# 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。

因为PA先运行，且PC最先被遍历到，所以PA之后紧接着运行的是PC，所以进程运行时序图如下：

T时刻： PA ------->

|

T+1时刻： PC------->

|

T+2时刻： PB ------->

|

T+3时刻 PA------->

（1）进程运行时序图：

T时刻： PA ------->

|

T+1时刻： PB ------->

|

T+2时刻： PC ------->

|

T+3时刻 PA------->

（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；

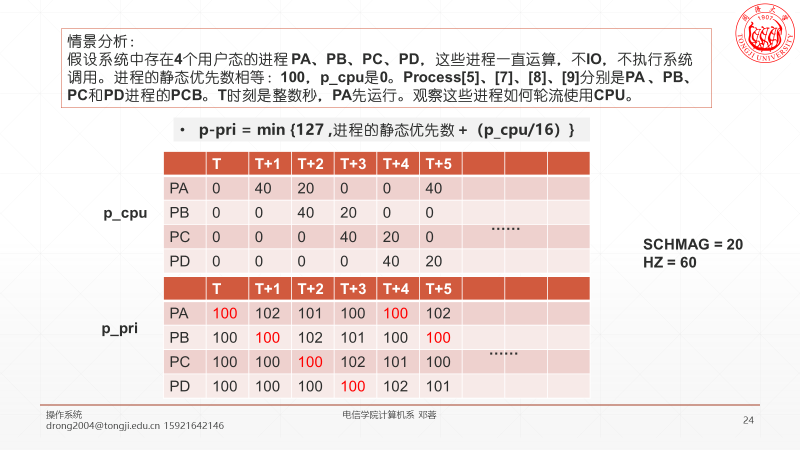
T+1时刻，当PA进程放弃CPU时系统需要保存PA进程的用户态执行现场，这通常包括保存CPU寄存器、程序计数器PC等信息，系统将这些信息保存在PA进程的系统控制块PCB中；

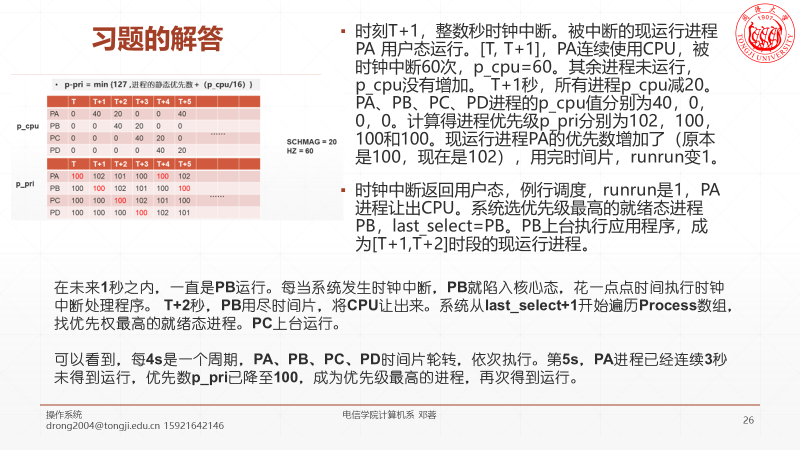
再次运行PA进程时，系统会从PA进程的PCB中恢复先前保存的用户态CPU执行现场，这包括将寄存器值和程序计数器设置为之前保存的值，以便PA进程可以从上一次中断的地方继续执行。

第一次现场保护中断响应过程将PA的用户态寄存器压入自己的核心栈，第二次现场保护中断返回用户态前，进行例行调度PA执行switch()放弃CPU将switch栈帧定位指针ESP，EBP存入自己的user结构，PA成为普通的就绪态进程

恢复现场时首先恢复核心态现场，PC放弃CPU后，switch函数选中PA，从它的user结构中取出T+1秒保护的switch栈顶定位指针并赋值ESP和EBP，第二步恢复用户态现场，switch函数返回，栈帧出栈，PA执行时钟中断处理入口函数例行调度之后的返回部分，这一步会弹出时钟中断入口程序栈帧中保存的用户态寄存器，之后IRET，PA返回用户态，继续执行应用程序。

参考：PPT24的表格和对这张PPT的讲解。





## Part 2、定时器服务

日程表

低可信度描述已自动生成

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。

1. 假设1025s先运行进程时PX，运行在用户态

响应时钟中断，PX执行时钟中断处理程序，先前用户态，调整系统时钟time,唤醒waketime到期的PA,PB进程。中断返回前，例行调度，PX优先级不及被唤醒的进程，被剥夺，放弃CPU，随后，PA、PB分别执行自己的sleep系统调用后半段。

PA执行系统调用后半段，waketime未到期，设置tout(1365)之后，再次入睡

PB执行系统调用后半段，waketime到期，sleep系统调用返回，恢复应用程序优先数，让出CPU

这两个系统调用下半段全部执行完毕后，CPU开始执行应用程序。PB，PX还有其他进程时间片轮转，直至有进程执行系统调用或1365s系统响应时钟中断唤醒PA

系统调用下半段执行期间，有可能 PA 先运行，也有可能 PB 先运行（1）如果 PA 先运行，入睡后 PB 执行 sleep 系统调用下半段，完成后，系统开始执行应用程序（2） 如果 PB 先运行，返回用户态前让出 CPU 给 PA 执行系统调用下半段，PA 入睡后系统 开始执行应用程序，PB、PX 和其它进程时间片轮转。

1. 假设1025s先运行进程PY运行在核心态

PY响应时钟中断，先前是核心态，直接返回，不调整系统时钟time，不会唤醒PA，上述操作延迟直至先前是用户态的第一个时钟中断

1. 1025s，系统idle

执行上述红色操作。时钟中断处理程序执行完毕后，0#进程选中 PA 或 PB，让出 CPU，恢复睡眠等待 RunIn 或 RunOut 的状态。

## 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。

1、先响应磁盘中断

1000s。内核任务执行次序：磁盘中断处理程序。。。被打断，时钟中断处理程序（没有调整 time，不会唤醒 PA），磁盘中断处理程序恢复执行，read 系统调用后半段。全部完成后， PX、PB 轮流使用 CPU 执行应用程序。

1000s 之后，先前是用户态的第一次时钟中断。内核任务执行次序：时钟中断处理程序（调 整 time，唤醒 PA），sleep 系统调用后半段。全部完成后，PX、PA、PB 轮流使用 CPU 执行 应用程序。

具体调度细节：

1000s。 现运行进程 PX 执行磁盘中断处理程序，唤醒 PB、 RunRun++。其间，嵌套执行中断优 先权更高的时钟中断处理程序。后者先前核心态，不调整 time，不会唤醒 waketime 到期的 PA。磁盘中断处理程序运行结束后，现运行进程 PX 被剥夺、放弃 CPU。SRUN 进程集合 { PX，优先数>=100；PB，优先数==-50 }。 PB 优先级高，被选中，执行 read 系统调用后半段。完成后恢复其应用程序优先数 （p\_pri >= 100），让出 CPU。SRUN 进程集合 { PX，优先数>=100；PB，优先数>=100 }。 PB 和 PX 轮流使用 CPU 执行应用程序。

1000s 之后，先前是用户态的第一次时钟中断。 现运行进程执行时钟中断处理程序，调整 time，唤醒 PA、 RunRun++。时钟中断处理 程序运行结束后，现运行进程被剥夺、放弃 CPU。SRUN 进程集合 { PX，优先数>=100；PA， 优先数==90；PB，优先数>=100 }。 PA 优先级最高，被选中，执行 sleep 系统调用后半段。waketime 到期，PA sleep 系统 调用返回，恢复应用程序优先数（p\_pri >= 100）之后，让出 CPU。SRUN 进程集合 { PX， 优先数>=100；PA，优先数>=100；PB，优先数>=100 }。PA、PB、PX 轮流使用 CPU 执行 应用程序

1. 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。

1000s，内核任务执行次序：时钟中断处理程序。。。被打断，硬盘中断处理程序，时钟中断 处理程序恢复执行，read 系统调用后半段，sleep 系统调用后半段。全部完成后，PX、PA 和 PB 轮流使用 CPU 执行应用程序。

具体调度细节： 现运行进程 PX 执行时钟中断处理程序, 更新 Time::time，STI、EOI 后, 中断嵌套，响应 磁盘中断。 磁盘中断处理程序唤醒 PB、 RunRun++。先前核心态，不调度，磁盘中断处理程序运 行结束后，PX 返回、继续执行被打断的时钟中断处理程序。 时钟中断处理程序唤醒 PA、 RunRun++(值是 2）。先前用户态，例行调度。RunRun 非 0，现运行进程 PX 被剥夺、放弃 CPU。SRUN 进程集合 { PX，优先数>=100；PA，优先数 =90；PB，优先数=-50 }。 PB 优先级最高，被选中，执行 read 系统调用后半段。完成后恢复其应用程序优先数 （p\_pri >= 100），让出 CPU。SRUN 进程集合 { PX，优先数>=100；PA，优先数=90；PB， 优先数>=100 }。 PA 优先级最高，被选中，执行 sleep 系统调用后半段。waketime 到期，PA sleep 系统 调用返回，恢复应用程序优先数之后，让出 CPU。SRUN 进程集合 { PX，优先数>=100；PA， 优先数>=100；PB，优先数>=100 }。PA、PB、PX 轮流使用 CPU 执行应用程序