操作系统2004年试卷

一.

1.系统调用：当用户程序调用操作系统所提供的函数时称为系统调用。
系统调用需要内核的参与，要在内核模式下执行，需要模式切换。
普通函数在用户模式下执行，不需要进行模式切换。

2.原因：高优先级的阻塞态进程很快就要就绪，因此换出一个低优先级的就绪态进程。
处于就绪/挂起态。

3.不一定。若这两个内核级线程来自同一进程，则开销会小些；但若这两个内核级线程来自不同的进程，则不仅仅进行模式切换，还要进行进程切换，开销反而变大。

4.阻塞等待方式比较好。当进程处于阻塞态时，处理器可以把资源分配给其他进程使用，这样便充分地利用了处理器。

5.最少需要3种消息：请求wait，请求signal，完成操作

6.最多允许4个进程同时运行。

7.f=p+n

8. 充分条件：

P1:C1=20,T1=100,D1=100

P2:C2=40,T2=150,D2=150

P3:C3=100,T3=350,D3=350

总使用率为：20/100+40/150+100/350＝0.753

$3\times(2^{1/3}-1)=0.78>0.753$

必要条件：

P1:C1=20,T1=100,D1=100

P2:C2=30,T2=145,D2=145

P3:C3=68,T3=150,D3=150

总使用率为:20/100+30/145+68/150=0.86

$3\times(2^{1/3}-1)=0.78<0.86$

9.为了有利于多进程读写磁盘，磁盘条带应该大些。
因为如果条带相对比较大，则一个I/O请求可能只包括对一个磁盘的访问，多个正在等待的I/O请求可以并行地处理，从而减少了每个请求的排队时间。

10.饥饿：一个可运行的进程尽管能继续执行，但被调度器无限期的忽视，而不能被调度执行。

死锁：两个或两个以上的进程因其中的每个进程都在等待其他进程做完某些事情而不能继续执行。

二.

证明：用反证法。

假设会出现死锁，出现死锁的情形为：进程i占有资源Ri,并请求资源Rj，而进程Rj占有资源Rj,并请求资源Ri。

由于对资源进行了排序，若进程占有资源R，则该进程只可能申请编号大于R的资源，

从而对于进程i,得到Ri<Rj

对于进程j,得到Rj<Ri

上述两式产生矛盾，因而死锁不会产生。

三.

1.While 语句保证了在比较票号之前没有人正在拿票，即当对 number[j]进行写操作时，不允许其他人对 number[j] 进行读操作，因此，这里实现了对 number[j]的读-写互斥。

2.证明：

每个要进入临界区的进程都要先得到一个票号，该票号是当前拿到票的号的最大值加 1。每个要想进入临界区的进程，都会用自己的票号和当前已拿到票的进程比较，只有满足(number[i],i)<(number[j],j)(对任意进程 j),的进程 i 才可以进入临界区，当它离开临界区后，它会将 number[i]置为 0，这保证了其他的进程中有一个进程的票号成为当前最小的，则它可以进入临界区。因而，在有限时间内，任何拿到票想进入临界区的进程最终都会成为最小的，因此可以进入临界区。综上，该算法是公平的。

四.

1.L.Lamport 算法中时间戳是用来排序的，每个进程在发送消息前，都会先将本地时钟加 1，然后发送一条打个时间戳的消息给其他进程，这保证了，消息的排序在各个进程中唯一的。

该算法中的时间戳只是用于更新，并无排序功能。例如,进程 Pi 中的 request[j]记录的只是最后一次收到来自进程 Pj 的请求消息的时间戳，而token[j]记录的只是令牌最近一次光顾进程 Pj 的时间戳。只有满足 request[j]>token[j]的进程 Pj 才可以进入临界区。

2.进程 Pi 中的 request[j]记录的是最后一次收到来自进程 Pj 的请求消息，该消息应该是最新的，Max 操作保证了收到的 request[j]一定是进程 Pj 最新发出的，而不是以前延迟的旧消息。