载系统启动组件到内存中：内核/执行体(通常是Ntoskrnl.exe)、HAL(hal.dll)，该文件包含系统储巢，Win32k.sys驱动包含Win32子系统的内核态部分，以及任何其他在系统储巢中作为启动驱动程序列出的驱动程序的镜像，这就意味着在系统启动时，它们是必需的。

一旦Windows启动组件加载到内存中，控制就转移给NTOS中的低级代码，来完成初始化HAL、内核和执行体、链接驱动像、访问/更新系统配置中的数据等操作。所有内核态的组件初始化后，第一个用户态进程被创建，使用运行着的smss.exe程序（如同UNIX系统中的/etc/init）。

Windows启动程序在遇到系统启动失败时，有专门处理常用问题的逻辑。有时安装一个坏的设备驱动程序，或运行一个像注册表一样的程序（能导致系统储巢损坏），会阻止系统正常启动。系统提供了一种功能来支持忽略最近的变化并启动到最近一次的系统正确配置。其他启动选项包括安全启动，它关闭了许多可选的驱动程序。还有故障恢复控制台，启动cmd.exe命令行窗口，它提供了一个类似UNIX的单用户态。

另一个常见的问题，用户认为，一些Windows系统偶尔看起来很不可思议，经常有系统和应用程序的（看似随机）崩溃。从微软的在线崩溃分析程序得到的数据，提供了许多崩溃是由于物理内存损坏导致的证据。所以Windows Vista启动进程提供了一个运行广义上的内存诊断的选项。也许未来的PC硬件将普遍支持ECC（或者部分）内存，但是今天的大多数台式机和笔记本电脑系统很容易受到攻击，即便是在它们所包含的数十亿比特的内存中的单比特错误。

11.3.3 对象管理器的实现

对象管理器也许是Windows可执行过程中一个最重要的组件，这也是为什么我们已经介绍了它的许多概念。如前所述，它提供了一个统一的和一致的接口，用于管理系统资源和数据结构，如打开文件、进程、线程、内存部分、定时器、设备、驱动程序和信号。更为特殊的对象可以表示一些事物，像内核的事务、外形、安全令牌和由对象管理器管理的Win32桌面。设备对象和I/O系统的描述联系在一起，包括提供NT名字空间和文件系统卷之间的链接。配置管理器使用一个Key类型的对象与注册配置相链接。对象管理器自身有一些对象，它用于管理NT名字空间和使用公共功能来实现对象。在这些目录中，有象征性的联系和对象类型的对象。

由对象管理器提供的统一性有不同的方面。所有这些对象使用相同的机制，包括它们是如何创建、销毁以及定额分配值的占有。它们都可以被用户态进程通过使用句柄访问。在内核的对象上有一个统一的协议管理指针的引用。对象可以从NT的名字空间(由对象管理器管理)中得到名字。调度对象（那些以信号事件相关的共同数据结构开始的对象）可以使用共同的同步和通知接口，如WaitForMultipleObjects。有一个共同的安全系统，其执行了以名称来访问的对象的ACL，并检查每个使用的句柄。甚至有工具帮助内核态开发者，在使用对象的过程中追踪调试问题。

理解对象的关键，是要意识到一个（执行）对象仅仅是内核态下在虚拟内存中可以访问的一个数据结构。这些数据结构，常用来代表更抽象的概念。例如，执行文件对象会为那些已打开的系统文件的每一个实例而创建。进程对象被创建来代表每一个进程。

一种事实上的结果是，对象只是内核数据结构，当系统重新启动时（或崩溃时）所有的对象都将丢失。当系统启动时，没有对象存在，甚至没有对象类型描述。所有对象类型和对象自身，由对象管理器提供接口的执行体的其他组件动态创建。当对象被创建并指定一个名字，它们可以在以后通过NT名字空间被引用。因此，建立对象的系统根目录还建立了NT名字空间。

对象结构，如图11-17所示。每个对象包含一个对所有类型的所有对象的某些共性信息头。在这个头的领域内包括在名字空间内的对象的名称，对象目录，并指向安全描述符代表的ACL对象。

对象的内存分配来自由执行体保持的两个堆（或池）的内存之一。在有（像内存分配）效用函数的执行体中，允许内核态组件不仅分配分页内核内存，也分配无分页内核内存。对于那些需要被具有CPU 2级以及更高优先级的对象访问的任何数据结构和内核态是对象，无分页内存都是需要的。这包括ISR和DPC（但不包括APC）和线程调度本身。该pagefault处理也需要由无分页内核内存分配的数据结构，以避免递归。

大部分来自内核堆管理器的分配，是通过使用每个处理器后备名单来获得的，这个后备名单中包含分配大小一致的LIFO列表。这些LIFO优化不涉及锁的运作，可提高系统的性能和可扩展性。