用来隐藏端点之间的传输方法和数据管理。Symbian操作系统使用套接字的概念在客户端和服务器端之间、线程到设备之间以及线程之间进行通信。

套接字模型也构成了设备I/O的基础。抽象再次成为使这一模型更加有效的关键。同一个设备进行数据交换的所有机制不是由应用程序管理的，而是由操作系统管理的。例如，网络环境中工作于TCP/IP上的套接字可以很容易地通过改变套接字使用的类型参数而适应于蓝牙环境。这种变换下，其他大部分的数据交换工作都是由操作系统完成的。

Symbian操作系统实现了通用操作系统上使用的标准同步原语。操作系统中广泛地使用了信号量和互斥量的一些形式。这些为进程和线程提供同步能力。

12.4 内存管理

诸如Linux和Windows系统中的内存管理使用了很多我们前面讲过的关于实现内存资源管理的概念。例如，从物理内存框架构建的虚拟内存页面、按需分页的虚拟内存以及动态页面置换，这些概念共同给出近乎无限的内存资源形象。这里物理内存是由诸如硬盘空间等支持和扩展的。

Symbian操作系统和实际的通用操作系统一样，也必须提供内存管理模式。然而，由于智能手机上的存储容量非常有限，内存模型受到限制，而且进行内存管理的时候不能使用虚拟内存/交换空间模型。但正是如此，Symbian操作系统使用了我们讨论过的内存管理的大多数机制，包括硬件MMU。

12.4.1 没有虚拟内存的系统

许多计算机系统没有提供成熟的支持按需分页的虚拟内存的设备。在这些平台上操作系统可以获得的惟一的存储设备就是内存，它们没有硬盘设备。正因为这样，大多数较小的系统，从PDA到智能手机，再到更高层次的掌上设备，都不支持按需分页的虚拟内存。

下面考虑大多数小的平台设备上使用的内存空间。这些系统一般都有两种类型的存储介质：RAM和闪存。RAM存储操作系统代码（当系统启动时使用）、闪存用作操作内存和永久性（文件）存储介质。通常，可以为一个设备（比如安全数据卡）增添额外的闪存，这些存储空间专门用作永久性存储。

没有支持按需分页的虚拟内存不代表缺少内存管理。实际上，大多数较小的平台构建在包含许多较大型系统的管理特征的硬件上。这些管理特征包含诸如分页、地址翻译以及虚拟/物理地址抽象。没有虚拟内存仅仅意味着页面不能从内存交换出去并存储在外部设备上，而内存页的抽象仍然在使用。页面被替换了，但是它们也只是被丢弃了。也就是说只有代码页可以被置换，因为只有它们备份在闪存上。

内存管理包含如下的任务：

- 应用程序大小的管理：应用程序的大小（所有的代码和数据）对如何使用内存有很大的影响。创建小的软件需要技巧和规则。使用面向对象的设计在这里成为一种阻碍（更多的对象意味着更多的动态内存分配，而这需要更大的堆尺寸）。大多数针对较小平台的操作系统非常不鼓励任何模块的静态链接。
- 堆的管理：堆（用来进行动态内存分配的空间）在较小的平台上必须严格地管理。堆空间在较小的平台上一般划定边界，以便程序员尽可能地回收和重用。冒险越界会导致内存分配的错误。
- 就地执行：没有磁盘设备的平台通常支持就地执行。这就是说闪存被映射到虚拟内存地址空间，程序可以直接从闪存上执行，而不需要首先复制到RAM上。这样做使加载时间减小为零，允许应用程序迅速启动，而且也不需要使用稀缺的RAM。
- 加载动态链接库：什么时候加载动态链接库的选择会明显影响系统性能。例如，当应用程序第一次加载到内存就加载所有的动态链接库，比在执行中不定时发生的加载更加容易接受。比起执行时应用程序发生延迟，用户更加能够接受启动它时有一些滞后。注意动态链接库可能并不需要加载，如果它们已经在内存中或者它们包含在外部闪存中（在这种情况下，它们可以就地执行）就是这种情况。
- 卸下内存管理给硬件：如果有MMU，尽可能地使用它。实际上，将越多的功能放入MMU，系统的性能越好。

即使使用就地执行的规则，较小的平台仍然需要保留一部分内存用作操作系统操作。这些内存与永