# 计算机的启动过程

Boot pull oneself up by one’s bootstraps

计算机的整个启动过程分为四个阶段

**一、第一阶段：BIOS**

上个世纪70年代初，”只读内存”（read-only memory，缩写为ROM）发明，开机程序被刷入ROM芯片，计算机通电后，第一件事就是读取它。

这块芯片里的程序叫做”基本輸出輸入系統”（Basic Input/Output System），简称为BIOS。

**1.1 硬件自检**

BIOS中主要存放的程序包括：自诊断程序（通过读取CMOS RAM中的内容识别硬件配置，并对其进行自检和初始化）、CMOS设置程序（引导过程中，通过特殊热键启动，进行设置后，存入CMOS RAM中）、系统自动装载程序（在系统自检成功后，将磁盘相对0道0扇区上的引导程序装入内存使其运行）和主要I/O驱动程序和中断服务（BIOS和硬件直接打交道，需要加载I/O驱动程序）。

BIOS程序首先检查，计算机硬件能否满足运行的基本条件，这叫做”硬件自检”（Power-On Self-Test），缩写为POST。

如果硬件出现问题，主板会发出不同含义的蜂鸣，启动中止。如果没有问题，屏幕就会显示出CPU、内存、硬盘等信息。

**1.2 启动顺序**

硬件自检完成后，BIOS把控制权转交给下一阶段的启动程序。

这时，BIOS需要知道，”下一阶段的启动程序”具体存放在哪一个设备。也就是说，BIOS需要有一个外部储存设备的排序，排在前面的设备就是优先转交控制权的设备。这种排序叫做”启动顺序”（Boot Sequence）。

打开BIOS的操作界面，里面有一项就是”设定启动顺序”。

**二、第二阶段：主引导记录**

BIOS按照”启动顺序”，把控制权转交给排在第一位的储存设备。即根据用户指定的引导顺序从软盘、硬盘或是可移动设备中读取启动设备的MBR，并放入指定的位置（0x7c000）内存中。

这时，计算机读取该设备的第一个扇区，也就是读取最前面的512个字节。如果这512个字节的最后两个字节是0x55和0xAA，表明这个设备可以用于启动；如果不是，表明设备不能用于启动，控制权于是被转交给”启动顺序”中的下一个设备。

这最前面的512个字节，就叫做”主引导记录”（Master boot record，缩写为MBR）。

**2.1 主引导记录的结构**

“主引导记录”只有512个字节，放不了太多东西。它的主要作用是，告诉计算机到硬盘的哪一个位置去找操作系统。

主引导记录由三个部分组成：

1. （1） 第1-446字节：调用操作系统的机器码。
2. （2） 第447-510字节：分区表（Partition table）。
3. （3） 第511-512字节：主引导记录签名（0x55和0xAA）。

其中，第二部分”分区表”的作用，是将硬盘分成若干个区。

**2.2 分区表**

硬盘分区有很多好处。考虑到每个区可以安装不同的操作系统，”主引导记录”因此必须知道将控制权转交给哪个区。

分区表的长度只有64个字节，里面又分成四项，每项16个字节。所以，一个硬盘最多只能分四个一级分区，又叫做”主分区”。

每个主分区的16个字节，由6个部分组成：

1. （1） 第1个字节：如果为0x80，就表示该主分区是激活分区，控制权要转交给这个分区。四个主分区里面只能有一个是激活的。
2. （2） 第2-4个字节：主分区第一个扇区的物理位置（柱面、磁头、扇区号等等）。
3. （3） 第5个字节：主分区类型。
4. （4） 第6-8个字节：主分区最后一个扇区的物理位置。
5. （5） 第9-12字节：该主分区第一个扇区的逻辑地址。
6. （6） 第13-16字节：主分区的扇区总数。

最后的四个字节（”主分区的扇区总数”），决定了这个主分区的长度。也就是说，一个主分区的扇区总数最多不超过2的32次方。

如果每个扇区为512个字节，就意味着单个分区最大不超过2TB。再考虑到扇区的逻辑地址也是32位，所以单个硬盘可利用的空间最大也不超过2TB。如果想使用更大的硬盘，只有2个方法：一是提高每个扇区的字节数，二是增加扇区总数。

**三、第三阶段：硬盘启动**

这时，计算机的控制权就要转交给硬盘的某个分区了，这里又分成三种情况。

**3.1 情况A：卷引导记录**

上一节提到，四个主分区里面，只有一个是激活的。计算机会读取激活分区的第一个扇区，叫做”卷引导记录”（Volume boot record，缩写为VBR）。

“卷引导记录”的主要作用是，告诉计算机，操作系统在这个分区里的位置。然后，计算机就会加载操作系统了。

**3.2 情况B：扩展分区和逻辑分区**

随着硬盘越来越大，四个主分区已经不够了，需要更多的分区。但是，分区表只有四项，因此规定有且仅有一个区可以被定义成”扩展分区”（Extended partition）。

所谓”扩展分区”，就是指这个区里面又分成多个区。这种分区里面的分区，就叫做”逻辑分区”（logical partition）。

计算机先读取扩展分区的第一个扇区，叫做”扩展引导记录”（Extended boot record，缩写为EBR）。它里面也包含一张64字节的分区表，但是最多只有两项（也就是两个逻辑分区）。

计算机接着读取第二个逻辑分区的第一个扇区，再从里面的分区表中找到第三个逻辑分区的位置，以此类推，直到某个逻辑分区的分区表只包含它自身为止（即只有一个分区项）。因此，扩展分区可以包含无数个逻辑分区。

但是，似乎很少通过这种方式启动操作系统。如果操作系统确实安装在扩展分区，一般采用下一种方式启动。

**3.3 情况C：启动管理器**

在这种情况下，计算机读取”主引导记录”前面446字节的机器码之后，不再把控制权转交给某一个分区，而是运行事先安装的”启动管理器”（boot loader），由用户选择启动哪一个操作系统。

Linux环境中，目前最流行的启动管理器是Grub。

对于grub而言，在MBR中的446字节的引导程序属于GRUB的开始执行程序，通过这段程序，进一步执行stage1.5或是stage2的执行程序，将在下面详细介绍执行过程。

其中stage1.5或是stage2便属于阶段2引导的过程了，stage2过程也是作为GRUB kernel的核心代码出现。Stage1.5过程（对于GRUB而言存在stage1.5，GRUB2则不存在）的功能很单一，主要就是为了引导stage2过程服务。由于stage2过程的代码存放在文件系统下的boot分区目录中，因此stage1.5过程就是需要提供一个文件系统的环境，而该文件系统环境需要保证系统可以找到stage2过程的文件，那么stage1.5阶段提供的文件系统需要是boot文件系统所对应的，这个在执行grub install过程中就已经确定了。stage2过程中，主要会把系统切换到保护模式，设置好C运行时环境，找到config文件（事实上就是menulist文件），如果没有找到就执行一个shell，等待用户的执行。然后的工作就变成了输入命令->解析命令->执行命令的循环中。当然该阶段引导的最终状态就是执行boot命令，将内核和initrd镜像加载进入内存中，进而将控制权转交给内核。

**四、第四阶段：操作系统**

控制权转交给操作系统后，操作系统的内核首先被载入内存。

以Linux系统为例，先载入/boot目录下面的kernel。内核加载成功后，第一个运行的程序是/sbin/init。它根据配置文件（Debian系统是/etc/initab）产生init进程。这是Linux启动后的第一个进程，pid进程编号为1，其他进程都是它的后代。

然后，init线程加载系统的各个模块，比如窗口程序和网络程序，直至执行/bin/login程序，跳出登录界面，等待用户输入用户名和密码。

至此，全部启动过程完成。

另外在大磊的[博客](http://www.dalei.org/linux-unix-boot-process/)中还有许多细节部分：

**BIOS启动细节：**

a) 按下电源开关，电源就开始向主板和其它设备供电；当芯片组检测到电源已经开始稳定供电了(当然从不稳定到稳定的过程只是一瞬间的事情)，它便撤去RESET信号(如果是手工按下计算机面板上的Reset按钮来重启机器，那么松开该按钮时芯片组就会撤去RESET信号)；CPU马上就从地址FFFF:0000H 处开始执行指令，放在这里的只是一条跳转指令，跳到系统BIOS中真正的启动代码处。

b) 系统BIOS的启动代码首先进行POST(Power－On Self Test，加电后自检)。POST的主要检测系统中一些关键设备是否存在和能否正常工作，例如内存和显卡等设备；由于POST是最早进行的检测过程，此时显卡还没有初始化，如果系统BIOS在进行POST的过程中发现了一些致命错误，例如没有找到内存或者内存有问题(此时只会检查640K常规内存)，那么系统BIOS就会直接控制喇叭发声来报告错误，声音的长短和次数代表了错误的类型；在正常情况下，POST过程进行得非常快，几乎无法感觉到它的存在。POST结束之后就会调用其它代码来进行更完整的硬件检测。

c) 接下来系统BIOS将查找显卡的BIOS。前面说过，存放显卡BIOS的ROM芯片的起始地址通常设在C0000H处，系统BIOS在这个地方找到显卡BIOS之后就调用它的初始化代码，由显卡BIOS来初始化显卡。此时多数显卡都会在屏幕上显示出一些初始化信息，介绍生产厂商、图形芯片类型等内容，不过这个画面几乎是一闪而过。系统BIOS接着会查找其它设备的BIOS程序，找到之后同样要调用这些BIOS内部的初始化代码来初始化相关的设备。

d) 查找完所有其它设备的BIOS之后，系统BIOS将显示出它自己的启动画面，其中包括有系统BIOS的类型、序列号和版本号等内容。

e) 接着系统BIOS将检测和显示CPU的类型和工作频率，测试所有的RAM，并同时在屏幕上显示内存测试的进度。可以在CMOS设置中自行决定使用简单耗时少或者详细耗时多的测试方式。

f) 内存测试通过之后，系统BIOS将开始检测系统中安装的一些标准硬件设备，包括硬盘、CD-ROM、串口、并口和软驱等设备，另外绝大多数较新版本的系统BIOS在这一过程中还要自动检测和设置内存的定时参数、硬盘参数和访问模式等。

g) 标准设备检测完毕后，系统BIOS内部支持即插即用的代码将开始检测和配置系统中安装的即插即用设备。每找到一个设备之后，系统BIOS都会在屏幕上显示出设备的名称和型号等信息，同时为该设备分配中断、DMA通道和I/O端口等资源。

h) 到这一步为止，所有硬件都已经检测配置完毕了，多数系统BIOS会重新清屏并在屏幕上方显示出一个表格，其中概略地列出了系统中安装的各种标准硬件设备，以及它们使用的资源和一些相关工作参数。

i) 接下来系统BIOS将更新ESCD(Extended System Configuration Data，扩展系统配置数据)。ESCD是系统BIOS用来与操作系统交换硬件配置信息的一种手段，这些数据被存放在CMOS(一小块特殊的RAM，由主板上的电池来供电)之中。通常ESCD数据只在系统硬件配置发生改变后才会更新，所以不是每次启动机器时都能够看到“Update ESCD… Success”这样的信息。不过，某些主板的系统BIOS在保存ESCD数据时使用了与Windows 9x不相同的数据格式，于是Windows 9x在它自己的启动过程中会把ESCD数据修改成自己的格式。但在下一次启动机器时，即使硬件配置没有发生改变，系统BIOS也会把ESCD的数据格式改回来。如此循环，将会导致在每次启动机器时，系统BIOS都要更新一遍ESCD，这就是为什么有些机器在每次启动时都会显示出相关信息的原因。

j) ESCD更新完毕后，系统BIOS的启动代码将进行它的最后一项工作：即根据用户指定的启动顺序从软盘、硬盘或光驱启动MBR。在这个过程中会按照启动顺序顺序比较其放置MBR的位置的结尾两位是否为0xAA55，通过这种方式判断从哪个引导设备进行引导。在确定之后，将该引导设备的MBR内容读入到0x7C00[1]的位置，并再次判断其最后两位，当检测正确之后，进行阶段1的引导。