**VMware虚拟机添加硬盘**

[root@cloud002 ~]# grep mpt /sys/class/scsi_host/host?/proc_name

/sys/class/scsi_host/host2/proc_name:mptspi

[root@cloud002 ~]# echo "- - -" > /sys/class/scsi_host/host2/scan

[root@cloud002 ~]# fdisk -l

# 磁盘及文件系统

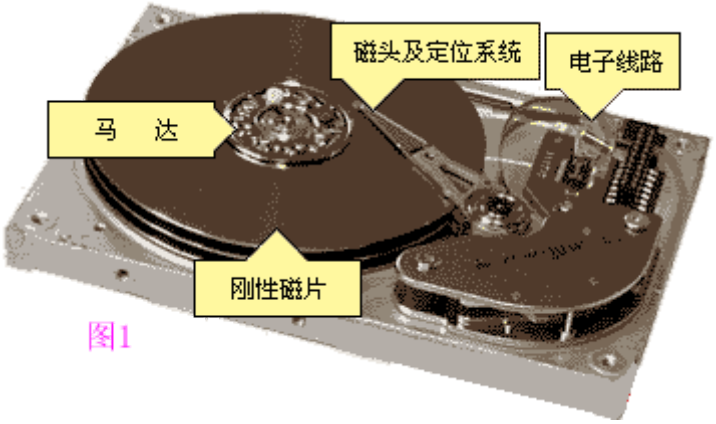
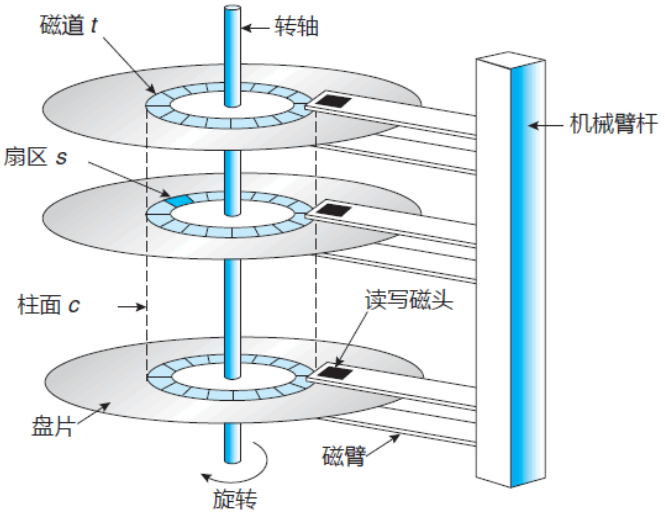



图1

硬盘的物理构成：

扇区（sector）、磁道（track）、磁头、柱面(cylinder)




分区：就是记录每一个分区的起始柱面和结束柱面。分区信息存放在0柱面0磁道1扇区上：MBR 主引导记录(446字节)+DPT磁盘分区表（64字节）+结束标志（2个字节）

硬盘的主引导记录（MBR）是不属于任何一个操作系统的，它先于所有的操作系统而被调入内存，并发挥作用，然后才将控制权交给主分区（活动分区）内的操作系统，并用主分区信息表来管理硬盘。

文件系统是操作系统用于明确存储设备（常见的是磁盘）或分区上的文件的方法和数据结构；即在存储设备上组织文件的方法。

block： 数据存储的最小单元

inode:索引节点，全局唯一编号，除了记录文件的属性外，同时还具有指针功能，指向文件内容放置的块

（里面保存的是文件的权限，所有者，所属主等基本信息）

1、文件的拥有者与用户组（owner/group）

2、文件的访问模式（read/write/excute）

3、文件的类型（type）

4、文件建立或状态改变的时间（ctime)、最近一次的读取时间（atime）、最近修改时间（mtime）

5、文件的大小

6、定义文件属性的标志（flag），如setUID...

7、文件真正内容的指针（pointer）

创建目录：分配一个inode和至少一个block

inode记录该目录的相关属性，并指向分配到的那个快；

block记录在这个目录下的相关文件（或目录）的关联性

创建文件：分配至少一个inode与相对于该文件大小的快数量

inode不记录文件名，而是记录文件的相关属性，文件名记录在目录所属的block区域

例如：读取/etc/crontab的流程如下：

1、操作系统根据根目录（/）的相关数据可获取/etc目录所在的inode，并读取/etc这个目录所有相关属性

2、根据/etc的inode的数据，可以获取/etc目录下所有文件的关联数据是放置在哪一个block中，并前往该block读取文件的关联性内容

3、由上步骤的block中，可以知道crontab文件的inode所在地，并前往该inode

4、由上步骤的inode中，可以获取crontab文件的所有属性，并前往由inode所指向的block区域，获取crontab文件内容

块组：

文件系统构成：（dumpe2fs 可以查看）

**superblock**(超级块)：记录整个文件系统相关信息

1、block与inode的总量

2、未使用和已使用的inode/block数量

3、文件系统的载入时间、最近一次写入数据的时间、最近一次检验磁盘（fsck）的时间等文件系统的相关信息

4、有效位的值，已载入为0，未载入为1

**block bitmap**(块位图)：此处记录block是否使用

在向硬盘存储数据时，文件系统需要知道哪些块是空闲的，哪些块是已经占用了的，位图只使用0和1标识对应block是空闲还是被占用，0和1在位图中的位置和block的位置一一对应，第一位标识第一个块，第二个位标识第二个块，依次下去直到标记完所有的block。1G的文件只需要128个block做位图就能完成一一对应。通过扫描这100多个block就能知道哪些block是空闲的，速度提高了非常多。但是要注意，bmap的优化针对的是写优化，因为只有写才需要找到空闲block并分配空闲block。对于读而言，只要通过inode找到了block的位置，cpu就能迅速计算出block在物理磁盘上的地址

**inode bitmap**(inode位图)：此处记录inode是否使用

标识inode号是否被分配的位图称为inode bitmap简称为imap。这时要为一个文件分配inode号只需扫描imap即可知道哪一个inode号是空闲的。

**inode table**(inode表)：为每个inode的数据存放区

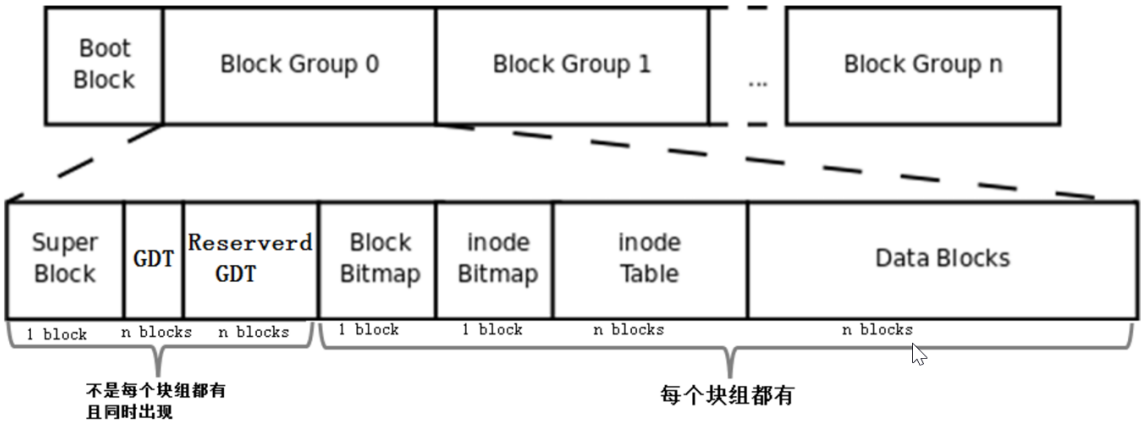
一个文件系统中可以说有无数多个文件，每一个文件都对应一个inode，难道每一个仅128字节的inode都要单独占用一个block进行存储吗？更优的方法是将多个inode合并存储在block中，对于128字节的inode，一个block存储8个inode，对于256字节的inode，一个block存储4个inode。这就使得每个存储inode的块都不浪费。在ext文件系统上，将这些物理上存储inode的block组合起来，在逻辑上形成一张inode表(inode table)来记录所有的inode。

**block group** 对用户是不可见的，子逻辑区域，有个超级标示区

为了解决bmap、inode table和imap太大的问题，引入块组概念，在物理层面上的划分是将磁盘按柱面划分为多个分区，即多个文件系统；在逻辑层面上的划分是将文件系统划分成块组。每个文件系统包含多个块组，每个块组包含多个元数据区和数据区：元数据区就是存储bmap、inode table、imap等的数据；数据区就是存储文件数据的区域。

**data block**(数据块)：为每个block的数据存放区

完整的文件系统图：




**boot block**部分，也称为boot sector。它位于分区上的第一个块，占用1024字节，并非所有分区都有这个boot sector，只有装了操作系统的主分区和装了操作系统的逻辑分区才有。里面存放的也是boot loader，这段boot loader称为VBR(主分区装操作系统时)或

EBR(扩展分区装操作系统时)，这里的Boot loader和mbr上的boot loader是存在交错关系的。开机启动的时候，首先加载mbr中的bootloader，然后定位到操作系统所在分区的boot serctor上加载此处的boot loader。

**group description**(组描述)：

既然文件系统划分了块组，那么每个块组的信息和属性元数据又保存在哪里呢？ext文件系统为每一个块组信息使用32字节描述，这32个字节称为块组描述符，所有块组的块组描述符组成块组描述符表GDT(group descriptor table)。但是不是每个块组中都存放了块组描述符。ext文件系统的存储方式是：将它们组成一个GDT，并将该GDT存放于某些块组中，存放GDT的块组和存放superblock和备份superblock的块相同，也就是说它们是同时出现在某一个块组中的。读取时也总是读取Group0中的块组描述符表信息。

创建文件：先在目录下创建一个条目，把名字和iNode对应上，回到源数据，指定块位置

删除文件：删除条目里面的名字和iNode对应关系，将对应的iNode标记为0

通过inode号码来删除文件：find ./ -inum 400935 -exec rm -rf {} \;

extundelete： linux下高效数据恢复工具 http://extundelete.sourceforge.net/ 安装时需要的依赖包 e2fsprogs-lib e2fsprogs-devel

复制文件：不同的inode和block

剪切文件：同一个分区，inode号和磁盘块位置都没有变，只是把条目里面的名字改变了一下

链接：硬链接和软链接（符号链接），多个文件指向同一个iNode，硬链接；符号连接，指向的是源文件的路径，而不是文件块

ln (link) [-s -v] 源文件 链接

硬链接：在某个目录下的block中增加一个文件关联数据，不会用到inode与磁盘空间

只能对文件进行创建，为了避免循环引用

不能跨文件系统，可以在不同目录下

创建硬链接可以增加文件被连接的次数

符号链接：建立一个独立的文件，这个文件会让数据的读取指向它连接的文件内容

可以对目录创建

可以跨文件系统

不会增加被链接文件的链接次数

其大小为指定路径所包含的字符个数

-b 删除，覆盖以前建立的链接

-d 允许超级用户制作目录的硬链接

-f 强制执行

-i 交互模式，文件存在则提示用户是否覆盖

-n 把符号链接视为一般目录

-s 软链接(符号链接)

-v 显示详细的处理过程

du 查看文件大小 [-s -h ]

-s 只分别计算命令列中每个参数所占的总用量

-h 做单位转换

df 查看硬盘使用情况

-i 查看inode的使用情况

-P 不换行显示

-h 做单位转换

-T 显示文件系统

fdisk 查看当前系统识别的硬盘和分区的情况

fdisk -l [/dev/to/smoe_device_file]

还可用来管理分区

fdisk /dev/sda

p:显示当前硬件的分区，包括没有保存的改动

n:创建一个新的分区

d：删除一个分区

w:保存并退出

q:不保存退出

t：修改分区类型（文件系统类型）

l:显示所支持的所有类型

cat /proc/partitions 查看分区

partprobe [/dev/sda] 通知操作系统分区表的变化

partx -a [/dev/sda] 告诉内核磁盘分区是否存在及其编号

cat /proc/filesystem 查看文件系统类型

高级格式化：mkfs -t ext4

mkfs -t msdos /dev/hdb3

mkfs.ext2 、mkfs.ext3 、mke2fs

-b：分区时设定每个数据区块占用空间大小，目前支持1024, 2048 以及4096 bytes每个块。

-i：设定inode大小

-N：设定inode数量，有时使用默认的inode数不够用，所以要自定设定inode数量。

-c：在格式化前先检测一下磁盘是否有问题，加上这个选项后会非常慢

-L：预设该分区的标签label

-j：建立ext3格式的分区，如果使用mkfs.ext3 就不用加这个选项了

查看磁盘超级块的信息：

tune2fs -l /dev/sda[n]　　调整ext2/ext3/ext4文件系统参数

dumpe2fs -h /dev/sda[n]　显示ext2/ext3/ext4文件系统信息

二.swap

swap分区：主要是用来内存过载使用，可以将内存上暂停进程的页面保存在swap分区中，再次启用时再分配页面。分区在系统的物理内存不够用的时候，把物理内存中的一部分空间释放出来，以供当前运行的程序使用。那些被释放的空间可能来自一些很长时间没有什么操作的程序，这些被释放的空间被临时保存到Swap分区中，等到那些程序要运行时，再从Swap分区中恢复保存的数据到内存中。

free　查看物理内存和交换空间使用情况

下面我们来看看swap分区如何手动创建。

有两种方式可以使用：

第一种单独用一个分区来作为swap

```
[root@server1 ~]# cat /proc/swaps
Filename                                Type            Size    Used    Priority
/dev/dm-1                               partition       4194300 0       -1
```

创建一个分区（如：/dev/sdb3），并更改分区ID为82

执行partx –a /dev/sdb命令，使分区修改生效

在分区上创建swap文件系统

```
[root@server1 ~]# mkswap /dev/sdb3
Setting up swapspace version 1, size = 10485756 KiB
no label, UUID=f8ef520f-8b61-4331-952c-bd9bbd37957c
[root@server1 ~]# swapon /dev/sdb3
[root@server1 ~]# cat /proc/swaps
Filename                                Type            Size    Used    Priority
/dev/dm-1                               partition       4194300 0       -1
/dev/sdb3                               partition       10485756        0       -2
```

修改fstab实现自动加载

```
/dev/sdb3               swap                    swap    defaults        0 0
```

第二种方式是创建一个文件块，这个文件所占有的空间作为swap使用

```
[root@server1 ~]# dd if=/dev/zero of=/swapfile bs=1M count=2048
2048+0 records in
2048+0 records out
2147483648 bytes (2.1 GB) copied, 11.637 s, 185 MB/s
[root@server1 ~]# mkswap /swapfile
Setting up swapspace version 1, size = 2097148 KiB
no label, UUID=be824cdc-8ed2-4bcd-8e5c-197f1f4a6417
[root@server1 ~]# ls -l /swapfile
-rw-r--r--. 1 root root 2147483648 May 11 10:58 /swapfile
[root@server1 ~]# chmod 600 /swapfile
[root@server1 ~]# swapon /swapfile
[root@server1 ~]# cat /proc/swaps
Filename                                Type            Size    Used    Priority
/dev/sdb3                               partition       10485756        0       -1
/dev/dm-1                               partition       4194300 0       -2
/swapfile                               file            2097148 0       -3
[root@server1 ~]#
```

mkswap 格式化为swap分区

1、通过fdisk创建swap分区或者使用dd命令创建一个虚拟内存的文件

dd if=/dev/zero of=/tmp/swapfile  bs=1M count=1024

2、mkswap进行swap分区格式化

3、swapon 激活swap分区

swapoff 关闭swap分区

mount：挂载命令

命令格式：mount [-t vfstype] [-o options] device dir

-a 挂载fstab中提到的所有文件系统

-o remount 重新挂载已挂载的文件系统

-o loop：用来把一个文件当成硬盘分区挂接上系统

-o ro：采用只读方式挂接设备

-o rw：采用读写方式挂接设备

挂接光盘镜像文件

1、从光盘制作光盘镜像文件。将光盘放入光驱，执行下面的命令。

#cp /dev/cdrom /home/sunky/mydisk.iso 或

#dd if=/dev/cdrom of=/home/sunky/mydisk.iso

2、挂载光盘镜像文件

#mount -o loop -t iso9660 /home/sunky/mydisk.iso /mnt/vcdrom

三.自动挂载

文件系统的配置文件/etc/fstab

OS在初始时，会自动挂载此文件中定义的每个文件系统

/dev/sda5　　/mnt/test　　ext4　　defaults　　0　0

mount -a：挂载/etc/fstab文件中定义的所有文件系统

/etc/fstab文件　实现开机自动挂载

设备 挂载点 文件系统类型 挂载参数 是否执行备份命令dump 是否启动使用fsck扫描磁盘

fuser: 验证进程正在使用的文件或套接字文件

-v: 查看某文件上正在运行的进程

-k:

-m：针对目录，也就是挂载点，一般和-k 一起使用

fuser -km MOUNT_POINT：终止正在访问此挂载点的所有进程

wodim --devices 查看设备名（没有挂载的光盘）

练习：

1、创建一个5G的分区，文件系统为ext4，卷标为MYDATA，块大小为1024，预留管理空间为磁盘分区的3%，要求开机后可以自动挂载至/data目录，并且自动挂载的设备要使用卷标进行引用；

2、创建一个本地回环文件/var/swaptemp/swapfile来用于swap，要求大小为512MB，卷标为SWAP-FILE，且开机自动启用此交换设备；

# mkdir /var/swaptemp

# dd if=/dev/zero of=/var/swaptemp/swapfile bs=1M count=512

# mkswap LABLE=SWAP-FILE /var/swaptemp/swapfile

/etc/fstab

/var/swaptemp/swapfile swap swap defaults 0 0

3、上述第一问，如何让其自动挂载的同时启用ACL功能；

/etc/fstab

LABEL='MYDATA' /data ext4 defaults,acl 0 0

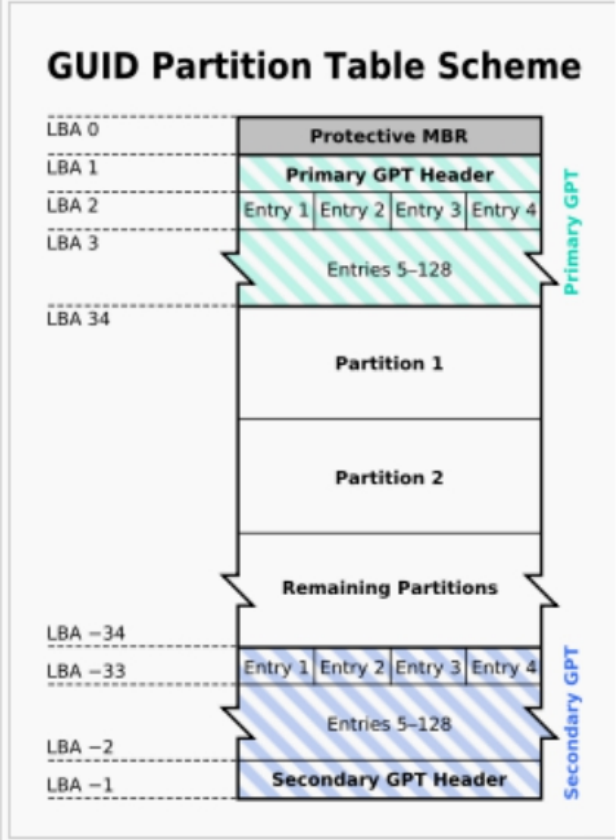
传统的硬盘分区都是MBR格式，MBR分区位于0扇区，他一共512字节，前446字节是grub引导程序，中间64字节是分区表，每个分区需要16个字节表示，因此主分区和扩展分区一共只能有4个分区，超过4个的分区只能从扩展分区上再设置逻辑分区来表示。每个分区的大小无法超过2T。 MBR的最后2个字节是结束符号。

GPT格式，打破了MBR的限制，可以设置多达128个分区，分区的大小根据操作系统的不同有所变化，但是都突破了2T空间的限制。支持高达 18EB (1EB=1024PB, 1PB=1024TB) 的卷大小，允许将主磁盘分区表和备份磁盘分区表用于冗余，还支持唯一的

磁盘和分区 ID (GUID)。与 MBR 分区的磁盘不同，GPT的分区信息是在分区中，而不象MBR一样在主引导扇区。为保护GPT不受MBR类磁盘管理软件的危害，GPT在主引导扇区建立了一个保护分区 (Protective MBR)的MBR分区表，这种分区的类型标识为0xEE，这个保护分区的大小在Windows下为128MB，Mac OS X下为200MB，在Window磁盘管理器里名为GPT保护分区，可让MBR类磁盘管理软件把GPT看成一个未知格式的分区，而不是错误地当成一个未分区的磁盘

在MBR硬盘中，分区信息直接存储于主引导记录(MBR)中（主引导记录中还存储着系统的引导程序）。但在GPT硬盘中，分区表的位置信息储存在GPT头中。但出于兼容性考虑，硬盘的第一个扇区仍然用作MBR，之后才是GPT头。

GPT的结构如下图：




GPT分区表的结构。此例中，每个逻辑块（LBA）为512字节，每个分区的记录为128字节。负数的LBA地址表示从最后的块开始倒数，-1表示最后一个块。

gdisk和fdisk非常类似

```
[root@server1 ~]# gdisk
GPT fdisk (gdisk) version 0.8.6

Type device filename, or press <Enter> to exit:
```

```
[root@server1 ~]# gdisk /dev/sdb
GPT fdisk (gdisk) version 0.8.6

Partition table scan:
  MBR: MBR only
  BSD: not present
  APM: not present
  GPT: not present


********************************************************************
Found invalid GPT and valid MBR; converting MBR to GPT format.
THIS OPERATION IS POTENTIALLY DESTRUCTIVE! Exit by typing 'q' if
you don't want to convert your MBR partitions to GPT format!
********************************************************************


Command (? for help): ?
b       back up GPT data to a file
c       change a partition's name
d       delete a partition
i       show detailed information on a partition
l       list known partition types
n       add a new partition
o       create a new empty GUID partition table (GPT)
p       print the partition table
q       quit without saving changes
r       recovery and transformation options (experts only)
s       sort partitions
t       change a partition's type code
v       verify disk
w       write table to disk and exit
x       extra functionality (experts only)
?       print this menu
```

n 添加一个新分区，创建新分区的时候可以看见可以有128个分区

```
Command (? for help): n
Partition number (1-128, default 1): 1
First sector (34-209715166, default = 2048) or {+-}size{KMGTP}:
Last sector (2048-209715166, default = 209715166) or {+-}size{KMGTP}: +10G
Current type is 'Linux filesystem'
Hex code or GUID (L to show codes, Enter = 8300):
Changed type of partition to 'Linux filesystem'

Command (? for help): p
Disk /dev/sdb: 209715200 sectors, 100.0 GiB
Logical sector size: 512 bytes
Disk identifier (GUID): 1E2EC40C-75E9-467D-B79B-CE8A4CBA64B9
Partition table holds up to 128 entries
First usable sector is 34, last usable sector is 209715166
Partitions will be aligned on 2048-sector boundaries
Total free space is 188743613 sectors (90.0 GiB)

Number  Start (sector)    End (sector)  Size       Code  Name
   1            2048        20973567   10.0 GiB    8300  Linux filesystem
```

parted，和前两个相比，更灵活，可以自行设定MBR或GPT格式和分区

```
[root@server1 ~]# parted /dev/sdb
GNU Parted 3.1
Using /dev/sdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) help
  align-check TYPE N                        check partition N for TYPE(min|opt) alignment
  help [COMMAND]                           print general help, or help on COMMAND
  mklabel,mktable LABEL-TYPE               create a new disklabel (partition table)
  mkpart PART-TYPE [FS-TYPE] START END     make a partition
  name NUMBER NAME                         name partition NUMBER as NAME
  print [devices|free|list,all|NUMBER]     display the partition table, available devices, free
        space, all found partitions, or a particular partition
  quit                                     exit program
  rescue START END                         rescue a lost partition near START and END
  rm NUMBER                                delete partition NUMBER
  select DEVICE                            choose the device to edit
  disk_set FLAG STATE                      change the FLAG on selected device
  disk_toggle [FLAG]                       toggle the state of FLAG on selected device
  set NUMBER FLAG STATE                    change the FLAG on partition NUMBER
  toggle [NUMBER [FLAG]]                   toggle the state of FLAG on partition NUMBER
  unit UNIT                                set the default unit to UNIT
  version                                  display the version number and copyright information of
        GNU Parted
(parted)
```

通过mklabel msdos可以设定为MBR格式，然后可以通过mkpart来划分分区

```
[root@server1 ~]# parted /dev/sdb
GNU Parted 3.1
Using /dev/sdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) mklabel msdos
Warning: The existing disk label on /dev/sdb will be destroyed and all data on this disk will be
lost. Do you want to continue?
Yes/No? yes
(parted) mkpart primary 0 10G
Warning: The resulting partition is not properly aligned for best performance.
Ignore/Cancel? I
(parted) p
Model: VMware, VMware Virtual S (scsi)
Disk /dev/sdb: 107GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start  End     Size    Type     File system  Flags
 1      512B   10.0GB  10.0GB  primary

(parted) set 1 lvm on
(parted) p
Model: VMware, VMware Virtual S (scsi)
Disk /dev/sdb: 107GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start  End     Size    Type     File system  Flags
 1      512B   10.0GB  10.0GB  primary              lvm

(parted)
```

msdos设定为MBR格式，gpt设定为GPT格式

primary代表主分区，extended代表扩展分区，logical代表逻辑分区。

set number flag state用于设置分区的用途，flag：boot、lvm、raid。State:on/off表示开启或关闭。

parted工具分完区后无需保存，输入q退出即可。

划分好分区之后，还需要格式化才能使用。可以通过 mkfs/mkswap来格式化文件系统

#mkfs.xfs /dev/分区设备名或#mkfs -t xfs /dev/分区设备名

```
[root@server1 ~]# mkfs -t xfs /dev/sdb1
meta-data=/dev/sdb1              isize=256    agcount=4, agsize=610352 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=0
data     =                       bsize=4096   blocks=2441406, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0 ftype=0
log      =internal log           bsize=4096   blocks=2560, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
```

可以修改fstab实现自动加载

```
#
# /etc/fstab
# Created by anaconda on Wed Jan 21 09:45:42 2015
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
/dev/mapper/rhel-root   /                       xfs     defaults        1 1
UUID=8903fe24-7c0d-42bd-bac1-5cc35773de2b /boot                   xfs     defaults        1 2
/dev/mapper/rhel-swap   swap                    swap    defaults        0 0
/dev/sdb1               /data                   xfs     defaults        0 0
```

测试是否能自动挂载

```
[root@server1 ~]# mount -a
[root@server1 ~]# df -h
Filesystem              Size  Used Avail Use% Mounted on
/dev/mapper/rhel-root    36G  3.2G   33G   9% /
devtmpfs                485M     0  485M   0% /dev
tmpfs                   494M     0  494M   0% /dev/shm
tmpfs                   494M  7.0M  487M   2% /run
tmpfs                   494M     0  494M   0% /sys/fs/cgroup
/dev/sda1               497M  119M  379M  24% /boot
/dev/sdb1               9.4G   33M  9.3G   1% /data
```

通过df -h查看已经挂载了的设备

-T选项可以显示设备的文件系统类

有的挂载点路径比较长，自动分2行显示，可以-P强制一行显示

如同进程有pid，用户有uid，每个文件系统也有自己的id，称为uuid，但是不是每个分区都有；如果某个分区没有文件系统，那么这个分区是没有uuid的。

可以通过 blkid（block id）来查看。注意uuid标记的是文件系统，而不是分区。uuid的好处在于可以通过uuid这个唯一值来挂载系统，这样可以避免因为删除硬盘造成的错位，sda6变成了sda5等等

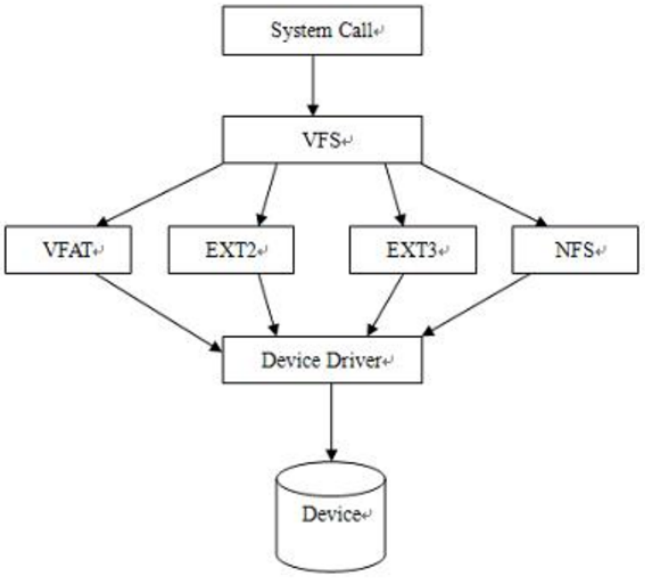
```
[root@server1 ~]# blkid
[ 5423.440343] end_request: I/O error, dev fd0, sector 0
/dev/sda1: UUID="8903fe24-7c0d-42bd-bac1-5cc35773de2b" TYPE="xfs"
/dev/sda2: UUID="YL1LDj-4PZv-q6b5-T7q8-oVCb-25zx-ePUk8w" TYPE="LVM2_member"
/dev/sdb1: UUID="4c83d212-3206-4fcc-8196-831ccc3d00be" TYPE="xfs"
/dev/sdb5: UUID="8e67bfd8-2594-41c6-a227-ea0aa15a35d0" TYPE="xfs"
/dev/sr0: UUID="2014-05-07-03-58-46-00" LABEL="RHEL-7.0 Server.x86_64" TYPE="iso9660" PTTYPE="dos"
/dev/mapper/rhel-root: UUID="fecc9c90-09ae-48fa-a8c9-2999d981fdeb" TYPE="xfs"
/dev/mapper/rhel-swap: UUID="61c24700-31d5-4325-9857-376fbea21c85" TYPE="swap"
```

我们可以通过xfs_admin -U 来手动更改文件系统的uuid

```
[root@server1 ~]# umount /dev/sdb1
[root@server1 ~]# uuidgen
e5db4010-59a8-4d0e-8d31-ff232ed886f9
[root@server1 ~]# xfs_admin -U e5db4010-59a8-4d0e-8d31-ff232ed886f9 /dev/sdb1
Clearing log and setting UUID
writing all SBs
new UUID = e5db4010-59a8-4d0e-8d31-ff232ed886f9
[root@server1 ~]# mount -a
[root@server1 ~]# blkid
[ 5597.723377] end_request: I/O error, dev fd0, sector 0
/dev/sda1: UUID="8903fe24-7c0d-42bd-bac1-5cc35773de2b" TYPE="xfs"
/dev/sda2: UUID="YL1LDj-4PZv-q6b5-T7q8-oVCb-25zx-ePUk8w" TYPE="LVM2_member"
/dev/sdb1: UUID="e5db4010-59a8-4d0e-8d31-ff232ed886f9" TYPE="xfs"
/dev/sdb5: UUID="8e67bfd8-2594-41c6-a227-ea0aa15a35d0" TYPE="xfs"
/dev/sr0: UUID="2014-05-07-03-58-46-00" LABEL="RHEL-7.0 Server.x86_64" TYPE="iso9660" PTTYPE="dos"
/dev/mapper/rhel-root: UUID="fecc9c90-09ae-48fa-a8c9-2999d981fdeb" TYPE="xfs"
/dev/mapper/rhel-swap: UUID="61c24700-31d5-4325-9857-376fbea21c85" TYPE="swap"
```

总体上说 Linux 下的文件系统主要可分为三大块：一是上层的文件系统的系统调用，二是虚拟文件系统 VFS(Virtual Filesystem Switch)，三是挂载到 VFS 中的各实际文件系统，例如 ext2，jffs 等。

VFS在内核中与其他的内核模块的协同关系：




VFS:virtual filesystem ,interface:system call

当前，除了Linux标准的文件系统Ext2/Ext3/Ext4外，还有很多种文件系统，比如reiserfs, xfs, Windows的vfat NTFS，网络文件系统nfs 以及flash 文件系统jffs2, yaffs/yaffs2 ubifs。linux通过叫做VFS的中间层对这些文件系统提供了完美的支持。

VFS存在的意义

1. 向上，对应用层提供一个标准的文件操作接口；

2. 对下，对文件系统提供一个标准的接口，以便其他操作系统的文件系统可以方便的移植到Linux上；

3. VFS内部则通过一系列高效的管理机制，比如inode cache, dentry cache 以及文件系统的预读等技术， 使得底层文件系统不需沉溺到复杂的内核操作，即可获得高性能；

4. 此外VFS把一些复杂的操作尽量抽象到VFS内部，使得底层文件系统实现更简单

VFS 是一种软件机制，也许称它为 Linux 的文件系统管理者更确切点，与它相关的数据结构只存在于物理内存当中。所以在每次系统初始化期间，Linux 都首先要在内存当中构造一棵 VFS 的目录树(在 Linux 的源代码里称之为 namespace)，实际上便是在内存中建立相应的数据结构。