# 概念介绍

**分区表**是在磁盘（存储介质）上的，用于描述该磁盘的分区情况，有**GPT**和**MBR**两种格式。

**BIOS**和**UEFI**是**固件接口标准**，功能包括开机自检、启动流程（如何找到引导程序）、给操作系统和引导程序提供系统服务等。

**MBR**是引导扇区，包括最多446个字节的引导程序（通常是引导程序的前部）和MBR分区表，其中可以包括4个主分区。

**启动方式**是指如何主板上的固件在开机自检后如何找到引导程序，有**Legacy模式**（BIOS + MBR）和**UEFI模式**（UEFI + GPT）

# 固件接口标准

**固件（Firmware）**是一种特殊的软件，提供了对于硬件设备的底层控制，包括控制、监控、数据操作等功能。固件存储在硬件设备附带的**非易失性存储器**（ROM、EPROM或者Flash）中，通常不可修改。

这里的**固件接口标准**特指的是PC主板上的固件接口的标准。PC主板固件是软件，因此它也是由（板载的）CPU来运行的。

## BIOS

**基本输入输出系统（Basic Input/Output System）**，IBM推出的业界标准的固件接口，存储于主板的**ROM（或EEPROM、flash）**中，BIOS是PC启动之后加载的第一个软件（现代的x86平台中，UEFI BIOS已经不是第一个被处理器运行的程序）。BIOS提供的功能包括：

1. 开机自检（Power on Self Test）
2. 加载引导程序（MBR中的，通常是bootloader的第一级）
3. 向OS提供抽象的硬件接口，隐藏具体硬件的变化。之前的程序（OS等）调用BIOS接口而非直接跟硬件打交道，而现在的操作系统会忽略BIOS接口直接控制硬件。

**CMOS**是PC上的另一个重要的存储器，用于保存BIOS的设置结果，CMOS是**RAM**，由板载的纽扣电池供电。

当计算机的电源打开，CPU的程序计数器会指向BIOS，开始执行BIOS中的程序，将芯片组和存储器子系统初始化。BIOS会把自己从闪存中，解压缩到系统的主存；并且从那边开始运行。PC的BIOS代码也包含诊断功能，以保证某些重要硬件组件，像是[键盘](https://zh.wikipedia.org/wiki/%E9%8D%B5%E7%9B%A4)、[磁盘](https://zh.wikipedia.org/wiki/%E7%A3%81%E7%A2%9F)（BIOS在引导时会侦测硬盘的[S.M.A.R.T.](https://zh.wikipedia.org/wiki/S.M.A.R.T.)状态）、输出输入端口等等，可以正常运作且正确地初始化。几乎所有的BIOS都包含设置程序（BIOS Setup），可以进行BIOS的设置，如设置系统时间，设置引导设备顺序，激活/禁用计算机的某些周边设备。主板的CMOS芯片（[UEFI](https://zh.wikipedia.org/wiki/UEFI)多使用[NVRAM](https://zh.wikipedia.org/wiki/NVRAM)存储UEFI设置值/硬件侦测信息）用于存储BIOS设置值及硬件侦测信息。主板上的钮扣电池用于让CMOS芯片存储BIOS设置值，以及计算机在断电时依然可以让系统时钟运作。CMOS在没有电力供应（CMOS所需电力由主板上的钮扣电池提供）的情况下会丢失存储的数据，而NVRAM则不会。

## EFI/UEFI

**统一可扩展固件接口，Unified Extensible Firmware Interface**，原名是**EFI**，由Intel研发，定义操作系统和系统固件之间的软件接口标准，用于替代BIOS接口，EFI是UEFI的前称。

相较于**BIOS**，**UEFI**是用模块化，[C语言](https://zh.wikipedia.org/wiki/C%E8%AF%AD%E8%A8%80)风格的参数堆栈传递方式，动态链接的形式构建的系统，较BIOS而言更易于实现，容错和调试特性更强，缩短了系统研发的时间。它可以运行于x86-64、IA32、IA64等架构上（在个人计算机上通常是x86-64平台），突破传统16位代码的[寻址能力](https://zh.wikipedia.org/w/index.php?title=%E5%AF%BB%E5%9D%80%E8%83%BD%E5%8A%9B&action=edit&redlink=1)，达到处理器的最大寻址。

一般认为，UEFI由以下几个部分组成：

1. Pre-EFI初始化模块
2. EFI驱动程序执行环境（DXE）
3. EFI驱动程序
4. 兼容性支持模块（CSM）
5. EFI高层应用
6. [GUID磁盘分区表](https://zh.wikipedia.org/wiki/%E5%85%A8%E5%B1%80%E5%94%AF%E4%B8%80%E6%A8%99%E8%AD%98%E5%88%86%E5%8D%80%E8%A1%A8)（GPT）

通常**初始化模块**和**DXE**被集成在一个ROM中；**Pre-EFI初始化程序**在系统开机的时候最先得到执行，完成存储器的初始化工作，然后加载**UEFI DXE（驱动程序执行环境）**。

当DXE被加载运行时，系统便具有了枚举并加载其他**UEFI驱动程序**的能力。在基于[PCI Express](https://zh.wikipedia.org/wiki/PCI_Express)架构的x64计算机系统中，系统会加载UEFI内置的驱动程序模块，完成UEFI固件、CPU、存储器、芯片组及主板的进一步初始化，然后初始化各PCIe控制器、PCIe适配器（如[RAID](https://zh.wikipedia.org/wiki/RAID)扩展卡或[显卡](https://zh.wikipedia.org/wiki/%E9%A1%AF%E7%A4%BA%E5%8D%A1)）及芯片组内置PCIe适配器（如芯片组内置的[SATA](https://zh.wikipedia.org/wiki/SATA)、[USB](https://zh.wikipedia.org/wiki/USB" \o "USB)、[网卡](https://zh.wikipedia.org/wiki/%E7%B6%B2%E5%8D%A1" \o "网卡)等功能）并加载这些PCIe设备的UEFI驱动程序（如果有的话，也有可能是加载PCIe设备的Legacy Option ROM）。UEFI驱动程序不仅可以包含在PCIe适配器的ROM中（作为PCIe设备的UEFI Option ROM），还可以以**.EFI**文件的形式被方便的加载。

**CSM**是在x86平台UEFI系统中的一个特殊的模块，它将为不具备UEFI引导能力的操作系统以及16位的传统Option ROM（即非EFI的Option ROM）提供类似于传统BIOS的系统服务。**Secure Boot**功能要求原生UEFI（即关闭CSM），因此在UEFI固件设置中打开CSM前，需要在UEFI固件设置中关闭Secure Boot。

**EFI驱动程序**一般在设备的ROM中，或者ESP中；**EFI高层应用**一般在**ESP**中。

# 启动方式

## Legacy mode

即通过MBR/BIOS进行引导的传统模式，流程如下：

1. BIOS加电自检（Power On Self Test -- POST）。
2. 读取主引导记录（MBR）。BIOS根据CMOS中的设置依次检查启动设备：将相应启动设备的第一个扇区（也就是MBR扇区）读入内存。
   1. 检查MBR的结束标志位是否等于55AAH，若不等于则转去尝试其他启动设备，如果没有启动设备满足要求则显示"NO ROM BASIC"然后死机。
   2. 当检测到有启动设备满足要求后，BIOS将控制权交给相应启动设备的MBR。
3. 根据MBR中的引导代码启动[引导程序](https://zh.wikipedia.org/wiki/%E5%BC%95%E5%AF%BC%E7%A8%8B%E5%BA%8F" \o "引导程序)。

## UEFI mode

UEFI启动不依赖于Boot Sector（比如MBR），大致流程如下：

1. **Pre-EFI初始化模块**运行，自检
2. 加载**DXE（EFI驱动程序执行环境）**，枚举并加载**EFI驱动程序**（设备ROM或ESP中）
3. 找到ESP中的引导程序，通过其引导操作系统。

## CSM mode

UEFI中的兼容性支持模块（Compatible Support Module）提供了引导UEFI固件的PC上的传统磁盘（MBR格式）的方法。

# 分区表

## MBR分区表

指的是**Master Boot Record（主引导记录）**中的分区表，由于大小限制，其中只能存有最多四个分区的描述（亦即4个主分区）。

MBR共512个字节，前446个字节是**主引导程序**，其后的64个字节为4个16字节的**分区描述符（Partition entry）**，分别描述4个**主分区（Primary Partition）**，包括**分区状态、起始扇区CHS（柱面Cylinder、磁头Head、扇区Sector）地址、分区类型、结束扇区CHS、逻辑块地址、扇区数量**。

## EBR分区表

传统的MBR分区表只有4个主分区，为了增加分区的数量，可以将其中一个（只能有一个）主分区指定为**扩展分区（Extended partition）**，于是该分区可以被分为若干**逻辑分区（Logical partition）**。这些逻辑分区由若干EBR描述，每个EBR只描述一个逻辑分区。

位于**Extended Boot Record（扩展引导纪录，EBR）**中的分区表，该分区表所描述的分区即逻辑分区。每个EBR仅表示了一个逻辑分区，该逻辑分区紧接在它的EBR后。EBR的前446个字节（MBR中为引导程序）为空，其后的四个分区描述符中的**第一个**表示当前EBR描述的分区，**第二个**描述符则指向下一个EBR（如果是最后一个则为空），也就是说，各个EBR串接成了一个EBR链表。

“扩展分区”这个词所表达的概念有时候有些混淆，逻辑分区在某些时候也会以“扩展分区”表示，也就是说扩展分区有时候表示主分区中的扩展分区（所有逻辑分区之和），有时候表示单个的逻辑分区。

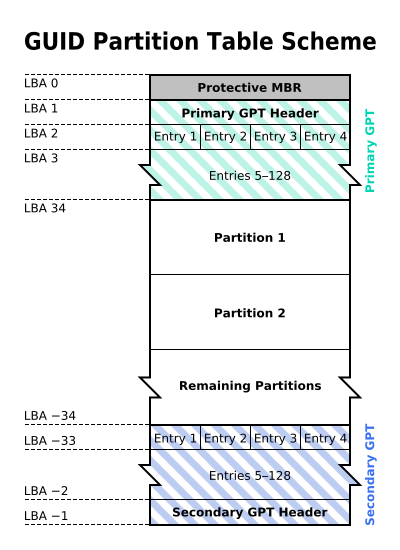
## VBR

**卷引导记录（Volume Boot Record）**，也是引导扇区的一种，在未分区的设备（比如老式的软盘）中，VBR就是第一个扇区。在分区的设备中，VBR位于分区的第一个扇区（设备的第一个扇区是MBR）。

VBR中并没有分区表，放在这里介绍只是防止混淆。

## GPT分区表

GUID Partition Table，是EFI标准的一部分，用于替代MBR分区表，相较起来有分区更大，数量更多（没有4个主分区的限制）等优势，GPT格式的硬盘结构如下，可以看到首部MBR的位置有个保护MBR（用于防止不识别GPT的硬盘工具错误识别并破坏硬盘中的数据），这个MBR中只有一个类型为0xEE的分区。GPT结构如下：



# 分区

分区可以是文件系统，启动kernel image，bootloader裸程序，或者参数等各种数据。

MBR Partition ID（分区类型）：<https://en.wikipedia.org/wiki/Partition_type>

## ESP（EFI系统分区）：

EFI System Partition，FAT格式，在MBR的分区类型ID是0xEF。主要目录是EFI。EFI/boot/bootx64.efi是EFI默认的启动项。安装的操作系统会建立相应的目录EFI/xxx，并将自己的启动项复制为到EFI/boot/bootx64.efi作为缺省启动项。

UEFI官网上注册的EFI的子目录：<http://www.uefi.org/registry>

比如安装Windows的时候，会在ESP分区中建立EFI/Microsoft子目录，并将EFI/Microsoft/bootmgr.efi复制到EFI/boot/bootx64.efi。

安装Ubuntu的时候，会在ESP分区中建立EFI/Ubuntu子目录，并将EFI/ubuntu/grubx64.efi（grub bootloader）复制为EFI/boot/bootx64.efi。因为Grub本身会扫描磁盘上的分区并找到windows启动程序（bootmgr.efi），因此先装windows后装ubuntu仍能通过grub让windows启动。

Linux也可以直接将编译出的Kernel及initrd（打开EFI Stub编译选项）作为efi文件复制到ESP中直接启动。

也可以在PC的系统设置中添加启动项。

维基百科的ESP条目：<https://en.wikipedia.org/wiki/EFI_system_partition>

## BIOS Grub分区

又称bios boot分区，用于在GPT格式的磁盘上存储grub的core.img（即步骤1.5的内容），在传统的MBR分区格式中，这个部分是存储在MBR之后的若干空闲扇区中，而GPT格式中没有这段空闲扇区簇。

注意这个不是Linux安装时候的boot分区

## Windows恢复分区

NTFS格式的分区，即Windows Recovery Environment，WinPE+工具集

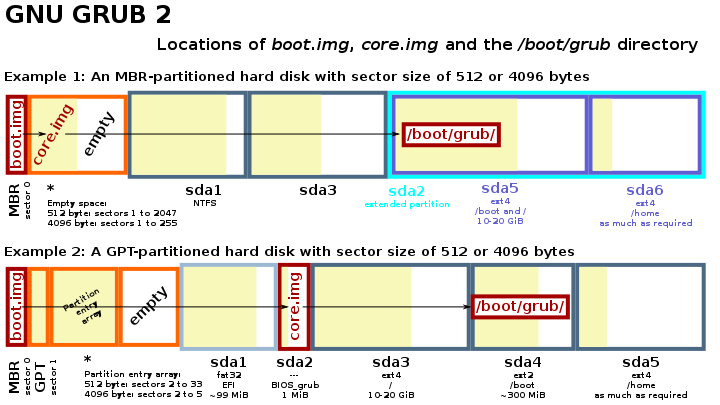
# Bootloader

Bootloader即上文中提到的引导程序，用于启动操作系统或者其它引导程序（比如Grub启动Windows Bootmgr）

## Grub

GNU的开源引导程序，可以用于引导Linux等操作系统，或者用于链式引导其它引导程序（比如Windows Boot Manager），分为三个部分，分别称为步骤1、1.5、2，看名字就可以知道，步骤1.5是可有可没有的，这三个步骤对应的文件分别是：

1. **Boot.img**：步骤1对应的文件，446个字节大小，步骤1可以引导步骤1.5也可以引导步骤2。MBR分区格式的磁盘中，放在**MBR**里（446也是为了符合MBR的启动代码区大小）； GPT分区格式的磁盘中，放在**Protective MBR**中。
2. **Core.img**：步骤1.5对应的文件，32256字节大小。MBR分区格式的磁盘中，放在紧邻MBR的若干扇区中；GPT分区格式的磁盘中，则放在34号扇区开始的位置（第一个分区所处的位置），而对应的GPT分区表中的第一个分区的entry被置空。通常其中包含文件系统驱动以便加载步骤2的文件。
3. /boot/grub：步骤2对应的文件目录，放在系统分区或者单独的Boot分区中



## Windows Boot Manager

是从**[Windows Vista](https://zh.wikipedia.org/wiki/Windows_Vista" \o "Windows Vista)**开始引进的新一代开机管理程序，用以取代**[NTLDR](https://zh.wikipedia.org/wiki/NTLDR" \o "NTLDR)**。

当电脑运行完开机自检后，传统型[BIOS](https://zh.wikipedia.org/wiki/BIOS)会根据[引导扇区](https://zh.wikipedia.org/wiki/%E5%95%9F%E5%8B%95%E7%A3%81%E5%8D%80" \o "引导扇区)查找开机硬盘中标记"引导"分区下的BOOTMGR文件；若是[UEFI](https://zh.wikipedia.org/wiki/UEFI" \o "UEFI)则是Bootmgfw.efi文件和Bootmgr.efi文件，接着管理程序会读取开机配置数据库（BCD, Boot Configuration Database）下的引导数据，接着根据其中的数据加载与默认或用户所选择的[操作系统](https://zh.wikipedia.org/wiki/%E4%BD%9C%E6%A5%AD%E7%B3%BB%E7%B5%B1)。

## NTLDR

是[微软](https://zh.wikipedia.org/wiki/%E5%BE%AE%E8%BD%AF)的[Windows NT](https://zh.wikipedia.org/wiki/Windows_NT)系列[操作系统](https://zh.wikipedia.org/wiki/%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F)（包括[Windows XP](https://zh.wikipedia.org/wiki/Windows_XP)和[Windows Server 2003](https://zh.wikipedia.org/wiki/Windows_Server_2003)）的引导程序。NTLDR可以从[硬盘](https://zh.wikipedia.org/wiki/%E7%A1%AC%E7%9B%98)以及[CD-ROM](https://zh.wikipedia.org/wiki/CD-ROM)、[U盘](https://zh.wikipedia.org/wiki/U%E7%9B%98)等移动存储器运行并引导Windows NT系统的启动。如果要用NTLDR启动其他操作系统，则需要将该操作系统所使用的[启动扇区](https://zh.wikipedia.org/w/index.php?title=%E5%90%AF%E5%8A%A8%E6%89%87%E5%8C%BA&action=edit&redlink=1)代码保存为一个文件，NTLDR可以从这个文件加载其它[引导程序](https://zh.wikipedia.org/wiki/%E5%BC%95%E5%AF%BC%E7%A8%8B%E5%BA%8F)。

NTLDR主要由两个文件组成，这两个文件必须放在系统分区（大多数情况下都是C盘）：

1. NTLDR，引导程序本身
2. boot.ini，引导程序的配置文件