来创建一个新进程，而不是通过映射一个新程序的段对象来新建进程。这种方式只用在实现POSIX的fork，而不是Win32的。

线程创建时传给新线程的参数包括：CPU的上下文信息（包括栈指针和起始指令地址）、TEB模板、一个表示线程创建后马上运行或以挂起状态创建（等待有人对线程句柄调用NtResumeThread函数）的标志。用户态下的栈的创建以及argv/argc参数的压入需要由用户态下的代码来解决，必须对进程句柄调用本地NT的内存管理API。

在Windows Vista的发行版中，包含了一个新的关于进程操作方面的本地API，这个接口将原来许多用户态下的步骤转移到了内核态下执行，同时将进程创建与起始线程创建绑定在一起进行。作这种改变的原因是支持通过进程划分信任边界。一般来说，所有用户创建的进程被同等信任，由用户决定信任边界在哪里。在Windows Vista中的这个改变，允许进程也可以提供信任边界，但是这意味着对于新进程句柄来说，创建者进程没有足够的权利在用户态下实现进程创建的细节。

1. 进程间通信

线程间可以通过多种方式进行通信，包括管道、命名管道、邮件槽、套接字、远程过程调用（RPC）、共享文件等。管道有两种模式：字节管道和消息管道，可以在创建的时候选择。字节模式的管道的工作方式与UNIX下的工作方式一样。消息模式的管道与字节模式的管道大致相同，但会维护消息边界。所以写入四次的128字节，读出来也是四个128字节的消息，而不会像字节模式的管道一样读出的是一个512字节的消息。命名管道在Vista中也是有的，跟普通的管道一样都有两种模式，但命名管道可以在网络中使用，而普通管道只能在单机中使用。

邮件槽是OS/2操作系统的特性，在Windows中实现只是为了兼容性。它们在某种方式上跟管道类似，但不完全相同。首先，它们是单向的，而管道则是双向的。而且，它们能够在网络中使用但不提供有保证的传输。最后，它们允许发送进程将消息广播给多个接收者而不仅仅是一个接收者。邮件槽和命名管道在Windows中都是以文件系统的形式实现，而非可执行的功能函数。这样做就可以通过现有的远程文件系统协议在网络上来访问到它们。

套接字也与管道类似，只不过它们通常连接的是不同机器上的两个进程。例如，一个进程往一个套接字里面写入内容，远程机器上的另外一个进程从这个套接字中读出来。套接字同样也可以被用在同一台机器上的进程通信，但是因为它们比管道带来了更大的开销，所以一般来说它们只被用于网络环境下的通信。套接字原来是为伯克利UNIX而设计的，它的实现代码很多都是可用的，正如Windows发布日志里面所写的，Windows代码中使用了一些伯克利的代码及数据结构。

远程过程调用（RPC）是一种进程A命令进程B调用进程B地址空间中的一个函数，然后将执行结果返回给进程A的方式。在这个过程中对参数的限制很多。例如，如果传递的是个指针，那么对于进程B来说这个指针毫无意义，因此必须把数据结构打包起来然后以进程无关的方式传输。实现RPC的时候，通常是把它作为传输层之上的抽象层来实现。例如对于Windows来说，可以通过TCP/IP套接字、命名管道、ALPC来进行传输。ALPC的全称是高级本地过程调用（Advanced Local Procedure Call），它是内核态下的一种消息传递机制，为同一台机器中的进程间通信作了优化，但不支持网络间通信。基本的设计思想是可以发送有回复的消息，以此来实现一个轻量级的RPC版本，提供比ALPC更丰富的特性。ALPC的实现是通过拷贝参数以及基于消息大小的临时共享内存分配。

最后，进程间可以共享对象，如段对象。段对象可以同时被映射到多个进程的虚拟地址空间中，一个进程执行了写操作之后，其他进程可以也可以看见这个写操作。通过这个机制，在生产者消费者问题中用到的共享缓冲区就可以轻松地实现。

2. 同步

进程间也可以使用多种形式的同步对象。就像Windows Vista中提供了多种形式的进程间通信机制一样，Vista也提供了多种形式的同步机制，包括信号量、互斥量、临界区和事件。所有的这些机制只在线程上工作，而非进程。所以当一个线程由于一个信号量而阻塞时，同一个进程的其他线程（如果有的话）会继续运行而并不会被影响。

使用Win32的API函数CreateSemaphore可以创建一个信号量，可以将它初始化为一个给定的值，