同步互斥机制

- 进程的并发执行
- 进程互斥
- 进程同步
- 信号量及PV操作
- 经典的IPC问题

在这部分里头我们主要介绍进程的并发执行

并发环境下进程的特征

进程并发执行

IPC 问题 我们首先来介绍进程的并发执行

问题的提出

并发是所有问题产生的基础

并发是操作系统设计的基础

并发是所有问题产生的基础 并发也是操作系统设计的一个基础

从进程的特征出发

并发

- 进程的执行是间断性的
- 进程的相对执行速度不可预测

共享

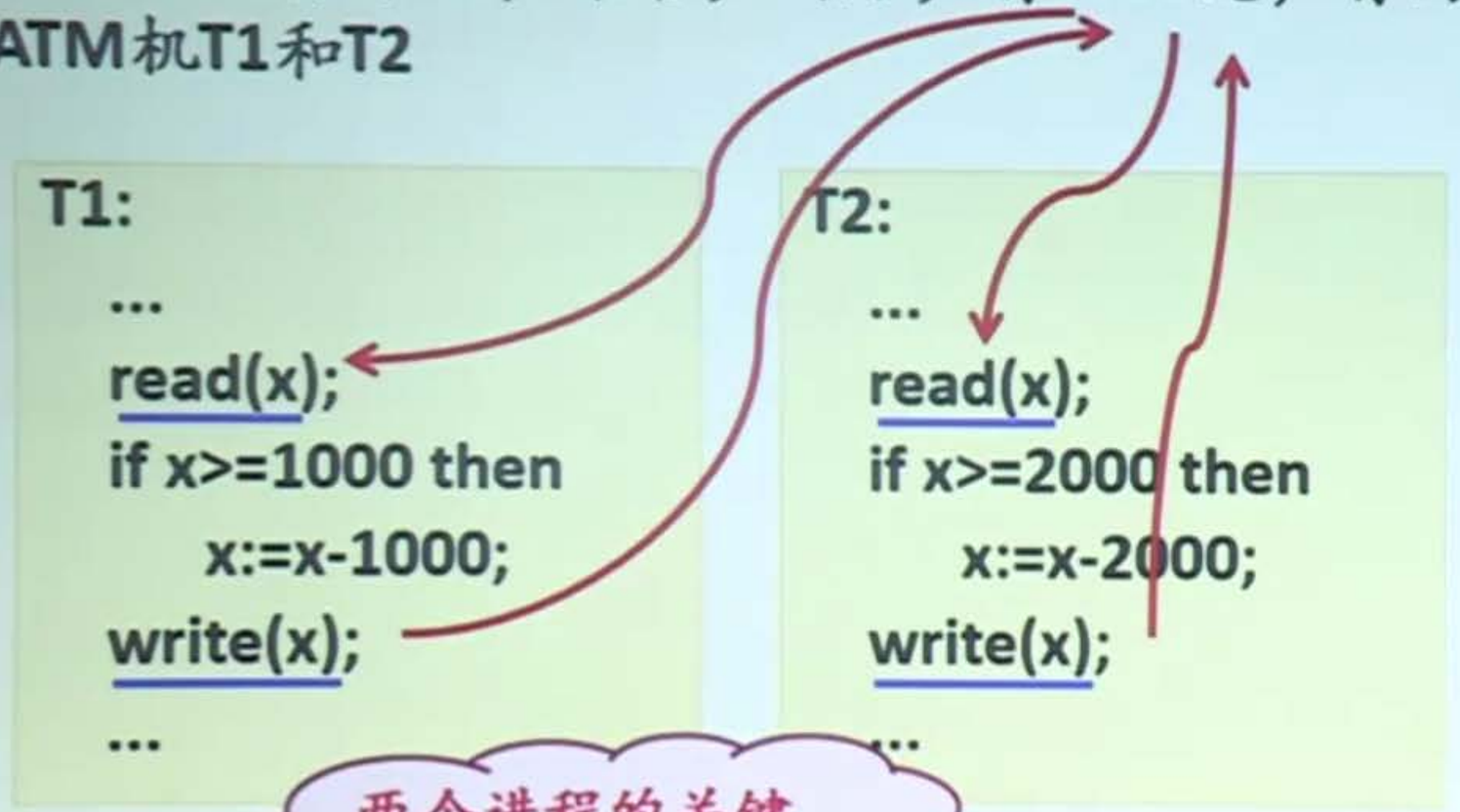
- 进程/线程之间的制约性

不确定性

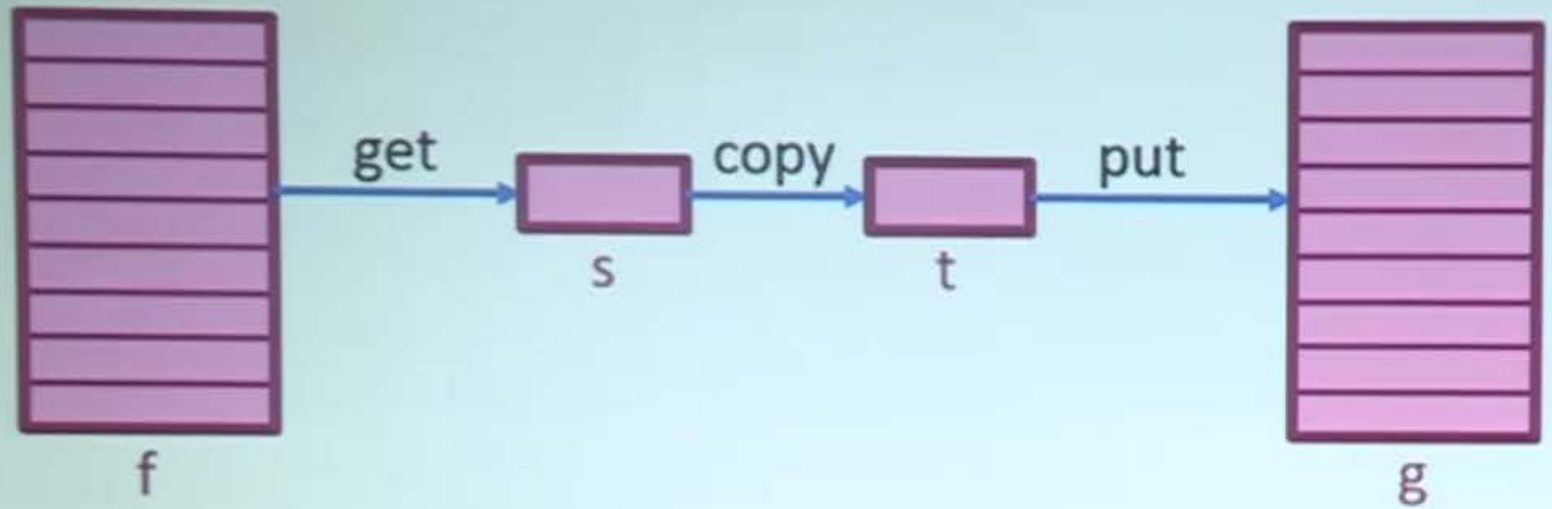
- 进程执行的结果与其执行的相对速度有关，是不确定的

与时间有关的错误---例子1

某银行业务系统，某客户的账户有5000元，有两个ATM机T1和T2



与时间有关的错误——例子2



场景

get、copy和put三个进程并发执行

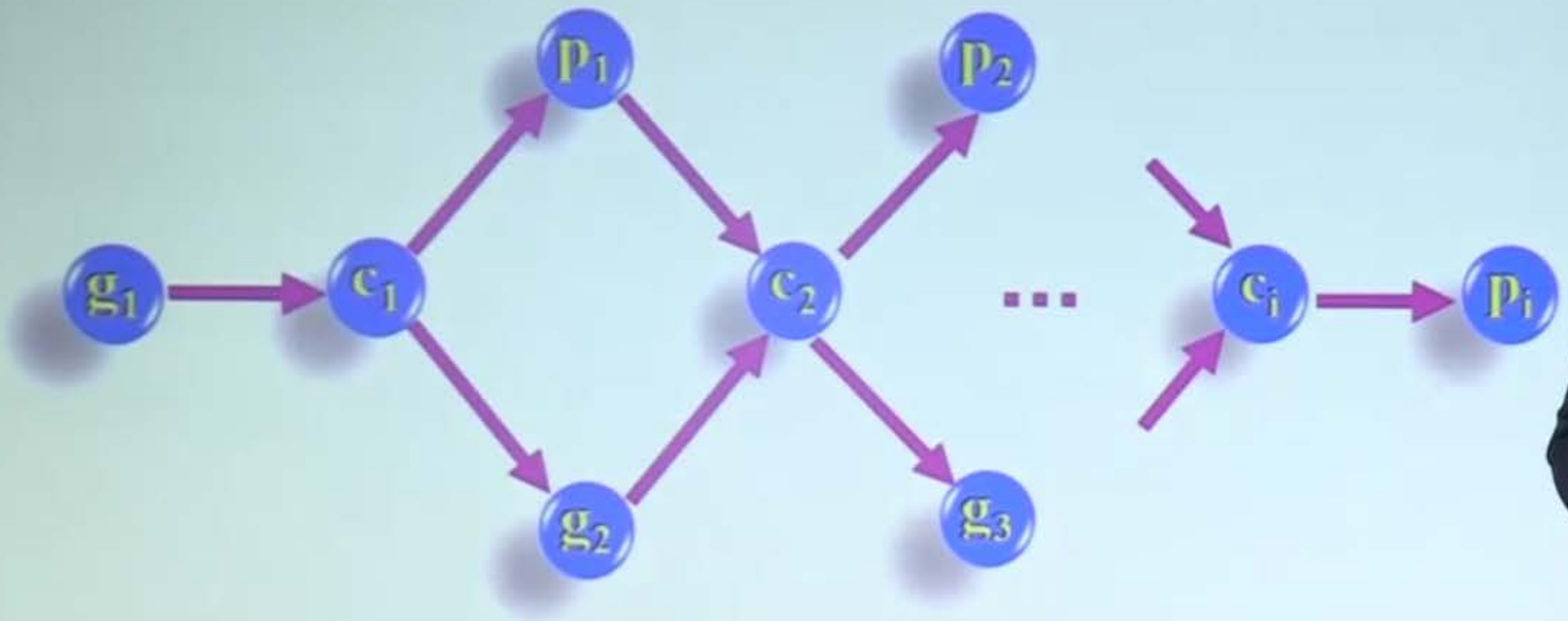
这个缓冲区里头 所以这是可能会出现各种各样的情况。

并发执行过程分析

	f	s	t	g	
当前状态	(3,4,...,m)	2	2	(1,2)	
可能的执行（假设g,c,p为get,copy,put的一次循环）					
g,c,p	(4,5,...,m)	3	3	(1,2,3)	√
g,p,c	(4,5,...,m)	3	3	(1,2,2)	X
c,g,p	(4,5,...,m)	3	2	(1,2,2)	X
c,p,g	(4,5,...,m)	3	2	(1,2,2)	X
p,c,g	(4,5,...,m)	3	2	(1,2,2)	X
p,g,c	(4,5,...,m)	3	3	(1,2,2)	X

设信息长度为m，有多少种可能性?

进程前趋图



并发环境下进程间的制约关系

制约关系应该满足这样一个 我们说的前趋图，这样才能保证不出错误

0:00 大家好，今天我给大家带来的是操作系统原理课的第五讲 同步互斥机制，今天我们讲同步互斥机制的第一部分 在这部分里头我们主要介绍进程的并发执行 介绍进程的同步和互斥这两个基本概念 然后我们介绍一种同步机制，信号量及 PV 操作 最后我们用信号量及 PV 操作来解决两个经典的 IPC 问题 我们首先来介绍进程的并发执行 我们这一讲的 问题都是由于并发所引起的 并发是所有问题产生的基础 并发也是操作系统设计的一个基础 因此我们从进程的这个并发特性 出发来看一下，在并发环境下 进程的执行会带来什么问题。 进程的几个基本特性 我们在以前介绍过：并发、共享 以及随机性所带来的一种不确定性。所谓并发呢 指的是进程的执行呢是一种间断性的 也就是每个进程在它的生命周期期间 一会儿上 CPU 执行，一会儿由于某种原因暂停执行 所以每个进程的执行是间断性的 由于这种间断性使得进程的相对执行速度是不可预测的 由于有进程调度，有其他事件的发生 每个进程上 CPU 执行 可能执行一段时间停止，然后再接着执行，所以整个执行的时间是不可预测的 另外在一个并发环境下 多个进程或者线程之间呢会共享某些资源 那么在这些资源的使用过程中 它会产生进程之间的一种制约性 比如说当一个进程享用打印机这个资源 那么另外一个进程在第一个进程没有释放 这个资源的前提之下，那么就得不到这个资源，那就得等待 因此在一个并发环境下，多个进程的执行呢会带来一种制约 那么进程的执行 的结果和它的相对执行速度是有关系的，因此 在不同的执行顺序的情况下，那么进程的执行结果也是不确定的 那么这是关于从进程的特征出发，我们看到带来了一些问题 下面呢我们举几个例子来介绍会带来哪些具体的问题 第一个例子呢，我们是一个银行的系统 那么这个银行系统里头，假设某个客户他的账户上有 5 千元 那么有两个 ATM 机，T1 和 T2 我们来看看 T1 和 T2 如果分别有两个 人在这两台机器上去取钱的话，那么会发生什么情况 好，那么假设 T1 首先 把这个账户的 5000 元的这个内容，值读入到自己的一个本地的变量里头 那么 T2 也可以这么做。 也就是说这 5000 元 都分别在 T1、 T2两个终端的这个变量 x 里头保存起来，然后它们分别判断 账户上的余额足够 不足够？好，那么如果足够的话，那么 T1 这个终端呢取了 1000 块钱，T2 取了 2000 块钱，然后分别把余额 要写回到这个账户上去。 那么如果在写回去的时候 不管是 T1 先写回去，还是 T2 先写回去 那么如果我们没有对这个账户加以控制 使得 T1、 T2 这两个程序当中的一些关键活动 它们中间出现了交叉，那么就可能会出现 与时间有关的错误。比如说 T1 先把 x 的值减完 1000 之后，再把它写回到账户上 那么这个时候账户上应该是 4000 元 可是 T2 呢因为刚才已经把

5000 元读到了 x，所以对 x 减去 2000 之后呢 那么再把它写回去的时候呢，就是相当于把 3000 元写回去 那么当然如果反过来，T2 先写 回去，T1 再写，那么账户上的余额呢就变成 4000 元 如果是 T1 先写，T2 再写，那么账户上余额呢就是 3000 元 那么不管是哪种情况，由于这两个进程它们的关键活动 中间出现了交叉，因此账户上的余额是不对的 那我们再看一个例子 这个例子呢是有这么三个进程 有这么一个场景，三个进程 那么这里头还有几个缓冲区 f 缓冲区和 g 缓冲区可以分别存放多个数据 而 s 和 t 缓冲区呢只能放一个数据 那么现在有三个进程 这三个进程呢都是循环执行的 我们来看第一个进程 get，get 的工作是 把 f 里头的数据取一个送到 s 里 那么 copy，copy 是从 s 里取一个数据送到 t 里 而 put 是从 t 里头取一个数据往 g 里去送 那么我们假设，那么 get 可以循环很多次 当它第一个把数据读到了 s 里头以后，那么再读第二次的时候 由于 s 里的数据假设还没有被 copy 给取走 那么第二次往 s 里送的数据实际上呢就把 前面送的数据给覆盖掉了。 那么为什么 会出现这种情况呢？那么因为我们这是一个在一个并发环境下执行的 那么很可能任何一个进程 都可能会被调度上 CPU，那么也可能会出现 put 先被调度上 CPU，而 t 里的内容还没有 准备好，那么 put 就去做了，那么就可以，就可能说把一个 没有的数据，不需要的数据送到了这个 g 这个缓冲区里头 所以这是可能会出现各种各样的情况。 我们来做一下具体的分析 假设当前的一个状态 是现在这么一个场景，也就是换句话说， 已经把这个 f 这个缓冲区当中的 1 和 2 这两个数据取出来，送到了 g 里头 那么这样呢，我们可以看到 f 里头还剩下 3 到 m 然后 s 和 t 里头分别是数据是 2 这是一个正确的一个结果。 我们从这个当前状态出发 看一看有哪些执行的路径的可能 那我们这里头先为了简单啊，我们假设 g,c,p 分别为 get,copy 和 put 的一次循环，一次循环过程 因此从当前状态出发，我们可能有这样一种 组合的执行过程。 那么 get, copy put 各执行一个循环。 那么得到的结果呢就是这样一个结果 就是把 3 这个数据运到了送到了 g 这个buffer 里头去 那么这是一个正确的结果，因为 s 和 t 里头分别也放了是 3 这个数据 但是如果还从这样一个当前状态出发，我们来看一下 如果是 g,p,c 这样一个执行的轨迹呢 那我们会看到最后 g 这个缓冲区里头得到的是 1,2,2 也就是得到两次 2 这个数据，当然前面的 几个缓冲区数据还是对的，但是这个结果就是错的 那么这是一种错误的情况，那么由于 g,c,p 这三个可能有各种各样的 组合，我们来看一下剩下的几种组合 分别都会带来一些错误，有的

是最后结果的错误 那么还有一些中间的错误。 所以这就是在一个并发环境下 三个并发的进程在执行过程中 由于调度，由于其他的因素会造成这样一个 错误，这样一个错误，当然我们这里头 只是按照这样一个循环来讨论这样一个执行轨迹的 如果是某一个进程，比如说 get，它可能循环两次， 或者是 copy 循环了 3次 put 循环了一次，那么各种组合加起来会有非常多的情况 那么我们这里头提一个问题，如果信息长度为 m，那么到底有多少种组合呢？ 大家可以去看一下。 那么这里头我们这是以语句为单位来讨论 那么还没有讨论到指令这一级，如果是指令这一级，可能的这种组合就会更多 那么在刚才这样一个场景下 那么是由于这三个进程它们之间是有制约关系的 而没有满足这种制约关系就会出现刚才的错误 现在我们来看一看这三个进程的制约关系是什么样子的 我们用一个进程的前趋图来表示 那么 g1 代表的是 get 执行一个循环 第一个循环，c1 呢表示的是 copy 执行的第一个循环。 那么我们可以看到当 get 执行完第一个循环之后，只能够 copy 执行它的第一个循环。 当 copy 执行完第一个循环之后呢，那么后面呢 是 put 执行第一个循环，还是 get 执行第二个循环，都可以，这两个可以顺序可以 颠倒，可以任意的顺序。 但是它们两 都执行完了，才能够去执行 copy 的第二个循环 因此这三个进程它们之间的这种 制约关系应该满足这样一个 我们说的前趋图，这样才能保证不出错误