1. MACH

Mach是一个由卡内基梅隆大学开发的用于支持操作系统研究的操作系统内核，为了用于操作系统之研究，特别是在分布式与并行运算上。是最早实现微核心操作系统的例子之一。

UNIX“把一切事物作为文件”， 在UNIX 中最关键的抽象是管(pipe)的概念。我们需要的是一个在一个更一般的水平上的管(pipe)的概念，从而使程序之间能传递各种各样的信息。这样的系统 使用进程间通信（inter-process communication，IPC）：一个管概念(pipe-like)的系统能在两个程序间交换信息，但不同于类似文件的信息。

Mach开始主要是作为一个明确定义的，基于UNIX的，高度可移植的成果。其包含以下的一般概念：  
　　1“ 任务 “是由一组对象组成的为”线程“提供运行能力的系统资源组成的  
　　2“ 线程 “是一个单一的执行单元，存在于一个任务的上下文和共享任务的资源  
　　3“ 端口 “是一个受保护的任务之间消息队列，任务之间的通信任务拥有每个端口的发送和接收的权利  
　　4“ “消息 “是数据对象的集合，他们只能被发送到端口，而不是特定的任务或线程  
　 　虽然马赫的开发是基于Accent的IPC的概念，但其使系统更加地类似于UNIX，甚至可以直接运行很少或根本没有修改的UNIX程序。要做到这一 点，马赫引入了一个端口（port)的概念，其每个端点的双向IPC。端口如UNIX下的文件一样有安全性和权限，允许类似UNIX的保护模式在其操作平 台上应用。此外，，为了让用户程序如和硬件交互般处理任务，马赫允许任何程序处理任务。

和UNIX操作系统一样，Mach再次成为主要实用工具的集合。与UNIX相比，马赫保持了处理硬件驱动程序的概念。因此，所有的硬件的驱动程序都必须包含在微内核中。其他基于硬件抽象层或exokernels的架构能将驱动程序移出微内核。  
　 　和UNIX的主要区别是，Mach不是处理文件的实用工具，他们更多的用于处理任何“任务”。更多操作系统的代码被出内核放入到用户空间，从而产生了更 小的内核及其崛起式的发展。与传统系统不同的是，在Mach中，程序或“任务”可以包含的多个线程。虽然在现代系统中，多线程是常见的，但马赫是第一个以 多线程方式处理任务的系统。  
　　端口概念和IPC的使用也许是马赫和传统的内核之间最根本的区别。在UNIX下，调用内核的操作被称为一个系统调 用（syscall）或陷阱（trap）。程序在共有的存储中存放数据，然后会导致中断（fault），或者说是某种类型的错误。当系统第一次启动时内核 处理所有的异常，所以当某个程序产生了异常，内核接管该异常，检查传递来的异常信息，然后进行处理。  
　　**IPC系统率先被使用于Mach内核。为了调用系统的功能，程序会向内核申请一个端口通道，然后使用IPC系统将消息发送到该端口。**  
　 　使用IPC传递消息受益于线程和并发性。由于任务是由运用IPC方法的多线程组成的，马赫能冻结和解冻线程从而同时处理消息。这使得系统能分布在多个处 理器上，可以直接使用共享内存，或添加代码到另一个处理器。在传统的内核中这是很难实现的：系统必须确保不同的程序没有从不同的处理器尝试写入到相同的内 存。在Mach中，这是定义良好，易于实现的：端口是一个良好的方法。

通过一个Mach端口发送一个消息调用一次mach\_msg\_send方法，但是这里需要做一些配置来构建待发送的消息。（消息）接收端稍微轻松点，调用mach\_msg\_receive(&message.header);