硬装入智能手机（有很大的难度），也不是使较大的操作系统适应于较小的平台。然而，它确实包含了许多其他大型操作系统所具有的特性，从多任务到内存管理再到安全问题。

Symbian操作系统继承了其前身的最佳的特性，具有由EPOC传承下来的面向对象特性。并且如版本6中所引入的，使用了微内核的设计方案，最小化了内核的开销，将不必要的功能移到了用户层进程。它模仿EPOC中应用的引擎模型，使用了客户机/服务器结构。它支持多种台式机功能，包括多任务和多线程，以及可扩展存储系统。它还继承了EPOC中强调的多媒体与通信。

12.2.1 面向对象

面向对象是一个意味着抽象的术语。在一个面向对象的设计中，针对某个系统成分的数据和功能，建立一个抽象的实体，称为对象。一个对象提供了具体的数据以及功能，但隐藏了具体实现。一个合理实现的对象可以被移除并被另外一个不同的对象代替，只要系统其他部分对这个对象的使用（也即其接口）保持不变。

当面向对象应用到操作系统设计中时，就意味着所有的系统调用以及内核端功能的使用均要通过接口，而不能直接获取实际数据或依靠其他类型的实现。一个面向对象的内核的实现通过对象来提供内核功能。使用内核端对象通常意味着一个应用程序具有一个对象的句柄，也就是对对象的一个引用，然后通过这个句柄来获得对该对象接口的访问。

Symbian操作系统采用了面向对象的设计。系统功能的实现是隐藏的，系统数据的使用通过系统对象已定义的接口完成。在Linux等操作系统中，构建一个文件描述符，并将这个描述符作为open调用的参数；而在Symbian操作系统中则会创建一个文件对象，然后调用该对象的open方法。举例来说，在Linux操作系统中，正如大家所知道的，文件描述符对应于系统内存中文件描述符表的索引的整数表示；而在Symbian操作系统中，文件系统表的实现是未知的，而所有的文件操作是通过一个特定的文件类的对象来实现的。

需要注意的是Symbian操作系统与其他在设计中运用了面向对象理念的操作系统不同。例如，许多操作系统设计使用了抽象数据类型，人们甚至可以说系统调用整个理念就是通过将系统实现细节对用户程序隐藏起来而实现了抽象。而对于Symbian操作系统，整个操作系统的结构均是面向对象设计的。操作系统功能以及系统调用都是与系统对象相联系的。资源分配以及保护则是对应于对象的分配，而不是系统调用的实现。

12.2.2 微内核设计

具有面向对象内在特性的Symbian操作系统的内核结构是微内核设计。内核中包括最小限度的系统功能以及数据，许多系统功能被放到了用户空间服务器端。服务器端通过获得系统对象的句柄并对这些对象进行必要的系统调用来完成各自的服务。用户空间应用程序与这些服务器端进行交互而不是采取系统调用。

典型的基于微内核的操作系统初始化引导时占用较少的内存，并且其结构也更加动态。当需要时可以启动服务器，而在启动时并不需要全部的服务器。微内核大多为可插拔结构，允许当需要时加载系统模块并插入到内核中。因此，微内核结构十分灵活：支持新功能的代码（例如，新硬件驱动程序）可以随时加载和插入。

Symbian操作系统被设计为基于微内核的操作系统。通过打开与资源服务器端的连接访问系统资源，资源服务器随后协同访问资源本身。Symbian操作系统支持对于新的实现的可插拔结构。对于系统功能的新的实现可以设计为系统对象，并动态插入到内核中。例如，可以实现新的文件系统并且在操作系统运行的同时添加到内核中。

这种微内核的设计也带来了一些需要探讨的话题。在传统的操作系统中一个系统调用便已足够时，微内核使用消息传递。性能可能会由于对象间通信所增加的花费而受到影响。在传统操作系统中位于内核的那些功能被移到用户空间时效率可能会降低。举例来说，与可以直接访问内核数据结构的Windows内核中的进程调度相比，进程调度的多函数调用的开销降低了性能。由于在用户空间与内核空间对象中传递消息，会经常发生特权级切换，这就更降低了它的性能。最后，在传统设计方案中只用到了一个地址空间的系统调用，而这种消息传递以及优先级转换意味着至少需要用到两个地址空间来完成一个微内核服务请求。