7.5.1 调度同质进程

最简单的一种视频服务器可以支持显示固定数目的电影，所有电影使用相同的帧率、视频分辨率、数据率以及其他参数。在这样的情况下，可以采用下述简单但是有效的调度算法。对每一部电影，存在一个进程（或线程），其工作是每次从磁盘中读取电影的一帧然后将该帧传送给用户。由于所有的进程同等重要，每一帧有相同的工作量要做，并且当它们完成当前帧的处理时将阻塞，所以采用轮转调度可以很好地做这样的工作。将调度算法标准化的惟一的额外要求是定时机制，以确保每一进程以恰当的频率运行。

实现适当定时的一种方式是有一个主控时钟，该时钟每秒滴答适当的次数，例如针对NTSC制式，每秒滴答30次。在时钟的每一滴答，所有的进程以相同的次序相继运行。当一个进程完成其工作时，它将发出suspend系统调用释放CPU直到主控时钟再次滴答。当主控时钟再次滴答时，所有的进程再次以相同的次序运行。只要进程数足够少，所有的工作都可以在一帧的时间内完成，采用轮转调度就足够了。

7.5.2 一般实时调度

不幸的是，这一模型在实践中几乎没有什么用处。随着观众的来来去去，用户的数目不断发生变化，由于视频压缩的本性（I帧比P帧或B帧大得多），帧的大小剧烈变化，并且不同的电影可能有不同的分辨率。因此，不同的进程可能必须以不同的频率运行，具有不同的工作量，并且具有不同的最终时限（在此之前所有工作必须完成）。

这些考虑导致一个不同的模型：多个进程竞争CPU，每个进程有自己的工作量和最终时限。在下面的模型中，我们将假设系统知道每个进程必须以什么样的频率运行、有多少工作要做以及下一个最终时限是什么。（磁盘调度也是一个问题，但我们将在后面考虑。）多个相互竞争的进程，其中若干进程或全部进程具有必须满足的最终时限的调度称为实时调度（real-time scheduling）。

作为实时多媒体调度程序工作环境的一个例子，我们考虑三个进程A、B和C，如图7-13所示。进程A每30ms运行一次（近似NTSC制式速度），每一帧需要10ms的CPU时间。在不存在竞争的情况下，进程A将在突发A1、A2、A3等中运行，每一突发在前一突发的30ms之后开始。每个CPU突发处理一帧并且具有一个最终时限：它必须在下一个突发开始之前完成。

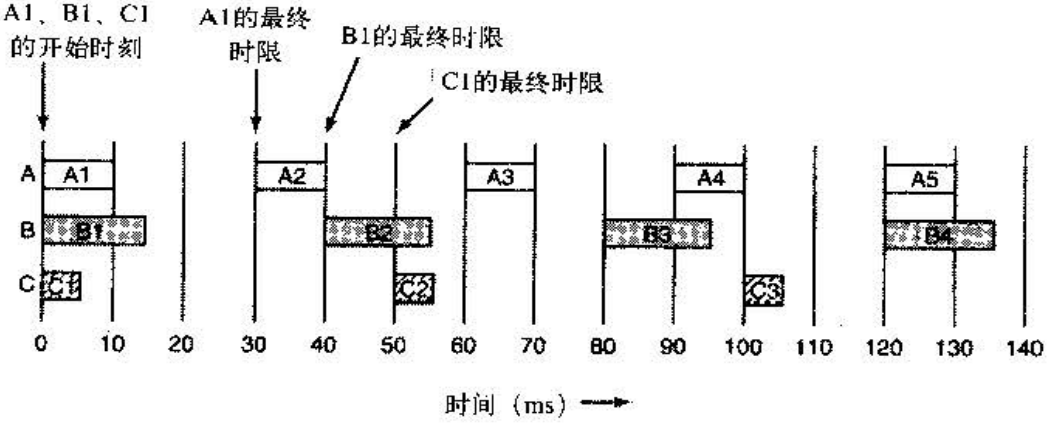

图7-13 三个周期性的进程，每个进程播放一部电影，每一电影的帧率以及每帧的处理需求有所不同

图7-13中还有另外两个进程：B和C。进程B每秒运行25次（例如PAL制式），进程C每秒运行20次（例如一个慢下来的NTSC或PAL流，意在使一个低带宽的用户连接到视频服务器）。每一帧的计算时间如图7-13中所示，进程B为15ms，进程C为5ms，没有使它们都具有相同的时间只是为了使调度问题更加一般化。

现在调度问题是如何调度A、B和C以确保它们满足各自的最终时限。在寻找调度算法之前，我们必须看一看这一组进程究竟是不是可调度的。回想2.4.4节，如果进程i具有P_i ms的周期并且需要C_i ms的CPU时间，那么系统是可调度的当且仅当

$$\sum_{i=1}^{m}\frac{C_i}{P_i}\leqslant 1$$