用程序需要的内存页面能够从操作系统中静态请求进入内存。对于堆（更确切地说是动态空间）是有界限的，因此静态请求也可以由动态空间来实现。从一个空闲页框的列表中分配页框给页面；如果没有空闲页框，那么就会出现错误。正在使用的页框不能被刚到的应用程序的页面替换，即使该页框是针对当前没有执行的应用程序的。这是由于Symbian操作系统中没有页交换，同时因为十分有限的闪存空间只给用户文件使用，也就没有空间来复制被收回空间的页面。

实际上Symbian操作系统使用的存储实现模型有四种不同的版本。每种模型都是为了特定类型的硬件配置。一个简要的列表如下：

- 移动模型：该模型是为早期的ARM体系结构设计的。移动模型中的页目录是4KB长，每个条目4字节，给出一块16KB大小的目录。通过与页框相关的存取位和使用域来标志存储器访问的方式保护存储页面。域信息记录在页表目录，MMU为每个域实现访问权限。尽管没有明确使用分段，但是在内存布局上有如下结构：有用户分配数据的数据区，也有内核分配数据的内核区。
- 复合模型：该模型是为ARM6或者之后的体系开发的模式。这些版本中的MMU与以前版本使用的不同。例如，由于页表目录能分成两部分，每个部分索引页表的不同部分，因此需要不同的处理方式。这两部分分别用作用户页表和内核页表。ARM体系中新的版本修订并增强了每个页框的访问位，但是不赞成使用域的概念。
- 直接模型：该模型假定根本就没有MMU。这一模型很少使用，并且在智能手机上禁止使用。没有MMU会导致严重的性能问题。由于某些原因，MMU被禁止的一些场合中该模式比较有用。
- 仿真模型：该模型是为了支持基于windows宿主机的Symbian操作系统仿真器。仿真器与实际的目标CPU几乎没有区别。仿真器作为一个单独的Windows进程运行，因此地址空间被限定为2GB，而不是4GB。为仿真器提供的所有内存可以被任何Symbian操作系统进程访问，因此不具有内存保护。Symbian操作系统库以Windows格式的动态链接库形式提供，因此Windows处理内存的分配和管理。

12.5 输入和输出

Symbian操作系统的输入/输出结构仿照其他操作系统的设计。本节会指出其中一些Symbian操作系统特有的基于自己目标平台的性质。

12.5.1 设备驱动

在Symbian操作系统中，设备驱动作为具有内核权限的代码运行，从而赋予用户级别的代码对系统保护资源的访问能力。同Linux与Windows一样，设备驱动程序代表软件去访问硬件。

Symbian操作系统中的设备驱动分为两层：一个是逻辑设备驱动（LDD），一个是物理设备驱动（PDD）。LDD为上层软件提供一个接口，而PDD直接与硬件进行交互。在这种模型下，LDD可以为一类特定的设备使用相同的实现，而PDD随着不同的设备改变。Symbian操作系统支持许多标准的LDD。有时，如果硬件非常标准或者常用，Symbian操作系统也提供PDD。

考虑串行设备的一个例子。Symbian操作系统定义了一个通用的串行LDD，该LDD定义了访问串行设备的程序接口。LDD给PDD提供一个接口，PDD提供串行设备访问接口。PDD实现有助于调节CPU和串行设备之间速度差异所必需的缓冲和流控制机制。一个单一的LDD（用户那边）可以连接任何用来运行串行设备的PDD。在某个特定的智能手机上，这些PDD可能包括一个红外端口或者一个RS-232端口。这两个是非常好的例子，它们使用相同的串行LDD，但是使用不同的PDD。

当LDD和PDD不在内存中时，它们可以由用户程序动态地加载进内存。程序编制工具能够检查是否需要加载。

12.5.2 内核扩展

内核扩展就是Symbian操作系统在引导时刻加载的驱动程序。由于它们是在引导时刻加载的，因此是与标准的设备驱动区别对待的特殊情况。

内核扩展与标准的设备驱动不同。大多数设备驱动是由LDD同成对的PDD实现的，在用户空间程序需要的时候加载。内核扩展在引导时刻加载，针对特定的设备，通常没有成对的PDD。