操作系统第一次作业

2019011008 无 92 刘雪枫

1. 有意义。例如 Intel 的系列 CPU 中，Intel Core i5、i7、i9 虽然性能不同，但是都采用相同的体系结构，指令集互相兼容，属于一个系列的 CPU 家族，这就是一种系列计算机思想。
2.
   a) 单纯从代码逻辑来看，库函数是否导致系统调用并不影响逻辑。
   b) 但是，系统调用存在 trap 指令，会导致程序陷入内核态。在进行用户态与内核态之间的交换时会有较大的性能开销。因此，当对程序的性能要求较高的情况下需要了解哪些库函数会导致系统调用并尽可能少地使用。
   c) 此外，系统调用同样可能会导致进程切换等，频繁的系统调用可能导致程序不会用满自己的时间片而抢占较少的 CPU 运行时间，因此在有较多进程一起运行的情况下，且对程序的运行速度要求较高，也需要了解库函数是否会导致系统调用。
3. 库函数和系统调用的内核函数不必具有相同的名字。对于编写用户态应用程序的程序员来说，库函数的名字更加重要。因为此类程序员直接调用的是库函数，而基本上不会去直接调用内核函数。因此了解库函数的名字更加重要。
4.
   a) 采用顺序执行，A 的周转时间为 20+10+10+20+10=70s，B 的周转时间为 70+20+20+10+10+20=150s。A 与 B 运行总共需要 150s，其中 CPU 运行 70s，使用设备 80s，因此 CPU 利用率为 46.7%，设备利用率为 53.3%。
   b) 采用并发执行时，两者同时载入内存开始运行。前 20s，A 利用 CPU 而 B 使用设备；20~30s，A 使用设备而 B 利用 CPU；30~40s，B 继续利用 CPU 而 A 等待 CPU；40~50s，A 利用 CPU 而 B 使用设备；50~60s，A 使用设备而 B 利用 CPU；60~70s，A 继续使用设备而 B 等待设备；70~80s，A 利用 CPU，A 执行完毕；70~90s，B 使用设备，B 执行完毕。

      因此，A 的周转时间为 80s，B 的周转时间为 90s。在这 90s 内，60~70s

和 80~90s 两个时间段内 CPU 空闲，因此 CPU 利用率为 1-20/90=77.8%；30~40s 设备空闲，因此设备利用率为 1-10/90=88.9%。

5. 在 Unix 中创建进程需要 fork 和 exec 两个调用而 Windows 则只需要 CreateProcess 一个调用。经过查阅资料发现，Unix 这样设计来源于早期的操作系统并不存在多进程的现象，在使用 shell 执行一个程序时，把当前的 shell 进行复制，然后再执行程序覆盖掉当前的 shell，在执行完毕后把 shell 原有的环境替换回来，因此才有了 fork 和 exec 两个调用。

   a) Unix 这两个调用的一个优点是使用方法简单，一个例子是有利于 shell 进行输入输出流的重定向：例如可以先进行 fork，再把子进程输入输出流重定向，然后再进行 exec；而使用 CreateProcess 如果要实现重定向则需要传递额外的参数，使得 CreateProcess 的参数增多，不利于使用。

   b) 而 Unix 的设计也有缺点。有研究指出 (Andrew Baumann, etc, 2019)，首先，Unix 的 fork 与多线程存在巨大的冲突问题：在多线程编程中，只有执行 fork 的线程才能第一时间了解到自身处于何种进程，而如果线程通信方式设计不周则可能导致其他线程由于辨识进程错误而误操作，因此使用 fork 的多线程与多进程之间协调难度较大。第二，由于 fork 的子进程继承了父进程的一切，即会承载父进程的几乎所有信息，因此在安全性上存在问题，等等。而使用 CreateProcess 则不存在这些问题。

   综合来看，两种方式都各有利弊，何种方式更优需要视具体情况而定。

参考文献

[1] Andrew Baumann, Jonathan Appavoo, etc. A fork() in the road. 2019/04. https://www.microsoft.com/en-us/research/uploads/prod/2019/04/fork-hotos19.pdf