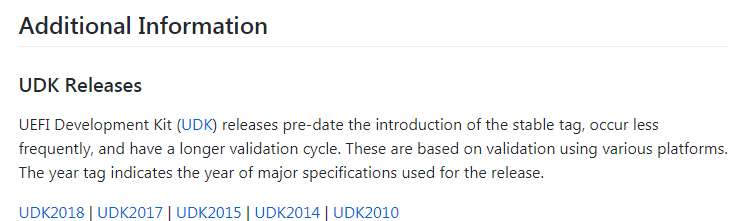
UEFI Development



<https://github.com/tianocore/tianocore.github.io/wiki/EDK-II>

<https://github.com/tianocore/tianocore.github.io/wiki/UDK2018-How-to-Build>

## HOW TO BUILD (LINUX-LIKE SYSTEM)

The below steps are verified on Ubuntu 16.04 LTS Desktop\*:

1. Setup Build Environment
   1. Follow instructions for setting up the build environment on tianocore.org. "<https://github.com/tianocore/tianocore.github.io/wiki/Using-EDK-II-with-Native-GCC>"
      1. Please notice that here the root is "~/src/MyWorkspace" instead of "~/src/edk2"
      2. Make sure BaseTools is built and required software like iASL compiler is installed well. a) Some operations need switch to user "root" to execute. b) At Ubuntu, you can type "sudo apt-get install git" under terminal "Cnt-Alt-T" to install git.
   2. Install NASM 2.12.01 or later. At Ubuntu, you can type "sudo apt-get install nasm" under terminal to install it. You can also download the source package from <http://www.nasm.us/>and install it following the instruction on the website.
   3. Install IASL 20150818 or later. At Ubuntu, you can type "sudo apt-get install iasl"under terminal to install it. You can also download the source package from <https://acpica.org/downloads> and install it following the instruction on the website.
2. Create the full Source Code directory for the UDK2018 release
   1. Create a working space directory in the build machine, for example, ~/src/MyWorkspace
   2. Download the official UDK2018 release .tar file from the [UDK2018 Release Page](https://github.com/tianocore/edk2/releases/tag/vUDK2018)
      1. Download - UDK2018 edk-vUDK2018 Workspace [Source code (tar.gz file)](https://github.com/tianocore/edk2/archive/vUDK2018.tar.gz)
      2. Extract files in [edk2-vUDK2018] to the working space directory ~/src/MyWorkspace.
   3. **OR** Checkout the vUDK2018 tag from GitHub with the following "git" command
      1. Run "git clone https://github.com/tianocore/edk2.git vUDK2018"
      2. Go to the vUDK2018 directory, and from there run: git checkout tags/vUDK2018 -b vUDK2018
      3. Move all files and folders under "vUDK2018" to "~/src/MyWorkspace"
3. Generate OpenSSL\* Crypto Library
   1. Open file "~/src/MyWorkspace/CryptoPkg/Library/OpensslLib/OpenSSL-HOWTO.txt" and follow the instruction to install OpenSSL\* for UEFI building. For this release, please use OpenSSL-1.1.0g. Download it from<https://github.com/openssl/openssl/archive/OpenSSL_1_1_0g.zip> Extract it to ~/src/MyWorkspace/CryptoPkg/Library/OpensslLib, and rename its directory name to openssl
4. Build Steps \*\*\* MdeModulePkg \*\*\*
   1. Open a terminal and type "cd ~/src/MyWorkspace" to enter the workspace directory.
   2. First build the BaseTools by typing "make -C BaseTools"
   3. Initialize the build environment by typing ". edksetup.sh".
   4. Type following command to build platforms using GCC v5.4.0   
      "build -p MdeModulePkg/MdeModulePkg.dsc -t GCC5"
   5. Upon the build completing successfully there should be the UEFI Application "HelloWorld.efi" in the ~/src/MyWorkspace/Build/MdeModule/DEBUG\_GCC5/IA32 directory

If you have questions please email the [edk2-devel](https://github.com/tianocore/tianocore.github.io/wiki/edk2-devel) mail list.

See also [Getting Started with EDK II](https://github.com/tianocore/tianocore.github.io/wiki/Getting-Started-with-EDK-II)

安装模拟器

Vs2008将UnixPkg废弃，迁移到EmulatorPkg模块

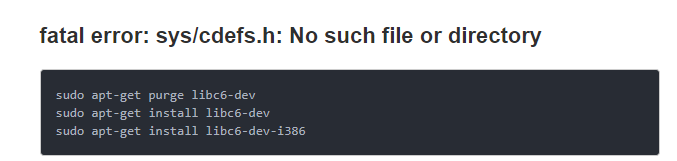
First build the BaseTools by typing "make -C BaseTools"

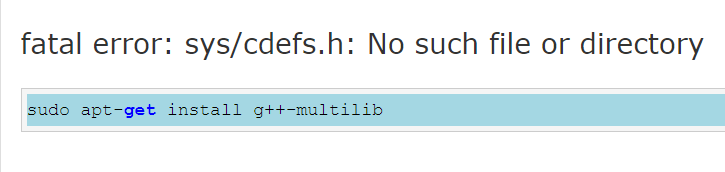
Initialize the build environment by typing ". edksetup.sh"

build -p EmulatorPkg/EmulatorPkg.dsc -t GCC5









$ sudo apt-get install build-essential uuid-dev iasl git gcc-5 nasm python3-distutils

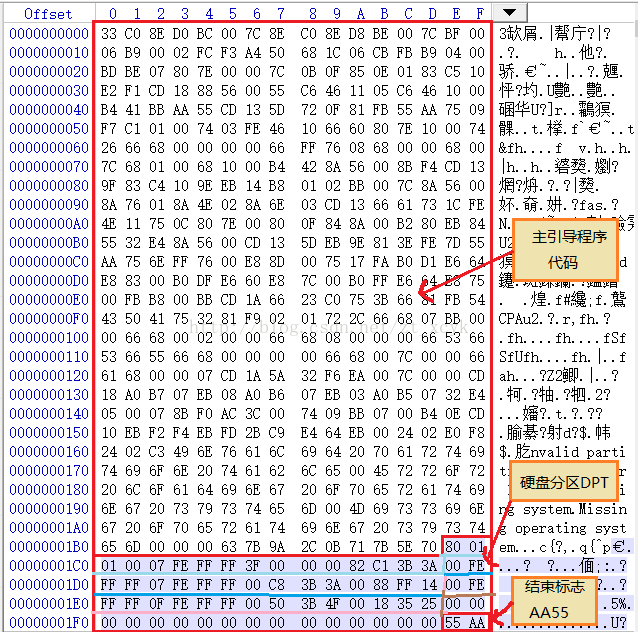
cd ../Build/EmulatorIA32/DEBUG\_GCC5/IA32/ && ./Host

MBR

# MBR分区表详解

2016年12月15日 13:37:46 [zt\_xcyk](https://me.csdn.net/zt_xcyk) 阅读数 5151

### 1.MBR分区

MBR（Main Boot Record 主引导记录区）位于整个硬盘的0磁道0柱面1扇区。在512字节的主引导扇区中，MBR只占用了其中的446个字节，另外的64个字节交给了 DPT（Disk Partition Table硬盘分区表），最后两个字节“55，AA”是分区的结束标志。这个整体构成了硬盘的主引导扇区。

### 2.硬盘分区DPT详解

分区表由4项组成，每项16个字节（Byte).共4×16 = 64字节(Byte)。每项描述一个分区的基本信息。(80) (01 01 00) (07) (FE FF FF) (3F 00 00 00) (82 C1 3B 3A)

|  |  |
| --- | --- |
| 存贮字节位 | 内容及含义 |
| 第1字节 | 引导标志。若值为80H表示活动分区，若值为00H表示非活动分区。 |
| 第2、3、4字节 | 本分区的起始磁头号、扇区号、柱面号。其中：      磁头号——第2字节；      扇区号——第3字节的低6位；      柱面号——为第3字节高2位+第4字节8位。 |
| 第5字节 | 分区类型符。      00H——表示该分区未用（即没有指定）；      06H——FAT16基本分区；      0BH——FAT32基本分区；      05H——扩展分区；      07H——NTFS分区；      0FH——（LBA模式）扩展分区（83H为[**Linux**](http://lib.csdn.net/base/linux)分区等）。 |
| 第6、7、8字节 | 本分区的结束磁头号、扇区号、柱面号。其中：      磁头号——第6字节；      扇区号——第7字节的低6位；      柱面号——第7字节的高2位+第8字节。 |
| 第9、10、11、12字节 | 逻辑起始扇区号 ，本分区之前已用了的扇区数。 |
| 第13、14、15、16字节 | 本分区的总扇区数。 |

分区表上有四项，每一项表示一个分区，所以一个分区表最多只能表示4个分区。[主分区](http://baike.baidu.com/view/1304004.htm)表上的4项用来表示主分区和[扩展分区](http://baike.baidu.com/view/55597.htm)的信息。因为扩展分区最多只能有一个，所以硬盘最多可以有四个主分区或者三个主分区，一个扩展分区。

余下的分区表是表示[逻辑分区](http://baike.baidu.com/view/143399.htm)的。这里有必要阐述一点：逻辑区都是位于扩展分区里面的，并且逻辑分区的个数没有限制。

### 3.实例讲解

**第一项：**

(80) (01 01 00) (07) (FE FF FF) (3F 00 00 00) (82 C1 3B 3A)

* （80）: 代表这个分区（C盘）为活动分区。即系统会从C盘启动。
* （01 01 00） :表示这个分区（C盘）的起始扇区为（0柱面，1磁头，1扇区）。
* （07）:表示这个分区的文件系统为NTFS。
* （ FE FF FF）:磁头号:254; 扇区号:(11 1111)2=(63)10; 柱面号：(11 1111 1111)2=(1023)10

(FE)16=(254)10 ;(FF)16=(1111 1111)2;(FF)16=(1111 1111)2;

故该分区(C盘)结束扇区为（1023柱面，254磁头，63扇区）。

* (3F 00 00 00) : 反向，（00 00 00 3F) 16 = (63) 10，为该分区(C盘)起始逻辑扇区号与逻辑0扇区号之差。表示该分区（C盘）前面已有63个扇区，这63个扇区为系统隐藏扇区。
* (82 C1 3B 3A) : 反向，(3A 3B C1 82)16=(976994690)10。表明该分区(C盘)有976994690个扇区。即（0柱面，1磁头，1扇区）至（1023柱面，254磁头，63扇区）共有976994690个扇区。
* 1个扇区512字节，所以该分区(C盘)大小为976994690/2/1024/1024=465G



GB与B的换算关系：

1B(byte 字节)=8bit

1KB(Kilobyte 千字节)=1024B，

1MB(Megabyte 兆字节 简称“兆”)=1024KB，

1GB(Gigabyte 吉字节 又称“千兆”)=1024MB，

  
**第二项：**(00) (FE FF FF) (07) (FE FF FF) (00 C8 3B 3A) (00 88 FF 14)  
和第一项解析一样。第三项：  
(00) (FE FF FF)(0F) (FE FF FF) (00 50 3B 4F) (00 18 35 25)

* (0F) :表示该分区为扩展分区
* (FE FF FF) :磁头号:（254) 10；扇区号:(11 1111) 2 = (63) 10 ; 柱面号:(11 1111 1111) 2 = (1023) 10

(FE) 16 = (254) 10 (FF) 16 = (1111 1111) 2 (FF) 16 = (1111 1111) 2;

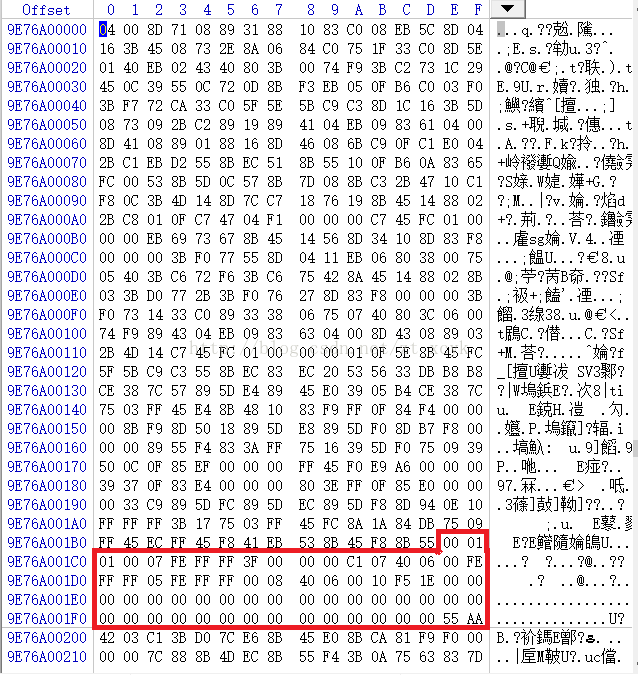
但这是不准确的，因为当柱面号的真实值超过1023时，表示柱面号的10位也依然是1023。

* (00 50 3B 4F) : 反向，(4F 3B 50 00)16=(1329287168)10,表示扩展分区共的起始扇区号为1329287168扇区。指向E盘的地址。
* (00 18 35 25): 反向，(25 35 18 00)16=(624236544)10,表示扩展分区有624236544个扇区。扩展分区的大小为624236544/2/1024/1024=297G，这是这个扩展分区的大小，里面又可以分无限分区。如图，我的扩展分区又分为E盘和G盘。

**第三项**

分区表链的查找

分区表链实际上相当于一个单向链表结构。第一个分区表，也即主分区表，可以有一项 描述扩展分区。而这一项就相当于指针，指向扩展分区。然后我们根据该指针来到扩展分 区起始柱面的0头1扇区，找到第二个分区表。对于该分区表，通常情况下：第一项描述了扩展分区中第一个分区的信息，第二项描述下一个分区，而这第二项就相当于指向第二个分 区的指针，第三项，第四项一般均为0。我们可以根据该指针来到扩展分区中第二个分区起 始柱面的0头1扇区，找到第三个分区表。以此类推，直到最后一个分区表。而最后一个分 区表只有第一项有信息，余下三项均为0.相当于其指针为空.所以只要找到了一个分区表就可 以推导找出其后面所有分区表。不过该分区表前面的分区表就不好推导出来了。但令人高兴 的是这个链表的头节点，也即主分区表的位置是固定的位于（0柱面，0磁头，1扇区）处， 我们可以很轻易的找到它，然后把剩下的所有分区表一一找到。

根据上面的信息，我们跳到1329287168扇区，这里有该分区的信息。如图

(00) (01 01 00) (07) (FE FF FF) (3F 00 00 00) (C1 07 40 06)

* (07)：说明是文件系统是NTFS。
* (C1 07 40 06)： 反向(06 40 07 C1)16=(104859585)10。改分区有104859585个扇区。大小为：104859585/2/1024/1024=50G。

对这个扇区的分析和上面类似，我们可以看到这个扩展分区里面只划分了两个分区，如果超过四个，需要继续链式查找。