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。
   1. 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。
   2. 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。
   3. 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。
2. 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。
   1. 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%。
   2. 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%。

1. 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。
   1. 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。
   2. 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>