时钟中断和时间片轮转调度

同济大学计算机系　操作系统作业　　2023-11-20
学号　2151769　　姓名　吕博文

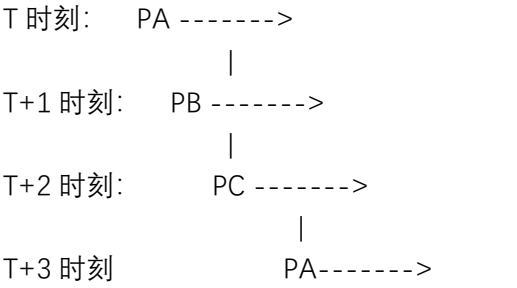
Part 1、Unix V6 时间片轮转调度的实现

习题： Unix V6++系统中存在 3 个 CPU bound 用户态进程 PA、PB 和 PC。3 个进程静态优先数相等：100，p_cpu 是 0。Process[8]、[5]、[9]分别是 PA 、PB、PC 进程的 PCB。T 时刻是整数秒，PA 先运行。

1、 画进程运行时序图。

2、 T+1 时刻，PA 进程用完时间片放弃 CPU。何时，PA 进程会再次得到运行机会？简述 T+1 时刻系统怎样保护 PA 进程的用户态 CPU 执行现场，下次再运行时系统如何恢复 PA 进程的用户态 CPU 执行现场。

（1）进程运行时序图：



（2）T+1 时刻，PA 进程放弃 CPU，等到 T+3 时刻再次得到运行机会；

T+1 时刻，当 PA 进程放弃 CPU 时系统需要保存 PA 进程的用户态执行现场，这通常包括保存 CPU 寄存器、程序计数器 PC 等信息，系统将这些信息保存在 PA 进程的系统控制块 PCB 中；

再次运行 PA 进程时，系统会从 PA 进程的 PCB 中恢复先前保存的用户态 CPU 执行现场，这包括将寄存器值和程序计数器设置为之前保存的值，以便 PA 进程可以从上一次中断的地方继续执行。

参考：PPT24 的表格和对这张 PPT 的讲解。

情景分析：
假设系统中存在4个用户态的进程PA、PB、PC、PD，这些进程一直运算，不IO，不执行系统调用。进程的静态优先数相等：100，p_cpu是0。Process[5]、[7]、[8]、[9]分别是PA、PB、PC和PD进程的PCB。T时刻是整数秒，PA先运行。观察这些进程如何轮流使用CPU。

- p-pri = min {127 ,进程的静态优先数 + （p_cpu/16）}

p_cpu

	T	T+1	T+2	T+3	T+4	T+5			
PA	0	40	20	0	0	40			
PB	0	0	40	20	0	0			
PC	0	0	0	40	20	0	.....		
PD	0	0	0	0	40	20			

SCHMAG = 20
HZ = 60

p_pri

	T	T+1	T+2	T+3	T+4	T+5			
PA	100	102	101	100	100	102			
PB	100	100	102	101	100	100			
PC	100	100	100	102	101	100	.....		
PD	100	100	100	100	102	101			

习题的解答

- p-pri = min {127 ,进程的静态优先数 + （p_cpu/16）}

p_cpu

	T	T+1	T+2	T+3	T+4	T+5			
PA	0	40	20	0	0	40			
PB	0	0	40	20	0	0			
PC	0	0	0	40	20	0	.....		
PD	0	0	0	0	40	20			

SCHMAG = 20
HZ = 60

p_pri

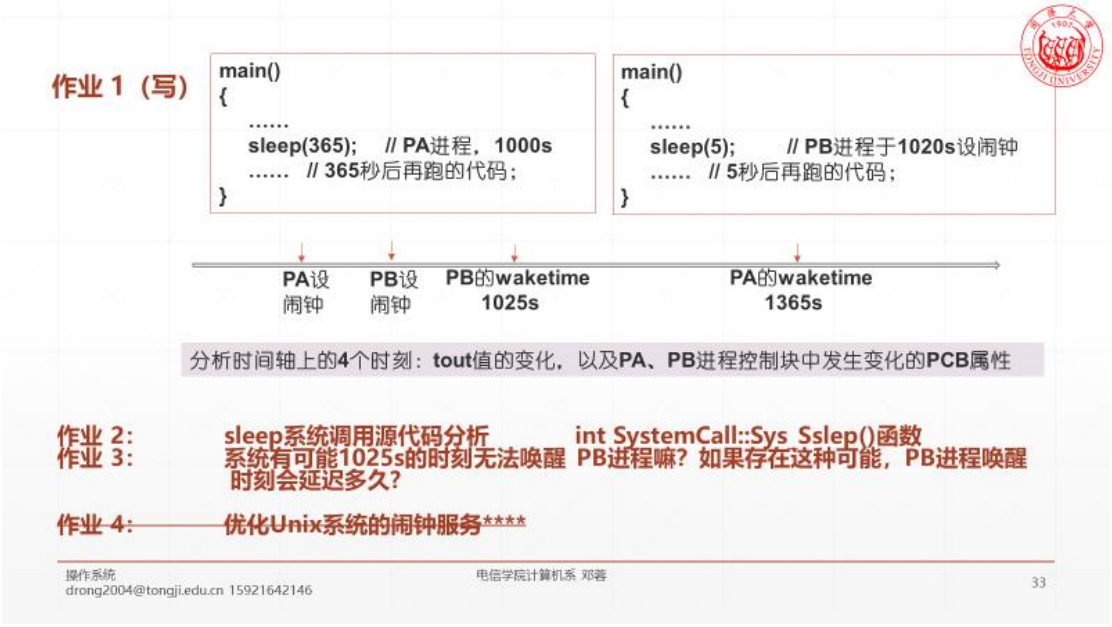
	T	T+1	T+2	T+3	T+4	T+5			
PA	100	102	101	100	100	102			
PB	100	100	102	101	100	100			
PC	100	100	100	102	101	100	.....		
PD	100	100	100	100	102	101			

- 时刻T+1，整数秒时钟中断。被中断的现运行进程PA 用户态运行。[T, T+1]，PA连续使用CPU，被时钟中断60次，p_cpu=60。其余进程未运行，p_cpu没有增加。 T+1秒，所有进程p_cpu减20。PA、PB、PC、PD进程的p_cpu值分别为40，0，0，0。计算得进程优先级p_pri分别为102，100，100和100。现运行进程PA的优先数增加了（原本是100，现在是102），用完时间片，runrun变1。
- 时钟中断返回用户态，例行调度，runrun是1，PA进程让出CPU。系统选优先级最高的就绪态进程PB，last_select=PB。PB上台执行应用程序，成为[T+1,T+2]时段的现运行进程。

在未来1秒之内，一直是PB运行。每当系统发生时钟中断，PB就陷入核心态，花一点点时间执行时钟中断处理程序。 T+2秒，PB用尽时间片，将CPU让出来。系统从last_select+1开始遍历Process数组，找优先权最高的就绪态进程。PC上台运行。

可以看到，每4s是一个周期，PA、PB、PC、PD时间片轮转，依次执行。第5s，PA进程已经连续3秒未得到运行，优先数p_pri已降至100，成为优先级最高的进程，再次得到运行。

Part 2、定时器服务



分析 1025 时刻（1）系统调度操作（2）Sleep 系统调用下半部对 tout 变量的维护。

（1）1025s，整数秒进入时钟中断，检查到 PB 进程设置的闹钟时间耗尽，系统唤醒 PB 进程，PB 进程进入可执行队列，等待系统选择它进入执行状态。

（2）Sleep 系统调用用于使进程进入休眠状态，等待一定的时间。在 UNIX 系统中，通常使用 tout 变量来维护所有进程的 waketime 的最小值。

tout 变量记录了下一个将唤醒的进程的触发时间。当一个进程调用 Sleep 时，它会将 waketime 设置为当前时间加上休眠的时间。

如果一个进程设置了一个比当前 tout 更早的 waketime，系统可能会更新 tout 的值，以便在该时间点之前唤醒该进程。

Part 3、综合题

全嵌套中断处理模式。低优先级中断处理程序运行时，系统响应高优先级中断处理请求。已知，时钟中断优先级高于磁盘中断优先级。假设：900s，PA 进程执行 sleep（100）入睡。998s，PB 进程执行 read 系统调用，读磁盘文件。1000s，现运行进程 PX 正在执行应用程序。PA 设置的闹钟到期、PB 读取的磁盘文件数据 IO 结束。分析 1000s，系统详细的调度过程。分两种情况考虑：

1、 先响应磁盘中断

T=1000s： 现运行进程为 PX。此时，PA 的 sleep(100)中断尚未到期，PB 正在执行 read 系统调用，导致磁盘 I/O 操作，而时钟中断优先级高于磁盘中断。

T=1001s： 由于磁盘中断请求，系统响应了磁盘中断。中断处理程序可能会暂停 PB 进程的执行，保存 PB 的执行现场，并开始处理磁盘中断。在中断处理期间，时钟中断可能被屏蔽，以防止中断嵌套。

T=1002s： 磁盘中断处理完毕后，系统可能会进行调度决策。由于此时 PA 的

sleep(100)中断到期，PA 可能被唤醒，被移动到可运行队列。由于时钟中断被屏蔽，时钟中断不会打断这一过程。

T=1003s： 调度程序选择运行 PA，PA 开始执行，PB 仍然处于阻塞状态。

总体而言，由于磁盘中断的优先级较低，系统在响应磁盘中断后，会立即检查更高优先级的时钟中断，确保及时唤醒 PA 进程。然后，PA 进程得到运行机会。

2、 先响应时钟中断

T=1000s： 现运行进程是 PX。此时，PA 的 sleep(100)中断已经到期，可能触发时钟中断。

T=1001s： 由于 PA 的 sleep(100)中断到期，系统可能响应时钟中断。中断处理程序会保存 PX 的执行现场，并进行时钟中断的处理。在中断处理过程中，PB 的 read 系统调用可能仍在进行。

T=1002s： 时钟中断处理完毕后，系统可能会进行调度决策。由于 PA 的 sleep(100)中断已经到期，PA 可能会被唤醒，移动到可运行队列。此时，系统可能选择运行 PA，因为时钟中断已经得到响应，可能继续 PX 的执行。PB 的 read 系统调用可能仍在进行。

T=1003s： PA 进程得到运行机会，继续执行。PB 的 read 系统调用可能仍在进行，PX 也可能继续执行。

总体而言，在这种情况下，由于 PA 的 sleep(100)中断到期，系统先响应时钟中断，保护 PX 的执行现场。然后，系统可能选择运行 PA 进程，继续 PX 的执行。PB 的 read 系统调用可能仍在进行。这种情况下，时钟中断的处理被插入到了两个不同进程的执行中。