Windows的消息系统及多任务机制

戴志远　(暨南大学)

一、前 言

在计算机系统中,消息的地位和作用越来越引人注目。以Microsoft公司的Windows操作系统来说,它是基于消息的一个操作系统,有关消息的组织、管理、传递是多任务机制的核心。所以本文简要讨论消息以及Windows的多任务机制,并给出了在Windows下编程的一些体会与经验。

二、消 息

"消息"这个词,是新闻界和通信系统中沿用已久的。信息论创立以后,也使用消息这个词,作为载有信息的某些文本、语句等的总称。

在计算机技术中,消息首先被应用于并发程序中,是一种程序构造,用以实现进程间的通信。在图形用户界面(GUI)中,广泛地使用消息,因为GUI的工作是事件驱动的,而每一事件总是与相应的消息联系在一起,是消息的到达引起事件的发生。在面向对象的系统(OOS)中,消息传递是程序运行的基本机制,对象只在消息到达时才予以响应,表现系统的正常运行。

Microsoft的Windows正是基于消息的操作系统。它的图形用户接口(GUI)、进程间的通信都是通过消息的传递和管理来实现的。它们定义了许多通用的消息,并且能生成新的消息。Windows屏蔽了底层硬件,用户不必与硬件打交道。系统把所有的硬件响应收集起来,进行统一分类处理,并给有关的应用程序发送它想要的消息。应用程序接到有关的消息后,根据不同的消息进行不同的处理。这样应用程序就不必关心底层的硬件细节,在Windows环境下就可以用少量的时间编出高质量的程序,因而达到了可移植的目的。

Windows的消息包括窗口管理消息、输入消息、动态数据交换消息、系统消息等。一般常用的消息有输入消息、窗口管理消息、系统消息。另外用户还可以定义自己的私有消息,供私有程序识别。私有消息可以用来在私有程序中传递简单的信息,标志某一事件的发生。用SendMessage()、PostMessage()等函数就可以方便地把这些消息送到目标窗口的消息队列或窗口函数中。

在Windows应用程序中,所有键盘、鼠标和计时器输入都由Windows系统截取,系统将这些输入放在相应的应用程序消息队列中。当应用程序准备检取输入时,它只是简单地从自己的消息队列中读取下一条输入消息。每次输入事件发生时,Windows系统就产生消息。一般说来,此消息首先被存储在系统队列里,接着被放入特定的应用程序队列内。应用程序队列是先进先出队列。但对于应用程序来说,某些消息被直接送到窗口函数。消息的送入没有任何特定的顺序,因此应用程序的运行不会取决于所接收消息的顺序。

对于应用程序来说,在完成一些必要的初始化工作后,其主要任务是获得与自己有关的消息并恰当地处理它们。处理完成后,就交回系统的控制权,以便和其它应用程序协调地工作。

Windows输入消息所包含的信息比标准的MS-DOS环境要多得多。这些消息包括系统时间、鼠标位置、键盘状态、键的扫描码(如果某键被按下)、鼠标键状态、设备所产生的消息。如键盘消息WM_KEYDOWN和WM_KEYUP,对应于一个键的按下和松开动作。在每一个键盘消息中,Windows系统提供了设备无关的虚键代码来标识键、键盘产生的设备相关的扫描码和键盘上其它键的状态,如SHIFT、CTRL和NUMLOCK键的状态。键盘、鼠标和计时器消息的形式相同并且处理逻辑相同。

图1是Windows输入消息的组织、管理、传递示意图。在这种情况下,Windows系统与应用程序一起完成消息的处理。当有输入时Windows接收输入响应。然

后 Windows 将输入消息从输入队拷贝到合适的应用程序队。消息循环接收这些输入消息，翻译之后将它们送到合适的窗口函数，由窗口函数进行相应的处理。另外 Windows 还可以直接将有关消息传给合适的窗口函数。如当 Windows 执行了窗口销毁操作后，系统越过应用程序队直接把 WM_DESTROY 消息发给窗口函数。然后窗口函数通知主函数窗口已经销毁，应用程序应该终止。这是通过用函数 PostQuitMessage()发送 WM_QUIT 消息到应用程序队来完成。

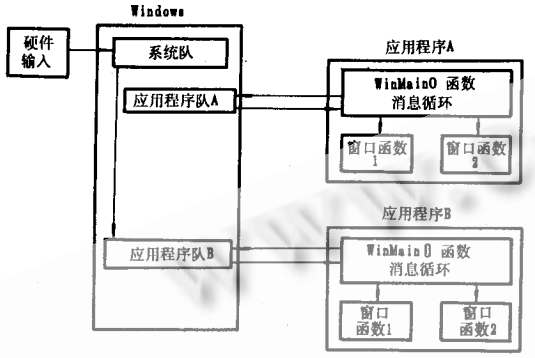

图 1 输入消息的传递处理

另外，应用程序可以调用 SendMessage()、PostMessage() 等函数在应用程序间传递自己定义的私有消息。

三、多任务机制

Windows 是一个多任务操作系统，它可以同时运行多个应用程序。标准的 MS-DOS 操作系统没有这种多任务机制。标准的 MS-DOS 应用程序通常独占计算机的所有资源，包括输入和输出设备、内存、屏幕甚至 CPU。而 Windows 应用程序必须与那些正在运行的应用程序共享这些资源。因此 Windows 系统必须仔细控制这些资源，并且要求 Windows 应用程序使用特定的程序接口以保证 Windows 系统正确地控制这些资源。

从原理上讲，有合作型和优先型两种多任务类型。在合作模式下，处理器只有在一个应用程序放弃其控制权时才能进行任务切换；而在优先模式下，处理器能在一个高优先级的任务到来时取得控制权并进行任务切换。Windows 的多任务是合作型的多任务，也就是非抢先的多任务。所谓"非抢先"是指 Windows 系统不能主动剥夺某个应用程序的控制权。一旦某个应用程序进行消息处理，就必须等到它处理完毕，并且让出系统的控制权之后，其它的应用程序才能得到系统的控制权。所以应用程序的消息处理过程要尽可能快，处理完毕立即让出系统的控制权。应用程序中不应当存在可能导致死循环(或较长时间循环，或者循环等待)的结构。

当一个应用程序进行消息处理时，系统把它从不活动状态变成活动状态；当它处理完毕并出让系统的控制权之后，系统把它从活动状态变成不活动状态。

而对其它的一些多任务操作系统，如 UNIX 则是用分时来实现的。从狭义来说，分时系统是把计算机的时间分成若干小的等分，并且在各个应用程序间分配这些时间。由于计算机是非常快速的设备，所以可以迅速地从一个作业转到另一个作业，从而造成了这样一种现象："计算机正在同时执行多个任务"。事实上计算机在一个任务上进行，然后转到下一个任务，再下一个，如此等等。

在一台分时的计算机系统中，系统时钟定期地产生中断。中断是一种硬件信号，它能够把计算机转向特定的应用程序。在系统时钟的服务程序运行过程中，进程的优先级数被重新计算，并且可能改变进程(一个程序正在运行，就称为一个进程。这和 Windows 中的"实例"类似)。在任何给定时刻，最多只有一个进程在活动，而其它所有进程都被挂起。一般分时系统动态地计算进程的优先级。以便决定在当前活动着的进程被挂起时，哪一个不在活动但已就绪的进程可投入运行的进程被挂起时，哪一个不在活动但已就绪的进程可投入运行。

由此可以看出，Windows 系统和其它多任务系统(UNIX 操作系统、iRMX 操作系统等)的主要区别是：Windows 系统必须在活动进程出让控制权之后才可以转换到其它的进程，是一种"非剥夺性"(或"非抢先性")的多任务系统。而 UNIX 操作系统则是通过"分时"来实现的，是一种"抢先性"多任务系统。

Windows 的应用程序可能一直占有系统的控制权，而使多任务机制失效，所以说 Windows 系统只提供基本的多任务能力。这对于一般的应用来说，是可以达到要求的。至于对多任务及实时能力要求较高的应用，可以选用分时系统。