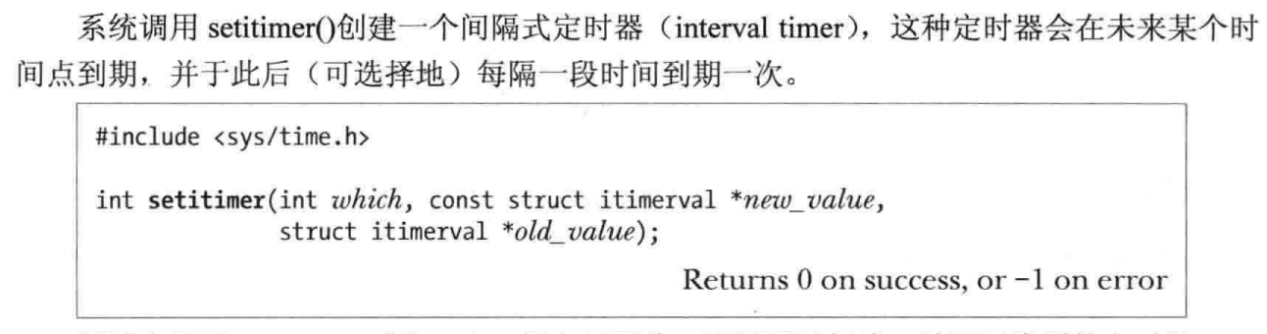
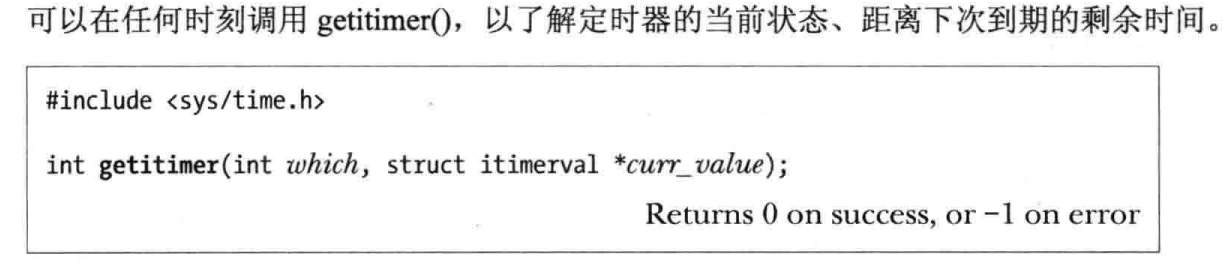
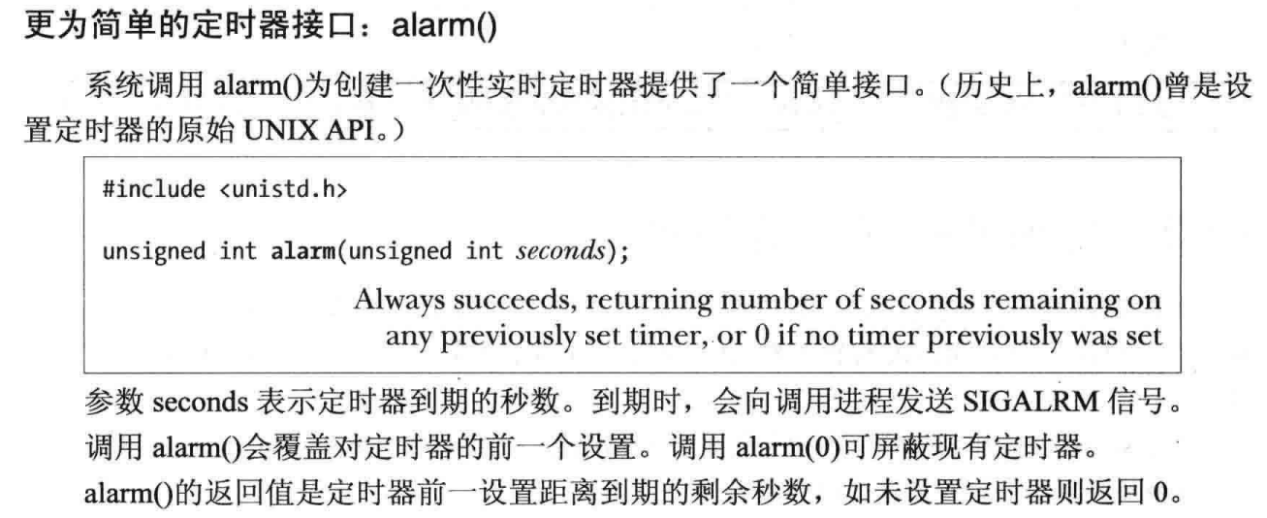
第二十三章

1. 间隔定时器：



注意，可以指定三种类型的定时器：真实时间，虚拟时间（用户模式下CPU时间）和进程时间。产生的信号也不同。



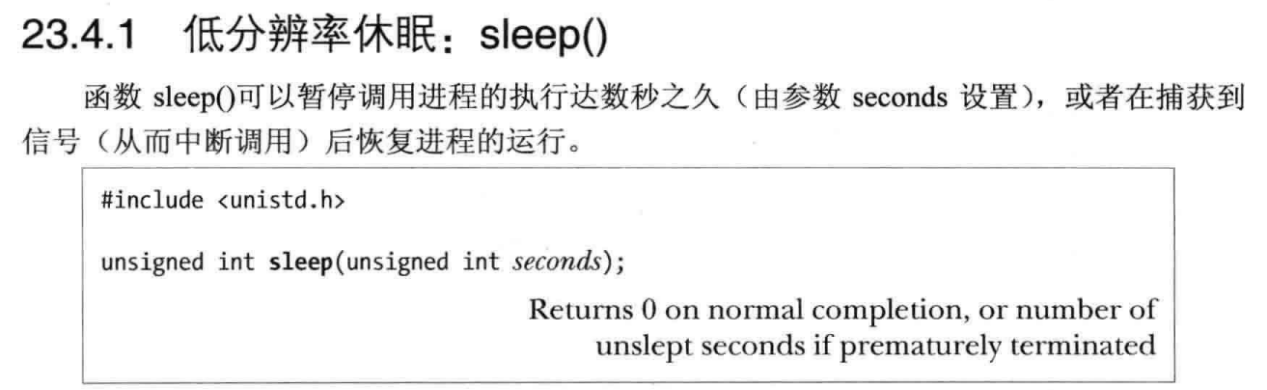


注意，这两个函数共享一个定时器，不要混合使用。有的版本的Linux里面的sleep函数也会由定时器实现，也不要混用。

1. 定时器的调度与精度：

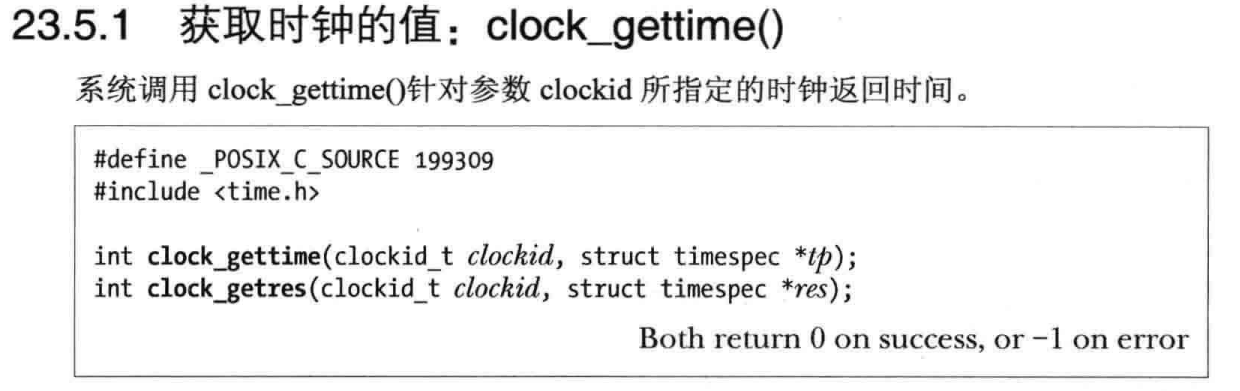
传统意义上的定时器受制与软件时钟的频率。系统的调度会导致定时器延误。

1. 定时器可以为阻塞操作设置超时设置
2. 暂停运行一段时间：

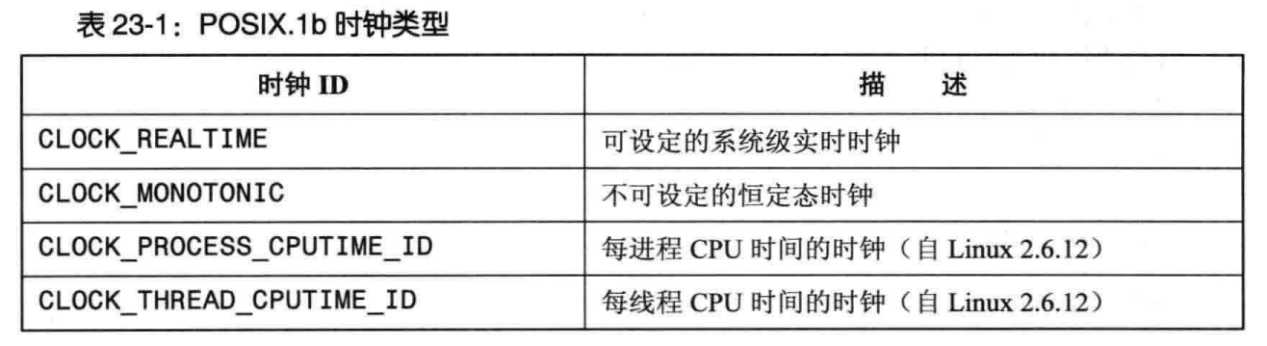


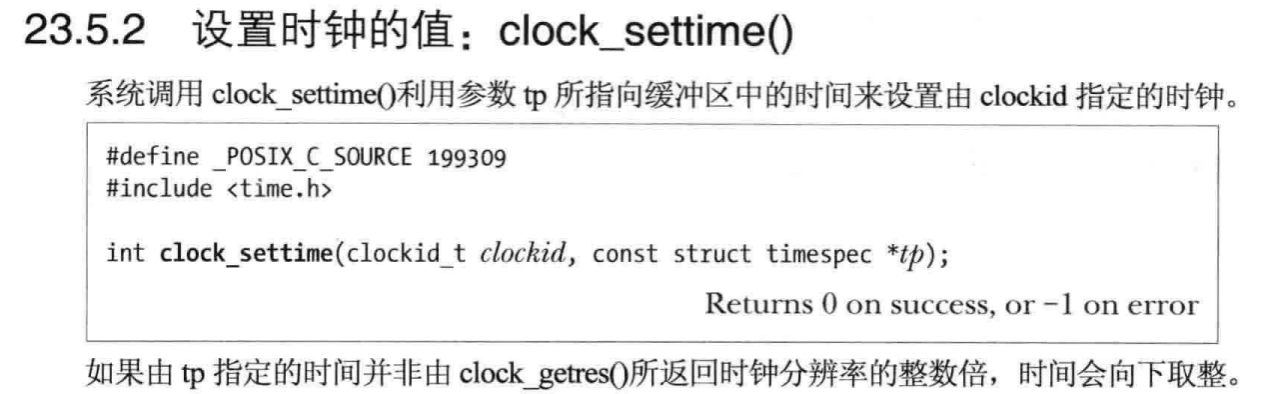


1. Posix时钟



第一个函数返回以clockid计算的时间，比如下图第一个是1997那个时间，可以理解为返回的时间戳。第二个函数返回时间精度。

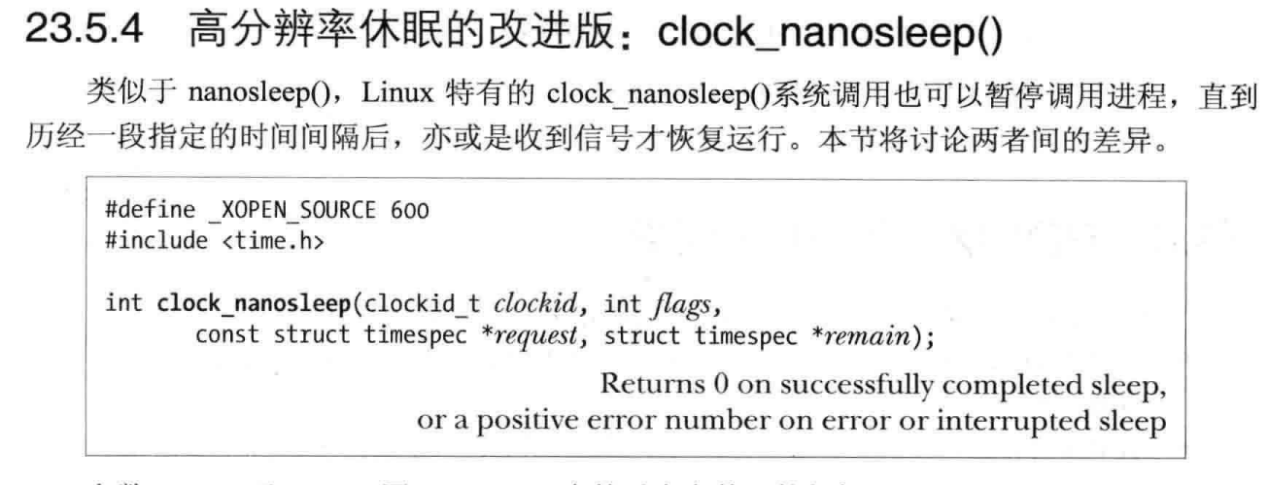




只能设置系统级实时时钟。



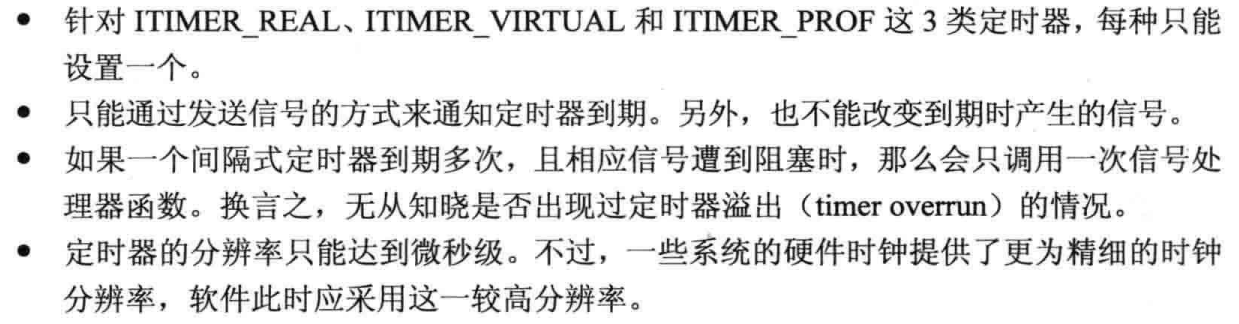
上述两个函数可以根据获取到的时钟ID调用clock\_gettime函数获取此进程的CPU时间。



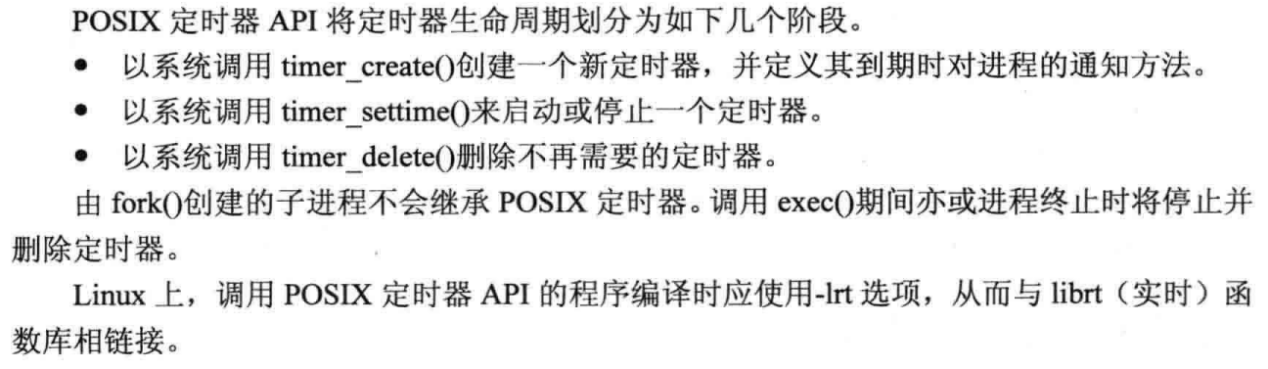
可以使用绝对时间的flag使得精度更高。

1. Posix间隔式定时器

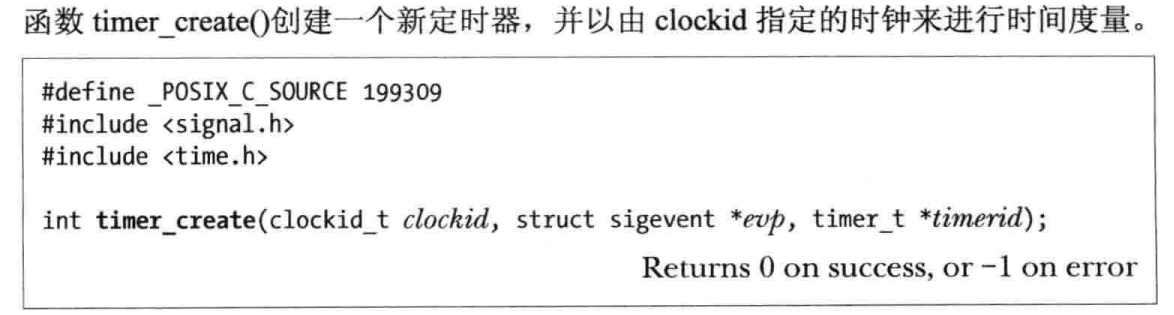
使用传统的setitimer定时器具有一定的弊端：



Posix定时器api的生命周期如下：

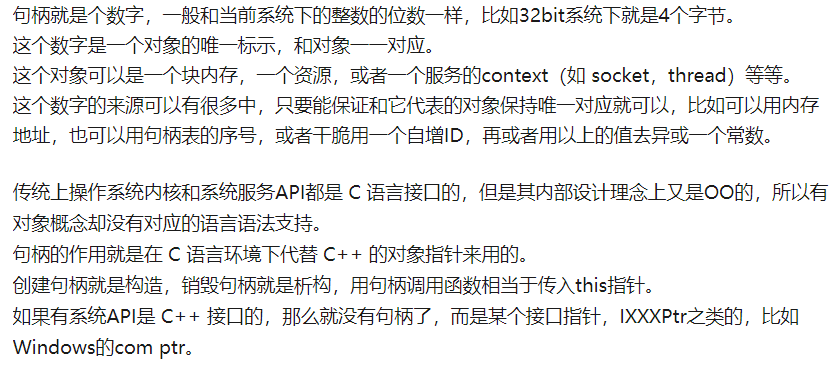


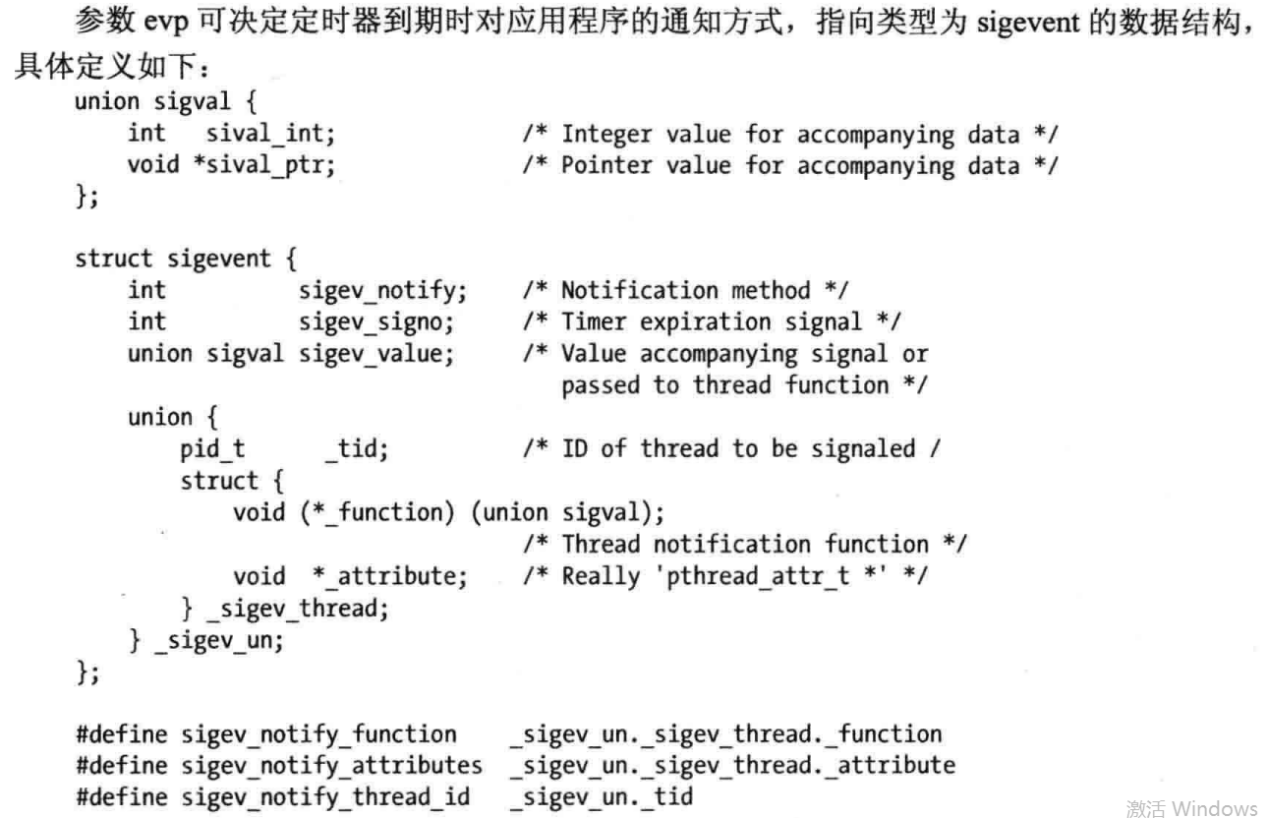
创建定时器：

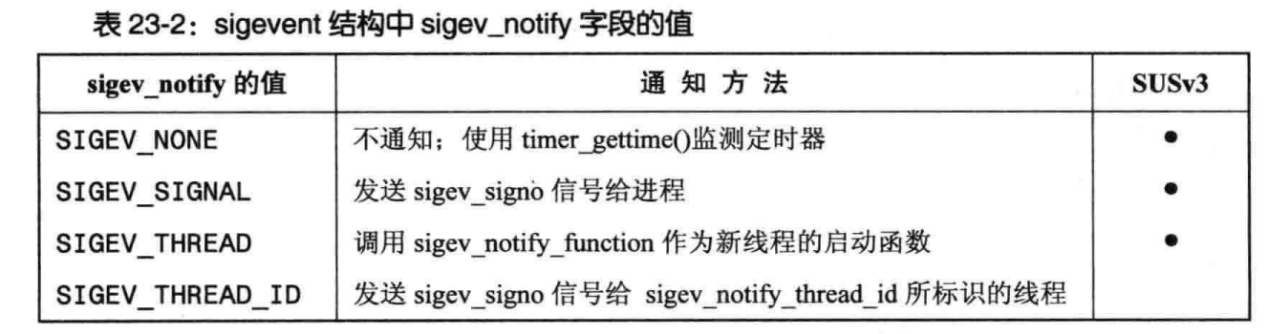


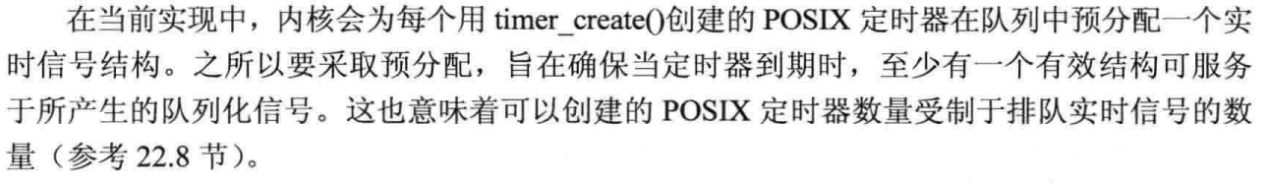
timerid，第三个参数是定时器句柄，供后续调用中指代该定时器来使用。

在这里，出现了句柄这个概念，什么是句柄？查阅资料比较简单的理解如下：

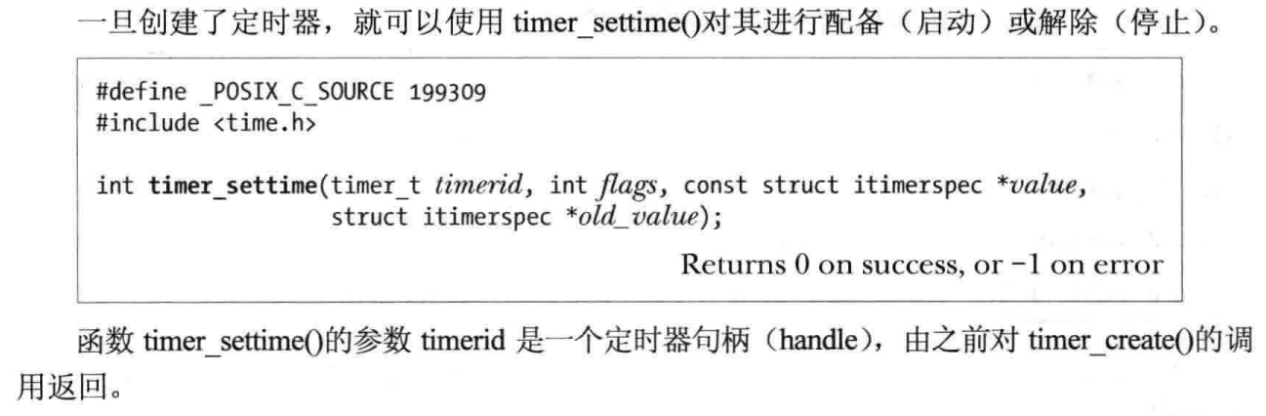


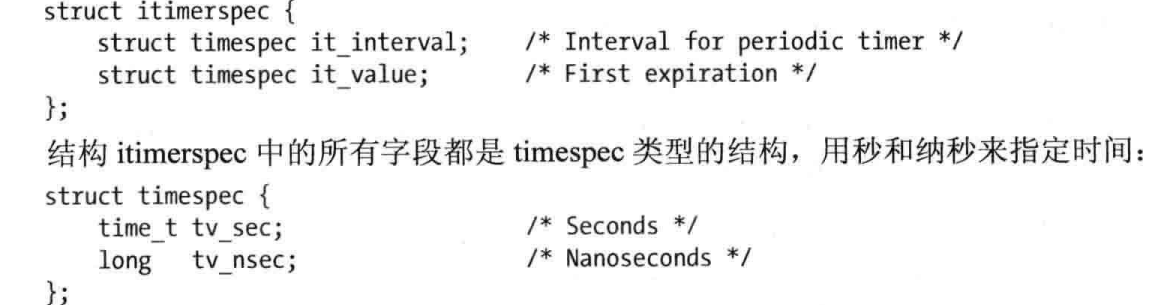




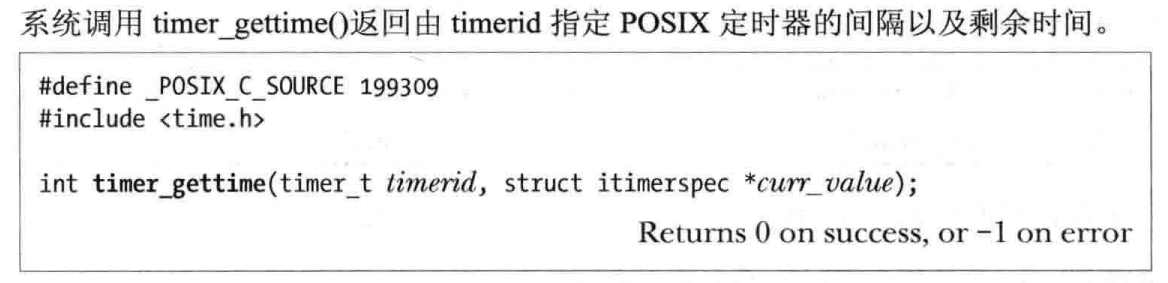


配备和解除定时器：

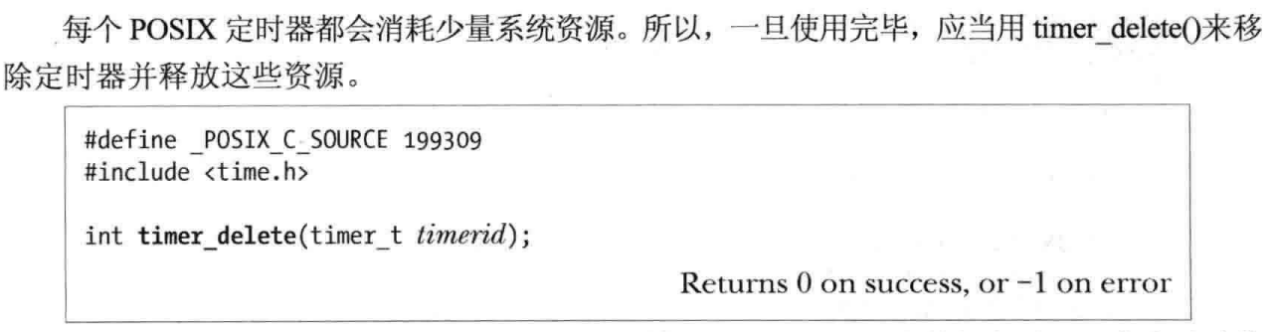




获取定时器的当前值：

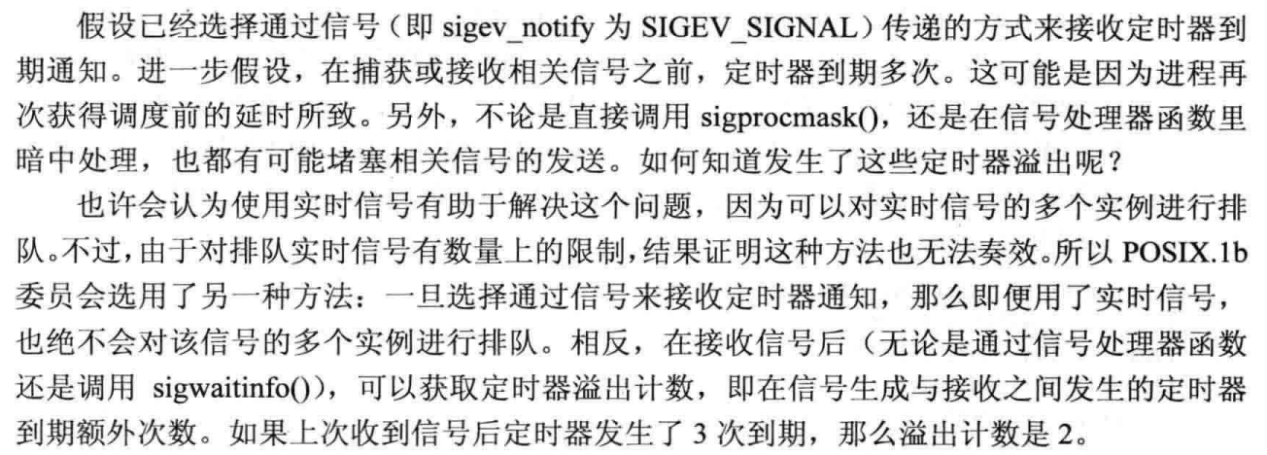


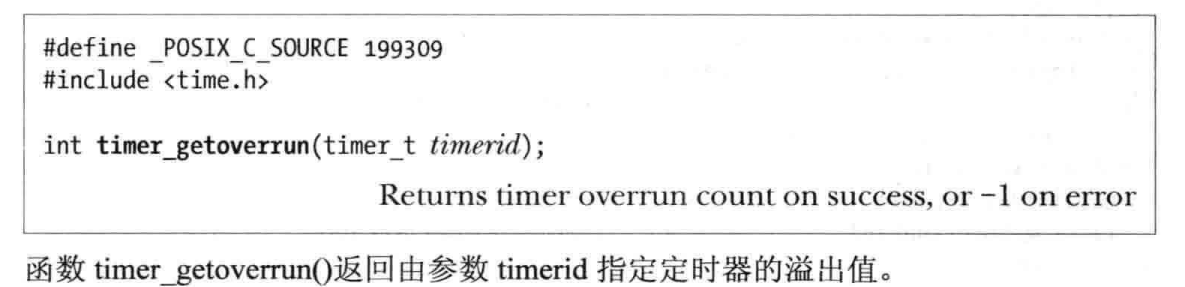
删除定时器：



定时器溢出：

Posix采用的方法导致即使是实时信号也不会对其进行排队，但可以获取到他的定时器溢出计数值。

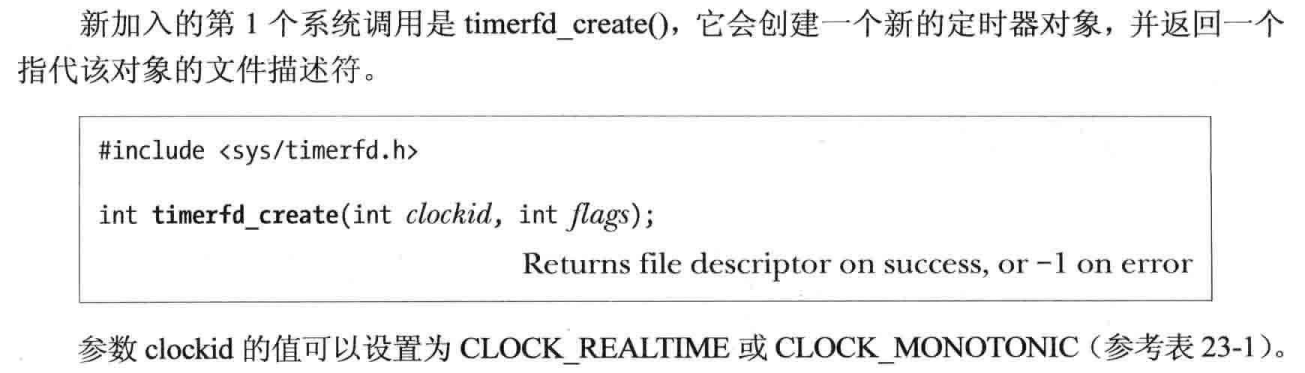


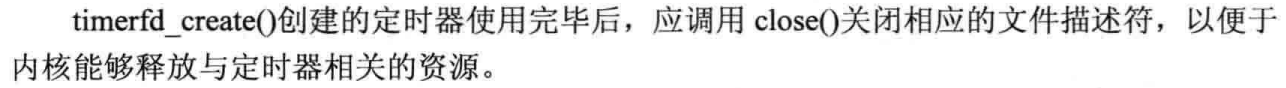


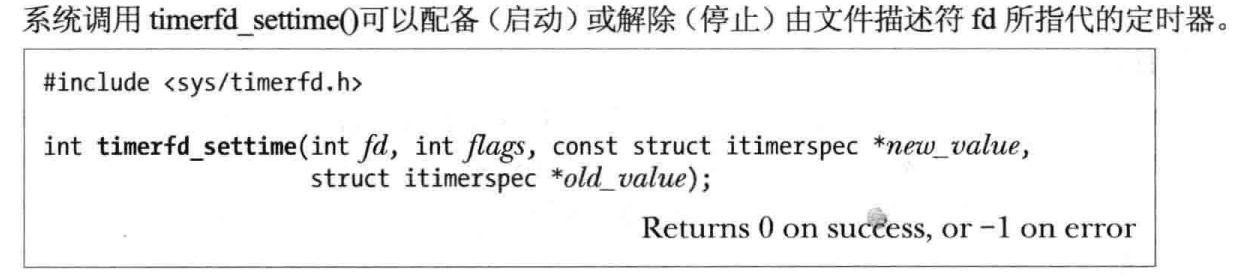
此函数是线程安全函数。

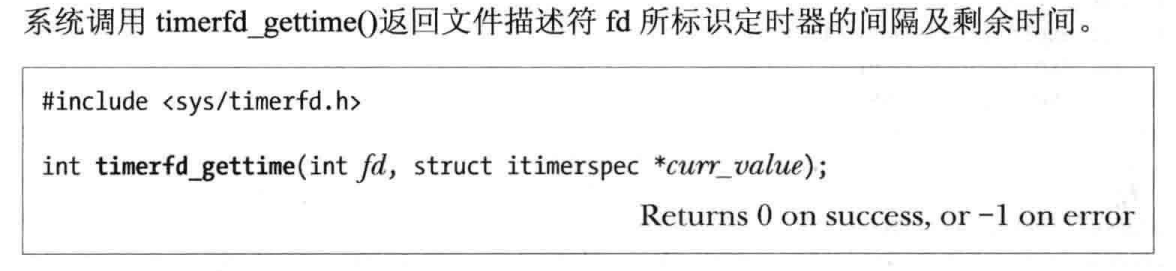
有一个与线程相关的例子没看懂，回头要看一看。

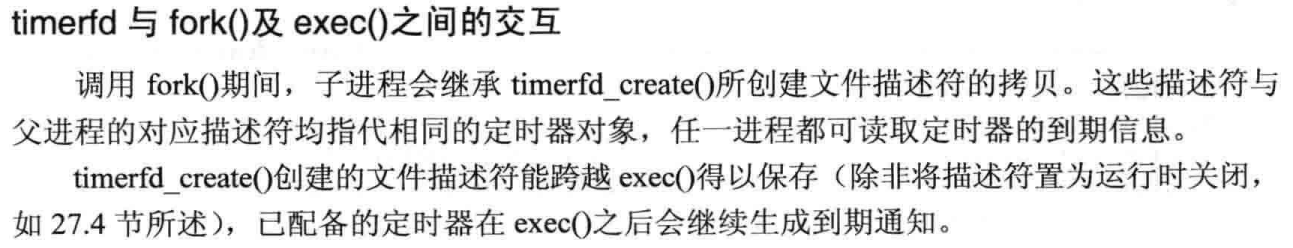
1. 利用文件描述符进行通知的定时器：











注意，一旦启动了定时器，就可以使用read读取文件描述符的到期信息。缓冲区要能容纳八字节的整形。（uint64\_t）