利可图（至少在 CPU 利用方面）。无论程序操作多少 I/O 操作，CPU 都是 100%的忙碌。当然，这里是假定主要的延迟是数据复制时的等待。如果 I/O 很慢的话，CPU 可以做其他工作。

3.
内核态和用户态的区别在于对硬件的访问权限不同:内核态下的程序---操作系统拥有对硬件的完全访问权限，能够执行机器支持的所有指令，例如影响机器控制和 I/O 操作的指令，具有绝对的特权;而用户态下的程序仅能使用机器指令的一个子集，不能执行属于内核态的特权指令
在处理器具备内核态和用户态两种工作模式的前提下设计操作系统，能够通过硬件机制保障运行在内核态下的操作系统的主导地位，保证系统的安全性。这两种工作模式的区分赋予了操作系统特权并提供保护，使得用户程序对硬件的访问受到操作系统的限制和管理:用户程序只能在用户空间操纵用户数据，调用用户程序中的过程而不能直接访问内核数据或者直接调用服务例程

4.
自陷指令是为了进行系统调用而引起处理器中断的机器指令，这是一条非特权指令，在用户态下执行时会将 CPU 的工作模式切换到内核态，使得操作系统的相应服务例程得以执行
应用程序执行系统调用，产生中断转向内核态，进入陷阱处理程序，它将按功能号来查询入口地址表，并转至对应服务例程执行；完成后退出中断，返回应用程序断点继续运行

5.
时分复用共享:CPU、网卡、打印机、键盘
空分复用共享:内存、磁盘
两者皆可:显示器

6.
就程序逻辑而言，库例程调用哪个系统调用是没有关系的。但是，如果需要考虑性能问题，无需系统调用就可以完成的任务将使进程运行更快。所有的系统调用都会导致用户环境和内核环境的切换开销。更进一步，在多用户系统中，在系统调用完成之前，操作系统可能调度到其他的进程，这将使得调用过程的处理更加迟缓。

7.

机制与策略分离原则是让程序中的独立部分分别来实现机制与策略，这有助于使内核保持短小和良好的结构，这种软件不但适应性好而且易于开发。通过将机制植入操作系统而将策略留给上层软件（如应用程序），即使存在改变策略的需要，系统本身也可以保持不变，即使策略模块必须保留在内核中，如果可能的话，它也应该与机制相隔离，这样策略模块中的变化就不会影响机制模块。

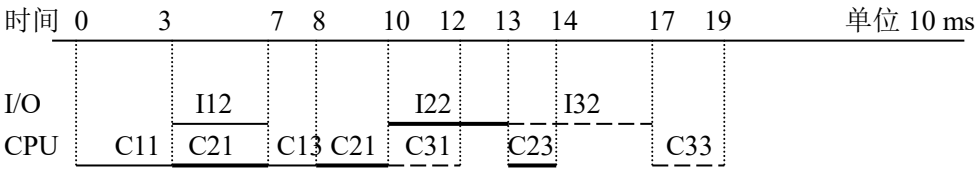
3. 设有三道程序，按 A、B、C 优先次序运行，其内部计算和 I/O 操作时间由表给出。

A	B	C
C_{11}＝30ms	C_{21}=60ms	C_{31}=20ms
|	|	|
I_{12}＝40ms	I_{22}=30ms	I_{32}=40ms
|	|	|
C_{13}＝10ms	C_{23}=10ms	C_{33}=20ms

试画出按多道运行的时间关系图（忽略调度执行时间）。完成三道程序共花多少时间？比单道运行节省了多少时间？若处理器调度程序每次进行程序转换化时 1ms，试画出各程序状态转换的时间关系图。

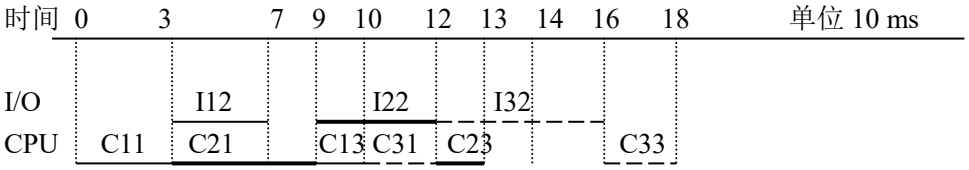
答：

（1）忽略调度执行时间，多道运行方式（抢占式）：



抢占式共用去 190ms，单道完成需要 260ms，节省 70ms。

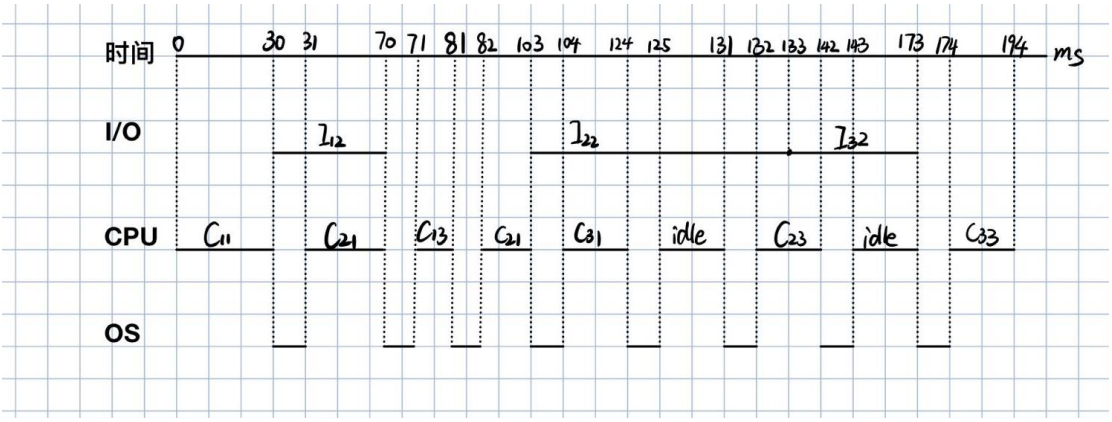
忽略调度执行时间，多道运行方式（非抢占式）：



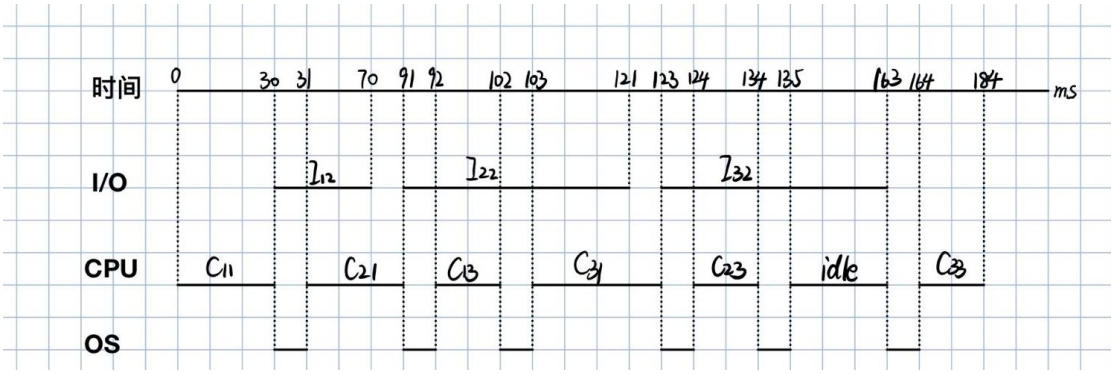
非抢占式共用去 180ms，单道完成需要 260ms，节省 80ms。

（2） 注意：不考虑 I/O 设备初始化时间和 I/O 中断处理时间，只考虑调度时间，没有可调度的进程时，选择 idle 进程执行。

调度执行时间 1ms，多道运行方式（抢占式）：



调度执行时间 1ms，多道运行方式（非抢占式）：



9. 在单机系统中，有同时到达的 A，B 两个程序，若每个程序单独执行，则需使用 CPU，DEV_1（设备 1），DEV_2（设备 2）的顺序和时间如表所示：

程序	运行情况(单位 ms)						
A	CPU	DEV_1	CPU	DEV_2	CPU	DEV_1	CPU
	25	39	20	20	20	30	20
B	CPU	DEV_1	CPU	DEV_2	CPU	DEV_1	CPU
	20	50	20	20	10	20	45

给定下列条件：

（1）DEV_1 和 DEV_2 为不同的 I/O 设备，它们能够同时工作。

（2）程序 B 的优先级高于 A。但是，当程序 A 占用 CPU 时，即使程序 B 需要使用 CPU，也不能打断程序 A 的执行而应等待。

（3）当使用 CPU 之后控制转向 I/O 设备，或者使用设备之后控制转向 CPU，由控制程序执行中断处理，但这段处理时间忽略不计。试解答下列问题：

（1）哪个程序先结束？

（2）程序全部执行结束需要多少时间？

（3）程序全部执行完毕时，CPU 的利用率为多少？

（4）程序 A 等待 CPU 的累计时间为多少？

（5）程序 B 等待 CPU 的累计时间为多少？

答：见运行图。

o ms　B 优先运行，占用 CPU 20 ms，其间 A 等待；

20ms　B 运行结束，并开始占用 DEV1，A 开始占用 CPU 25ms；

45 ms　A 占用 CPU 25ms 结束，B 继续占用 DEV1；

70 ms　B 第二次占用 CPU，A 开始占用 DEV1；

90 ms　B 第二次占用 CPU 20ms 结束，B 第一次占用 DEV2；

109 ms　A 第一次占用 DEV1 结束，A 第二次占用 CPU，B 继续占用 DEV2；

110ms　B 第一次占用 DEV2 结束，B 开始空等，A 继续占用 CPU；

l29 ms　B 空等 CPU 19ms 结束，开始第三次占用 CPU，A 第二次占用 CPU 结束，A 第一次开始占用 DEV2；

139 ms　B 第三次占用 CPU l0ms 结束，B 第二次占用 DEV1 开始，此时 A 第一次继续占用 DEV2；

149 ms　A 第一次继续占用 DEV2 结束，并开始第三次占用 CPU，B 继续占用 DEV1；

159 ms　B 第二次占用 DEV1 结束，开始空等 CPU，此时 A 继续第三次占用 CPU；

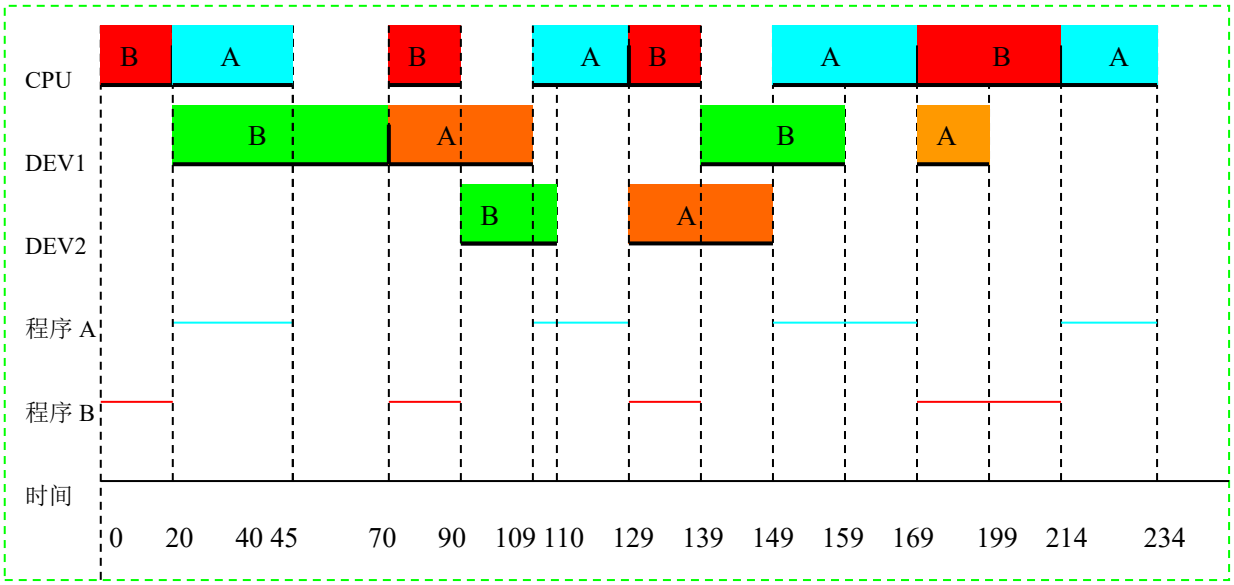
169 ms　A 第三次占用 CPU 结束，并开始第二次占用 DEV1，B 空等 CPU l0ms 结束，开始第四次占用 CPU；

199 ms　A 第二次占用 DEV1 结束，时间为 30ms，并开始空等 CPU，此时 B 正占用 CPU；

214 ms　B 第四次占用 CPU 结束，至此 B 全部结束。而 A 开始第四次占用 CPU，时间为 20ms；

234 ms　A 占用 CPU 结束，至此 A 全部结束。

根据以上分析可知，程序 B 先结束。全部程序运行结束需要 234ms。CPU 的利用率为：（20＋20＋10＋45 十 25＋20＋20＋20）／234＝76.92％。程序 A 等待 CPU 的累计时间为 35 ms(0ms 起等了 20ms，199ms 起等了 15ms)；程序 B 等待 CPU 的累计时间为 29ms(110ms 起等了 19ms，199 起等了 10ms)。


CPU
DEV1
DEV2
程序 A
程序 B
时间
B
A
B
A
B
A
B
A
B
A
B
A
0
20
40
45
70
90
109
110
129
139
149
159
169
199
214
234