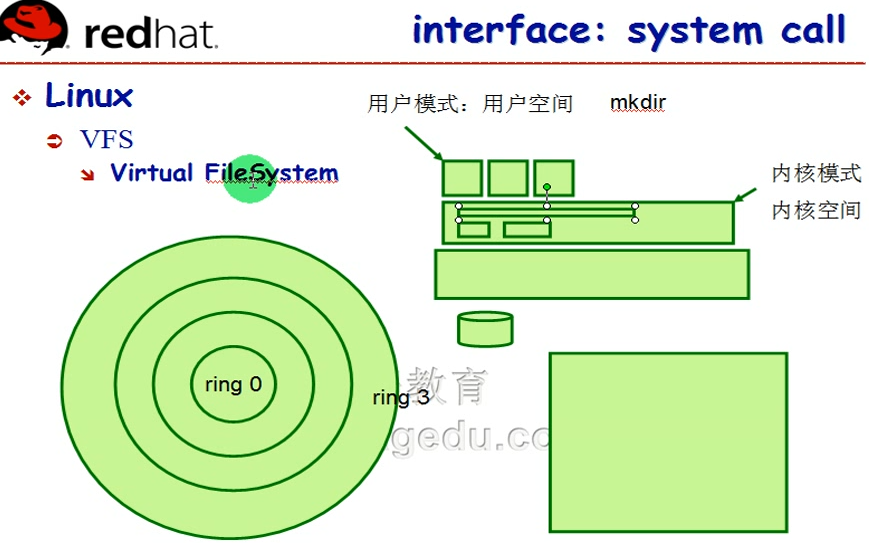
linux 基础

## 程序运行模式空间



程序运行模式

用户模式 用户空间

系统模式 系统空间

CPU 有4个级别

ring0 运行系统进程 享有特权的操作指令,操作硬件的指令,比如IO

ring3 运行用户进程 没有特权,不能跟硬件设备打交道, 如果有需要操作硬件的指令,必须转入ring0 进行system call.

内核模式/内核空间

文件系统管理就工作在内核空间,是内核应该具备的功能, 都需要同硬件设备打交道的.

操作文件 : **open close read write delete create** 这些动作都是内核的文件系统管理的功能,需要同硬件设备打交道的,内核向上提供了这些动作的功能接口或者功能函数,system call

然而不同厂商开发的文件系统不尽相同,对于上层的调用开发者来说就困难了,所以VFS诞生了,屏蔽了底层的多种文件管理系统,向上提供统一的调用接口,传入不同的参数即可,用起来就方便多了. 所以VFS也是内核种的功能

## ext2

元数据是存储在inode中的, 数据块存储数据. 一个数据块的大小通常 1024,2048,4096 反正是2的n次方. 1k 2k 4k 这是因为内存分页 页框一次装载是2的n次方

使用1K的好 还是2k的好 还是4K的好?怎么存最合理 ?

要看单个存储文件的大小,如果有许许多多的小于1k的文件,那么用1k吧,如果使用2k的块 ,那么2k的块就只能利用1k ,那1k就不被利用浪费

**一般文件占用空间比文件本身大小要大,因为总会存在浪费的地方.**

**磁盘块 - 文件管理系统对磁盘的逻辑划分**

符合链接一般不需要磁盘块

设备文件 不需要磁盘块

套接字文件 管道文件 设备文件 都是没有大小的.

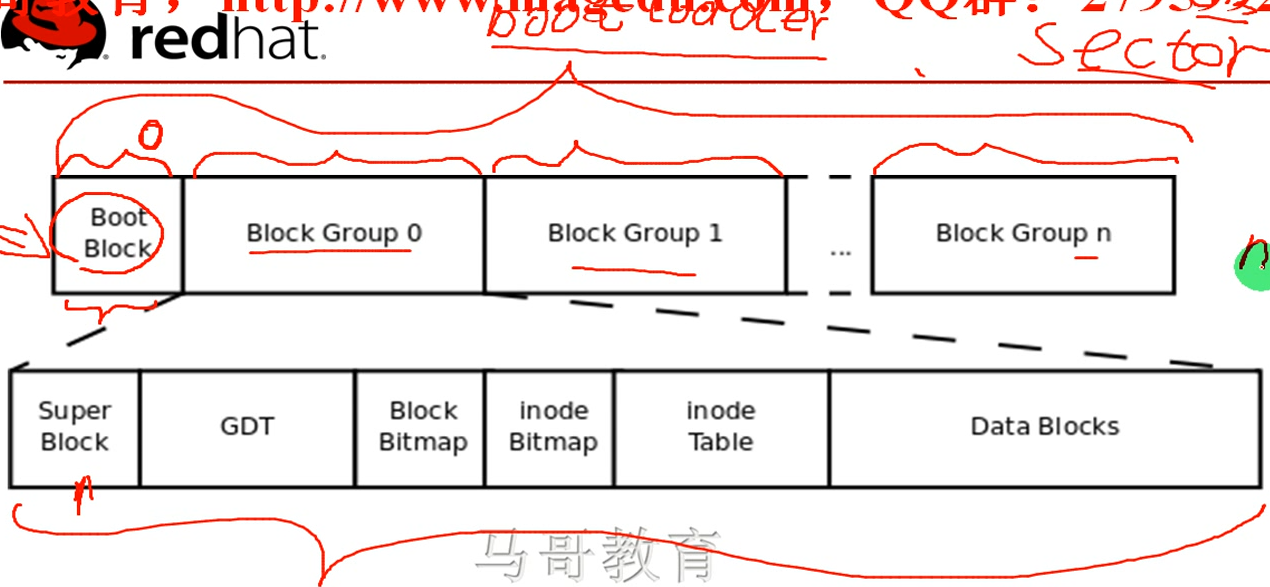
占据磁盘块的一般都是**目录文件**和**普通文件**

**bitmap 位图**

**block位图** 和 **inode位图** 扫描块组下inode Bitmap 速度非常快

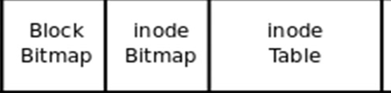
oracle中就使用了位图索引

整个磁盘空间逻辑划分



**block group**

元数据区



数据区



**super block**

需要备份 **不是每个 block group中都有super block,但是必须有备份,**

**如果bolock group0 中的super block坏了有备份,可以手动修复.**

一个分区里面有多少block group 取决于空间的大小

一个组有多大 取决于块位图的大小,inode位图的大小,一般一个位图用一个块来标识,一个块通常1k, 1024\*8(bit) 所以一个块组中的块的个数 1024\*8 个

有多少个块组,在创建文件系统的时候,在元数据区就保存了一个super block 超级块中,如果超级块坏了 基本上整个分区就挂了,但是超级块可以有多个备份,就不怕了.

超级块有多个备份: 包含分区中的全局信息,有多少块组,每个块组包含多少块,块大小,空闲磁盘块,已用磁盘块,inode使用情况等.空闲inode,已用inode.

**GDT 块组描述符表**

需要备份,如果坏了可以手动修复

块组描述符表保存着当前系统的块的描述信息. 如果挂了,整个分区也完了.所以需要备份.

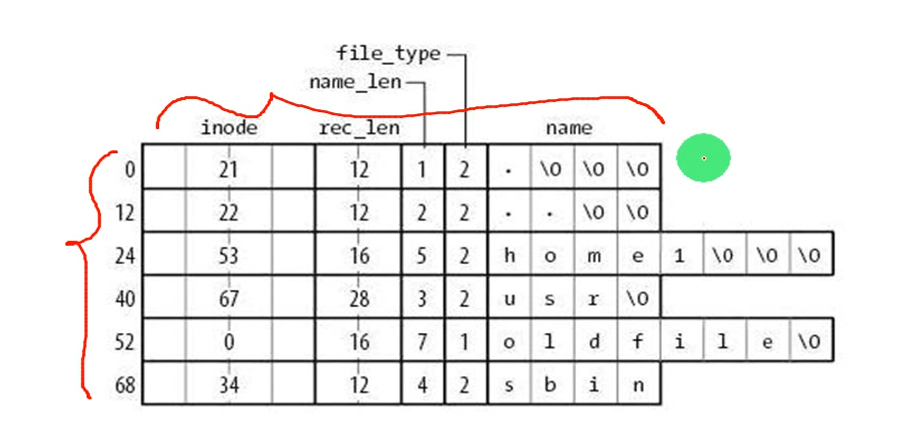
包含当前分区上每一个块组的块组名,每一个块组的起始范围.

**Boot Block** **引导块**

**不能被使用(不能存数据或者元数据)**,要预留出来,如果有多系统互存的时候需要用到,操作系统的BootLoader放在里面.如果只有一个操作系统,那么操作系统BootLoader放在MBR中,如果有多个操作系统,那么其他操作系统的BootLoader就放在自己所在分区中的BootBlock中

**注意**: MBR(主引导记录)不属于任何分区是一个扇区sector(512字节),和逻辑分区没有关系.

目录数据块如下图



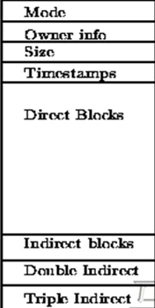
/var/log/messages

**/目录文件**, **var目录文件,log目录文件** 上图

每次都回到inode表中通过name去找对应的inode,就能找到messages文件

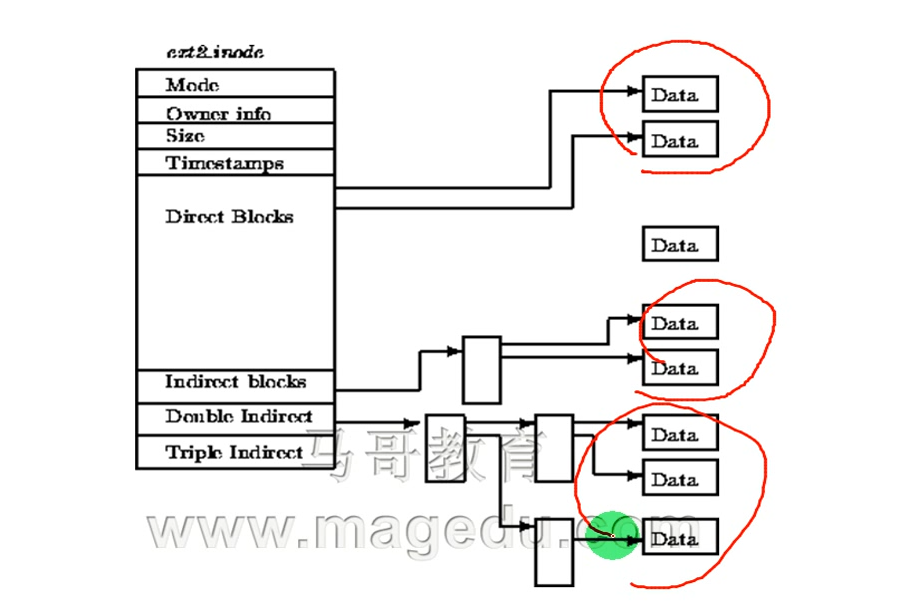
找到messages后,那么怎么引用磁盘块,找到数据呢?

我们知道inode中存放的元数据:



根据文件数据块的大小做了如下的逻辑划分

(直接 ,二级, 间接) 磁盘块指针 引用磁盘块



以上这就是ext2文件系统

## ext3

**journal file system 日志文件系统**

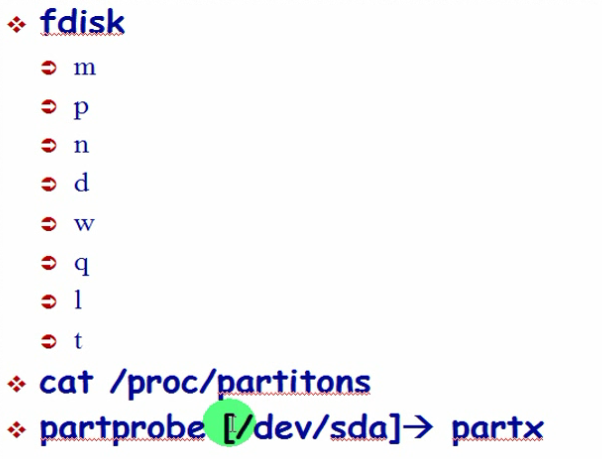
为了应对写的时候,突然掉电, 加快修复文件系统的速度,而存在.写的时候比ext2多一次写的操作,

原理:分成3个区域, inode最先创建在日志区域,掉电重启的时候,只去扫描修复日志区域,不用全部扫描修复,(有inode而数据块不完整的删除),在将日志的中的inode写到元数据区域.

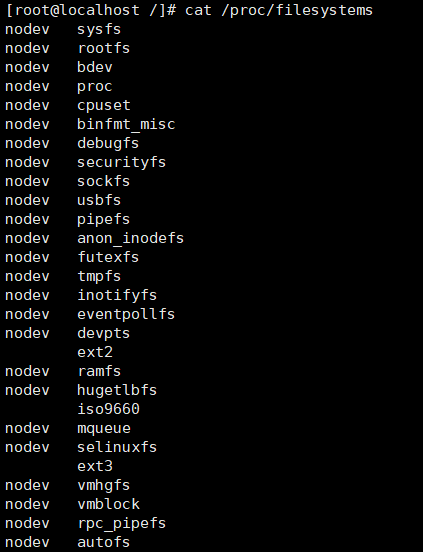
加快了文件系统修复的速度，但是要多写一次。

inode中元数据区的数据块编号存在而数据块不存在.错误的指示,修复就是将这个inode给删除.

其实linux 系统 ext2 是非日志文件系统,其他都是日志文件系统.



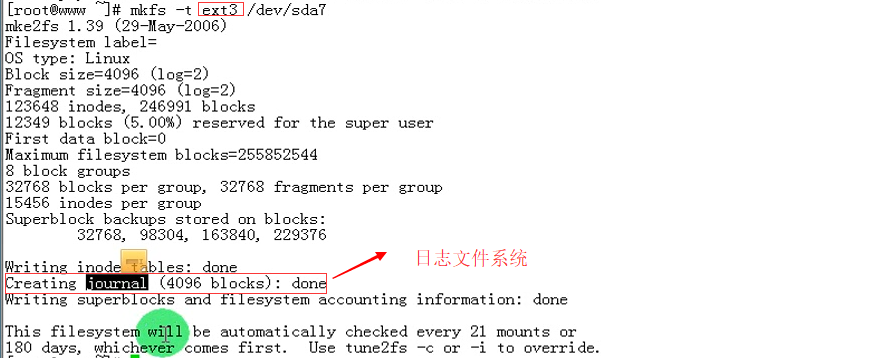
查看当前内核所支持的文件系统，内核编译阶段有没有指定哪些文件系统



## mkfs

**mkfs -t type 哪个分区**

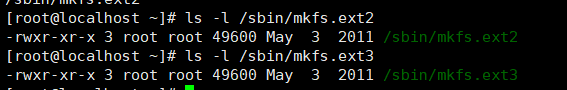
创建指定的文件系统类型



mkfs -t ext3 就是调用/sbin/mkfs.ext3这个命令完成操作

mkfs -t ext3 = mkfs.ext3

mkfs -t ext2 = mkfs.ext2



mkfs是一个统一调用接口

专门管理ext系列文件的命令

## mke2fs

-j : 指定ext系列文件的类型

-b: # 指定块大小,默认4096,可以取值 1024, 2048,4096

-L: 指定分区卷标有了卷标就可以使用卷标来引用分区, 挂载的时候就可以使用卷标来挂载

-m : # 指定预留个管理员数据块的百分比

-i : 用于指定多少个字节的空间创建一个inode,默认为8192,这里给出的数值是2的n

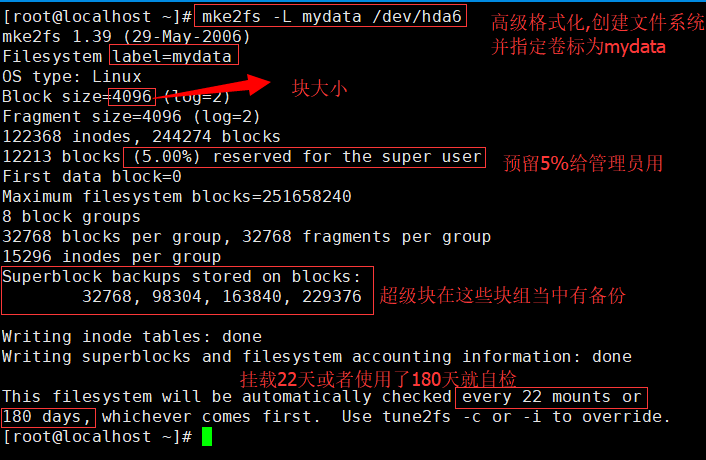
次方倍

-n : 指定inode的个数

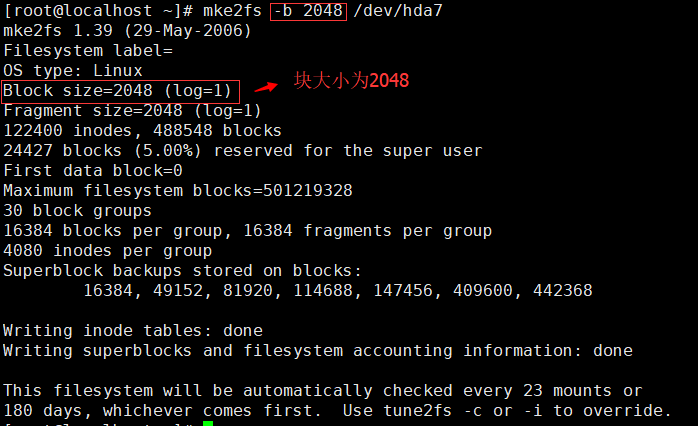
-F : 强制创建文件系统

-E : 用户指定额外的文件系统属性

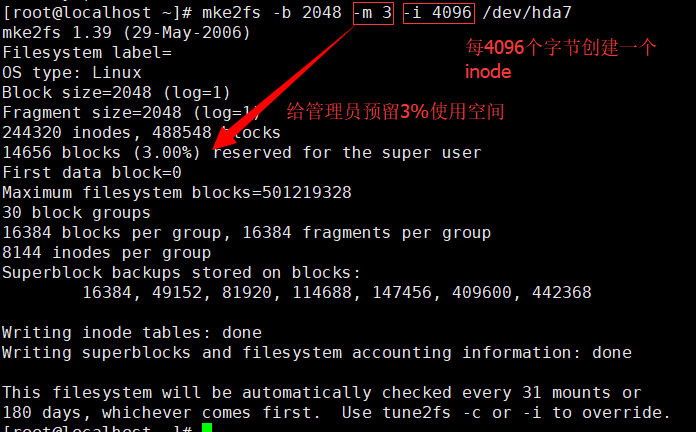
指定卷标



指定块大小



管理员使用的比例 ,根据硬盘的大小来定



## blockid

查看磁盘设备的相关属性



label 如果指定了卷标,显示卷标信息

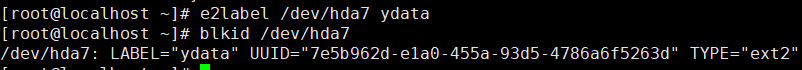
uuid 全局唯一标识符标识这个磁盘设备

type 文件系统类型

sec\_type 安全类型

## e2label

用于查看卷标信息,也可以指定卷标



## tune2fs

调整文件系统的相关属性

-j : 不损坏原有数据将ext2 升级为 ext3 ,不能降级,不能调整块数大小,是物理特性,要改重新格式化

-L : 设定或者修改卷标;

-m # : 调整预留百分比

-r # : 指定预留块数

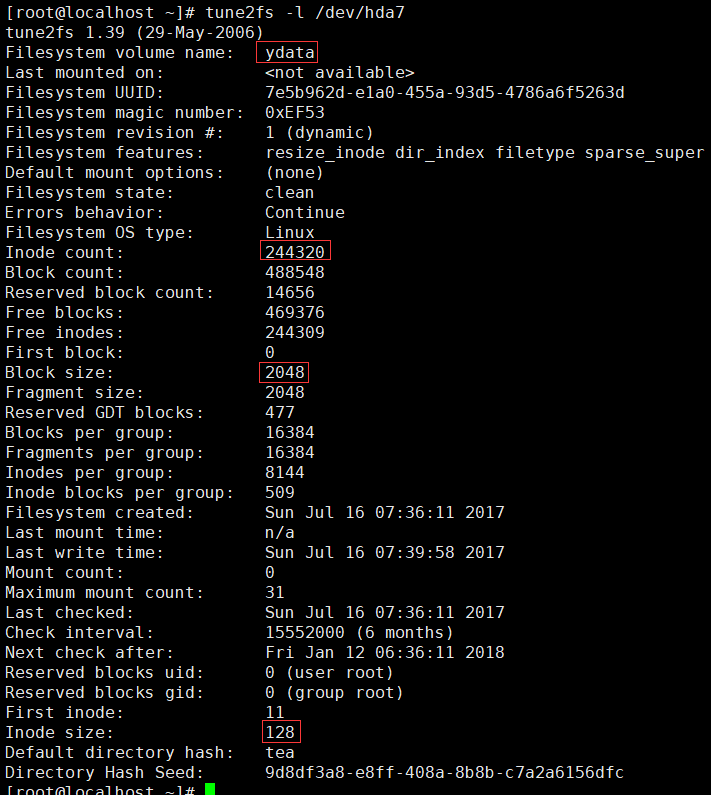
-o : 设定默认挂载选项

acl 要指定

-c # : 指定挂载次数达到#次后进行自检,0 和 -1 表示关闭此功能

-i # : 指定挂载使用多少天后进行自检, 0 和 -1 表示关闭此功能

-l : 显示超级块中的信息

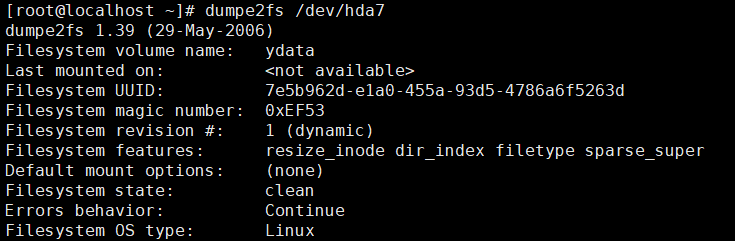


## dump2fs

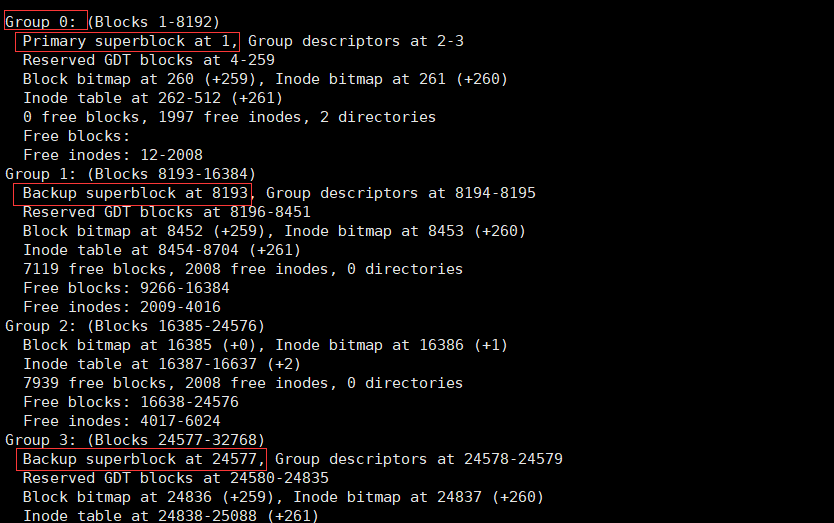
显示详细的文件系统属性

dump2fs 设备文件

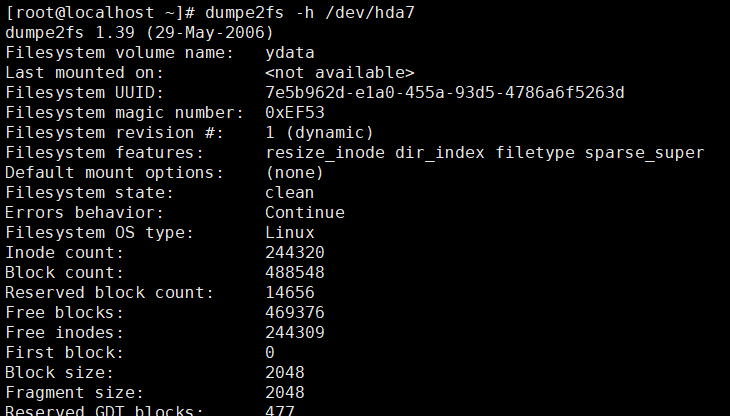
-h 只显示超级块信息 与tune2fs -I 功能一样



块组0 中primary superblock 默认使用, 如果坏了 可以到 组1 组3 中复制



dumpe2fs -h /dev/sda1



## fsck

检查并修复文件系统

-t 文件系统类型 不要指定错了

-a 不询问,自动修复

## e2fsck

专门用于ext系列文件系统

-f : 强制检查修复

-p : 自动修复 也可以使用-a

## mount

挂载: 将新的文件系统关联至根文件系统

mount 设备 挂载点

设备:

设备文件: /dev/sda2

卷标 : LABEL=””

UUID : UUID=””

挂载点: 目录

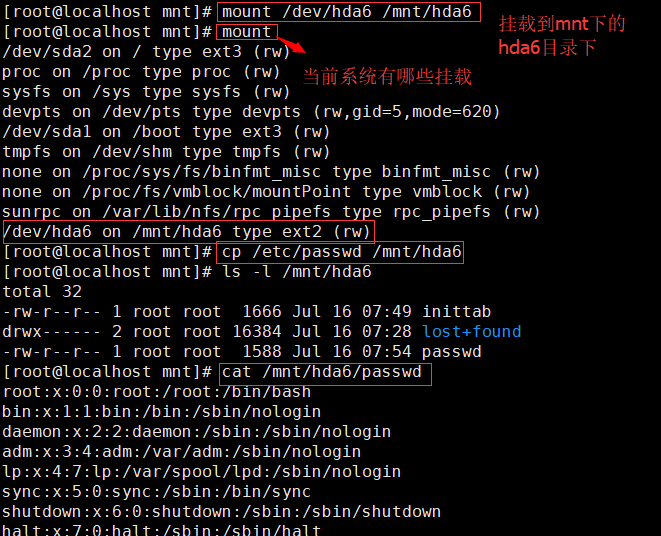
要求:

1.此目录没有被其他进程使用

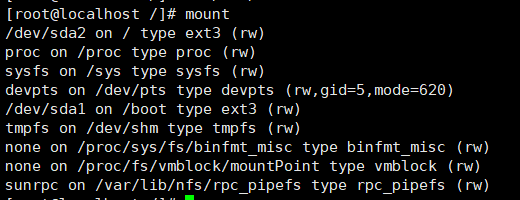
2.目录得事先存在

3.目录中的原有文件将会占时隐藏;卸载后就可以看到原有文件

挂载完成后,要通过挂载点访问对应文件系统上的文件



mount不带任何参数 显示当前系统已经挂载的设备及挂载点



mount [options] [ -o options] device mount\_point

**-a** : 表示挂载/etc/fstab文件中定义的所有文件系统

**-n** : 默认情况下,mount命令每挂载一个设备,都会把挂载设备信息保存在/etc/mtab文件中,使用-n选项意味着挂载设备时,不把信息写入此文件

**-t** : 指定正在挂载设备上的文件系统类型,不使用此选项时,mount会调用blkid命令获取对应文件系统的类型

**-r** : 只读挂载,挂载光盘常用此选项

**-w**: 读写挂载,默认使用

**-o** : 指定额外的挂载选项,即指定文件系统启用的属性

**sync** 异步默认

**atime** 更新时间戳,关闭,产生IO

**auto** 使用-a选项

**default** 默认使用选项

**dev** 允许挂载的设备有其他设备,允许其他设备可以启用,作为其他设备的访问入口

**exec** 允许挂载的设备中有执行权限的文件执行

**noexec** 挂载设备中可以执行的文件不允许执行,优盘挂载优盘,只读,不允许执行

**netdev** 网络映射挂载 如果网络不可用,就不行

**owner** 允许普通用户挂载,一般不会让普通用户挂载

**remount** 重新挂载当前文件系统

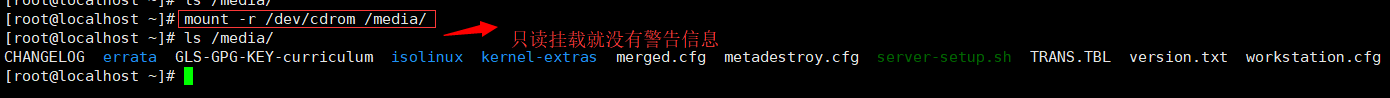
**ro**: 挂载为只读 -r = -o ro 是一个意思

**rw** : 读写挂载 -w = -o rw

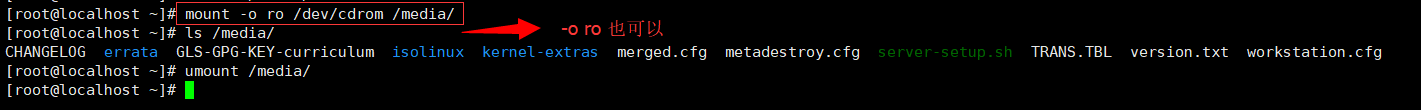
mount /dev/cdrom /media/



只读挂载

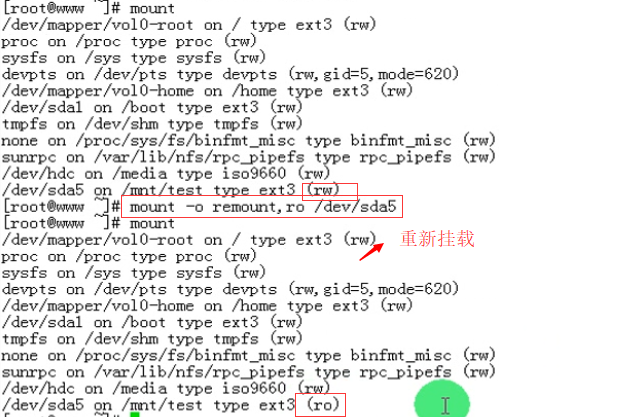


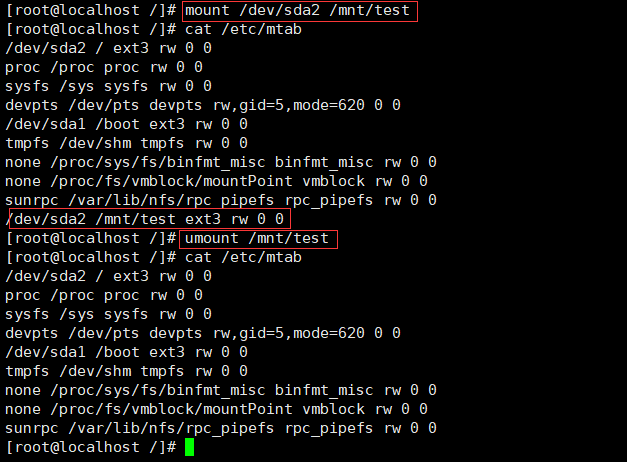
只读挂载

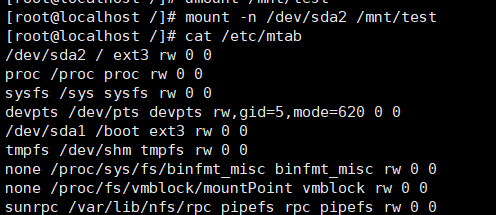


mount -o remount,ro /dev/sda5

重新挂载为ro,原来是rw的







## umount

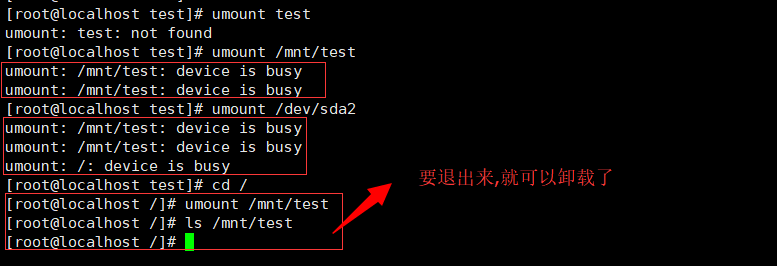
卸载: 将某文件系统与当前根文件系统的关联关系移除

umount 设备

umount 挂载点

卸载注意事项

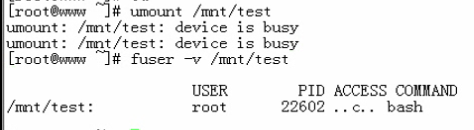
1. 挂载的设备没有进程使用



如果另一个用户在使用设备,你这边就会报devic is busy,可以通过一个命令来查看,这个设备当前是谁在使用mnt

## fuser

**fuser -v /mnt**



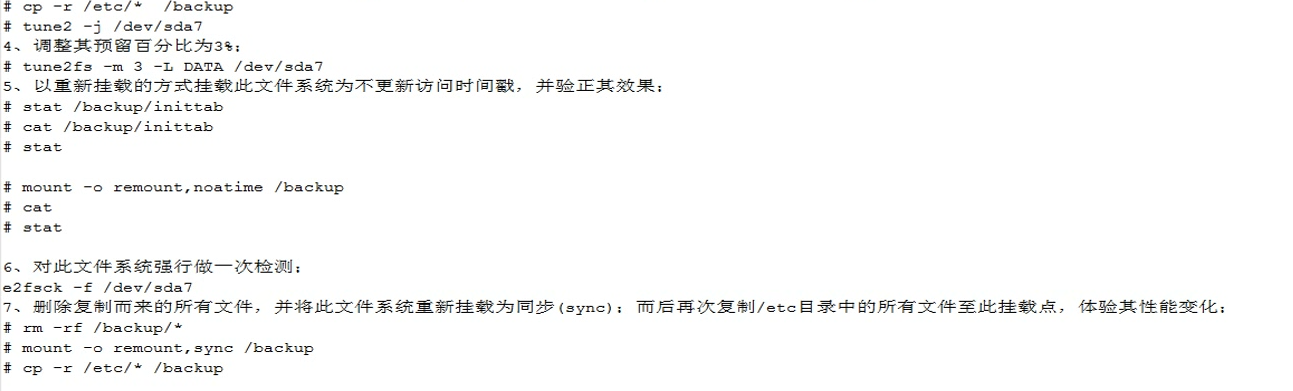
fuser验证进程正在使用的文件和套接字

-v 查看某文件上正在运行的进程

-k

-m

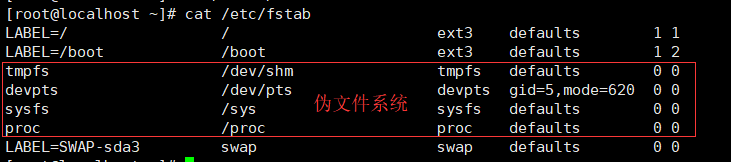
fuser -km 将对方踢出去



文件系统挂载的配置文件/etc/fstab

如果定义在fstab中的挂载,在os启动的时候,系统会自动根据fstab中的配置进行挂载,

当有些挂载没有定义在fstab中,那么系统重启,挂载就会失效.



字段的意思

**要挂载的设备**  **挂载点** **文件系统类型**  **挂载选项** **转储频率** **文件系统检查次序**

如:想要开机自动挂载,在fstab中做如下添加即可

/dev/sda5 /mnt/test ext3 defaults 0 0

然后使用命令 **mount -a** ,就会将fstab中定义的挂载给挂载上.(没挂的挂上,挂了的就不管)

交换分区的挂载点**swap**

**转储频率**: 每隔多少天完全备份一次 0 表示不备份

dump 备份 完全备份 增量备份 差异备份

**文件系统检查次序**: 根为1 最先检查 0表示不检查

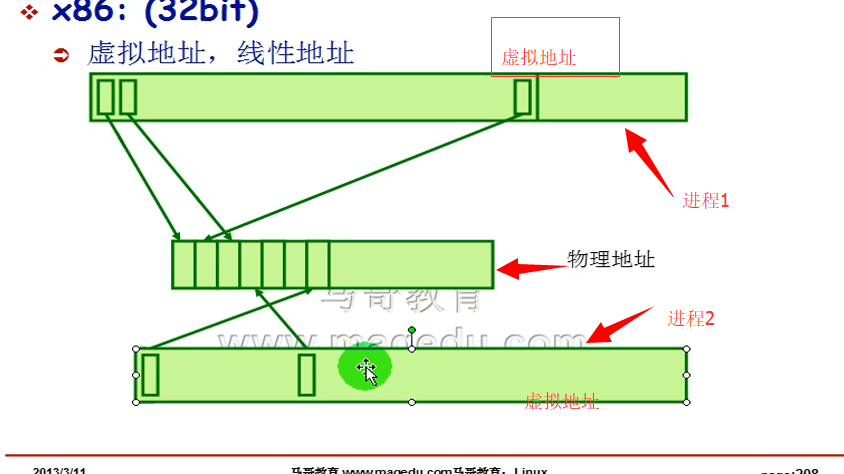
## 交换空间

操作系统就是一个虚拟机,将底层硬件资源虚拟成软件接口,通过timeshareing , multiplex

时间片,多路复用,来完成计算任务.

cpu : time slice 时间片

memory : x86 虚拟地址 ,线性地址



虚拟内存(一个进程)是分页的,物理内存分页框.

对于一个进程来说,他认为自己总有4G的空间,有1G留给操作系统, 最后映射到物理内存中,对与物理内存来说一个进程的数据是不连续的.

不停的有进程启动,物理内存一定,总有耗尽时, 这时就拿硬盘上的一块空间来当作物理内存,将最近很少使用的页面放入到硬盘空间中的页框中. 同时,内核会维护一张表,这表维护了进程及其相关的数据被移到硬盘上的哪个位置.

突然一个进程去找他的数据,就是被弄到硬盘上的那个数据的进程.此时,内核对这个进程说,你等会儿,于是内核将其他最近没有使用的数据同样的方法放到硬盘上,再根据那自己维护的那种表中的地址,从硬盘上找到数据,载入内存,分配地址. 这时才对要数据的那个进程说,你要的数据在哪个地方,为你准备好了,去找吧. 这个过程就叫**换进换出,pageout pagein** 而硬盘上的那块空间就叫做**交换空间.**

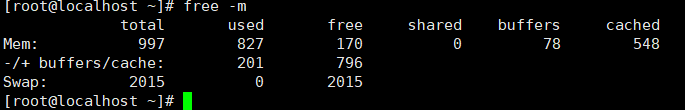
交换空间允许了内存**过载**使用 ,万不得已可以使用.一般物理内存要给够, 频繁换进换出性能就影响了.

Linux 系统交换空间是一个独立的分区.

## free

查看物理内存和交换空间使用的情况.

-m 兆字节显示



total 总共

used 使用了的

free 空想的

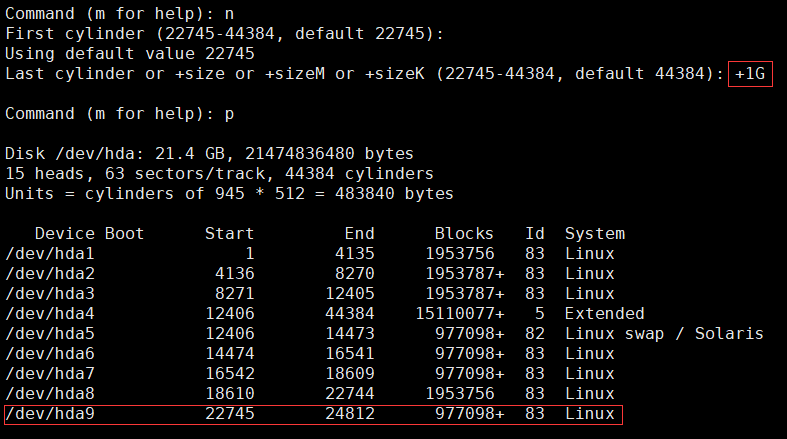
shared 共享

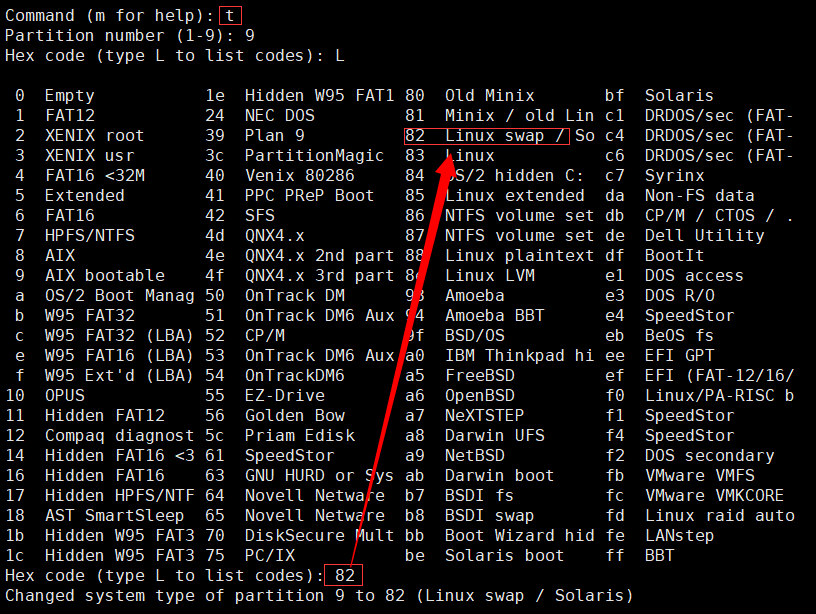
buffers 缓存 为了解决两个设备的读取速度的差异

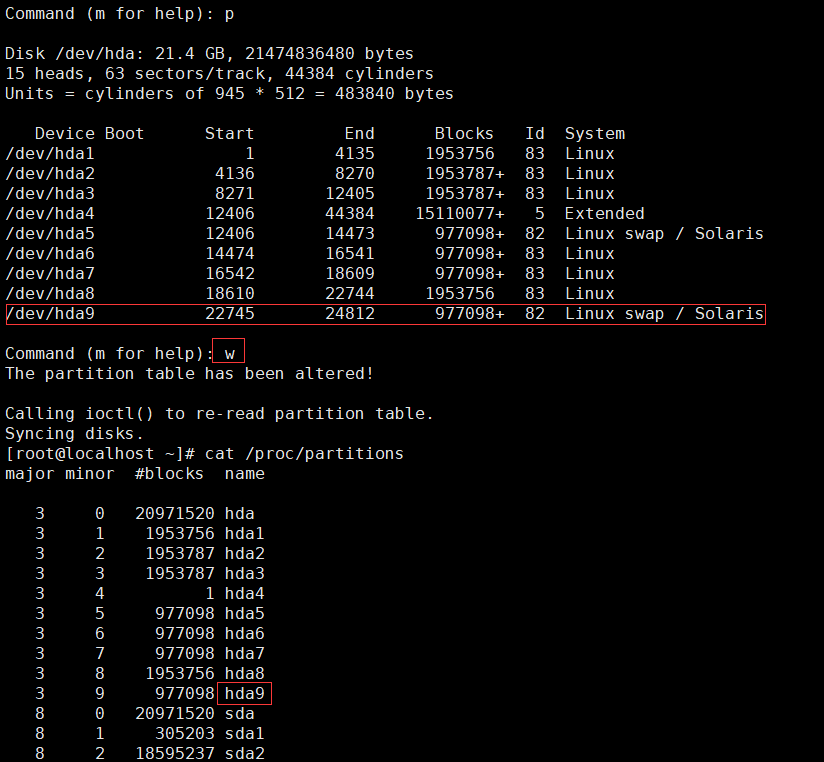
cached 缓冲

Swap : 交换空间没有使用

如果要增加交换分区,(window中叫虚拟内存),首先要新建一个分区,将分区类型调整为 Linux swap







然后创建交换分区文件系统

## mkswap

创建交换分区文件系统之前,先要调整分区类型为Linux swap

mkswap /dev/sda8

-L : 指定卷标



## swapon

开启交换空间,不能使用mount命令挂载交换空间

-a 启用所有定义在/etc/fstab中的交换设备

## swapoff

关闭交换空间,不能使用umont命令卸载交换空间



## 回环设备

lookback,**使用软件来模拟实现硬件**

## dd

dd

if=数据来源

of=数据存储目标

bs=# 以多少个字节为单位进行复制

count=# 一共复制多少个这样的字节

seek=# 创建数据文件时,跳过空间的大小

与cp命令差不多,但是他们复制文件的形式有区别,cp按照文件以文件为单位,dd复制的是底层的文件流(10100101010011001…)

可以不完整复制,在有些地方强大到你不能想象

**备份MBR**

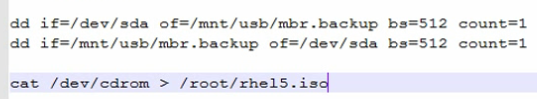
dd if=/dev/sda of=/mnt/usb/mbr.backup bs=512 count=1

**如果MBR坏了,可以还原MBR**

dd if=/nnt/usb/mbr.backup of=/dev/sda bs=512 count=1

创建一个光驱映像

cat /dev/cdrom > /root/rhe15.iso



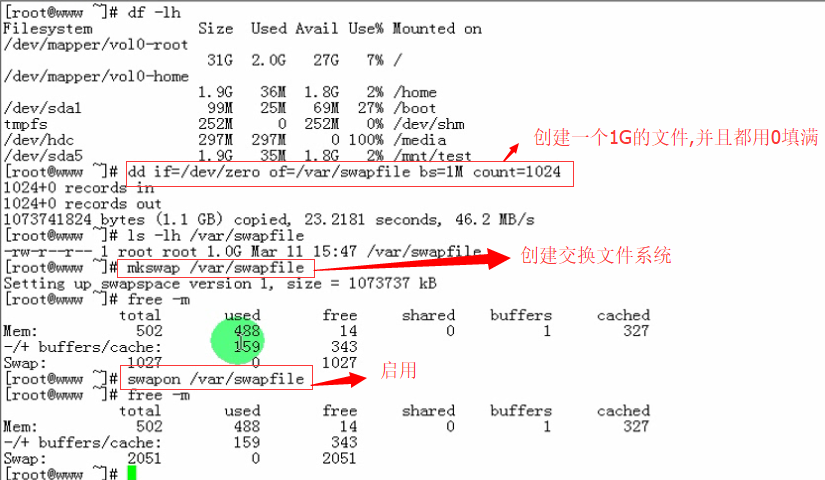
如果硬盘分区已经满了,一些分区还有没有用完的空间,也可以这样解决问题.临时救急

/dev/null 数据黑洞 给多少要多少

/dev/zero 泡泡设备 要多少给多少

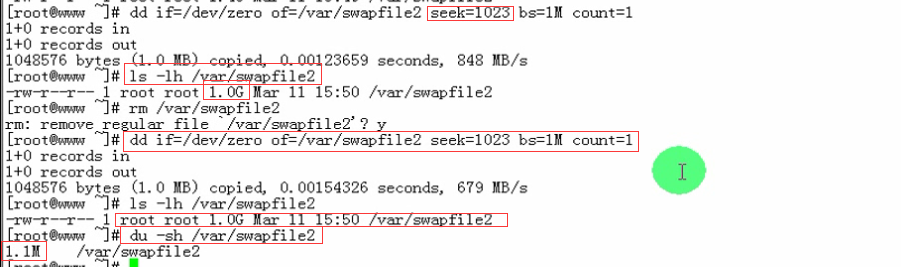
loop 回环设备

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



dd if=/dev/zero of=/var/swapfile2 seek=1023 bs=1M count=1

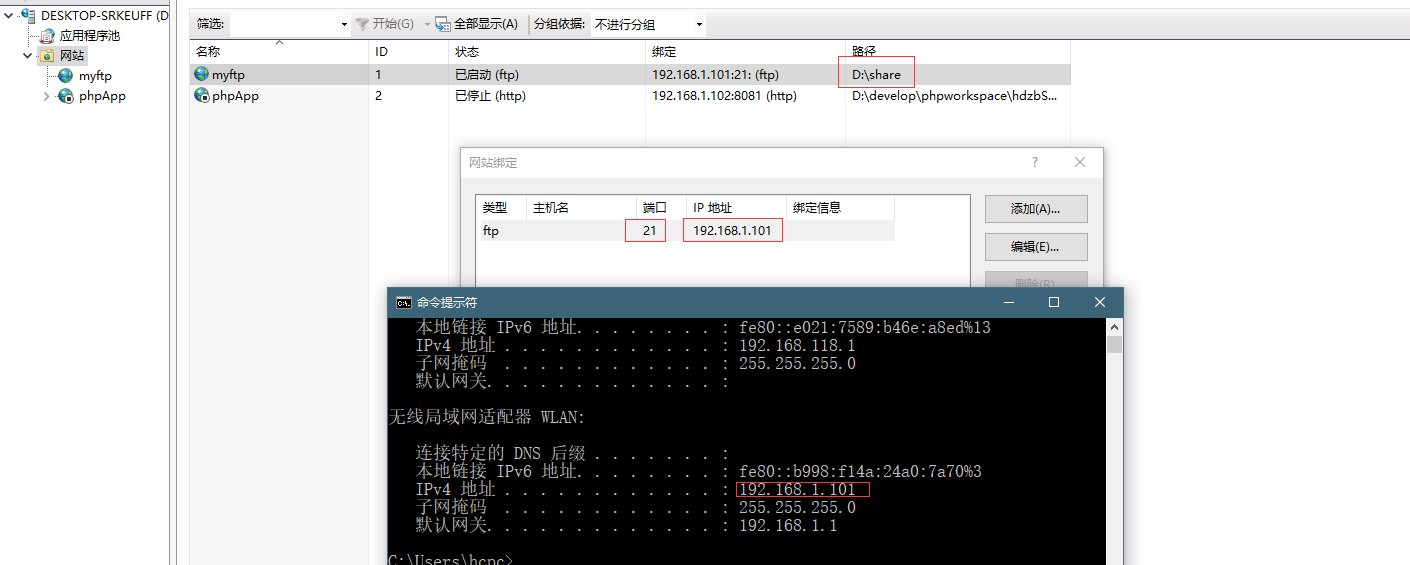
**跳过1023**,只有1M ,文件看上去有1G但是实际文件大小1.1M



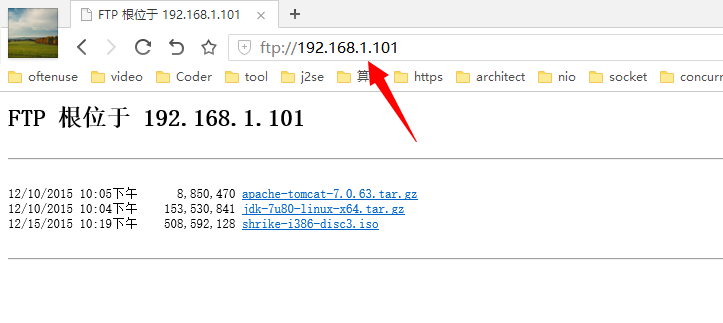
mount一般用来挂载磁盘,光驱,也可以通过回环设备来挂载ISO文件

回环设备模拟了硬件设备,就可以挂载ISO文件并对其进行访问

1.windows搭建ftp服务器



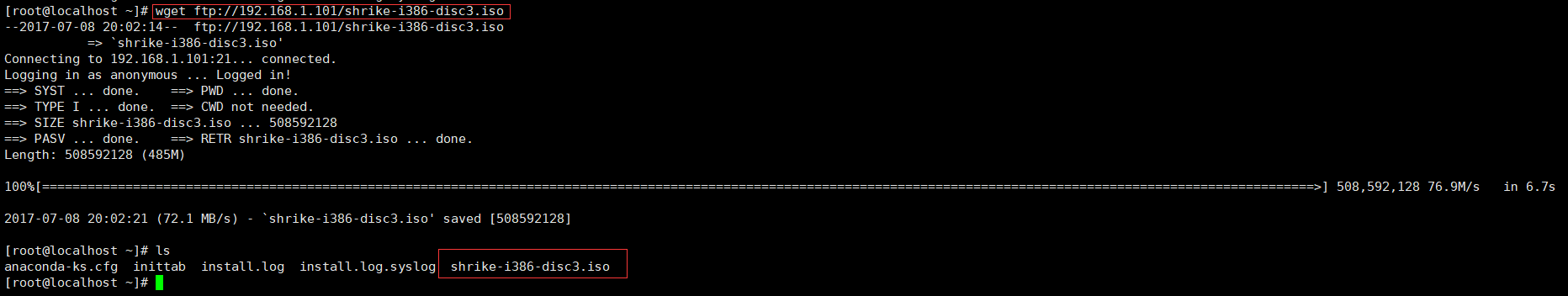
将文件放入share目录,打开浏览器测试



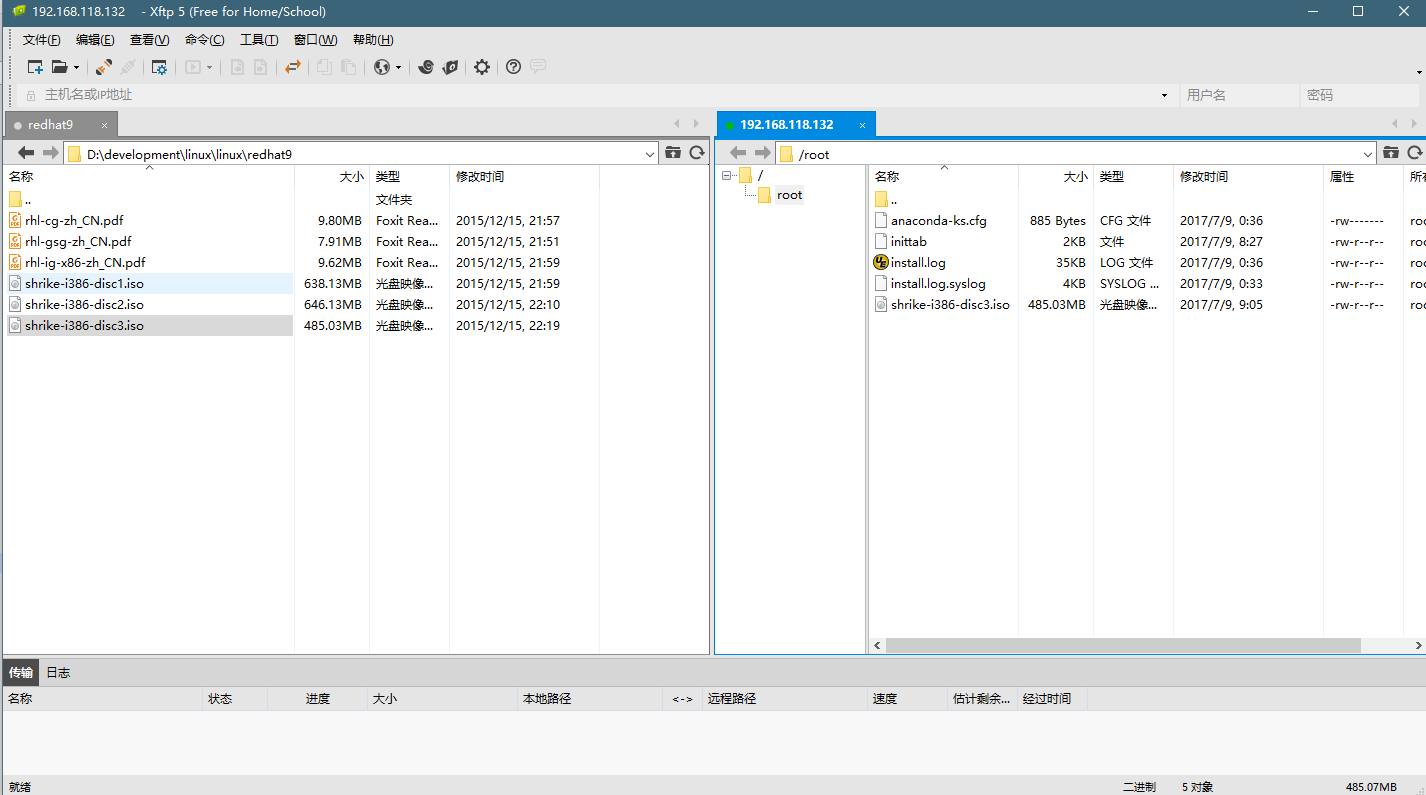
wget <ftp://192.168.1.101/xxxxxx>

Linux可以通过ftp传输获取文件

wget ftp://192.168.1.101/sharike-i386-disc3.iso



也可以通过xftp上传文件



对\*.iso 文件进行挂载, 说iso不是一个块设备 可以使用 -o loop



使用回环设备挂载

