

# Linux 系統開機加電後發生了什麼？

點擊關注  馬哥Linux運維 2021-12-15 22:06



关注蓝字  收获每日技术干货

匠 心 精 神 良 心 教 育

## linux系統的啟動流程

關於linux系統的啟動流程我們可以按步進行劃分為如下：

### BIOS

- POST自檢
- BIOS(Boot Sequence)

### 引導操作系統

- 加載對應引導上的MBR(bootloader)
- 主引導設置加載其BootLoader

### 加載操作系統

## 啟動BIOS，準備實模式下的中斷向量表和中斷服務程序

電腦啟動後，CPU邏輯電路被設計為只能運行內存中的程序，沒有能力直接運行存在於軟盤或硬盤中的操作系統，如果想要運行，必須要加載到內存（RAM）中。

BIOS是如何啟動的，CPU硬件邏輯設計為在加電瞬間強行將CS值置為0XF000，IP為0XFFF0，這樣CS:IP就指向0XFFFF0這個位置，這個位置正是BIOS程序的入口地址。

BIOS程序被固化在計算機主機板上的一塊很小的ROM芯片裡。現在CS:IP已經指向了0XFFFF0這個位置，意味著BIOS開始啟動。

## POST自檢

BIOS的第一步動作就是進行上電自檢（POST）

POST的工作是檢查硬件設備。隨著BIOS程序的執行，屏幕上會顯示顯卡的信息，內存的信息等，

## 初始化設備

BIOS的第二步動作就是枚舉本地設備並初始化

有一項對啟動操作系統至關重要的工作，那就是BIOS在內存中建立中斷向量表和中斷服務程序

BIOS程序在內存最開始的位置（0x00000）用1KB的內存空間（0x00000~0x003FF）構建中斷向量表，在緊挨著它的位置用256KB的內存空間構建BIOS數據區（0x00400~0x004FF），並在大約57KB以後的位置（0x0e05b）加載了8KB左右的與中斷向量表相應的若干中斷服務程序。

中斷向量表有256個中斷向量，每個中斷向量佔4個字節，其中兩個字節是CS值，兩個字節是IP值。每個中斷向量都指向一個具體的中斷服務程序。

## BIOS-runtime服務按照boot啟動順序搜索設備，尋找BBR

由於BIOS功能使用上的不同，它由兩個部分組成：POST和runtime服務。POST完成後，它將從存儲器中被清除，但是BIOS runtime服務會被保留，用於目標操作系統。

為了啟動操作系統，BIOS的runtime服務將搜索那些激活狀態的或是可引導啟動的設備，搜索的順序則由CMOS設置決定（也就是我們平時所謂的在BIOS中設置的啟動順序）。一個軟驅，一台光驅，一個硬盤上的分區，網絡上的設備甚至一個usb 閃存盤都可以作為一個啟動設備。

當然，linux通常是從硬盤啟動的。硬盤上的MBR（主啟動記錄）包含有基本的boot loader，它是一個512字節大小的扇區，位於磁盤的第一個扇區（0磁頭0磁道1扇區）。當MBR被裝載到RAM中後，BIOS就會將控制權轉交給MBR。

## 引導操作系統內核並為保護模式做準備

位於MBR中的主boot loader 是一個512字節的鏡像，其中不僅包含了bootload 程序代碼，還包含了一個小的分區表。

最初的446字節是主boot loader，它裡面就包含有可執行代碼以及錯誤消息文本。接下來的64字節是分區表，其中包含有四個分區的各自的記錄（一個分區佔16字節）。MBR 通過特殊數字0xAA55（譯者註：在電子界中AA55 確實是具有傳奇色彩的數字，想知道為什麼麼？將它展開成二進制形式，看看有什麼規律）作為兩個字節的結束標誌。0x55AA 同時也是MBR 有效的校驗確認。

首先對CPU發送int 0x19 中斷，使CPU運行int 0x19 中斷對應的中斷服務程序，這個中斷服務程序的作用就是把軟盤第一個扇區的程序加載到內存的指定位置。

主boot loader 的工作是尋找並加載次boot loader（內核加載程序）

它通過分析分區表，找出激活分區來完成這個任務，當它找到一個激活分區時，它將繼續掃描剩下的分區表中的分區，以便確認他們都是未激活的。

確認完畢後，激活分區的啟動記錄（次boot loader）從設備中被讀到RAM，並被執行。

其中加載過程需要藉助BIOS 提供的int 0x13中斷向量指向的中斷服務程序來完成。該程序將軟盤第二個扇區開始的4個扇區，即setup.s 對應的程序加載至內存的SETUPSEG（0x90200）處。

把第一階段和第二階段的boot loaders 聯合起來，就是在x86個人電腦中，我們所說的linux loader（LILO）或者GRand Unified Bootloader(GRUB)。由於GRUB 修正了一些LILO 中存在的缺陷，因此下面就讓我們來看看GRUB（如果你希望得到更多的關於GRUB，LILO 和與之相關話題的討論資源，請見文後的參考資料）

對於GRUB 來說，一個比較好的方面就是它包含了linux 文件系統的知識。與LILO使用裸扇區不同的是，GRUB 能夠從ext2 或者ext3 文件系統中加載linux 內核。它是通過將本來兩階段的boot loader 轉換成三個階段的boot loader。在第一階段（MBR）中會啟動stage1.5的boot loader 來理解linux 內核鏡像中的特殊的文件系統格式，例如，reiserfs\_stage1-5(用於從reiserfs日誌文件系統中進行加載)或e2fs + stage1\_5 (用於從wxt2或ext3文件系統進行加載)。當stage1.5 的boot loader 被加載並運行時，stage2 的boot loader 才能被加載。當stage2 被加載時，GRUB能根據請求的情況顯示一個可選內核的清單

（在/etc/grub.conf 中進行定義，同時還有幾個軟符號鏈接/etc/grub/menu.lst 和/etc/grub.conf）。你可以選擇一個內核，修改其附加的內核參數。同時，你可以選擇使用命令行的shell來對啟動過程進行更深層次的手工控制。

在次boot loader 存在與內存中後，就可以對文件系統進行查詢了，同時將默認的內核鏡像以及初始化內存盤鏡像也被加載到內存中。

一切準備完畢之後，次boot loader 就會調用內核鏡像，完成操作系統的加載。

## 加載內核並從實模式轉換為保護模式

當內核映像被加載到內存中（加載過程仍然用int 0x13中斷向量），並且次引導加載程序釋放控制權之後，內核階段就開始了。

## 加載內核鏡像

內核映像並不是一個可執行的內核，而是一個壓縮過的內核映像。通常它是一個zImage（壓縮映像，小於512KB）或一個bzImage（較大的壓縮映像，大於512KB），它是提前使用zlib 進行壓縮過的。在這個內核映像前面是一個例程，它實現少量硬件設置，並對內核映像中包含的內核進行解壓，然後將其放入高端內存中，如果有初始RAM 磁盤映像，就會將它移動到內存中，並標明以後使用。然後該例程會調用內核，並開始啟動內核引導的過程。

當bzImage（用於i386 映像）被調用時，我們從./arch/i386/boot/head.S 的start 彙編例程開始執行。

這個例程會執行一些基本的硬件設置，並調用./arch/i386/boot/compressed/head.S 中的startup\_32，設置一個基本的環境（堆棧等），並清除Block Started by Symbol（BSS）。然後調用一個叫做decompress\_kernel 的C 函數（在./arch/i386/boot/compressed/misc.c 中）來解壓內核。當內核被解壓到內存中之後，就可以調用它了。這是另外一個startup\_32 函數，但是這個函數在./arch/i386/kernel/head.S 中。

## 進入保護模式並初始化

- 進入保護模式
- 設置中斷描述附表和全局描述符表
- 創建了內存分頁機制

## 啟動內核

- start\_kernel啟動內核
- 創建init進程

## BIOS階段–準備實模式下的中斷向量和中斷服務程序

## BIOS是什麼

上個世紀70年代初，“只讀內存”（read-only memory，縮寫為ROM）發明，開機程序被刷入ROM芯片，計算機通電後，第一件事就是讀取它。計算機，啟動這塊芯片裡的程序叫做“基本輸出輸入系統”（Basic Input/Output System），簡稱為BIOS。

它是一組固化到計算機內主板上一個ROM芯片上的程序，它保存著計算機最重要的基本輸入輸出的程序、開機後自檢程序和系統自啟動程序，它可從CMOS中讀寫系統設置的具體信息。其主要功能是為計算機提供最底層的、最直接的硬件設置和控制。

## BIOS存儲的信息

BIOS芯片中主要存放：

- 自診斷程序：通過讀取CMOSRAM中的內容識別硬件配置，並對其進行自檢和初始化；
- CMOS設置程序：引導過程中，用特殊熱鍵啟動，進行設置後，存入CMOS RAM中；
- 系統自舉裝載程序：在自檢成功後將磁盤相對0道0扇區上的引導程序裝入內存，讓其運行以裝入DOS系統；
- 主要I/O設備的驅動程序和中斷服務：由於BIOS直接和系統硬件資源打交道，因此總是針對某一類型的硬件系統，而各種硬件系統又各有不同，所以存在各種不同種類的BIOS，隨著硬件技術的發展，同一種BIOS也先後出現了不同的版本，新版本的BIOS比起老版本來說，功能更強。

BIOS：計算機加電自檢完成後第一個讀取的地方就是就是BIOS（Basic Input Output System，基礎輸入輸出系統），BIOS裡面記錄了主機板的芯片集與相關設置，如CPU與接口設備的通信頻率、啟動設備的搜索順序、硬盤的信息、系統時間、內存信息、時鐘信息、PnP特性、外部總線、各種接口設備的I/O地址、已經與CPU通信的IRQ中斷信息，所以，啟動如果要順利啟動，首先要讀取BIOS設置。

計算機會首先加載BIOS信息，BIOS信息是如此的重要，以至於計算機必須在最開始就找到它。

電腦啟動後，CPU邏輯電路被設計為只能運行內存中的程序，沒有能力直接運行存在於軟盤或硬盤中的操作系統，如果想要運行，必須要加載到內存（RAM）中。

## BIOS是如何啟動的

CPU硬件邏輯設計為在加電瞬間強行將CS值置為0XF000，IP為0XFFF0，這樣CS:IP就指向0XFFFF0這個位置，這個位置正是BIOS程序的入口地址。

## BIOS需要在內存中加載中斷向量表和中斷服務程序

BIOS程序被固化在計算機主機板上的一塊很小的ROM芯片裡。現在CS:IP已經指向了0xFFFF0這個位置，意味著BIOS開始啟動。隨著BIOS程序的執行，屏幕上會顯示顯卡的信息，內存的信息，說明BIOS程序在檢測顯卡，內存，這個就是POST開機自檢期間，有一項對啟動操作系統至關重要的工作，那就是BIOS在內存中建立中斷向量表和中斷服務程序

BIOS程序在內存最開始的位置（0x00000）用1KB的內存空間（0x00000~0x003FF）構建中斷向量表，在緊挨著它的位置用256KB的內存空間構建BIOS數據區（0x00400~0x004FF），並在大約57KB以後得位置（0x0e05b）加載了8KB左右的與中斷向量表相應的若干中斷服務程序。

中斷向量表有256個中斷向量，每個中斷向量佔4個字節，其中兩個字節是CS值，兩個字節是IP值。每個中斷向量都指向一個具體的中斷服務程序。

## BIOS階段的工作

### POST開機自檢

BIOS程序首先檢查，計算機硬件能否滿足運行的基本條件，這叫做“硬件自檢”（Power-On Self-Test），縮寫為POST。

如果硬件出現問題，主板會發出不同含義的蜂鳴，啟動中止。如果沒有問題，屏幕就會顯示出CPU、內存、硬盤等信息。

電腦主機打開電源的時候，隨後會聽到滴的一聲，系統啟動開始了開機自檢（POST-power on self test）自檢開始）

這個過程中主要是檢測計算機硬件設備比如：CPU，內存，主板，顯卡，CMOS等設備是否有故障存在

如果有硬件故障的話將按兩種情況理：

- 對於嚴重故障(致命性故障)則停機，此時由於各種初始化操作還沒完成，不能給出任何提示或信號；
- 對於非嚴重故障則給出提示或聲音報警信號，等待用戶處理），如果沒有故障，POST完整自己的接力任務，將尾部工作交接給BIOS處理



```

Diskette Drive B : None
Pri. Master Disk : LBA,ATA 100, 250GB
Pri. Slave Disk : LBA,ATA 100, 250GB
Sec. Master Disk : None
Sec. Slave Disk : None

Serial Port(s) : 3F0 2F0
Parallel Port(s) : 370
DDR at Bank(s) : 0 1 2

Pri. Master Disk HDD S.M.A.R.T. capability ... Disabled
Pri. Slave Disk HDD S.M.A.R.T. capability ... Disabled

PCI Devices Listing ...
Bus Dev Fun Vendor Device SVID SSID Class Device Class IRQ
-----
0 27 0 8086 2668 1458 A005 0403 Multimedia Device 5
0 29 0 8086 2658 1458 2658 0C03 USB 1.1 Host Cntrlr 9
0 29 1 8086 2659 1458 2659 0C03 USB 1.1 Host Cntrlr 11
0 29 2 8086 265A 1458 265A 0C03 USB 1.1 Host Cntrlr 11
0 29 3 8086 265B 1458 265A 0C03 USB 1.1 Host Cntrlr 5
0 29 7 8086 265C 1458 5006 0C03 USB 1.1 Host Cntrlr 9
0 31 2 8086 2651 1458 2651 0101 IDE Cntrlr 14
0 31 3 8086 266A 1458 266A 0C05 SMBus Cntrlr 11
1 0 0 10DE 0421 10DE 0479 0300 Display Cntrlr 5
2 0 0 1283 8212 0000 0000 0180 Mass Storage Cntrlr 10
2 5 0 11AB 4320 1458 E000 0200 Network Cntrlr 12
ACPI Controller 9
```

加載BIOS

BIOS把控制權轉交給下一階段的啟動程序。

這時，BIOS需要知道，“下一階段的啟動程序”具體存放在哪一個設備。也就是說，BIOS需要有一個外部儲存設備的排序，排在前面的設備就是優先轉交控制權的設備。這種排序叫做“啟動順序”(Boot Sequence)。打開BIOS的操作界面，裡面有一項就是“設定啟動順序”。



在此之後，計算機心裡就有譜了，知道應該去讀取哪個硬件設備了。

## 引導操作系統

硬件自檢完成後，我們期望能否啟動操作系統，但是問題出來了

- 操作系統存放在哪？
- BIOS如何找到操作系統？
- BIOS如何加載操作系統？

## 背景知識

### 多操作系統時的啟動順序

為了尋找操作系統，BIOS按照“啟動順序”，把控制權轉交給排在第一位的儲存設備。

這時，計算機讀取該設備的第一個扇區，也就是讀取最前面的512個字節。

如果這512個字節的最後兩個字節是0x55和0xAA，表明這個設備可以用於啟動；

如果不是，表明設備不能用於啟動，控制權於是被轉交給“啟動順序”中的下一個設備。

這最前面的512個字節，就叫做**主引導記錄（Master boot record，縮寫為MBR）**

## 主引導記錄MBR

位於MBR中的主boot loader是一個512字節的鏡像，其中不僅包含了程序代碼，還包含了一個小的分區表。

最初的446字節是主boot loader，它裡面就包含有可執行代碼以及錯誤消息文本。接下來的64字節是分區表，其中包含有四個分區的各自的記錄（一個分區佔16字節）。MBR通過特殊數字0xAA55（譯者註：在電子界中AA55確實是具有傳奇色彩的數字，想知道為什麼麼？將它展開成二進制形式，看看有什麼規律）作為兩個字節的結束標誌。0x55AA同時也是MBR有效的校驗確認。

主boot loader的工作是尋找並加載次boot loader。它通過分析分區表，找出激活分區來完成這個任務，當它找到一個激活分區時，它將繼續掃描剩下的分區表中的分區，以便確認他們都是未激活的。確認完畢後，激活分區的啟動記錄從設備中被讀到RAM，並被执行。



“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：

- 第1-446字节：调用操作系统的机器码。
- 第447-510字节：分区表（Partition table）。
- 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。

## 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：

- 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。
- 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）。
- 第5个字节：主分区类型。
- 第6-8个字节：主分区最后一个扇区的物理位置。
- 第9-12字节：该主分区第一个扇区的逻辑地址。
- 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。

MBR：第一个可开机设备的第一个扇区内的主引导分区块，内包含引导加载程序

引导加载程序 (Boot loader)：一支可读取内核文件来执行的软件

内核文件：开始操作系统的功能

## 引导操作系统的过程

由硬盘启动时，BIOS通常是转向第一块硬盘的第一个扇区，即主引导记录(MBR)。装载GRUB和操作系统的过程，包括以下几个操作步骤：

### 装载记录

基本引导装载程序所做的唯一的事情就是装载第二引导装载程序。

### 装载Grub

这第二引导装载程序实际上是引出更高级的功能，以允许用户装载一个特定的操作系统。

### 装载系统

如linux内核。GRUB把机器的控制权移交给操作系统。

不同的是，微软操作系统都是使用一种称为链式装载的引导方法来启动的，主引导记录仅仅是简单地指向操作系统所在分区的第一个扇区。

## 加载主引导加载程序-基本装载程序

众所周知，硬盘上第0磁道第一个扇区被称为MBR，也就是Master Boot Record，即主引导记录，它的大小是512字节，别看地方不大，可里面却存放了预启动信息、分区表信息。

按照BIOS所设定的系统启动流程，如果检测通过，则根据引导次序(Boot Sequence)开始在第一台设备上支持启动程序，我们的启动设备主要包括硬盘、USB、SD等，我们一般用的是硬盘，然后进行读取第一个设备就是硬盘，第一个要读去的就是该硬盘的主引导记录MBR (Master Boot Record)，然后系统可以根据启动区安装的引导加载程序 (Boot Loader) 开始执行核心识别的工作。

MBR程序只是找到只是硬盘分区内最前面的446个字节的Boot Loader，然后查找相关配置和定义。

然后将控制权交给主引导代码。主引导代码的任务包括

- 扫描分区表，找到一个激活(可引导)分区；
- 找到激活分区的起始扇区；
- 将激活分区的引导扇区装载到内存7C00处；
- 将控制权交给引导扇区代码；

## 加载次引导记载程序-高级装载程序bootload如GRUB

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。

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。

### 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。

### 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。

### 情况C：启动管理器

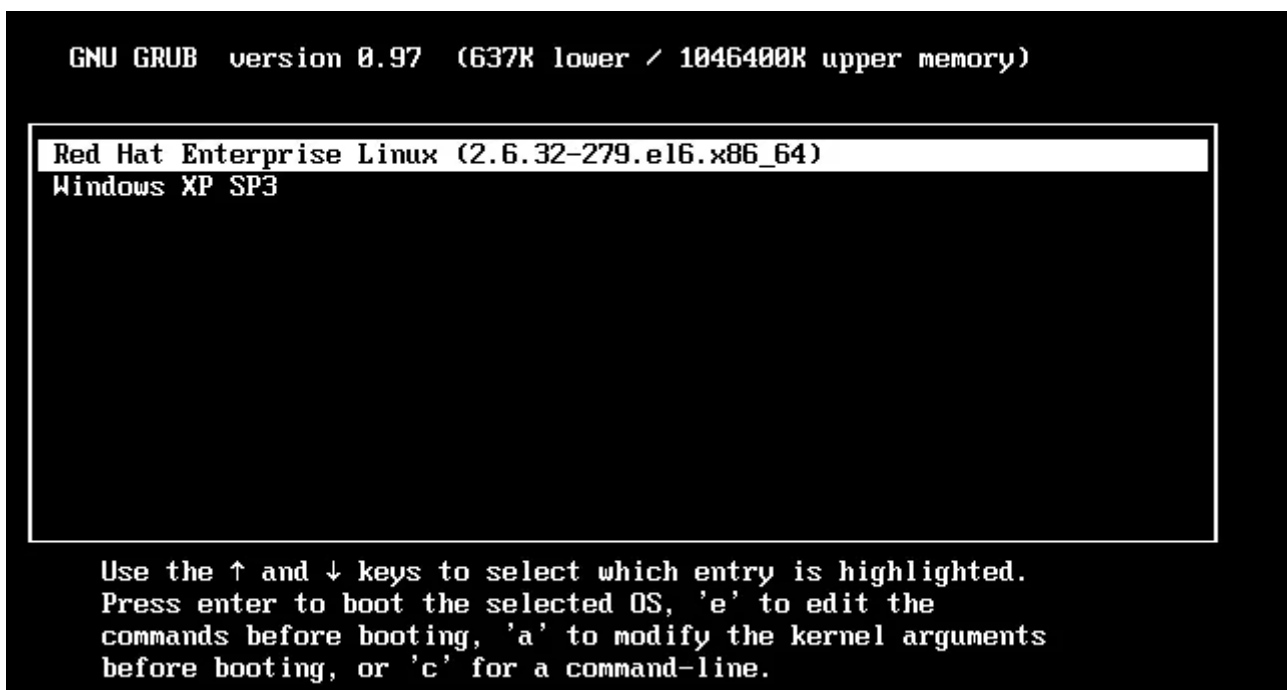
在这种情况下，计算机读取“主引导记录”前面446字节的机器码之后，不再把控制权转交给某一个分区，而是运行事先安装的“启动管理器”（boot loader），由用户选择启动哪一个操作系统。

Linux环境中，目前最流行的启动管理器是Grub。

Boot Loader 就是在操作系统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。

Boot Loader有若干种，其中Grub、Lilo和spfdisk是常见的Loader。

我们以Grub为例来讲解吧，毕竟用lilo和spfdisk的人并不多。



## 为什么这么复杂

早期的操作系统并没有那么复杂，当然bootload也没有那么多功能，但是如今我们的操作系统越来越复杂，bootload也越来越庞大，而且如今在一台电脑上安装多系统变得那么平常，因此之前简单的bootload已经无法满足这些功能。

BIOS和MBR都是硬件本身会支持的功能，至于Boot Loader则是操作系统安装在MBR上面的一套软件。由于MBR仅有446bytes而已，因此这个引导加载程序是非常小而完美的。这个BootLoader的主要任务如下

- 提供菜单：用户可以选择不同的开机选项，这也是多重引导的重要功能

- 载入内核文件：直接指向可开机的程序段来开始操作系统。
- 转交其他Loader：将引导加载功能转交给其他loader负责

上面的前两点还容易理解，但是第三点很有趣！那表示你的计算机系统里面可以具有两个以上的引导加载程序呢。有可能吗？我们的硬盘不是只有一个MBR而已？但是引导加载程序除了可以安装在MBR之外，还可以安装在每个分区的引导扇区。

举一个例子来说，假设你的个人计算机只有一块硬盘，里面分成4个分区。其中第一，二分区分别安装了Windows及Linux，你要如何在开机的时候选择用Windows还是Linux开机呢？假设MBR内安装的是可以同时识别Windows和Linux操作系统的引导加载程序，那么整个流程如下

做个总结就是这样：

- 每个分区都有自己的启动扇区
- 系统分区为第一及第二分区
- 实际可开机的内核文件是放置到各分区内的
- loader只会认识自己的系统分区内的可开机内核文件，以及其他的Loader而已
- loader可直接指向或者是间接将管理权交给另一个管理程序

现在想一下，为什么人家常说：“如果要安装多重引导，最好先安装Windows再安装Linux呢”？

这是因为Linux在安装时，你可以选择将引导加载程序安装在MBR或个别分区的启动扇区，而且Linux的Loader可以手动设置菜单，所以你可以在Linux的Boot Loader里面加入Windows开机选项

Windows在安装的时候，他的安装程序会主动覆盖掉MBR以及自己所在分区的启动扇区，你没有选择的机会，而且他没有让我们自己选择菜单功能

## 加载操作系统内核

用户选择要加载的内核之后，次引导加载程序（GRUB）就会根据/boot/grub.conf配置文件中所设置的信息，从/boot/所在的分区上读取Linux内核映像，然后把内核映像加载到内存中并把控制权交给Linux内核。

linux内核获得控制权之后开始干自己的事

- 检测硬件

- 解压缩自己并安装必要驱动
- 初始化与文件系统相关的虚拟设备，LVM或RAID
- 装载根文件系统，挂在根目录下面
- 完成之后，linux在进程空间里面加载init程序，下面轮到init干活

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。

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

关于/etc/inittab文件的学问，其实还有很多

## 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，里面的脚本够你看几天的

## 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。

## 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。

## 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。你可以把你想要设置和启动的东西放到这里。

## 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。

文章转载：Linux学习

(版权归原作者所有，侵权)



底层原理+项目实战双轮驱动，要学就学专业的

# Linux云计算SRE工程师

双轨教学/20+实战项目/引入多云案例/优化30+技能模块


系统稳定性建设 | 微服务 | Kubernetes | 阿里云  
ELK日志系统 | 持续集成CI/CD | 指标采集

更多课程内容  
请扫码咨询 >>>



 **大家都在看**  
想看一场奥运会比赛太不容易了! 推荐

用腾讯视频观看

 点击下方“阅读原文”查看更多

阅读原文

喜欢此内容的人还喜欢

单片机C语言学习架构  
C语言与C++编程



收藏：梳理Linux内核概念和学习路线  
架构师技术联盟



带你真正认识Linux 系统结构  
技术让梦想更伟大

