

0#进程和盘交换区

例题： 假设某UNIX 系统中只存在4 个进程，分别是：睡眠中的0#进程，正在CPU 上执行的pa 进程，内存中就绪状态的pb 进程（CPU bound，不会IO）， 盘交换区上低优先级睡眠的pc 进程。已知，当前时刻T=1000.5 s，内存空间已满。请尽量详细地分析以下时刻系统中与进程调度和中断相关的行为，并修改下表中的相关字段。

- (1) 1000.5 s，现运行进程pa 执行read（读文件）系统调用
- (2) 1001.5 s， pc等待的IO操作完成

| 序号 | 占用空间 | 优先数 | 状态 | 位置 | p_time |
|----|------|------|------------|--------|--------|
| 0# | - | -100 | 高睡（RunOut） | SLOAD | - |
| pa | 100K | 103 | 执行 | SLOAD | 2 |
| pb | 10K | 101 | 就绪 | SLOAD | 2 |
| pc | 60K | 90 | 低睡 | ~SLOAD | 3 |

解：

- (1) 1000.5 s，现运行进程pa 执行read（读文件）系统调用
- 1000.5s，现运行进程pa执行read（读文件）系统调用，高优先权入睡（p_stat = SSLEEP, p_pri = -50），放弃CPU。随后，系统选中pb进程，因为它是内存中就绪的唯一进程。表格修改如下：

| 序号 | 占用空间 | 优先数 | 状态 | 位置 | p_time |
|----|------|------|------------|-------|--------|
| 0# | - | -100 | 高睡（RunOut） | SLOAD | - |
| pa | 100K | -50 | 高睡 SSLEEP | SLOAD | 2 |

| | | | | | |
|----|-----|-----|----|--------|---|
| pb | 10K | 101 | 执行 | SLOAD | 2 |
| pc | 60K | 90 | 低睡 | ~SLOAD | 3 |

1001.0s, 现运行进程 pb 执行整数秒时钟中断处理程序, 会++所有进程的 p_time。修正用户态进程的优先数, pb此轮执行了 0.5s, p_cpu增加30, 整数秒时减20。总的效果: p_cpu加了10。整除16后, p_pri可能是102, 也可能是101。表格修改如下:

| 序号 | 占用空间 | 优先数 | 状态 | 位置 | p_time |
|----|------|---------|-------------|--------|--------|
| 0# | - | -100 | 高睡 (RunOut) | SLOAD | - |
| pa | 100K | -50 | 高睡 SSLEEP | SLOAD | 3 |
| pb | 10K | 101/102 | 执行 | SLOAD | 3 |
| pc | 60K | 90 | 低睡 | ~SLOAD | 4 |

(2) 1001.5 s, pc等待的IO操作完成

1001.5 s。

- 现运行进程pb执行中断处理程序, 唤醒pc, 同时唤醒 0#进程。
- 中断处理程序执行完毕后, pb将CPU让给0#进程执行sched ()。0#进程为盘交换区上的pc进程分配内存空间。首先找内存中低睡 (SWAIT) 或SSTOP (暂停) 的进程, 未果。pc的p_time达到4s, 值得为它换出内存中高优先级或就绪的进程。pa的p_time达到3s, 入选。0#进程首先换出pa, 之后换入盘交换区上的pc (放在pa原先占据的内存区域)*。
- 完成对换操作后, 0#进程 sleep(&Runout,-100)入睡, 执行swtch () 将CPU让给内存中就绪的pc进程。pc执行系统调用后半部。完成后,
- pc和pb时间片轮转, 轮流执行应用程序。

表格修改如下:

*处的细节, 不要求掌握。就是换入、换出过程中, 0#进程会睡眠等待IO结束的, 此时pb使用CPU, 是现运行进程。一旦IO完成, pb立即将CPU让给0#进程, 因为它优先级最高。

| 序号 | 占用空间 | 优先数 | 状态 | 位置 | p_time |
|----|------|---------|-------------|--------|--------|
| 0# | - | -100 | 高睡 (RunOut) | SLOAD | - |
| pa | 100K | -50 | 高睡 SSLEEP | ~SLOAD | 0 |
| pb | 10K | 101/102 | 执行 SRUN | SLOAD | 3 |
| pc | 60K | 100+ | 就绪 SRUN | SLOAD | 0 |

习题： T0 时刻，某 UNIX V6++系统进程状态如下表所示。内存空间已满，除图示空间外，其余空间不可用。请尽量详细地分析以下时刻系统中与进程调度和对换操作（swap in, swap out）相关的行为，并修改下表中的相关字段。本题不考虑时钟中断。1 小题，2 小题相关。

| 序号 | 占用空间 | 状态 | 位置 | 内存起始地址 |
|----|------|------------|--------|------------|
| 0# | - | 高睡（RunOut） | SLOAD | **** |
| p1 | 40K | 低睡 | SLOAD | 0x00408000 |
| p2 | 30K | 执行 | SLOAD | 0x00430000 |
| p3 | 30K | 低睡 | ~SLOAD | 0x00450000 |

（1）T0 时刻，现运行进程 p2 执行 read 系统调用读磁盘文件（磁盘高速缓存中没有 p2 需要的文件数据）

T0 时刻，现运行进程 p2 执行 read（读文件）系统调用，高优先权入睡（p_stat = SSLEEP, p_pri = -50），放弃 CPU。但是内存中并没有就绪的进程供以进程调度，表格修改如下：

| 序号 | 占用空间 | 状态 | 位置 | 内存起始地址 |
|----|------|-----------------------------------|--------|------------|
| 0# | - | 高睡（RunOut） | SLOAD | **** |
| p1 | 40K | 低睡 | SLOAD | 0x00408000 |
| p2 | 30K | 高睡（ p_stat = SSLEEP, p_pri = -50） | SLOAD | 0x00430000 |
| p3 | 30K | 低睡 | ~SLOAD | 0x00450000 |

（2）T1 时刻，已完成 read 系统调用的 p2 进程运行在用户态。p3 等待的 I/O 操作完成。

T1 时刻，正在执行的进程是 p2，此时 p3 等待的 I/O 操作完成，会执行中断处理程序，唤醒 p3 进程，并唤醒 0#进程，在中断处理程序结束之后，由于之前是用户态，会发生例行调度，并将 CPU 让给优先级更高的 0#进程。

0#进程会执行 sched()函数，为盘交换区上就绪的 p3 进程分配内存空间，会在内存中查找低睡（SWAIT）或 SSTOP（暂停）的进程。此时会发现进程 p1 的状态为低优先级睡眠，因此 p1 入选。0#进程会换出 p1 进程，并换入盘交换区的 p3 进程(放在 p1 进程原先占据的内存区域)。

完成上述的换入换出的操作之后，0#进程会 sleep(&Runout,-100)入睡，并执行 swtch()函数将 CPU 让给进程 ，然后进程会执行下去。

之后遇到时间片到的时钟中断，p2 进程和 p3 进程会时间片轮转，轮流执行下去。

| 序号 | 占用空间 | 状态 | 位置 | 内存起始地址 |
|----|------|------------|--------|------------|
| 0# | - | 高睡（RunOut） | SLOAD | **** |
| p1 | 40K | 低睡 | ~SLOAD | 0x00450000 |
| p2 | 30K | 执行 | SLOAD | 0x00430000 |
| p3 | 30K | 就绪 | SLOAD | 0x00408000 |