2.3.2 临界区

怎样避免竞争条件？实际上凡涉及共享内存、共享文件以及共享任何资源的情况都会引发与前面类似的错误，要避免这种错误，关键是要找出某种途径来阻止多个进程同时读写共享的数据。换言之，我们需要的是互斥（mutual exclusion），即以某种手段确保当一个进程在使用一个共享变量或文件时，其他进程不能做同样的操作。前述问题的症结就在于，在进程A对共享变量的使用未结束之前进程B就使用它。为实现互斥而选择适当的原语是任何操作系统的主要设计内容之一，也是我们在后面几节中要详细讨论的主题。

避免竞争条件的问题也可以用一种抽象的方式进行描述。一个进程的一部分时间做内部计算或另外一些不会引发竞争条件的操作。在某些时候进程可能需要访问共享内存或共享文件，或执行另外一些会导致竞争的操作。我们把对共享内存进行访问的程序片段称作临界区域（critical region）或临界区（critical section）。如果我们能够适当地安排，使得两个进程不可能同时处于临界区中，就能够避免竞争条件。

尽管这样的要求避免了竞争条件，但它还不能保证使用共享数据的并发进程能够正确和高效地进行协作。对于一个好的解决方案，需要满足以下4个条件：

1) 任何两个进程不能同时处于其临界区。

2) 不应对CPU的速度和数量做任何假设。

3) 临界区外运行的进程不得阻塞其他进程。

4) 不得使进程无限期等待进入临界区。

从抽象的角度看，我们所希望的进程行为如图2-22所示。图2-22中进程A在T_1时刻进入临界区。稍后，在T_2时刻进程B试图进入临界区，但是失败了，因为另一个进程已经在该临界区内，而一个时刻只允许一个进程在临界区内。随后，B被暂时挂起直到T_3时刻A离开临界区为止，从而允许B立即进入。最后，B离开（在时刻T_4），回到了在临界区中没有进程的原始状态。

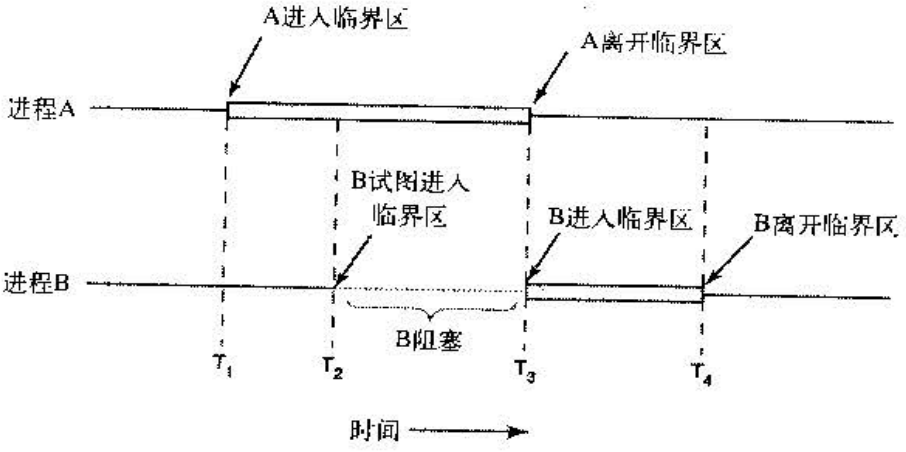

图2-22 使用临界区的互斥

2.3.3 忙等待的互斥

本节将讨论几种实现互斥的方案。在这些方案中，当一个进程在临界区中更新共享内存时，其他进程将不会进入其临界区，也不会带来任何麻烦。

1. 屏蔽中断

在单处理器系统中，最简单的方法是使每个进程在刚刚进入临界区后立即屏蔽所有中断，并在就要离开之前再打开中断。屏蔽中断后，时钟中断也被屏蔽。CPU只有发生时钟中断或其他中断时才会进行进程切换，这样，在屏蔽中断之后CPU将不会被切换到其他进程。于是，一旦某个进程屏蔽中断之后，它就可以检查和修改共享内存，而不必担心其他进程介入。

这个方案并不好，因为把屏蔽中断的权力交给用户进程是不明智的。设想一下，若一个进程屏蔽中断后不再打开中断，其结果将会如何？整个系统可能会因此终止。而且，如果系统是多处理器（有两个或可能更多的处理器），则屏蔽中断仅仅对执行disable指令的那个CPU有效。其他CPU仍将继续运行，