一种多任务应用系统的设计和实现

曾大亮 詹建桥 (华中理工大学)

摘要:本文介绍了一种在 *DOS* 环境下,多任务应用系统的设计技术和方法。它使用数据区交换的方法突破 *DOS* 不支持多任务的限制。文中着重讨论了任务切换时,堆栈切换、数据区交换和实现控制转移的技术。

一、引 言

当前,人们对多任务应用程序的需求不断增加,新的多任务操作系统也在不断出现。但是,这些操作系统为支持多任务付出了沉重的代价。它们失去了象 DOS 这样的单任务操作系统的结构紧凑,内存占用少,环境要求不苛刻等优点。在 DOS 这样的单任务操作系统环境下,开发能运行多任务应用系统的软件的问题,仍然是热门的话题。

本文讨论了在 DOS 环境下,如何设计一个多任务应用系统的问题。为了克服 DOS 不支持多任务的限制,使用了数据区交换的方法,使得在单任务的环境下能安全地运行多任务程序。本文还着重讨论了任务切换时,堆栈切换,数据区切换和实现控制转移的技术。

二、突破 DOS 不支持多任务的限制

DOS 不能支持多任务,是因为 DOS 是不可重入的。可以用二种方法回避它:一种是在编写应用程序时完全采用低级语言,并且只使用 BIOS 调用,不使用 DOS 系统调用。这种方法对于编写稍大一点的程序几乎是不可能的;另一种是采用延时法,当探测到任务切换时正在进行 DOS 调用,就不马上进行切换,而是推迟到本次 DOS 调用结束,然后再去切换。这种推迟时间的方法使得由此构成的多任务应用程序的运行效果受到严重伤害。

经分析,只要在切换任务之前,将 DOS 数据区保存下来,待下次往回切换时再恢复,回切以后的任务就可以继续运行下去。使用这种方法,DOS 就可在任何时候重入了。

事实上,调用 DOS 功能 5D06H(DOS 3.X 版本)或 50DBH(DOS 4.X 版本)可以存取 DOS 数据区。数据区共有 1852 个字节,它的前 24 个字节包含了任务运行时的重要数据,如当前 PSP、DTA、扩展错误码等,但它不包括 DOS 的三个栈。因此,当任务切换时,如果此刻任务没有进行 DOS 调用(可用 INDOS 监视),只须交换数据区前 24 个字节,否则就交换整个数据区。使用这种方法,就可以使 DOS 支持多任务成为可能的了。

下面讨论的多任务应用系统,就是基于这种方法。

三、多任务应用系统的设计思想

对于一个真正的多任务操作系统,它有一个相当重要的机构—任务调度机构。由于它的重要地位,设计时放在操作系统的底层。与此相反,我们开发的多任务应用系统,恰恰是以 DOS 这个单任务的操作系统为原始操作系统,任务调度机构所在的多任务系统,必须构造在 DOS 的上层。我们必须设计好以弥补它所带来的影响。图 1 是整个的设计思想结构图。

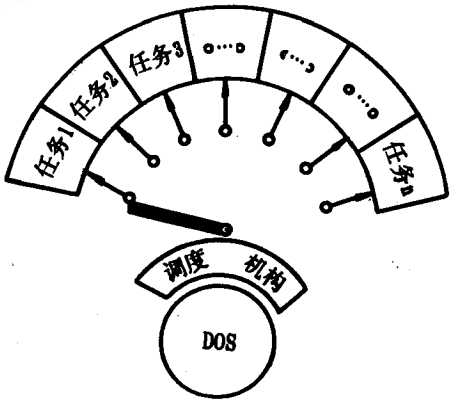

图 1

通常使用事件驱动的中断来触发调度模块,然后再由

调度模块根据调度策略启动相应的应用任务程序。这个调度策略是根据各任务所完成的目标及它们之间的关系来决定的。

由于我们使用了 DOS 数据交换区的技术,DOS 的重入就不再成为设计中的真正的障碍了。只要能妥善地保存好 DOS 数据交换区的数据,任务切换所引起的环境重建问题就容易解决了。

四、任务状态转换和任务切换方法

调度程序要能不断地根据调度策略去驱动相应的任务,这样才能协同完成某项工作。任务之间的转换,可以用一个状态机代替。我们根据调度策略构造一个状态机,其中每一状态是一个任务。图 2 是假设的一个状态机。例如状态 S1 若输入 T 时转换到 S2,若输入 Q 就转换到 S4,S4 若输入 Q 则状态不变,若输入 T 则转为 S1。如果我们假定 T 是时间片完成。Q 是某一控制键输入,那么,可知这个调度策略是键盘抢夺时间片轮转多任务调度方式。

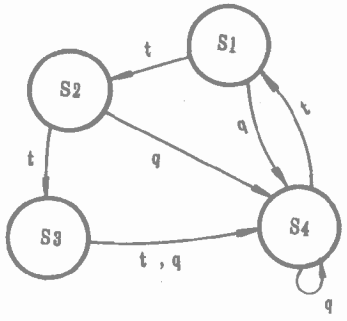

图　2

为了完成任务的正确切换,每一任务都必须有一独立的堆栈。这些堆栈的地址结合成一个堆栈地址表,如图 3 所示。我们可以在状态转换时,利用切换堆栈的方法来实现任务的切换。下面就来讨论这种方法。

在调度程序中包含一张所有任务的堆栈地址表,表中每一表项记录该任务被切换下来时的堆栈当前段地址和偏移地址。在任务切换时,被切换下来的任务栈地址填入对应地址表中,再将换上来运行的任务栈对应的表项内容填入堆栈寄存器(SS) 和栈顶指示器(SP)中,从而完成了堆栈的切换过程。

堆栈切换和任务切换有什么关系呢?首先,因为栈中保存有任务运行时的数据,恢复任务运行就必须恢复它的数据。其次,我们把每个任务分别设计成一个子程序,它从运行状态被切换下来时的断点地址就保存在这个栈的栈顶,所以通过切换栈就可以从堆栈顶中得到它对应的任务的断点地址,从而让控制转到这个地址上去。

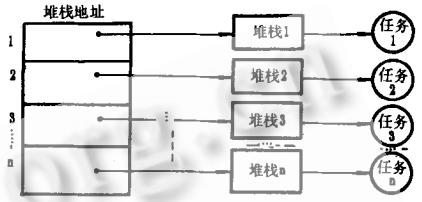

图　3

为了说明简单起见,假定只有两个任务被调度,即任务Ⅰ和任务Ⅱ。设任务Ⅰ正在运行,由于事件触发中断(例如时间片完所引发的时钟中断),控制从任务Ⅰ转给调度程序,任务的断点地址和状态标志字就被推入栈顶保存。调度模块接收到控制后,为了将控制转给任务Ⅱ,先进行栈切换,将任务Ⅱ的堆栈变为活跃态。然后进行中断返回。按道理,本应返回到任务Ⅰ,但此时堆栈已切换为堆栈Ⅱ。而堆栈Ⅱ的栈顶存放的是先前它被切换下去时它的断点地址和状态标志字,因此,中断返回就从堆栈Ⅱ

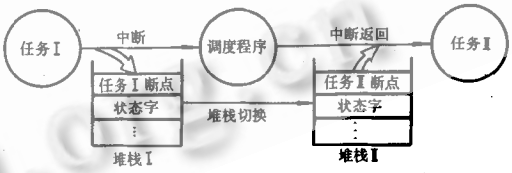

图　4

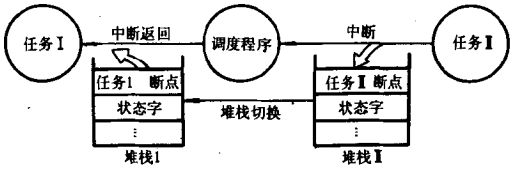

图　5

栈顶取地址返回,自然而然控制就转给任务Ⅱ去了,如图 4 所示。反之,从任务Ⅱ转到任务Ⅰ也是这样的,见图 5 所示。总而言之,任务的切换是通过堆栈的切换实现的。

五、多任务中 DOS 数据区的切换方法

前面已经提及,为了使得任务切换后各个任务的运行状态下不致于被破坏,就必须保存妥各个任务的 DOS 数据区的数据。为此,应为每一任务申请数据交换缓冲区。例如,对于 DOS 版本 3 如果系统中有两个任务,就必须申请两个缓冲区。假如系统中任务Ⅰ正在运行,调度程序正欲调入任务Ⅱ运行,而欲将任务Ⅰ挂起。为此,首先将 DOS 数据区的数据保存到任务Ⅰ缓冲区中,尔后,再将任务Ⅱ缓冲区的内容恢复到 DOS 数据区中。这样就恢复了任务Ⅱ的运行状态,从而能使任务Ⅱ正确运行,如图 6 所示。此后,在下一次任务切换时,先把 DOS 数据区的数据保存到任务Ⅱ缓冲区中,再将任务Ⅰ缓冲区的内容恢复到 DOS 数据区中。这样又恢复了任务Ⅰ的运行状态,如图 7 所示。这里必须注意的是,图中的交换顺序不能改变。

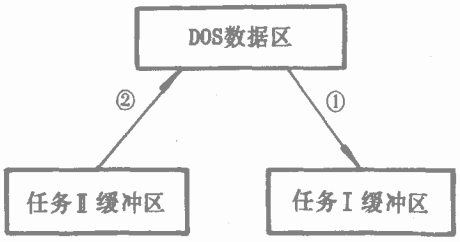

图 6 任务Ⅰ挂起,任务Ⅱ运行的数据交换

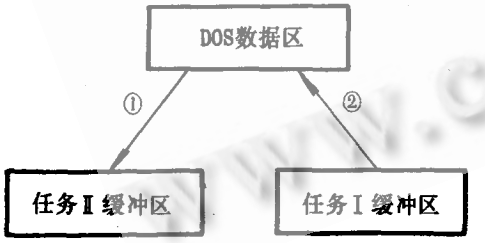

图 7 任务Ⅱ挂起,任务Ⅰ运行的数据交换

数据区切换的工作可通过定义下面的两个函数来实现:

```
void suspend__task1() / * 挂起任务Ⅰ * /
{
save__dos__swap1(); / * 保存任务Ⅰ的 DOS 数据区 * /
restore__dos__swap2(); / * 恢复任务Ⅱ的 DOS 数据区 * /
}
void suspend__dask2() / * 挂起任务Ⅱ * /
{
save__dos__swap2(); / * 保存任务Ⅱ的 DOS 数据区 * /
restore__dos__swap1(); / * 恢复任务Ⅰ的 DOS 数据区 * /
}
```

六、任务睡眠

当活跃的任务存取磁盘等硬资源时,由于时间太长,例如磁盘驱动器需要几秒钟的稳定时间,此时,任务需要等待而进入睡眠,并且应该释放它所占的 CPU 和其它资源给另一个任务使用。在实现时,让任务设置一个几秒钟的全局性的时间变量 SLEEP 的值作为睡眠时间长度,然后才将控制转给其它任务。睡眠时间通过倒数计数法,并通过修改时钟中断处理程序来实现控制。例如,使用如下语句:

```
if (sleep > 0)
  sleep = sleep-1;
```

时钟中断每 55ms 中断次,就执行一次这个 IF 语句,直到 SLEEP 为 0,再唤醒睡眠的任务。如果系统中只有两个任务,那么可以让唤醒的任务立即运行。如果系统中有两个以上的任务,则可以根据调度策略,或者立即让唤醒的任务运行,或者让它进入就绪队列重新排队。

七、结束语

由于 DOS 普及简单,人们能比较容易在它的基础上开发。如果开发的目标具有相同的调度策略,则这样的程序就可以通用,或者稍作修改就可以应用到不同目标的程序中去,只不过是不同的应用目标编写不同的任务而已。文中所讨论的方法,对于使用微机开发用于检测、控制等多任务应用程序也具有一定的参考价值。

参考文献:

[1]DOS 内存驻留技术,李振略等编译,中科院希望电脑公司,1991

[2]实时系统软件设计初步,沈昌祥,人民邮电出版社,1981

[3]尚未公开的 DOS 秘密,吴双编译,海洋出版社,1991.5

[4]The Design of UNIX Operating System, Bach M. J., Prentice-Hall International Inc.,1986