属于操作系统本身，它通过运行在NTOS层顶层的子系统（subsystem）来实现的。

最早的NT支持三个个性化子系统：OS/2、POSIX、Win32。OS/2在 Windows XP中已经不使用了。POSIX也同样不使用了，但是客户可以得到一个叫做Interix的改进版POSIX的子系统，它是微软面向UNIX的服务(SFU)的一部分，因此所有设备都支持系统中原有的POSIX。尽管微软支持其他的API，但大多数Windows的应用软件都是用Win32写的。

不同于Win32，.NET并不是原来NT的内核接口上的正式的子系统。相反，.NET是建立在Win32编程模型之上的。这样就可以使.NET与现有的Win32程序很好地互通，而不必关心POSIX和OS/2子系统。WinFX API包含了很多Win32的功能，而实际上WinFX基本类库（Base Class Library）中大多数的功能都是Win32 API的简单包装器。WinFX的优点是有丰富的对象类型支持、简单一致的界面、使用.NET公共语言运行库（CLR）和垃圾收集器。

如图11-7所示，NT子系统建立了四个部分：子系统进程、程序库、创建进程（CreateProcess）钩子、内核支持。一个子系统进程只是一个服务。它唯一特殊的性质就是通过smss.exe程序（一个由NT启动的初始用户态程序）开始，以响应来自Win32的CreateProcess或不同的子系统中相应的API的请求。

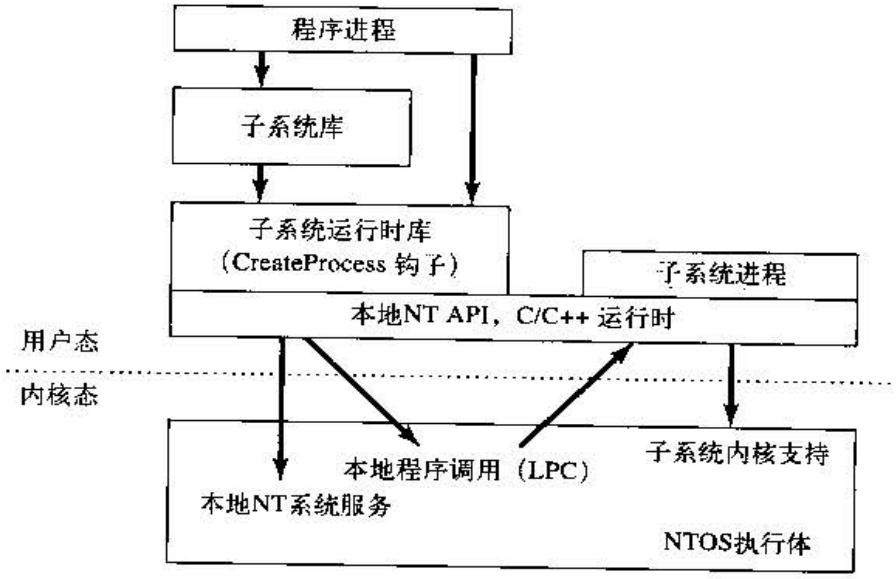

图11-7 用于构建NT子系统的模块

程序库同时实现了高层的操作系统功能和特定的子系统进程。这些高层的操作系统功能是特定于子系统以及子系统所包含的桩程序（stub routine）的。桩程序是进行不同的使用子系统的进程间通信的。对子系统进程的调用通常是利用内核态的本地过程调用LPC（Local Procedure Call）所提供的功能。LPC实现了跨进程的进程调用。

在Win32 CreateProcess中的钩子函数（hook）通过查看二进制图像来检测子系统中每个程序请求。（如果它没有运行）通过smss.exe启动子系统进程csrss.exe。然后子系统进程开始加载程序。在其他子系统中也有类似的钩子函数（例如POSIX中的exec系统调用）。

NT内核有很多一般用途的设备，可以用来编写操作系统特定的子系统。但是为了准确地执行每一个子系统还需要加入一些特殊的代码。例如，本地NtCreateProcess系统调用通过重复使用进程实现POSIXF fork函数调用，内核提供一个Win32特殊类型串表(叫atoms)，通过进程有效实现只读字符串的共享。

子系统进程是本地端NT程序，其使用NT内核和核心服务提供的使用本地系统调用，例如smss.exe和lsass.exe（本地安全管理）。本地系统调用包括管理虚拟地址的跨进程功能（facility）、线程、句柄和为了运行用来使用特定子系统的程序而创建的进程中的异常。

11.2.1 内部NT应用编程接口

像所有的其他操作系统一样，Windows Vista也拥有一套系统调用。它们在Windows Vista的 NTOS层实施，在内核态运行。微软没有公布内部系统调用的细节。它们被操作系统内部一些底层程序使用，