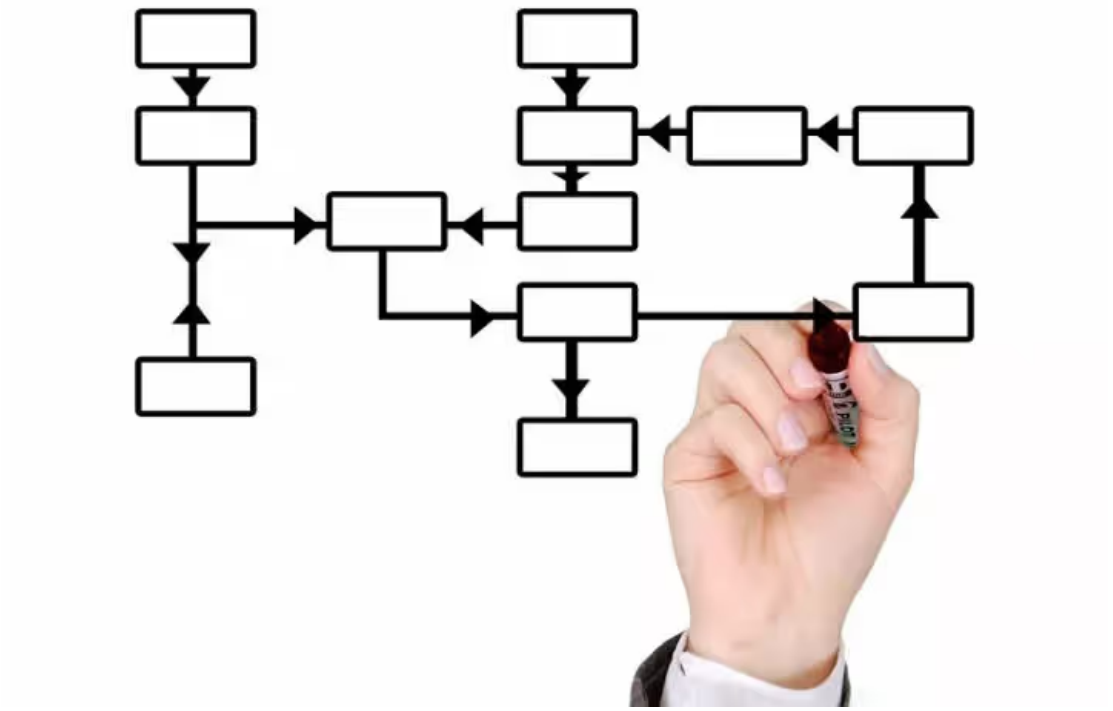


# 用了五年Linux，三分钟带你揭开Linux过程内幕

夏威廉
你所看到的不一定是真的，如果你不去努力那失败就是真的

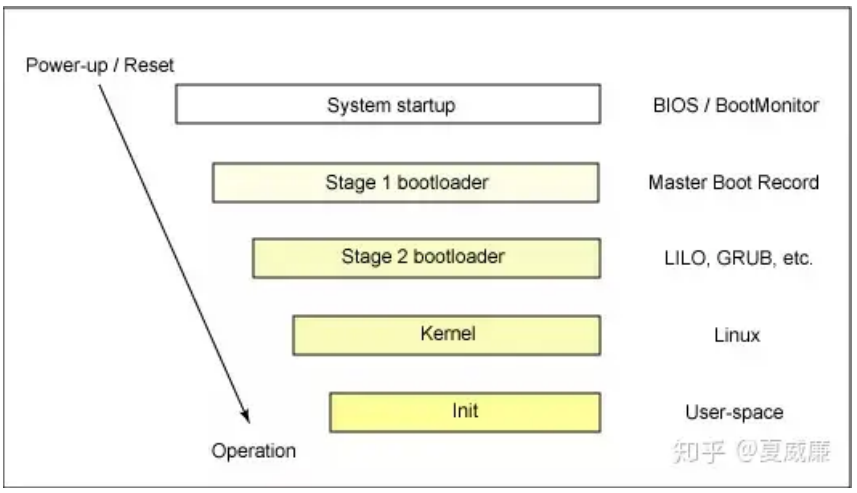


早期时，启动一台计算机意味着要给计算机喂一条包含引导程序的纸带，或者手工使用前端面板地址/数据/控制开关来加载引导程序。尽管目前的计算机已经装备了很多工具来简化引导过程，但是这一切并没有对整个过程进行必要的简化。

让我们先从高级的视角来查看Linux引导过程，这样就可以看到整个过程的全貌了。然后将回顾一下在各个步骤到底发生了什么。在整个过程中，参考一下内核源代码可以帮助我们更好地了解内核源代码树，并在以后对其进行深入分析。

## 1、概述

图 1 是我们在20,000 英尺的高度看到的视图。




当系统首次引导时，或系统被重置时，处理器会执行一个位于已知位置处的代码。在个人计算机(PC)中，这个位置在基本输入/输出系统(BIOS)中，它保存在主板上的闪存中。嵌入式系统中的中央处理单元(CPU)会调用这个重置向量来启动一个位于闪存/ROM中的已知地址处的程序。在这两种情况下，结果都是相同的。因为PC提供了很多灵活性，BIOS必须确定要使用哪个设备来引导系统。稍后我们将详细介绍这个过程。

当找到一个引导设备之后，第一阶段的引导加载程序就被装入RAM并执行。这个引导加载程序在大小上小于512 字节(一个扇区)，其作用是加载第二阶段的引导加载程序。

当第二阶段的引导加载程序被装入RAM 并执行时，通常会显示一个动画屏幕，并将Linux 和一个可选的初始 RAM磁盘(临时根文件系统)加载到内存中。在加载映像时，第二阶段的引导加载程序就会将控制权交给内核映像，然后内核就可以进行解压和初始化了。在这个阶段中，第二阶段的引导加载程序会检测系统硬件、枚举系统链接的硬件设备、挂载根设备，然后加载必要的内核模块。完成这些操作之后启动第一个用户空间程序(init)，并执行高级系统初始化工作。

这就是 Linux引导的整个过程。现在让我们深入挖掘一下这个过程，并深入研究一下Linux引导过程的一些详细信息。

**2、系统启动**

系统启动阶段依赖于引导Linux系统上的硬件。在嵌入式平台中，当系统加电或重置时，会使用一个启动环境。这方面的例子包括U-Boot、RedBoot和Lucent的MicroMonitor。嵌入式平台通常都是与引导监视器搭配销售的。这些程序位于目标硬件上的闪存中的某一段特殊区域，它们提供了将Linux内核映像下载到闪存并继续执行的方法。除了可以存储并引导Linux映像之外，这些引导监视器还执行一定级别的系统测试和硬件初始化过程。在嵌入式平台中，这些引导监视器通常会涉及第一阶段和第二阶段的引导加载程序。

在 PC 中，引导Linux 是从 BIOS 中的地址0xFFFF0 处开始的。BIOS的第一个步骤是加电自检(POST)。POST的工作是对硬件进行检测。BIOS的第二个步骤是进行本地设备的枚举和初始化。

给定 BIOS 功能的不同用法之后，BIOS由两部分组成：POST 代码和运行时服务。当POST 完成之后，它被从内存中清理了出来，但是BIOS运行时服务依然保留在内存中，目标操作系统可以使用

这些服务。

要引导一个操作系统，BIOS运行时会按照CMOS的设置定义的顺序来搜索处于活动状态并且可以引导的设备。引导设备可以是软盘、CD-ROM、硬盘上的某个分区、网络上的某个设备，甚至是USB闪存。

通常，Linux都是从硬盘上引导的，其中主引导记录(MBR)中包含主引导加载程序。MBR是一个512字节大小的扇区，位于磁盘上的第一个扇区中(0道0 柱面1 扇区)。当 MBR 被加载到RAM 中之后，BIOS 就会将控制权交给MBR。

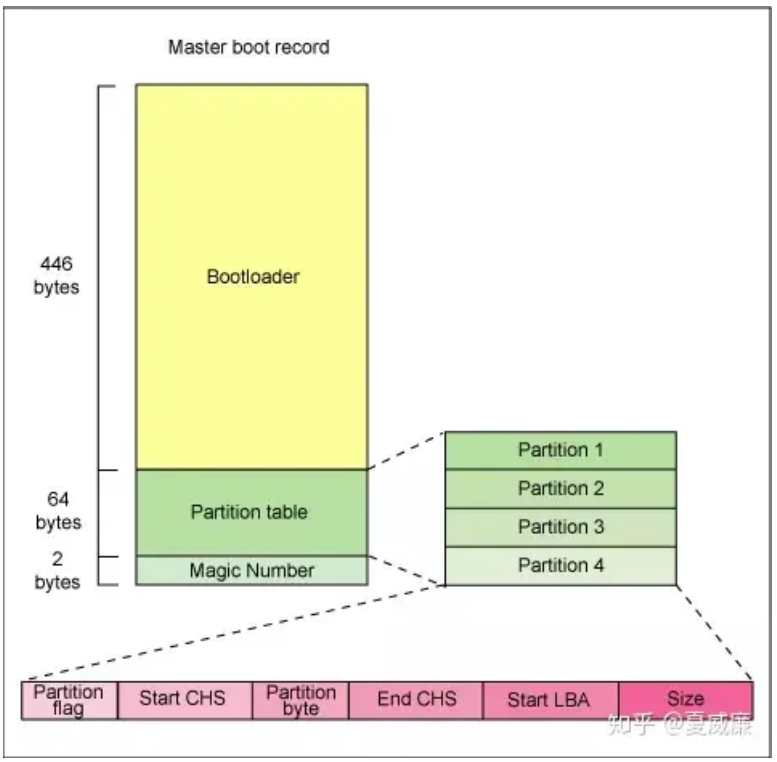
注意：要查看 MBR的内容，请使用下面的命令：

# ddif=/dev/hda of=mbr.bin bs=512 count=1#od -xambr.bin

这 个dd 命令需要以root用户的身份运行，它从/dev/hda(第一个IDE 盘)上读取前 512 个字节的内容，并将其写入mbr.bin 文件中。od命令会以十六进制和ASCII码格式打印这个二进制文件的内容。

MBR 中的主引导加载程序是一个512 字节大小的映像，其中包含程序代码和一个小分区表(参见图2)。前 446个字节是主引导加载程序，其中包含可执行代码和错误消息文本。接下来的64个字节是分区表，其中包含4 个分区的记录(每个记录的大小是16 个字节)。MBR以两个特殊数字的字节(0xAA55)结束。这个数字会用来进行MBR的有效性检查。

图2. MBR剖析



主引导加载程序的工作是查找并加载次引导加载程序(第二阶段)。它是通过在分区表中查找一个活动分区来实现这种功能的。当找到一个活动分区时，它会扫描分区表中的其他分区，以确保它们都不是活动的。当这个过程验证完成之后，就将活动分区的引导记录从这个设备中读入RAM中并执行它。

**3、第二阶段引导加载程序**

次引导加载程序(第二阶段引导加载程序)可以更形象地称为内核加载程序。这个阶段的任务是加载Linux内核和可选的初始 RAM磁盘。

在 x86 PC环境中，第一阶段和第二阶段的引导加载程序一起称为Linux Loader(LILO)或GRand Unified Bootloader(GRUB)。由于LILO有一些缺点，而 GRUB克服了这些缺点，因此下面让我们就来看一下GRUB。

关于 GRUB，很好的一件事情是它包含了有关Linux文件系统的知识。GRUB不像LILO 一样使用裸扇区，而是可以从ext2 或 ext3 文件系统中加载Linux 内核。它是通过将两阶段的引导加载程序转换成三阶段的引导加载程序来实现这项功能的。阶段1 (MBR)引导了一个阶段1.5 的引导加载程序，它可以理解包含Linux 内核映像的特殊文件系统。这方面的例子包括reiserfs_stage1_5(要从Reiser日志文件系统上进行加载)或e2fs_stage1_5(要从ext2或 ext3 文件系统上进行加载)。当阶段1.5 的引导加载程序被加载并运行时，阶段2的引导加载程序就可以进行加载了。

当阶段2加载之后，GRUB就可以在请求时显示可用内核列表(在/etc/grub.conf中进行定义，同时还有几个软符号链接/etc/grub/menu.lst和/etc/grub.conf)。我们可以选择内核甚至修改附加内核参数。另外，我们也可以使用一个命令行的shell对引导过程进行高级手工控制。

GRUB阶段引导加载程序：/boot/grub目录中包含了stage1、stage1.5和stage2引导加载程序，以及很多其他加载程序(例如，CR-ROM使用的是iso9660_stage_1_5)。

将第二阶段的引导加载程序加载到内存中之后，就可以对文件系统进行查询了，并将默认的内核映像和initrd映像加载到内存中。当这些映像文件准备好之后，阶段2 的引导加载程序就可以调用内核映像了。

**4、内核**

当内核映像被加载到内存中，并且阶段2 的引导加载程序释放控制权之后，内核阶段就开始了。内核映像并不是一个可执行的内核，而是一个压缩过的内核映像。通常它是一个zImage(压缩映像，小于512KB)或一个bzImage(较大的压缩映像，大于512KB)，它是提前使用zlib进行压缩过的。在这个内核映像前面是一个例程，它实现少量硬件设置，并对内核映像中包含的内核进行解压，然后将其放入高端内存中，如果有初始RAM磁盘映像，就会将它移动到内存中，并标明以后使用。然后该例程会调用内核，并开始启动内核引导的过程。

当 bzImage(用于i386映像)被调用时，我们从./arch/i386/boot/head.S的start 汇编例程开始执行(主要流程图请参看图3)。这个例程会执行一些基本的硬件设置，并调用./arch/i386/boot/compressed/head.S中的startup_32例程。此例程会设置一个基本的环境(堆栈等)，并清除Block Started by Symbol(BSS)。然后调用一个叫做decompress_kernel的C 函数

(在./arch/i386/boot/compressed/misc.c中)来解压内核。当内核被解压到内存中之后，就可以调用它了。这是另外一个startup_32函数，但是这个函数在./arch/i386/kernel/head.S中。

在这个新的 startup_32函数(也称为清除程序或进程0)中，会对页表进行初始化，并启用内存分页功能。然后会为任何可选的浮点单元(FPU)检测CPU的类型，并将其存储起来供以后使用。然后调用start_kernel函数(在init/main.c中)，它会将您带入与体系结构无关的Linux 内核部分。实际上，这就是Linux 内核的 main函数。

图3. Linux内核i386引导的主要函数流程


注意函数decompress_kernel就是显示我们通常看到的解压消息的地方：

Uncompressing Linux... Ok, booting thekernel.

通过调用start_kernel，会调用一系列初始化函数来设置中断，执行进一步的内存配置，并加载初始RAM磁盘。最后，要调用kernel_thread(在arch/i386/kernel/process.c中)来启动init函数，这是第一个用户空间进程(user-spaceprocess)。最后，启动空任务，现在调度器就可以接管控制权了(在调用cpu_idle之后)。通过启用中断，抢占式的调度器就可以周期性地接管控制权，从而提供多任务处理能力。

在内核引导过程中，初始 RAM磁盘(initrd)是由阶段2引导加载程序加载到内存中的，它会被复制到RAM 中并挂载到系统上。这个initrd 会作为RAM中的临时根文件系统使用，并允许内核在没有挂载任何物理磁盘的情况下完整地实现引导。由于与外围设备进行交互所需要的模块可能是initrd的一部分，因此内核可以非常小，但是仍然需要支持大量可能的硬件配置。在内核引导之后，就可以正式装备根文件系统了(通过pivot_root)：此时会将initrd根文件系统卸载掉，并挂载真正的根文件系统。

initrd函数让我们可以创建一个小型的Linux内核，其中包括作为可加载模块编译的驱动程序。这些可加载的模块为内核提供了访问磁盘和磁盘上的文件系统的方法，并为其他硬件提供了驱动程序。由于根文件系统是磁盘上的一个文件系统，因此initrd函数会提供一种启动方法来获得对磁盘的访

问，并挂载真正的根文件系统。在一个没有硬盘的嵌入式环境中，initrd可以是最终的根文件系统，或者也可以通过网络文件系统(NFS)来挂载最终的根文件系统。

### 5、Init

当内核被引导并进行初始化之后，内核就可以启动自己的第一个用户空间应用程序了。这是第一个调用的使用标准C库编译的程序。在此之前，还没有执行任何标准的C 应用程序。

在桌面 Linux 系统上，第一个启动的程序通常是/sbin/init。但是这不是一定的。很少有嵌入式系统会需要使用init所提供的丰富初始化功能(这是通过/etc/inittab进行配置的)。在很多情况下，我们可以调用一个简单的shell脚本来启动必需的嵌入式应用程序。

### 6、结束语

与Linux本身非常类似，Linux的引导过程也非常灵活，可以支持众多的处理器和硬件平台。最初，加载引导加载程序提供了一种简单的方法，不用任何花架子就可以引导Linux。LILO引导加载程序对引导能力进行了扩充，但是它却缺少文件系统的感知能力。最新一代的引导加载程序，例如GRUB，允许Linux 从一些文件系统(从Minix 到 Reise)上进行引导。

发布于 2019-09-28 11:56

Linux　操作系统　Linux 开发

写下你的评论...

2 条评论　默认　最新

**云颠下的烛火**
+2
2020-03-01　回复　赞

**卢集**
+1
2019-10-05　回复　赞

**文章被以下专栏收录**

**Linux之家**
让Linux语言变成最简单的学习

**推荐阅读**

**2020年完全使用Linux工作**

零一学堂 发表于零一学堂


**14门Linux课程，打通你Linux的任督二脉！**

监桥云课 发表于编程教室

**写在使用 Linux 工作一年后**

诺阿卡