

附录B 计算机时钟

既然本书中的大多数的例子都需要测量一个时间间隔，我们需要更仔细地介绍一下当前 Unix 系统所采用的记录时间的方法。下面的描述适用于本书中例子所使用的系统，也适用于大多数的 Unix 系统。[Leffler et al. 1989] 的 3.4 节和 3.5 节给出了另外的细节。

硬件按照一定的频率产生一个时钟中断。对于 Sun SPARC 和 Intel 80386，时钟中断每 10 ms 产生一次。

应该注意到大多数的计算机使用一种无补偿的晶体振荡器来生成这些时钟中断。正如 RFC1305 [Mills 1992] 的表 7 指出的，你不要想知道这种振荡器一天的偏差有多少。这就意味着几乎没有计算机能维持精确的时间（即，中断并不是精确地每 10 ms 发生一次）。一个 0.01% 的误差就会产生一个每天 8.64 秒的差错。为了得到更好的时间测量需要：（1）一个更好的振荡器；（2）一个外部的更精确的时间资源（如，全球定位卫星提供的时间资源）；或者（3）通过因特网访问一个具有更精确时钟的系统。后者通过 RFC1305 定义的网络时间协议实现，对该协议的描述超出了本书的范围。

Unix 系统中引起时间差错的另一个公共的原因是 10 ms 的中断只是引起内核给一个记录时间的变量增 1。如果内核丢失了一个中断（也就是说两个连续中断之间间隔 10 ms 对于内核来说太快了），时钟将失去 10 ms。丢失这种类型的中断经常引起 Unix 系统丢失时间。

尽管时间中断近似于每 10 ms 到达一次，更新的系统，如 SPARC，提供了一个更高精度的定时器来测量时间差异。通过 NIT 驱动程序，tcpdump（在附录 A 中描述）已经访问了这个更高精度的定时器。在 SPARC 上，这个定时器提供了微秒级的精度。用户进程也可以通过 `gettimeofday(2)` 函数来访问这个更高精度的定时器。

作者做了下面的试验。我们运行了一个程序，这个程序在一个循环里调用了 10 000 次 `gettimeofday` 函数，并将每次的返回值保存在一个数组中。在循环结束后，打印了 9999 个时间差。对于一个 SPARC ELC，时间差的分布如图 B-1 所示。

微 秒	次数
36	4 914
37	4 831
38	167
39	8
其他	79

图 B-1 在 SPARC ELC 上调用 `gettimeofday` 函数 10 000 次所需要的时间分布

在一个空闲的系统中，运行这个程序所花的时钟时间为 0.38 秒。根据这一点，我们可以说进程调用 `gettimeofday` 所花的时间大约 37 微秒。既然 ELC 的速度是 21 MIPS（MIPS 表示每秒 100 万指令），37 微秒相应于大约 800 个指令。这些指令对于内核处理一个用户进程的调用、执行系统调用、复制 8 个字节的結果及返回给用户进程看起来是合理的（MIPS 速度是不可靠的，很难测量当前系统的指令时间。我们试图做的只是得到一个粗略的估计来评价一下上面的值是否有意义）。

从这个简单的试验，我们可以说 `gettimeofday` 返回的值确实包含了微秒级的精度。

如果我们在SVR4/386上进行类似的测试，结果是不同的。这是因为很多 386 Unix 系统，如 SVR4，只计数 10 ms 的时钟中断，而没有提供更高的精度。图 B-2 是运行在 25 MHz 80386 上的 SVR4 中 9999 个时间差的分布。

这些值是无意义的，因为时间差一般小于 10 ms，都被认为是 0 了。在这些系统中，我们所能做的就是在一个空闲的系统上测量时钟时间，除以循环的次数。这个结果提供了一个上界，因为它包含了调用 `printf` 函数 9999 次的时间和将结果写入一个文件的时间（在 SPARC 的情况，图 B-1，时间差没有包括 `printf` 的时间，因为所有 10 000 个值都是首先获得的，然后才打印结果）。在 SVR4 的时钟时间为 3.15 秒，每个系统调用消耗了 315 微秒。这个大约比 SPARC 慢 8.5 倍的系统调用时间看来是正确的。

BSD/386 1.0 版提供了类似于 SPARC 的微秒级的精度。它读 8253 时钟寄存器，计算从上次时钟中断以来的微秒次数。调用 `gettimeofday` 的进程和内核模块，如 BSD 分组过滤器，可以使用这个精度。

和 `tcpdump` 联系起来，这些数字意味着我们可以相信在 SPARC 和 BSD/386 系统上打印的毫秒和亚毫秒 (submillisecond) 的值。而在 SVR4/386 上，`tcpdump` 打印的值总是 10 ms 的倍数。对于其他打印往返时间的程序，如 `ping` (第 7 章) 和 `traceroute` (第 8 章)，在 SPARC 和 BSD/386 系统上，我们可以相信它们输出的毫秒值，但在 SVR4/386 上，打印的值总是 10 的倍数。为了在 LAN 上测量某个 `ping` 的时间，在第 7 章中我们显示的时间是 3 ms，所以需要在 SPARC 和 BSD/386 系统上运行 `ping` 程序。

本书中的一些例子是运行在 BSD/386 0.9.4 版上，它类似于 SVR4，只提供了 10 ms 的时钟精度。在我们显示这个系统的 `tcpdump` 输出时，只显示到小数点后面两位，因为这就是所提供的精度。

微 秒	次 数
0	9 871
10 000	128

图B-2 在SVR4/386上调用`gettimeofday`函数
10000次所需要的时间