系统的内存ram断电后就丢失数据，所以操作系统内核不能放在ram中，要放在一些能保存数据的设备商，如flash或者硬盘，启动时从硬盘中取出内核到ram中，所以cpu一开机就要执行一段程序来完成这个工作，cpu都被设计成一上电就会从制动的地址开始执行指令，i386的这个线性地址就是0xffff

所以这个地址上要有能引导操作系统的指令的开始

硬盘上的第一个扇区是主引导扇区，即MBR，共512字节，MBR在linux里面并不直接引导操作系统，而是类似于一种中间过程，用它来引导LILO，GRUB等系统引导程序，用他们来引导系统

下面开始从系统上电讲起

1. 上电->BIOS

系统上电后最先BIOS开始自检，扫描硬件以及确定初始启动设备等，然后从硬盘从读出主引导块，即MBR，MBR被读到内存以后，BIOS就完成任务，退出工作，将控制权交给MBR

BIOS的主要工作：

检测硬件，探测系统安装有哪些设备以及工作是否正常，即“上电自检”

对硬件初始化

从软驱，光驱，硬盘等启动设备查找可以引导的扇区

发现合法设备，将设备的第一个扇区的内容复制到ram中，然后跳到这段内存中并执行这段扇区里面的内容

1. MBR

如果是从硬盘启动linux，则第一个扇区就是MBR，有512字节，放着分区表和一个小程序，这个程序用来引导操作系统，如果是linux系统的话，这个小程序就是LILO或者GRUB，

但LILO或者GRUB太大，MBR容纳不下，于是LILO或者GRUB就把自己拆成两部分，第一部分放在MBR中，另外一部分放在另外的山区中中

启动的时候，BIOS把LILO或者GRUB的位于MBR中的部分载入到内存中，然后这部分程序把自己剩余的部分也拷贝到ram中，第二部分的GRUB会从磁盘里面读入setup()函数和内核镜像

（3）setup

首先是检测系统可用内存的总量,然后探测、初始化显卡，总线等设备，把CPU从实模

式切换到保护模式，为内核执行以及内核的解压缩作准备，最后跳转到startup\_32()函数,进入内核

1. startup\_32()

/arch/i386/kernel/head.S中

为第一个Linux进程（process 0）设置环境，如设置PROCESS0的内核模式栈，初始化GDT,IDT，最后跳转到start\_kernel（）函数

1. start\_kernel（）

位于/init/main.c中

继续执行内核的初始化，并且是高层次的初始化

具体函数分析

asmlinkage void \_\_init start\_kernel(void)

{

…

printk(linux\_banner);

打印内核信息，linux\_banner这个字符串位于init/version.c中

setup\_arch(&command\_line);

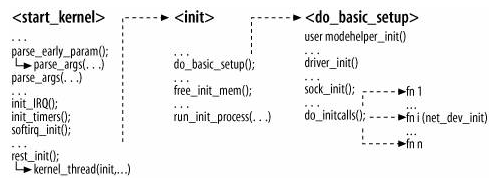
这个函数中会为调用init\_bootmem为物理内存页面管理机制的建立做准备，为整个物理内存建立页面位图，即把内核映像终点end上方的若干个页面作为物理页面位图，

Paging\_init初始化页表

Mem\_init初始化页描述符

Kernel\_thread创建内核线程

Cpu\_idle无穷循环，



Do\_initcalls初始化所有\_\_init宏带头的需要在内核启动的时候初始化的函数，执行完以后会释放这些模块占用的内存