# BIOS->MBR 和 UEFI->GPT

BIOS全称为:Basic Input Output System,基本输入输出系统

UEFI全称为: Unified Extensible Firmware Interface,统一可扩展的固件接口，用于取代BIOS

MBR全称为:Master Boot Record,主引导记录

GPT全称为:GUID Partion TABLE,全局唯一标识分区表，用于取代MBR

特殊说明:

一般将MBR分为广义和狭义两种：广义的MBR包含整个扇区(引导程序、分区表及分隔标识),也就是上面所说的主引导记录；而狭义的MBR仅指引导程序而言

BIOS文章: <http://www.w3cschool.cn/bioswqsc/ygleo7.html>

UEFI文章:<https://en.wikipedia.org/wiki/Unified_Extensible_Firmware_Interface>

MBR文章: <http://baike.baidu.com/subview/9485/9079238.htm>

GUID Partion Table文章:

中文:

<https://www.ibm.com/developerworks/cn/linux/l-gpt/>

<https://wiki.archlinux.org/index.php/GUID_Partition_Table_(%E7%AE%80%E4%BD%93%E4%B8%AD%E6%96%87)#GPT_.E7.9A.84.E4.BC.98.E7.82.B9>

英文:

<http://www.anchor.com.au/blog/2012/10/the-difference-between-booting-mbr-and-gpt-with-grub/>

BIOS与Boot loader的关系:

BIOS是位于主板ROM芯片上的固化程序，用于开机硬件设备的自检，设备的启动(寻找硬盘上MBR的Boot loader去加载内核)

Boot loader:位于硬盘MBR中，用于加载内核到内存

# FS加载顺序

Linux系统启动流程:<http://www.linux178.com/linux/linuxinit.html>

bootfs 🡪rootfs🡪diskfs

boot file system(bootfs）：包含 boot loader 和 kernel。用户不会修改这个文件系统。实际上boot完成之后，整个内核都会被boot loader加载到内存， bootfs 会被卸载掉从而释放出所占用的内存。同时也可以看出，对于同样内核版本的不同的 Linux 发行版的 bootfs 都是一致的。

rootfs:与特定linux发行版有关，与linux发行版相关的目录，如/dev, /proc, /bin, /etc, /lib, /usr, and /tmp和要运行用户应用的配置文件和二进制文件，库文件。在启动时为只读模式，启动后读写模式。

Diskfs:磁盘文件系统，如ext2,ext3,xfs等。

# 磁盘信息统计 && 目录信息统计

#df -h

*Filesystem Size Used Avail Use% Mounted on*

*/dev/mapper/centos-root 48G 13G 36G 26% /*

*devtmpfs 1.9G 0 1.9G 0% /dev*

*tmpfs 1.8G 0 1.8G 0% /dev/shm*

*tmpfs 1.8G 65M 1.7G 4% /run*

*tmpfs 1.8G 0 1.8G 0% /sys/fs/cgroup*

*/dev/xvda1 497M 165M 333M 34% /boot*

*tmpfs 354M 0 354M 0% /run/user/1000*

#cd 到某目录,查看该目录下的统计汇总信息，并按排序输出

#du -sh \* | sort -nr

*448K specs*

*328K releasenotes*

*228K doc*

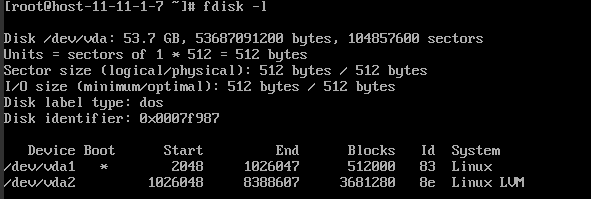
*144K kolla*

*104K tools*

# MBR格式的硬盘分区(硬盘容量<2T)

## 查看硬盘分区状况

#fdisk -l #同时也可以通过df -h 查看挂载点以及使用百分比情况 mount -l:挂载详情



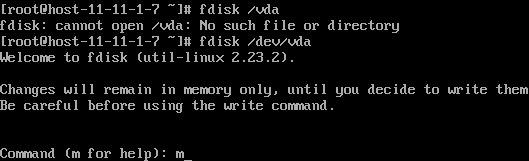
*结果解释:*

*/dev/vda1作为启动分区，从开始柱面2048到结束柱面2026047*

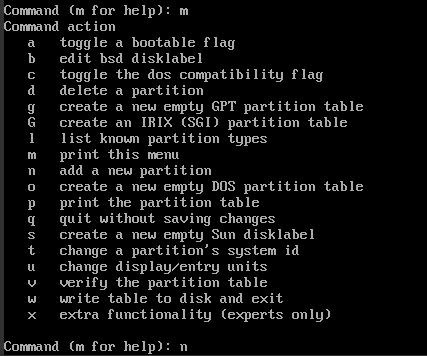
*从展示结果可以看出，有很多空间没有被利用，已用来分区*

## 建立扩展分区、逻辑分区

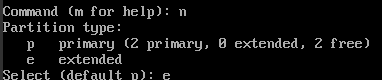
#fdisk /dev/vda



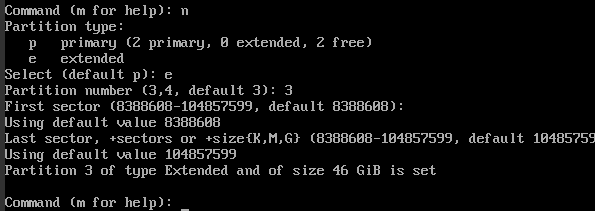
输入m,可以看到各种帮助



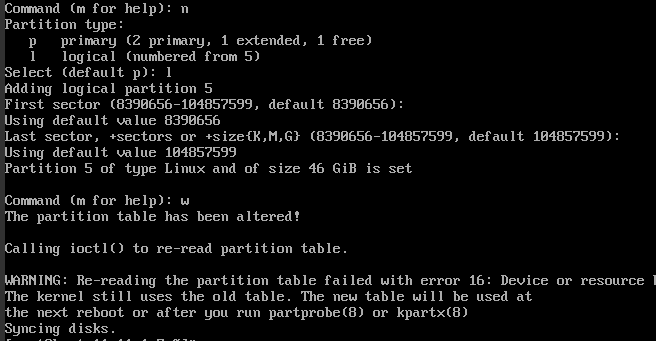
输入n,建分区



输入e,选择扩展分区，输入开始Sector和结束Sector



在扩展分区上建逻辑分区且将逻辑分区的systemid调整为8e，最后输入w保存



*备注:MBR分区的分区表只有64字节，而每个分区占用16字节，一般采用4主或3主+1扩展的分区结构*

## 分区表立即生效

#partprobe

## 物理化(建立PV)

#pvcreate /dev/vda5

*备注:里面涉及到LVM的相关知识，请参考:*

*http://baike.baidu.com/link?url=OnwEJuKHzieixWjzsWyUBsKi0-xKADDZVNnPSr1uSTy4Pf8fTPSolpLpPKRCIQk3WrVT8qRlnzVK3eTtPSwJmK*

## 生成卷组(建立VG)

#vgcreate appvg /dev/vda5

*其他相关命令:*

*在VG上追加PV:#vgextend appvg /dev/vda5*

## 在卷组上划分逻辑卷(建立LV)

#lvcreate -L 15G -n app appvg #app为lv,appvg为卷组

#lvcreate -L 15G -n applog appvg

#lvcreate -L 15G -n data appvg

*其他相关命令:*

*在LV上扩展容量(以下两种方式能扩展，前者表示增加，后者表示增加后，是调整后的容量)*

*#lvextend -L +2G /dev/mapper/centos-root #与lvextend相反的是lvreduce，如:*

#*lvreduce -L -5G /dev/mapper/centos-root*

*#lvresize -L 5G /dev/mapper/centos-root*

*如果是针对xfs的文件系统，为了避免lvextend之后，通过df -h查看不了文件系统变化，需要执行:xfs\_growfs /dev/mapper/centos-root（有的资料也提供这种方法：resize2fs -p /dev/mapper/centos-root）*

## 为逻辑卷建立文件系统

#mkfs.ext3 /dev/mapper/appvg-app

#mkfs.ext3 /dev/mapper/appvg-applog

#mkfs.ext3 /dev/mapper/appvg-data

## 挂载逻辑卷

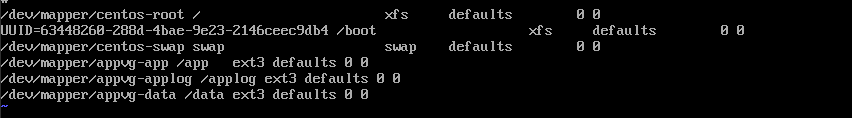
#mount /dev/mapper/appvg-app /app

#mount /dev/mapper/appvg-applog /applog

#mount /dev/mapper/appvg-data /data

## 挂载持久化

将挂载信息写入/etc/fstab，执行mount -a重新挂载(否则重启后挂载信息将丢失)



*备注:以后挂载能够使用uuid的，就使用uuid*

# GPT格式的硬盘分区(硬盘容量>=2T)

参考:

<http://www.cnblogs.com/zhangpengme/archive/2011/12/29/2305963.html>

<http://man.linuxde.net/parted>

## 查看硬盘情况

#parted /dev/xvdb

*GNU Parted 3.1*

*Using /dev/xvdb*

*Welcome to GNU Parted! Type 'help' to view a list of commands.*

*(parted) print*

*Model: Xen Virtual Block Device (xvd)*

*Disk /dev/xvdb: 53.7GB*

*Sector size (logical/physical): 512B/512B*

*Partition Table: gpt*

*Disk Flags:*

*Number Start End Size File system Name Flags*

*(parted)*

## 创建分区表(谨慎)

(parted) mklabel

New disk label type? gpt

*Warning: The existing disk label on /dev/xvdb will be destroyed and all data on this disk will be lost. Do you want to continue?*

Yes/No? yes

## 创建分区

(parted) mkpart

Partition name? []? myp1

File system type? [ext2]? ext3

Start? 0

End? 20G

*Warning: The resulting partition is not properly aligned for best performance.*

Ignore/Cancel? Ignore

(parted) mkpart

Partition name? []? myp2

File system type? [ext2]? ext3

Start? 20G

End? 25G

(parted) print

*Model: Xen Virtual Block Device (xvd)*

*Disk /dev/xvdb: 53.7GB*

*Sector size (logical/physical): 512B/512B*

*Partition Table: gpt*

*Disk Flags:*

*Number Start End Size File system Name Flags*

*1 17.4kB 20.0GB 20.0GB myp1*

*2 20.0GB 25.0GB 5000MB myp2*

## 删除分区

(parted) rm 2

(parted) print

*Model: Xen Virtual Block Device (xvd)*

*Disk /dev/xvdb: 53.7GB*

*Sector size (logical/physical): 512B/512B*

*Partition Table: gpt*

*Disk Flags:*

*Number Start End Size File system Name Flags*

*1 17.4kB 20.0GB 20.0GB myp1*

## 为分区建立文件系统

#mkfs.ext3 /dev/xvdb1

*mke2fs 1.42.9 (28-Dec-2013)*

*warning: 376 blocks unused.*

*Filesystem label=*

*OS type: Linux*

*Block size=4096 (log=2)*

*Fragment size=4096 (log=2)*

*Stride=0 blocks, Stripe width=0 blocks*

*1222992 inodes, 4882432 blocks*

*244121 blocks (5.00%) reserved for the super user*

*First data block=0*

*Maximum filesystem blocks=4294967296*

*149 block groups*

*32768 blocks per group, 32768 fragments per group*

*8208 inodes per group*

*Superblock backups stored on blocks:*

*32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,*

*4096000*

*Allocating group tables: done*

*Writing inode tables: done*

*Creating journal (32768 blocks): done*

*Writing superblocks and filesystem accounting information: done*

## 挂载分区到文件夹

#mkdir /mnt1

#mount /dev/xvdb1 /mnt1

#df -h

*Filesystem Size Used Avail Use% Mounted on*

*…*

*/dev/xvdb1 19G 45M 18G 1% /mnt1*

## 卸载

#umount /mnt1