《openEuler内核编程》

课程讲稿

第二章 第1讲

引导程序

软件所制

第二章 第1讲 引导程序

**学时：**1学时

**教学目的：**知晓计算机加电初始化的过程，对计算机启动产生感性认识。了解BIOS产生的历史及BIOS在计算机启动过程中关键的作用。了解现代UEFI产生的原因，及其相对传统BIOS的优势。了解Linux系统启动引导流程。

**课程时间线：**

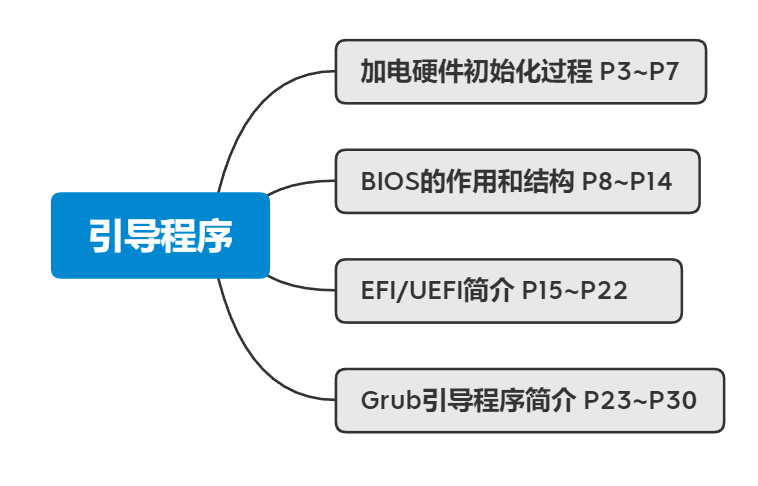


**课外参考读物：**

<https://zhuanlan.zhihu.com/p/25281151> （UEFI历史）

<https://blog.csdn.net/celiaqianhj/article/details/6777599（UEFI>结构）

**知识框图：**

****

**PPT讲稿：**

3. 任何一台计算机，或者说是计算设备，比如手机，打印机，路由器，都算，启动的时候第一步一定是加电自检。英文叫Power-On Self Test。简称POST。加电自检是针对硬件的，确保我所有的硬件运行正常。如果POST出错，计算机就会有蜂鸣声，不同的蜂鸣声代表的意义不同。同时计算机也不会进一步启动。比如内存、CPU插不牢都会触发POST错误。硬件的稳定是系统正常启动和运行的基石。
4. POST过程基本分为以下一些步骤。第一检查CPU的寄存器，第二检查BIOS代码是否完整。不完整后面根本没有办法引导启动。第三,检查DMA，也就是直接存储器访问。它允许不同速度的硬件之间进行数据交换。还要检查定时器，中断控制器。第四检查内存是否正常。第五检查系统总线和外部的设备。第六才开始初始化BIOS。接着跳转到下一级bios执行并返回。最后识别可启动的设备，比如硬盘、U盘，光盘等等。
5. 这一张图非常详细的计算机硬件加电初始化图。简单说一下。右上角是RTC电池，实际上就是一个纽扣电池，如果你拆开过台式机，里面的主板上一定会有一块这样的电池。这块电池能量很小，只能维持RTC时钟芯片以及CMOS芯片运转。一旦介入系统电源后。就不需要它了。然后看序号3左上角，这里的意思就是接入电源，这里不管是台式机还是笔记本，不做区分。紧接着一系列的通信，把一些芯片唤醒。然后我们按下了开机键，看上面的序号8。EC控制器开始通知南桥芯片，南桥芯片紧接着通知其他芯片工作，最终在北桥芯片的配合下同时给CPU发送信号，CPU正式启动，开始启动系统。后面就是bios的事了。
6. 这是一张上电时序图，对应刚才的进程，不过在时间的维度展示给大家。
7. CPU开始运作之后，其实他自己也是蒙的，不知道该干嘛。因为必须先运行程序，计算机才能启动，但是只有计算机启动了才能运行程序。这是一个相互矛盾的事情，为了解决这个问题，人们发明了bios，也就是基本输入输出系统。
8. bios第一次出现是在CP/M操作系统中。这个操作系统有着比较好的层次结构，也就是模块化的设计理念。这样做的好处就是系统的移植性好。BIOS就是这个操作系统的一个模块。要知道，我们在配置一台新电脑的时候，可选的配件品牌是非常多的，不同的主板，不同的CPU。内存、硬盘都有很多不同的品牌。以及各种各样的pci设备，比如显卡、网卡、声卡等等。这些设备千差万别，那么如何用一个软件安装包服务于这些多种设备呢？我们就需要在中间搭建一层软件抽象层来进行硬件的封装。BIOS的作用就在这里。BIOS的主要功能就是：初始化硬件和提供硬件的软件抽象。通过BIOS的抽象，操作系统才能做到一个安装包适应各种各样的硬件环境。BIOS的发明人就是图片中这位，加里 基尔代尔。
9. 但BIOS真正的流行起来，靠的是IBM。当年IBM靠自己的PC产品赚了一大笔钱，他们的电脑硬件是英特尔的，软件是微软的，只有bios是自己的。自信的IBM在计算机的说明书上对bios进行了详细的说明。但这个说明竟然埋下了隐患。由于bios在当时是非常核心和关键的一项技术，有着很高的商业价值。很多公司都会照着IBM的bios进行仿制，山寨。有的高级一点的，为了防止版权纠纷，会独立的开发bios，但是呢，参考资料就是ibm的PC说明书。所以在当时，bios成了一个实质上的标准。
10. 这两张图都是bios芯片。右边的phoenix公司就是采用“净室“的方法，参考IBM的手册，开发出bios芯片的，甚至达到了bios领域霸主的地位。
11. BIOS的作用有以下几个方面。首先是开机时候对操作系统的各个组件进行检查。然后加载引导程序以及操作系统。接着向操作系统硬件配置信息。同时给操作系统提供硬件的抽象，隐藏最底层的硬件信息。这一步非常重要，有了bios，使得操作系统不用对不同的硬件进行特殊的配置了，极大地方便了操作系统的移植和普适性。
12. BIOS中主要存放了四类程序。第一是自诊断程序。也就是完整的POST自检，包括CPU,640K基本内存检测，1M扩展内存检测，主板、显卡等等。出错的话会峰鸣警告。第二是CMOS设置程序，通过键盘的热键启动，一般是F2。这个我想大家应该都经历过，有的时候想改变一下磁盘启动顺序，还有电脑启动不了，都会先进到这里看一下，这也是一个独立的给用户配置BIOS的程序。第三是系统自检装载程序将磁盘0磁道0扇区的引导程序装入内存，运行后在把操作系统装入内存。第四主要I/O设备的驱动程序以及提供中断服务。
13. bios本身分为两部分，一部分是ROM，这个是存储BIOS代码的。另一部分是CMOS，这一部分是存储BIOS数据的。想刚才说的，进入设置界面，改变启动盘顺序，就是将数据改写在CMOS中。下面是BIOS的代码结构，看下有个基本印象就可以。
15. BIOS 作为一个软件，他的生命周期非常长，从七十年代，一直使用到九十年代末。在二十世纪初我们国家计算机开始普及的时候，绝大多数电脑也是用的BIOS。有的同学说，老师，不对啊，现在我的电脑用的也是BIOS啊，而且界面比以前的好看多了，还是图形化的。现在我们用的电脑所说的BIOS已经不是经典的BIOS了，而是新一代的UEFI。属于是BIOS的继承者，但是出于习惯，还是会叫它BIOS，但是已经不是经典的BIOS程序了。
16. 经典的BIOS程序全部用汇编语言编写，一用就是几十年。一直到97年的时候，英特尔准备开发基于一款新处理器的服务器，所有的配套硬件软件都是新的，原本想用经典的BIOS，但还是不行。于是英特尔打算自己开发一套新的BIOS，这一次他们用的是C语言写的。写完之后，可以和他们的新服务器完美匹配。于是他们就想，为什么不把这套新的BIOS应用到自己其他计算机上呢？于是两年后的1999年，他们发布了全新设计的BIOS，命名为EFI，全称可扩展式固件接口。
17. 我们看这个时间表，99年的时候是EFI发布。后续又更新了几版。在2005年的时候，人们还是觉得EFI有一点不太给力，于是各大bios厂商，操作系统厂商以及芯片生产企业共同成立了一个UEFI论坛，来共同发展一个新的接口。名字就叫UEFI，统一的可扩展固件接口。Intel作为论坛的一员，提供了EFI的源代码。在第二年，论坛发布了UEFI 2.0规范。要注意，这里的UEFI只是一个规范标准，并不负责具体的代码实现。
18. 图中是微星主板的uefi设置界面。可以看到图形化做的非常好了，还支持鼠标的操作。由于模块化的设计，使得uefi更新也比较容易。现在有的UEFI还支持在线更新。
19. 不论是传统BIOS，还是UEFI，都有三大任务。1. 初始化硬件 2. 提供硬件的软件抽象 3. 启动操作系统。从这一根本的角度来讲，两者并没有任何的区别。那为什么UEFI可以代替传统BIOS呢。他最大的优势，就在于，标准接口，开放统一，并且开源。完全没有了最早BIOS时代，各个厂商竞争激烈，接口互不相容的状态了。UEFI非常利于行业的发展。
20. 同时UEFI支持32位，64位模式，比传统16位BIOS的寻址能力更强。其驱动也是单独加载并识别硬件的。另外这些驱动使用一种专门用于UEFI的虚拟机器语言编写的，在运行环境下才会被解释运行，保证了兼容性。这些都是UEFI的优点
21. UEFI追求的是模块化、标准化、简单化。为了达成这一目标，就需要一个设计优秀的框架结构。目前看到的这个框架就是Intel提出的一个框架，也是UEFI唯一的框架。最底层就是硬件。网上绿色的就是FrameWork，H形状。上下两端分别用来和操作系统以及硬件进行沟通，分别有一套协议。两套协议之间进行沟通的桥梁就是中间的DXE 和 PEI模块。上方蓝色的Platform Drivers是和具体硬件平台相关的驱动。由于UEFI在设计的时候就考虑到了跨平台的需求，这里蓝色部分，切换平台的时候，只替换这一部分就可以了。橙色EFI Drivers，失职符合EIF驱动标准的驱动程序。粉色的CSM。这个模块是为了向前兼容而存在的。上面黄色的薄薄的一层是EFI，表示的是接口。上面的灰色就是操作系统了，我们看到分了好几个部分，左边的就是兼容EFI标准的操作系统 ，右边的两个就是不兼容EFI标准的操作系统。
22. 最后我们来看一下Grub引导程序
23. 在BIOS完成硬件检测，资源分配等工作后，会把磁盘中的主引导扇区中的BootLoader读取到系统内存中。随后控制权边交接给了BootLoader。BootLoader的主要使命就是将操作系统加载到内存中，并启动操作系统。
24. 我们以Linux启动为例，看一下启动顺序。BIOS运行完成之后，回去MBR主引导扇区加载GRUB,启动之后的GRUB会查找并加载系统内核。运行中的内核，会装载驱动。然后执行init初始化程序，此时操作系统开始初始化，然后执行runlevel程序，启动指定级别的服务，一般有gui的操作系统，会进入level 5的级别。只有命令行的系统，比如服务器，会进入level3的级别。到这，操作系统就算是启动完了。BIOS讲过了，下面这几个步骤，给大家简单讲一讲。
25. MBR又叫主引导记录。它存储在主分区的第一扇区中。存储了bootloader的程序，分区表，以及一个表示MBR结束的魔术16进制数55AA。MBR在现在看来也已经是一个过时的技术了，在他的分区表规则中，最大只能支持2T的硬盘。现在更常用的是GPT分区表，可以支持近乎无限大的硬盘容量和分区数量。顺便说一下GPT也是UEFI标准中的一项。
26. MBR中存储的BootLoader程序可以有很多种，其中Grub是Linux上最常用的启动器。GRUB支持多操作系统，并且在启动的时候，用户可以自己手动选择想启动的系统。GRUB不只支持所有的Linux文件系统，对于Windows的FAT和NTFS也都支持。GRUB还支持图形界面，甚至可以使用鼠标操作。Grub还支持丰富的终端命令，用户可以灵活的进行自定义配置。
27. 引导过后，就会载入系统的内核。内核是Linux系统最主要的程序。内核其实非常小，只包含启动系统所需要最基本的模块。kernel一开始是压缩的，解压后才放入内存中。kernel会以只读的方式挂在根文件系统，随后装载第一个进程，也就是init程序。
28. init程序要执行的操作也有很多，这里看到至少又要11个主要步骤。
29. 初始化过后运行runlevel程序，Linux一共有6中不同的级别。第0级表示关机状态。第1级是单一用户模式，可以看成是保护模式。第2级是多用户模式，但是没有网络连接。第3级是命令行模式。也就是我们没有GUI界面的操作系统会进入的启动模式，一般没有GUI的服务器操作系统都是这个级别。第4级没有设置。第5级是图形界面。我们常用的一些面向终端用户的Linux发行版，比如Ubuntu，有漂亮的操作界面，就是属于第5级。第6级表示重启系统。每个级别都对应有一个脚本，runlevel会根据不同的需求来运行指定的脚本。
30. 当然，最新的Linux内核版本一级很少使用inittab的方式执行大量shell脚本来启动服务，而是使用systemd进行初始化。systemd是Linux下的一种init软件，由Lennart Poettering带头开发，其开发目标是提供更优秀的框架以表示系统服务间的依赖关系，并依此实现系统初始化时服务的并行启动，同时达到降低Shell的系统开销的效果，最终代替现在常用的System V与BSD风格init程序。传统sysvinit使用inittab来决定运行哪些shell脚本，大量使用shell脚本被认为是效率低下无法并行的原因。systemd使用了Linux专属技术，不再顾及POSIX兼容。systemd是一个服务管理器，能解决服务间的依赖关系，实现服务并行启动。linux上的服务，启动过程通常要做初始化工作，如：挂载文件系统，过程的每一步都被systemd抽象为一个配置单元，即：unit。如：服务A启动时服务B必须已启动，systemd使用unit管理和解决服务依赖问题。systemd将启动和运行相关的对象包装成unit，每个unit对应一个unit配置文件。systemd默认从/etc/systemd/system/目录中读取配置文件。定义unit的文件可以放在多个地方，每个地方都有不同的优先级和含义。系统的unit文件副本通常存放在/lib/systemd/system目录中，安装软件时对应的unit文件默认存放在此目录。存储在此目录的unit文件可在会话期间按需启动和停止，通常不应编辑这个目录下的文件。下面是一个system unit典型示例。这个示例，在服务启动时，使用ethtool命令禁用以太网接口(ens5f5)上的“wake-on-lan”特性，并在服务停止时重新启用“wake-on-lan”特性。
31. 至此，我们就大致了解了操作系统的启动情况。