开机过程指的是从打开计算机电源直到LINUX显示用户登录画面的全过程。分析LINUX开机过程也是深入了解LINUX核心工作原理的一个很好的途径。

　　启动第一步--加载BIOS

　　当你打开计算机电源，计算机会首先加载BIOS信息，BIOS信息是如此的重要，以至于计算机必须在最开始就找到它。这是因为BIOS中包含了CPU的相关信息、设备启动顺序信息、硬盘信息、内存信息、时钟信息、PnP特性等等。在此之后，计算机心里就有谱了，知道应该去读取哪个硬件设备了。在BIOS将系统的控制权交给硬盘第一个扇区之后，就开始由Linux来控制系统了。

　　启动第二步--读取MBR

　　硬盘上第0磁道第一个扇区被称为MBR，也就是Master[Boot](http://www.chinabyte.com/keyword/Boot/" \t "http://blog.csdn.net/tzy3013218117/article/details/_blank) Record，即主引导记录，它的大小是512字节，可里面却存放了预启动信息、分区表信息。可分为两部分：第一部分为引导(PRE-BOOT)区，占了446个字节;第二部分为分区表(PARTITION PABLE)，共有66个字节，记录硬盘的分区信息。预引导区的作用之一是找到标记为活动(ACTIVE)的分区，并将活动分区的引导区读入内存。

　　系统找到BIOS所指定的硬盘的MBR后，就会将其复制到0×7c00地址所在的物理内存中。其实被复制到物理内存的内容就是Boot Loader，而具体到你的电脑，那就是lilo或者grub了。

　　启动第三步--Boot Loader

　　Boot Loader 就是在[操作系统](http://soft.chinabyte.com/os/" \t "http://blog.csdn.net/tzy3013218117/article/details/_blank)内核运行之前运行的一段小程序。通过这段小程序，我们可以初始化硬件设备、建立内存空间的映射图，从而将系统的软硬件环境带到一个合适的状态，以便为最终调用操作系统内核做好一切准备。通常，BootL oade:是严重地依赖于硬件而实现的，不同体系结构的系统存在着不同的Boot Loader。

　　Linux的引导扇区内容是采用汇编语言编写的程序，其源代码在arch/i386/boot中(不同体系的CPU有其各自的boot目录)，有4个程序文件：

　　◎bootsect.S，引导扇区的主程序，汇编后的代码不超过512字节，即一个扇区的 大 小

　　◎setup.S， 引导辅助程序

　　◎edd.S,辅助程序的一部分，用于支持BIOS增强[磁盘](http://www.chinabyte.com/keyword/%E7%A3%81%E7%9B%98/" \t "http://blog.csdn.net/tzy3013218117/article/details/_blank)设备服务

　　◎video.S,辅助程序的另一部分，用于引导时的屏幕显示

　　Boot Loader有若干种，其中Grub、Lilo和spfdisk是常见的Loader，这里以Grub为例来讲解吧。

　　系统读取内存中的grub配置信息(一般为menu.lst或grub.lst)，并依照此配置信息来启动不同的操作系统。

　　启动第四步--加载内核

　　根据grub设定的内核映像所在路径，系统读取内存映像，并进行解压缩操作。此时，屏幕一般会输出“Uncompressing Linux”的提示。当解压缩内核完成后，屏幕输出“OK, booting the kernel”。

　　系统将解压后的内核放置在内存之中，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，完成Linux核心环境的建立。至此，Linux内核已经建立起来了，基于Linux的程序应该可以正常运行了。

　　start\_kenrel()定义在init/main.c中，它就类似于一般[可执行程序](http://www.chinabyte.com/keyword/%E5%8F%AF%E6%89%A7%E8%A1%8C%E7%A8%8B%E5%BA%8F/" \t "http://blog.csdn.net/tzy3013218117/article/details/_blank)中的main()函数，系统在此之前所做的仅仅是一些能让内核程序最低限度执行的初始化操作，真正的内核初始化过程是从这里才开始。函数start\_kerenl()将会调用一系列的初始化函数，用来完成内核本身的各方面设置，目的是最终建立起基本完整的Linux核心环境。

　　start\_kernel()中主要执行了以下操作：

　　(1) 在屏幕上打印出当前的内核版本信息。

　　(2) 执行setup\_arch()，对系统结构进行设置。

　　(3)执行sched\_init()，对系统的调度机制进行初始化。先是对每个可用CPU上的runqueque进行初始化;然后初始化0号进程(其task struct和系统空M堆栈在startup\_32()中己经被分配)为系统idle进程，即系统空闲时占据CPU的进程。

　　(4)执行parse\_early\_param()和parsees\_args()解析系统启动参数。

　　(5)执行trap\_in itQ，先设置了系统中断向量表。0-19号的陷阱门用于CPU异常处理;然后初始化系统调用向量;最后调用cpu\_init()完善对CPU的初始化，用于支持进程调度机制，包括设定标志位寄存器、任务寄存器、初始化程序调试相关寄存器等等。

　　(6)执行rcu\_init()，初始化系统中的Read-Copy Update互斥机制。

　　(7)执行init\_IRQ()函数，初始化用于外设的中断，完成对IDT的最终初始化过程。

　　(8)执行init\_timers()， softirq\_init()和time\_init()函数，分别初始系统的定时器机制，软中断机制以及系统日期和时间。

　　(9)执行mem\_init()函数，初始化物理内存页面的page数据结构[描述符](http://www.chinabyte.com/keyword/%E6%8F%8F%E8%BF%B0%E7%AC%A6/" \t "http://blog.csdn.net/tzy3013218117/article/details/_blank)，完成对物理内存管理机制的创建。

　　(10)执行kmem\_cache\_init()，完成对通用slab缓冲区管理机制的初始化工作。

　　(11)执行fork\_init()，计算出当前系统的物理内存容量能够允许创建的进程(线程)数量。

　　(12)执行proc\_caches\_init() , bufer\_init()， unnamed\_dev\_init() ,vfs\_caches\_init()， signals\_init()等函数对各种管理机制建立起专用的slab缓冲区队列。

　　(13 )执行proc\_root\_init()Wl数，对虚拟文件系统/proc进行初始化。

　　在 start\_kenrel()的结尾，内核通过kenrel\_thread()创建出第一个系统内核线程(即1号进程)，该线程执行的是内核中的init()函数，负责的是下一阶段的启动任务。最后调用cpues\_idle()函数：进入了系统主循环体口默认将一直执行default\_idle()函数中的指令，即CPU的halt指令，直到就绪队列中存在其他进程需要被调度时才会转向执行其他函数。此时，系统中唯一存在就绪状态的进程就是由kerne\_hread()创建的init进程(内核线程)，所以内核并不进入default\_idle()函数，而是转向init()函数继续启动过程。

　　启动第五步--用户层init依据inittab文件来设定运行等级

　　内核被加载后，第一个运行的程序便是/sbin/init，该文件会读取/etc/inittab文件，并依据此文件来进行初始化工作。

　　其实/etc/inittab文件最主要的作用就是设定Linux的运行等级，其设定形式是“：id:5:initdefault:”，这就表明Linux需要运行在等级5上。Linux的运行等级设定如下：

　　0：关机

　　1：单用户模式

　　2：无网络支持的多用户模式

　　3：有网络支持的多用户模式

　　4：保留，未使用

　　5：有网络支持有X-Window支持的多用户模式

　　6：重新引导系统，即重启

　　启动第六步--init进程执行rc.sysinit

　　在设定了运行等级后，Linux系统执行的第一个用户层文件就是/etc/rc.d/rc.sysinit脚本程序，它做的工作非常多，包括设定PATH、设定网络配置(/etc/sysconfig/network)、启动swap分区、设定/proc等等。如果你有兴趣，可以到/etc/rc.d中查看一下rc.sysinit文件。

　　线程init的最终完成状态是能够使得一般的用户程序可以正常地被执行，从而真正完成可供应用程序运行的系统环境。它主要进行的操作有：

　　(1) 执行函数do\_basic\_setup()，它会对外部设备进行全面地初始化。

　　(2) 构建系统的虚拟文件系统目录树，挂接系统中作为根目录的设备(其具体的文 件系统已经在上一步骤中注册)。

　　(3) 打开设备/dev/console，并通过函数sys\_dup()打开的连接复制两次，使得文件号0,1 ,2 全部指向控制台。这三个文件连接就是通常所说的“标准输入”stdin,“标准输出”stdout和“标准出错信息”stderr这三个标准I/O通道。

　　(4) 准备好以上一切之后，系统开始进入用户层的初始化阶段。内核通过系统调用execve()加载执T子相应的用户层初始化程序，依次尝试加载程序"/sbin/initl"," /etc/init"," /bin/init'，和“/bin/sh。只要其中有一个程序加载获得成功，那么系统就将开始用户层的初始化，而不会再回到init()函数段中。至此，init()函数结束，Linux内核的引导 部分也到此结束。

　　启动第七步--启动内核模块

　　具体是依据/etc/modules.conf文件或/etc/modules.d目录下的文件来装载内核[模块](http://www.chinabyte.com/keyword/%E6%A8%A1%E5%9D%97/" \t "http://blog.csdn.net/tzy3013218117/article/details/_blank)。

　　启动第八步--执行不同运行级别的脚本程序

　　根据运行级别的不同，系统会运行rc0.d到rc6.d中的相应的脚本程序，来完成相应的初始化工作和启动相应的服务。

　　启动第九步--执行/etc/rc.d/rc.local

　　你如果打开了此文件，里面有一句话，读过之后，你就会对此命令的作用一目了然：

　　# This script will be executed \*after\* all the other init scripts.

　　# You can put your own initialization stuff in here if you don’t

　　# want to do the full Sys V style init stuff.

　　rc.local就是在一切初始化工作后，Linux留给用户进行个性化的地方。你可以把你想设置和启动的东西放到这里。

　　启动第十步--执行/bin/login程序，进入登录状态

　　此时，系统已经进入到了等待用户输入username和password的时候了，你已经可以用自己的帐号登入系统了。

　　1: 启动电源后，主机第一步先做的就是查询BIOS(全称：basic input/output system 基本输入输出系统)信息。了解整个系统的硬件状态，如CPU，内存，显卡，网卡等。嗯，这一步windows算和它是一家。不分彼此。

　　2: 接下来，就是主机读取MBR(硬盘的第一个扇区)里的boot loader了。这个可是重点哦，据说[troubleshooting](http://www.chinabyte.com/keyword/Troubleshooting/" \t "http://blog.csdn.net/tzy3013218117/article/details/_blank)里就会考这点，给个坏了的loader，叫你修正。windows不支持linux的分区格式。所以，用windows的boot。ini是查不到linux的系统的。一般我装系统都是先装 windows再装linux，然后用grub来做boot loader。两个字：省心!因为linux不像windows那么小气。grub可是支持windows分区格式的哦。

　　3: 接上一步，主机读取boot loader后，会读取里面的信息，知道谁跟谁是待在哪，假如主机想进入linux系统，读取到linux核心是在/boot文件目录中后，将此核心加载到内存中。开始了接下来的分析启动之旅。

　　4: OK，第一个运行程序是谁？就是/sbin/init程序。不信，就用top程序看下，是不是PID为1的就是这个东东，它，可是万物之祖啊，我简称它是女娲娘娘(不喜欢亚当夏娃)。

　　· 5: init首先查找启动等级(run-level)。因为启动等级不同，其运行脚本(也就是服务)会不同。默认的等级有以下几项：

　　0 - halt (系统直接关机)

　　1 - single user mode (单人模式，用于系统维护时使用)

　　2 -[Multi](http://www.chinabyte.com/keyword/MULTI/" \t "http://blog.csdn.net/tzy3013218117/article/details/_blank)-user， without[NFS](http://www.chinabyte.com/keyword/NFS/" \t "http://blog.csdn.net/tzy3013218117/article/details/_blank) (类似3模式，不过少了NFS服务)

　　3 - Full multi-user mode (完整模式，不过，是文本模式)

　　4 - unused (系统保留功能)

　　5 - X11 (与3模式类似，不过，是X终端显示)

　　6 - reboot (重新开机)

　　(不要选择0或4，6 否则，进步了系统的)

　　· 6: OK。系统知道自己的启动等级后，接下来，不是去启动服务，而是，先设置好主机运行环境。读取的文件是/etc/rc。d/rc。sysinit文件。那究竟要设置哪些环境呢？

　　· 设置网络环境/etc/sysconfig/network，如主机名，[网关](http://www.chinabyte.com/keyword/%E7%BD%91%E5%85%B3/" \t "http://blog.csdn.net/tzy3013218117/article/details/_blank)，IP，DNS等。

　　· 挂载/proc。此文件是个特殊文件，大小为0，因为它是在内存当中。里面东东最好别删。

　　· 根据内核在开机时的结果/proc/sys/kernel/modprobe。开始进行周边设备的侦测。

　　· 载入用户自定义的模块/etc/sysconfig/modules/\*。modules

　　· 读取/etc/sysctl。conf文件对内核进行设定。

　　· 设定时间，终端字体，硬盘LVM或RAID功能，以fsck进行磁盘检测。

　　· 将开机状况记录到/var/log/dmesg中。(可以用命令dmesg查看结果)

　　· 7: OK，接下来，就是启动系统服务了，不同的run-level会有不同的服务启动。到/etc/rc。d目录中，不同的level会有不同的目录。如启动 3模式，会有个rc3。d目录，里面就保存着服务。其中，S(start)开头的表明开机启动，K(kill)开头的表明开机不启动。数字表示启动顺序。数字越小，启动越早。

　　注意，他们都是连接到etc/rc。d/init。d/目录中的相关文件。所以，想手工启动某一服务，可以用"/etc/rc。d/init。 d/某个服务 start"启动哦。相反，我们也可以把某个服务ln(链接命令)到不同run-level的目录中。记得打上S或者K+数字哦。

　　· 8: 读取服务后，主机会读取/etc/rc。d/rc。local文件。所以，如果需要什么开机启动的话，可以写个脚本或命令到这里面来。就不用像上面那么麻烦。以后删除也方便。

　　OK，经过一番长途跋涉后，系统终于可以安心的开启shell了。