第11章 时钟中断

过程“ clock ”(3725)处理来自线频时钟(KW11-L型，中断矢量地址100)或可编程实时时钟(KW11-P型，中断矢量地址104)的中断。

UNIX要求这两种时钟中至少有一种是可用的。(如果两种都存在，则只使用线频时钟。)

不管使用哪一种时钟，都以线频产生中断(若电源频率为50Hz，则每20ms产生一次中断)。时钟中断优先级为6，在我们的典型系统中，这高于任何其他外部设备，所以时钟控制器一旦请求中断时，通常很快就会得到响应和处理。

11.1 clock(3725)

“ clock ”的功能是一般的内务处理：

- 更新显示寄存器(仅PDP11/45和PDP11/70具有此种功能)。
- 维护各种与计算相关的值，例如时间、累计处理时间以及执行简要表。
- 唤醒睡眠时间已到预定值的进程。
- 每秒一次启动内存交换活动。

“ clock ”并不遵守外设处理程序的很多惯例：它引用当前“ u ”结构；有时它以较低优先级运行。如果中断前的执行尚未完成，则“ clock ”会压缩其活动的某些部分。

3740：“ 显示 ”在PDP11/40机上并不执行任何有效操作。

3743：“ callout ”数组(0265)是长度为NCALL(0143)的“ callo ”型(0260)结构数组。“ callo ”结构包含3个成员：一个增量时间、一个参数和一个函数地址。若函数成员非空，则在指定的时间后执行该函数，并将参数传递给它。

对于我们的样板系统，以这种方式执行的唯一一个函数是“ ttrstrt ”(8486)，它是电传打字机处理程序的一部分。

3748：如果该列表的第1个成员为空(null)，则整个列表为空。

3750：“ callout ”列表按所希望的执行顺序前后编排。“ callo ”结构中记录的时间(在增量时间成员中)是相邻两事件之间的时钟滴答数。除非列表中第 1个时间(在下一个事件之前的时间)已为0(表示已到应执行的时间)，否则该时间值应当减1。

若该时间已倒计数为0，则对于下一个时间值减1，若它也为0，则继续向后，直至找到一个非0值并对其减1。在列表前部具0时间值的各项表示它们指定的操作都已到达应执行的时间。(这些操作的实际执行时间一定会稍有延迟。)

3759：检查前处理机状态字，若其优先级非 0，则不执行下一段代码，该段代码执行已到预定时间的操作。

3766：将处理机优先级降为5(现在可以响应其他6级中断请求)。

3767：搜索“ callout ”数组，查看已到执行时间的各操作并执行这些操作。

3773：将未到期的操作向数组头部移动。

3787：只要前处理机优先级为5或6，则执行从此开始至3797行之间的代码。

3788：若前状态是“ 用户态 ”，则对用户时间计数器执行增 1操作，若需对执行状况简要表进行累计处理，则调用“ incupc ”(0895)，以便按用户态程序计数器 (PC)在直方图中对相应项进行增1处理。

“ incupc ”是用汇编语言编写的，估计这是为了效率和方便。在 UPM的PROFIL(II)部分可以找到对其功能的说明。也请参见“ profil ”过程(3667)。

3792：若前状态并非用户态，则增加该进程的系统 (核心)时间计数器。

刚说明的代码段执行基本时间计算。每个时钟滴答对某个进程 (时钟中断前的当前运行进程)的“ u.u_utime ”或“ u.u_stime ”增1。在“ fork ”(3322)时，对所创建新进程的“ u.u_utime ”和“ u.u_stime ”赋初值0。在“ wait ”(3270)中将取用这两项的值。在 32K滴答后(约10小时)，这些时间值将变为负值。

3795：“ p_cpu ”用于计算进程优先数。这是一个总被解释为正整数 (0～255)的字符值。当将其移送至一特定寄存器中时，进行符号位扩展，所以 255就会变成－1。对该值加1，就会变成0，对0减1又成为－1，并以不带符号的255存储。注意，在引用“ p_cpu ”的其他各处(2161、3814)，在将其值移送到一特定寄存器后掩蔽其高 8位。

3797：对“ lbolt ”增1，若其值超过“ HZ ”，亦即已经过1s或更多一些时间…

3798：然后假定处理机以前在0优先级运行，则进行大量内务处理操作。

3800：从“ lbolt ”中减去“ HZ ”。

3801：使时间累计器值加1。

些时间，但在必要情况下可以中断这些处理，以便对其他
降为 1，它低于所有外设优先级(外设的最低优先级为4)。
过程之前，可能产生另一次时钟中断。在这里，将处理机
2次“ clock ”不会企图执行从 3804行开始的代码。但是应
功能上是和优先级0相同的。
少)的值等于存放在“ tout ”中的值，则唤醒调用“ sleep ”系
相关进程，而该系统调用在核内是由“sleep ”(5979)执行的。
被唤醒的时间。若因“ tout ”而睡眠的进程有几个，则除
进程也将被唤醒，因此应将它们的睡眠唤醒时间进行比
够起到应起的作用，但是这种进程的数量如果较大，则其
使用类似于“ callout ”数组的机制。(将这两种机制合并
么?)
2位为0亦即每隔4s时，重新设置“ runrun ”调度标志，唤
“ lbolt ”表示每4s就应处理一次的一般事件，用其启动杂
机制。)
：
127，若其值已为127，则不再增1，(p_time是一个字符型

对“ p_cpu ”减“ SCHMAG ”(3707)，但不使其成为负值。注意，正如前面所讨论过的(3795行)，“ p_cpu ”被处理为0~225之间的正整数。

注意，“ setpri ”(2156)在计算进程优先数“ p_pri ”时，使用了“ p_cpu ”。“ swtch ”(2209)选择一个进程占用处理机，其选择条件是：进程映像在内存中 (SLOAD)，已准备运行“ SRUN ”，若满足此条件的进程有若干个，则从中选择“ p_pri ”最小者。

“ p_time ”用于度量一个进程本次驻在内存或换出至磁盘所经过的时间 (单位：秒)。“ newproc ”(1869)、“ sched ”(2047)和“ xswap ”(4386)将“ p_time ”设置为0。“ sched ”用“ p_time ”决定将哪一个进程换入或换出。

3820：如果调度进程正等待对进行图像交换作出重新安排，则将其唤醒。作出调度决策的正常速率是每秒1次。

3824：若中断之前的状态是“ 用户态 ”，则将“ r0 ”的地址存放在标准位置，如果该进程已接收到一个“ 信号 ”，则调用“ psig ”(4043)以执行适当的动作。

11.2 timeout(3845)

本过程在“ callout ”数组中构造新项。在本样板系统中，只在“ ttstart ”(8505)例程调用“ timeout ”，调用时的一个参数是“ ttrstrt ”过程(8486)。注意，“ ttrstrt ”调用“ ttstart ”，而它又可调用“ timeout ”，这是一种连续不断循环调用的特殊关系！

也请注意，“ timeout ”大部分在处理机优先级7级条件下运行，以避免时钟中断。