#### 操作系统科普

在早期想要编写程序并在电脑上运行，就得要参考一堆硬件功能函数（可理解成厂商放在硬件上的固件程序），而且还要学习机器语言。而如果能够将所有的硬件都驱动，并且提供一些功能调用接口给工程师开发软件的话，那发展软件就会变得非常简单，这个想法就是操作系统。

**操作系统内核功能：**操作系统其实也是一组程序（在内存中的区块是受保护的，开机后就一直存在于内存中），这组程序的重点在于管理电脑的所有活动以及驱动系统中的所有硬件。只有内核有提供的功能，你的电脑系统才能帮你完成。举例来说，你的核心并不支持TCP/IP协议，那么无论是什么网卡，操作系统都无法提供网络功能。

内核提供核心功能，包括内存管理、进程管理、硬件管理（声卡，显卡，网卡，串口等等这些）、文件系统管理。提一下硬件管理，Linux系统需要与之通信的设备都必须在内核代码中插入驱动程序代码，驱动程序代码是内核能够像设备传输数据。在Linux中插入设备驱动的程序有两种方法：

1. 在内核中编译驱动程序；
2. 向内核添加驱动程序模块（常用）。

Linux将硬件设别标识为特殊文件，称为设备文件，一共分为3种：1、字符；2、块；3、网络。字符设备文件用于那些一次仅处理一个字符的设备。块文件用于那些一次可处理大量数据块的设备。网络文件用于那些使用数据包发送和接收数据的设备。

**系统调用（System Call）：**既然我的硬件都是由核心管理，那么如果我想开发软件的话，自然就得要去参考这个核心的相关功能！如此一来就从原本的参考硬件函数变成核心功能，还是很麻烦。

为了解决这个问题，操作系统通常都会提供一整组的开发接口给工程师来开发软件！比如C语言的printf函数，我们只要了解该函数的接口参数定义相关即可，内核的系统调用接口会将该函数交由底层实现。

电脑系统主要由硬件构成，然后核心程序主要在管理硬件，提供合理的电脑系统资源分配（包括CPU资源、内存使用资源等等）， 因此只要硬件不同（如x86架构与RISC架构的CPU），核心就得要进行修改才行。 而由于核心只会进行电脑系统的资源分配，所以在上头还需要有应用程序的提供，使用者才能够操作系统的。为了保护核心，并且让程序设计师比较容易开发软件，因此操作系统除了核心程序之外，通常还会提供一整组开发接口， 那就是系统调用层。软件开发工程师只要遵循公认的系统调用参数来开发软件，该软件就能够在该核心上头运行。 所以你可以发现，软件与核心有比较大的关系，与硬件关系则不大！硬件也与核心有比较大的关系！至于与使用者有关的，那就是应用程序啦！

操作系统关键词：硬件驱动、软件开发接口（printf，socket这些）、资源管理、进程调度；

###### 总结

1. 操作系统的内核直接参考硬件规格写成，所以同一个操作系统不能在不一样的硬件架构下运行（比如Windows不能在手机上运行）；
2. 操作系统只是在管理整个硬件资源，包括CPU、内存、输入输出设备及文件系统文件。如果没有其他的应用程序辅助，操作系统只能让电脑主机准备妥当而已；
3. 应用程序的开发都是参考操作系统的开发接口（显示各种界面、调用网络接口、申请内存、读取硬盘、控制设备等等），所以应用软件一般都会针对不同的操作系统开发多个版本；
4. 硬件厂商在推出硬件时，一般会自行参考操作系统驱动程序开发接口开发驱动程序，所以驱动程序是设计的，与操作系统开发人员无关；
5. BIOS是主板厂商开发的程序，主要工作是在操作系统加载之前的一段时间，它的功能包括引导操作系统；它永久的嵌在一个只读内存中（掉电不失）。在学习微机原理时，DOS操作系统绝大部分系统调用功能底层实现也是调用BIOS功能。

#### Linux

POSIX是可携式操作系统接口（Portable Operating System Interface）的缩写，重点在规范核心与应用程序之间的接口， 这是由美国电器与电子工程师学会（IEEE）所发布的一项标准。

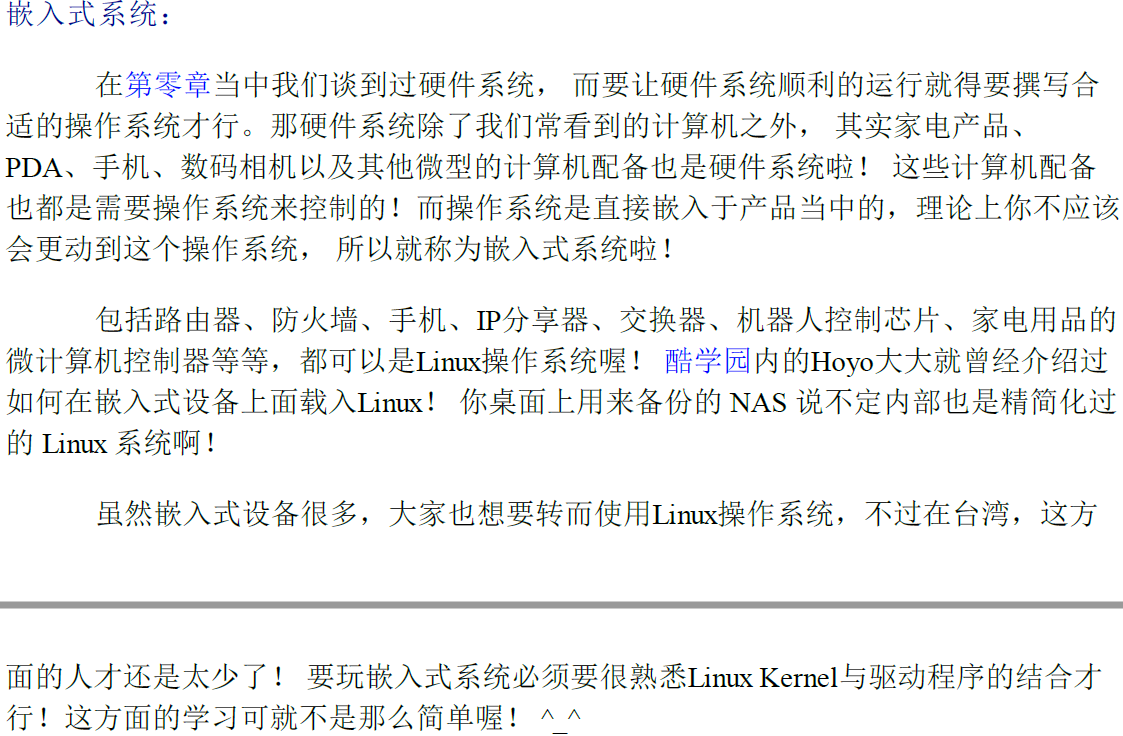


图１　以后的主要方向

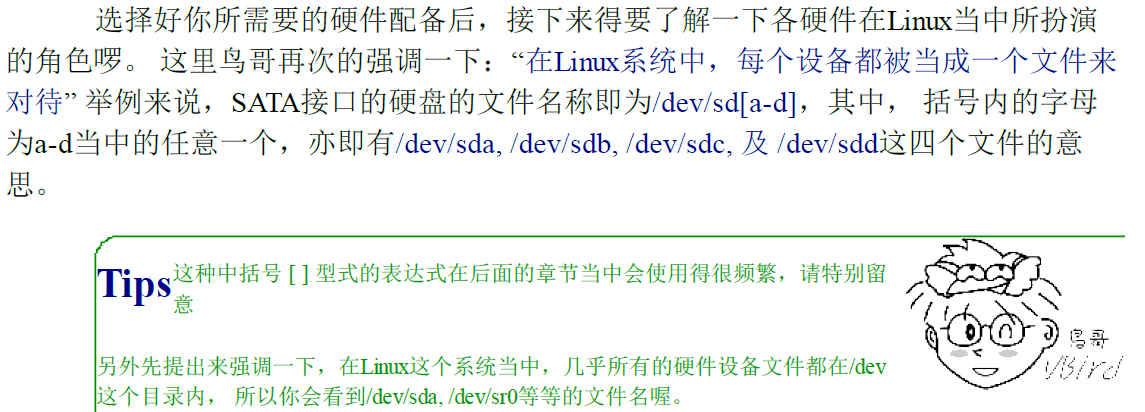


图２　一切皆文件