# 系统开机过程

## 开机过程的概念

* 1. Linux开机过程指的是从按下电源按键开始，到显示用户登录界面为止的一个过程，这个过程中系统在做什么，是这里要研究的，而且，在这个过程中，我们可以自定义什么，哪些部分是可控的，也是这里讨论的重点；

## 启动过程：

### 第一步：启动BIOS

* + 1. 打开电源，计算机会先启动BIOS，计算机必须而且立刻要找到BIOS设置，因为这里存放着CPU的相关信息，设备启动顺序，硬盘信息，内存信息等重要特性；然后，计算机就知道下一步该去哪个设备继续操作；然后BIOS将系统的控制权交给硬盘的第一个扇区，开始由Linux控制系统；
       1. Bios：基本输入输出系统，在IBM兼容PC系统上，是一种业界标准的固件接口，主要有两个功能：检测系统各部分和加载引导程序；如果在检测系统各部分中是否有问题，则会打印出相关的错误信息，检测主要包括初始化硬件，检测硬件功能；如果检测通过，则加载引导程序或者操作系统；程序会直接使用bios功能间接控制硬件，而现代的操作系统则是直接控制硬件，忽略bios提供的抽象层；
       2. Bios存放在一个断电后内容不会丢失的只读内存中，当系统被重置或者过电时，处理器的第一条指令的地址就会被传递到bios的内存中，让初始化程序开始运行；2000年，英特尔公司开发了“可扩展接口”，即EFI，用来规范bios设置界面，后来，英特尔公司宣布开放EFI，统一制定bios的标准，就有了后来的UEFI；它的作用位置如下图所示：



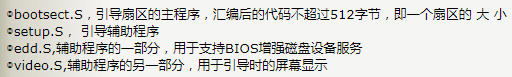
* + - 1. UEFI的产生，本质上是为系统硬件和预启动软件之间提供接口规范，它可以操作所有的硬件；它类似于一个低阶的操作系统，但设计之初的要求就是不能够威胁到操作系统的地位；所以，有很多操作系统的功能，它并没有设计开发；
      2. 当UEFI所有组件加载完毕，则开始引导操作系统的引导程序，或者启动一个shell，也称为EFIshell，它可以调用EFI应用程序，EFI驱动程序等进行硬件检测，开机管理软件等功能；EFI格式的应用程序可以使PE格式的.efi文件，用C语言编写；一旦引导程序将控制权交给操作系统，则引导程序的代码全部停止运行；
      3. UEFI固件区分架构，当使用UEFI固件开机时，通常只能运行特定架构的操作系统；比如：使用64位UEIF固件的pc，则只能使用64位操作系统；而使用Legacy开机模式（bios兼容开机模式），则可以运行16,32,64位的操作系统；

### 第二步：读取MBR

* + 1. 硬盘上的第0个磁道的第一个扇区，存放着MBR，即主引导记录，大小为512字节，里面存放着预启动信息，分区表信息；预启动信息是引导区，占446个字节，作用之一是找到标记为活动（active）的分区，将活动分区的引导区读入内存；分区表是存放着分区信息，占66个字节；
    2. 当系统找到BIOS指定的硬盘的MBR时，就会将一些信息复制到0x7c00地址的物理内存中；而被复制的内容就是boot loader， 具体就是grub或者是Lilo；

### 第三步：启动boot loader

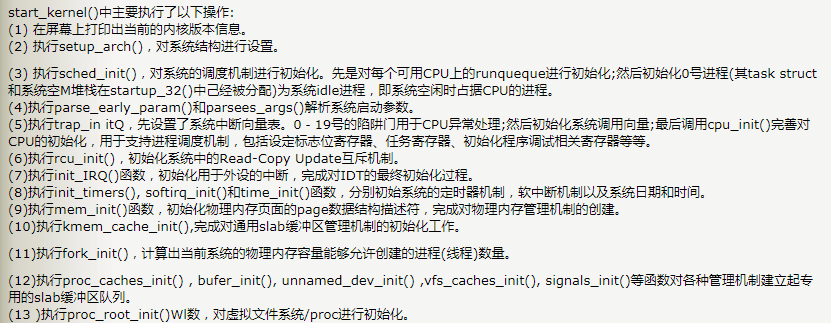
* + 1. 其实boot loader就是一段用汇编语言编写的小程序，通过运行这段小程序，可以初始化设备，建立内存空间的映射图，目的是将系统软硬件带到一个合适的状态中，最终为调用系统内核做好一切准备；通常情况下，boot loader是严重依赖于硬件配置的，不同体系的系统存在不同的boot loader;linux的引导扇区文件是由汇编语言编写的，源代码放在arch/i386/boot中，当然，不同的体系有不同的位置；有四个文件，如下图所示：



* + 1. Boot loader 有很多种，不同系统采用不同的方式，其中，grub,lilo,spfdisk是最常见的；这里以grub为例（因为它是使用最广泛的）；
       1. 系统读取内存中的grub信息（一般为menu.lst或grub.lst），依照此配置来启动操作系统；
       2. Grub 简介：
          1. Grub，原来是GNU项目上的一个启动引导程序(**Grand Unified Bootloader**)，它用来启动不同的操作系统或者内核文件，允许用户拥有多个操作系统；
          2. 它是独立于系统之外，所以，即使硬盘中未安装操作系统，也可以使用grub来引导安装；
          3. 它可以安装在硬盘中，也可以安装在U盘，移动设备上引导一个无操作系统的机器；
          4. Grub可以引导多种操作系统，有专门的教程和命令，有时间再探讨这个；

### 第四步：加载系统内核

* + 1. 根据grub所设定的内核映像路径，系统读取内核映像，并进行解压缩操作；一般屏幕上会显示“uncompressing linux”的提示，当解压缩完成后，屏幕会输出“OK，boot the kernel”
    2. 系统将解压缩的内核文件放在内存中，并调用start\_kernel函数来启动一系列的初始化函数并初始化各种设备，来建立Linux的核心环境；至此，Linux内核已经完全建立起来，而基于Linux的程序也可以正常运行；
    3. Start\_kernel函数定义在init/main.c中，它是类似于main的一般函数，系统真正的初始化操作从这里开始，它会调用一系列初始化函数，用来完成内核本身各方面的设置，最终建立起Linux内核的核心体系；而在调用这个函数之前的操作，仅仅是系统为启动内核而做的铺垫；
    4. Start\_kernel函数主要执行了以下操作，如图所示：



* + 1. 在start\_kernel函数的结尾，内核通过kernel\_thread()创建了第一个系统内核线程（即1号进程），该进程将执行内核中的init函数，负责下一个阶段的启动任务；最后调用cpus\_idle函数，它将会进入系统主循环体，默认将一直执行defalut\_idle函数中的指令，就是CPU中的halt指令，直到其他进程需要被调度是才去执行其他操作；

### 第五步：init根据inittab文件设定运行等级

* + 1. 内核被加载后，第一个运行的程序是/sbin/init，它会调用/etc/inittab，读取其文件，根据此文件进行初始化操作；其实/etc/inittab中最重要的目的是设定Linux的运行等级，设定格式类似“id:5:initdefault:”表示默认运行等级为5；Linux有七个运行等级，在另一个文件中讨论过；

### 第六步：init进程执行rc.sysinit文件

* + 1. 在设定了运行等级后，Linux内核执行的第一个用户层文件就是/etc/rc.d/rc.sysinit，这是一个脚本程序，它会做很多工作，包括设置环境变量，设置网络配置，启动swap分区，设置虚拟内存等；
    2. 线程init最终完成的标志是可以让一般用户的应用程序可以被正常执行，从而搭建起一个可供软件运行的系统环境，因此，这个线程主要进行的操作有
       1. 执行do\_basic\_setup函数，它会对外部设备进行全面的初始化；
       2. 构建系统的虚拟文件系统目录树，挂载系统中作为根目录的设备；
       3. 打开设备dev/console,并将函数sys\_dup（）打开的连接复制两次，使得文件号0,1,2三个文件全部指向控制台，这就是我们的标准输入，标准输出，出错信息这三个标准的I/O通道；
       4. 准备好上述操作，内核进入用户层初始化阶段；内核通过系统调用execve()加载执行相应的用户层初始化程序；依次开始加载"/sbin/initl"," /etc/init"," /bin/init'，和“/bin/sh。，
       5. 只要上述的文件，有一个被加载成功，则会开始进行用户层的初始化，而不会再回到init的函数段中；至此，init函数结束，内核引导部分结束；

### 第七步：启动内核模块

* + 1. 具体是根据/etc/modules.conf和/etc/modules.d下的文件来进行装载内核模块；

### 第八步：执行不同运行级别的脚本程序

* + 1. 根据系统运行级别的不同，系统会运行rc0.d到rc6.d不同运行级别中相应的脚本程序，来完成相应的初始化操作和启动相应的服务；

### 第九步：执行/etc/rc.d/rc.local

* + 1. 其实rc.local文件就是在一切系统启动完成后，留给用户进行自定义的地方，可以把想启动和设置的东西都会放在这里；

### 第十步：执行/bin/login，进入登录状态

* 1. 此时，系统进入登录界面，等待用户输入账号和密码；