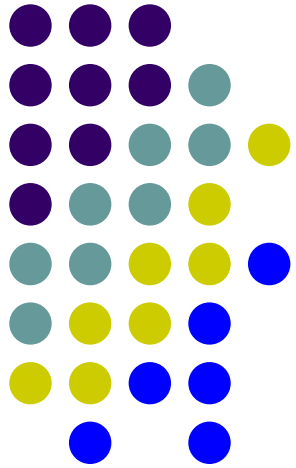
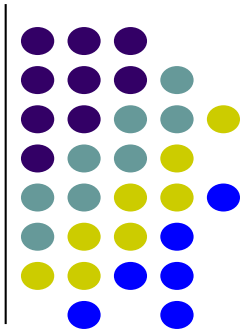
第7章

Linux文件系统

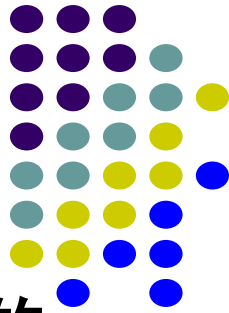


本章内容



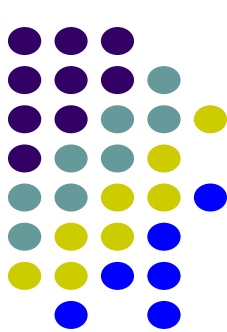
- 文件系统基础
- 文件系统的挂载与卸载
- 管理磁盘的Shell命令
- 文件管理
- 虚拟文件系统

7.1 Linux文件系统基础



- 数据处理是计算机的主要功能之一，与数据处理相关的数据管理和数据保存是必不可少的重要的环节。在计算机中，大量的数据和信息是通过文件存储和管理的。
- 从资源管理角度来看，操作系统是计算机中软、硬件资源管理者。其中软件资源管理就是文件系统，文件系统负责管理文件，为用户提供文件的操作手段（存储、检索、更新、共享和保护）。同时，文件系统隐藏了最为纷繁复杂的硬件设备特征，为用户以及操作系统的其他子系统提供一个统一、简洁的接口，使得用户方便地使用计算机软硬件资源。

Linux文件系统

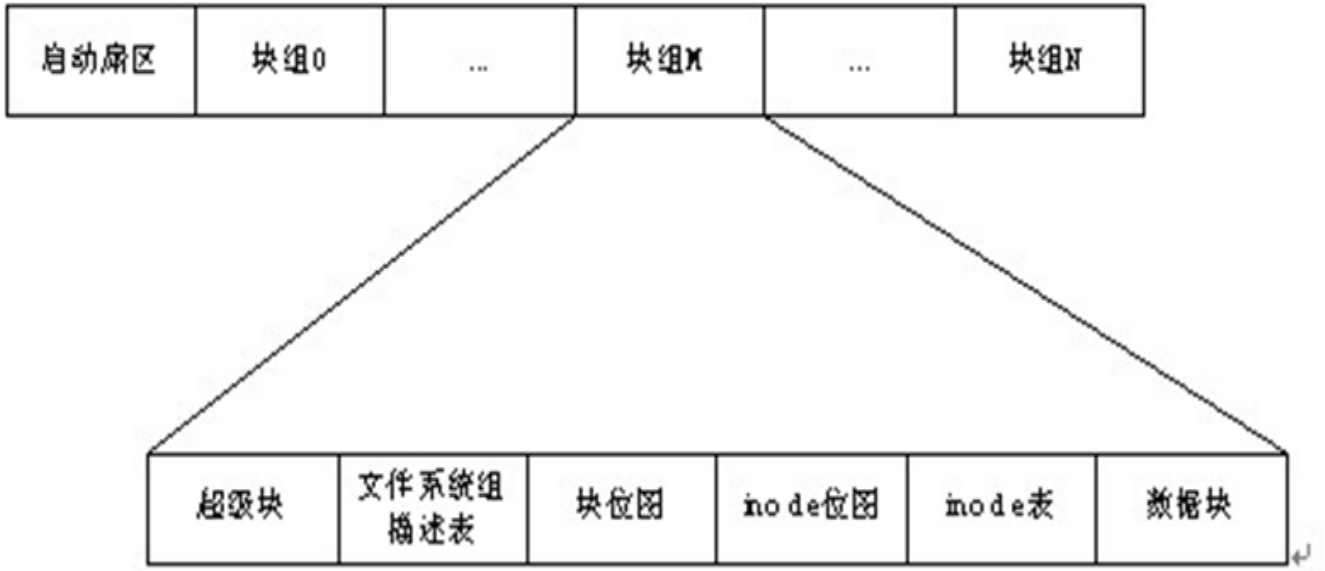


- Linux系统的一个重要特点就是“一切皆文件”，从这个特点看出文件的重要性。Linux的文件系统继承了UNIX文件系统的优秀设计，并融入了现代文件系统的先进技术，在开放性、可扩展性等方面都十分出色，不仅具备普通的文件管理功能，还有许多特殊的功能，文件管理非常灵活，功能强大。
- Linux 最早的文件系统是Minix，Minix文件系统由MINIX操作系统定义，有一定的局限性，如文件名最长14个字符，文件最长64M字节。第一个专门为Linux设计的文件系统是EXT（Extended File System）。

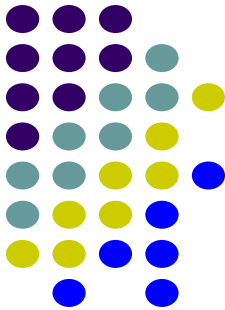
Linux基本文件简介

ext2文件系统

- 扩展文件系统第二版ext2对Linux产生了重大影响。ext2文件系统功能强大、易扩充。它的设计思想是由一系列逻辑上线性排列的数据块构成，每个数据块大小相同。所有的数据块被划分成若干个分组，每个组包含相同个数的数据块。

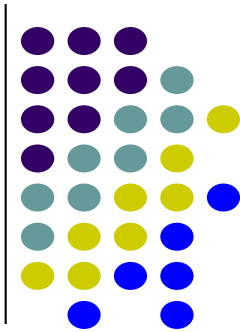


Linux基本文件简介



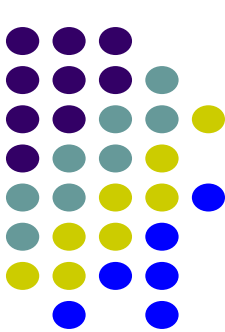
- ext3是ext2的升级版本，完全兼容ext2，加入日志技术，ext3文件系统已经非常稳定可靠，支持16T分区。
- ext4是一种针对ext3系统的扩展日志式文件系统。ext3升级到ext4能为系统提供更高的性能，消除存储限制，并且不需要重新格式化分区，ext4会在新的数据上用新的文件结构，旧的文件保留原状。
- Btrfs文件系统是最新的Linux文件系统。
- Ubuntu发行版当前默认使用ext4文件系统。理论支持1024PB大小（1PB=1024TB）的存贮设备，支持文件的连续写入，减少文件碎片，提高磁盘的读写性能。

Linux基本文件系统简介



- swap文件系统
  - swap文件系统用于Linux的交换分区。在Linux中，使用整个交换分区来提供虚拟内存，其分区大小一般应是系统物理内存的2倍。
  - 在安装Linux操作系统时，就应创建交换分区，它是Linux正常运行所必需的，其类型必须是swap。交换分区由操作系统自行管理。

Linux支持的文件系统

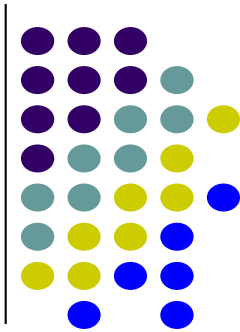


- Linux采用虚拟文件系统（VFS）技术，可支持多种常见的文件系统，并允许用户在不同的磁盘分区上安装不同的文件系统，Linux支持的文件系统类型主要有：
  - msdos：MS DOS采用的FAT文件系统。
  - vfat：　Windows的FAT32文件系统。
  - Ntfs:　Windows的NTFS文件系统。
  - sysV：UNIX中最常用的system V文件系统。
  - nfs：网络文件系统（Network File System）。
  - iso9660：CD-ROM或DVD-ROM的标准文件系统。

Linux文件的类型

- Linux系统按照文件中数据的特点对文件划分不同的类别，称作文件类型。文件划分类型后，系统处理文件可以分类处理。Linux内核把文件类型归类如下：
  1. 普通文件
  2. 目录文件
  3. 链接文件
  4. 设备文件
  5. 管道文件
  6. 套接字文件

文件的类型



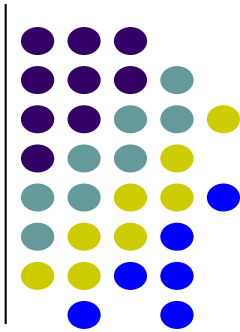
1. 普通文件

普通文件是最常使用的一类文件，其特点是不包含文件系统的结构信息。用户所接触到的文件，如图像文件、数据文件、文档文件、声音文件等都属于普通文件。按内部结构可细分为两类：

文本文件，采用ASCII编码方式，可编辑可修改；

二进制文件，不可查看，不可修改；（所有的命令等）

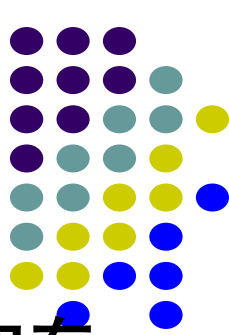
文件的类型



2. 目录文件

存放的内容是目录中的文件名和子目录名，目录文件可以包含下一级目录文件或普通文件。在linux中，目录文件是一种文件，它是内核组织文件系统的基本节点。Linux的目录文件和其他操作系统中的“目录”的概念不同，它是linux文件中的一种。

文件的类型



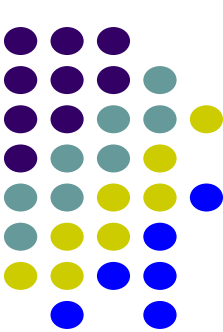
3. 链接文件

链接文件是一种特殊的文件，实际上是指向一个真实存在的文件的链接。根据链接文件的不同，它又可以细分为

硬链接文件：硬链接文件保留所链接文件的索引节点（磁盘的具体物理位置）信息，即使被链接文件更名或移动，硬链接文件仍然有效。Linux要求硬链接文件和被链接的文件必须属于同一分区并采用相同的文件系统。

软链接文件：也称符号链接，类似Windows下的快捷方式，其本身并不保存文件内容，而只记录所链接文件的路径。如果被链接文件更名或移动，符号链接文件就无任何意义。源文件与链接文件可以跨越索引节点。

文件的类型



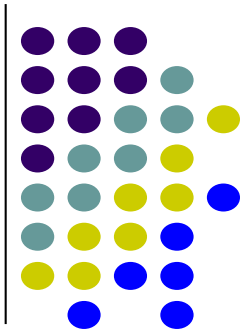
4. 设备文件

设备文件是Linux中最特殊的文件，用于用户访问物理设备所用。Linux系统为外部设备提供一种标准接口，将外部设备视为一种特殊的文件，使用户可以像访问普通文件一样访问任何外部设备。通常Linux系统将设备文件放在“/dev”目录下。设备文件使用设备的主设备号和次设备号来指定某个外部设备。根据访问数据方式的不同，设备文件可以分为

字符设备：键盘、鼠标；

块设备：硬盘、光驱；

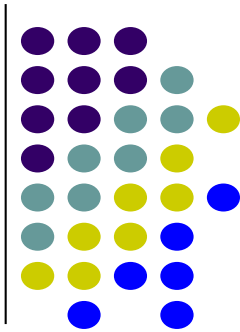
文件的类型



5. 管道文件

管道文件主要用于不同进程间的消息传递。当两个进程间需要进行数据或信息传递时，可以使用管道文件。一个进程将需传递的数据或信息写入管道的一端，另一进程则从管道的另一端取得所需的数据或信息。前一个命令的输出作为后一个命令的输入。

文件的类型

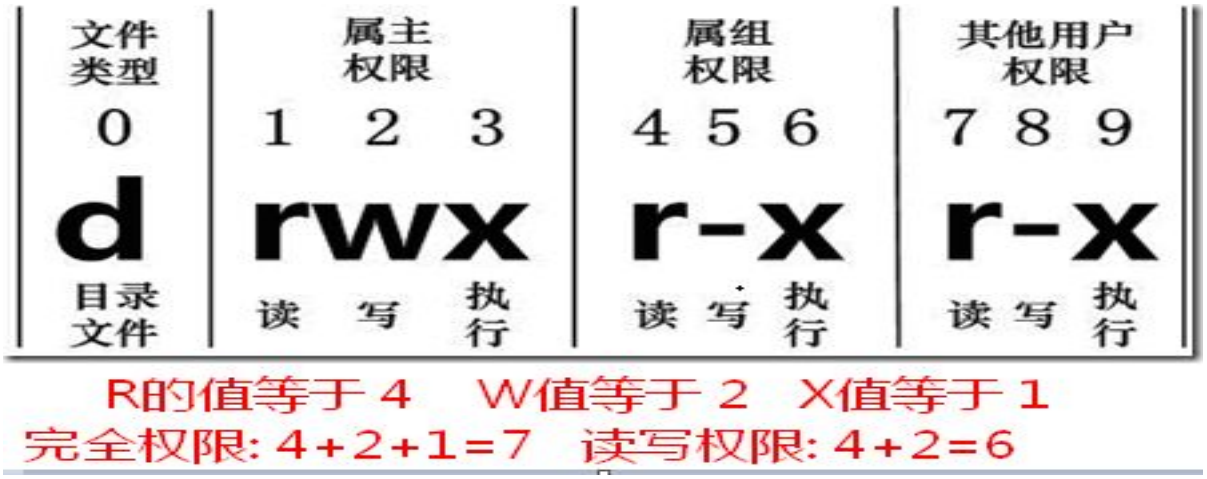


6.套接字文件

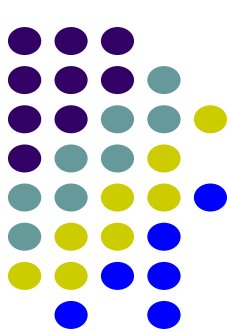
套接字文件系统是一个用户不可见的， 高度简化的，用于汇集网络套接字的内存文件系统，它没有块设备， 没有子目录， 没有文件缓冲，它借用虚拟文件系统的框架来使套接字与文件描述字具有相同的用户接口。当用户用socket(family,type,protocol)创建一个网络协议族为family， 类型为type，协议为protocol的套接字时, 系统就在套接字文件系统中为其创建了一个名称为其索引节点编号的套接字文件。

存取权限和文件模式

- 为保证文件信息的安全，Linux设置了文件保护机制，其中之一就是给文件都设定了一定的访问权限。当文件被访问时，系统首先检验访问者的权限。
- Linux对文件设定了三级权限：文件所有者、与文件所有者同组的用户、其他用户。对文件的访问主要是三种处理操作：读取、写入、执行。



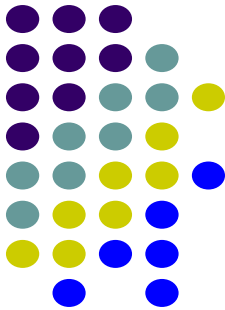
Linux文件的命名



文件的命名：

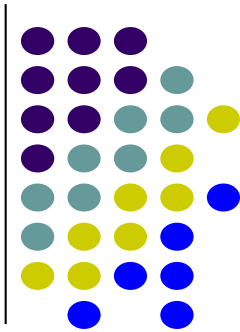
- 由字母（可用汉字）、数字、下划线、圆点等字符构成。
- 长度不超过255个字符，避免使用特殊字符 ？＊ \ $ 等。
- 同一目录下不能有相同的文件名，不同目录下可以同名。
- 圆点 · 在第一位置时表示隐含文件。
- 文件的属性与取名无关，文件名中不规定扩展名。
- 区分英文字符的大小写。比如myfile, Myfile 和myFILE表示的是三个不同的文件。

Linux文件目录



- Linux以文件目录的方式来组织和管理系统中的所有文件，将所有文件的说明信息采用树形结构组织起来。整个文件只有一个“根”（root），然后在根上分权。任何一个分权都可以再分权，权上可以长叶子。“根”和“权”在Linux中被称为“目录”或“文件夹”，“叶子”则是文件。
- 用户可可以设置目录和文件的管理权限，以便允许或拒绝其他人对其进行访问。文件目录结构的相互关联性使分享数据变得十分容易。

Linux文件目录



- 目录常见的概念：

（1）路径：从树形目录中的某个目录层次到某个文件的一条道路，路径的构成是目录名称，中间用“/”隔开。路径分为相对路径和绝对路径，绝对路径从“根”开始的路径，也称完全路径；相对路径是从用户工作目录开始的路径。

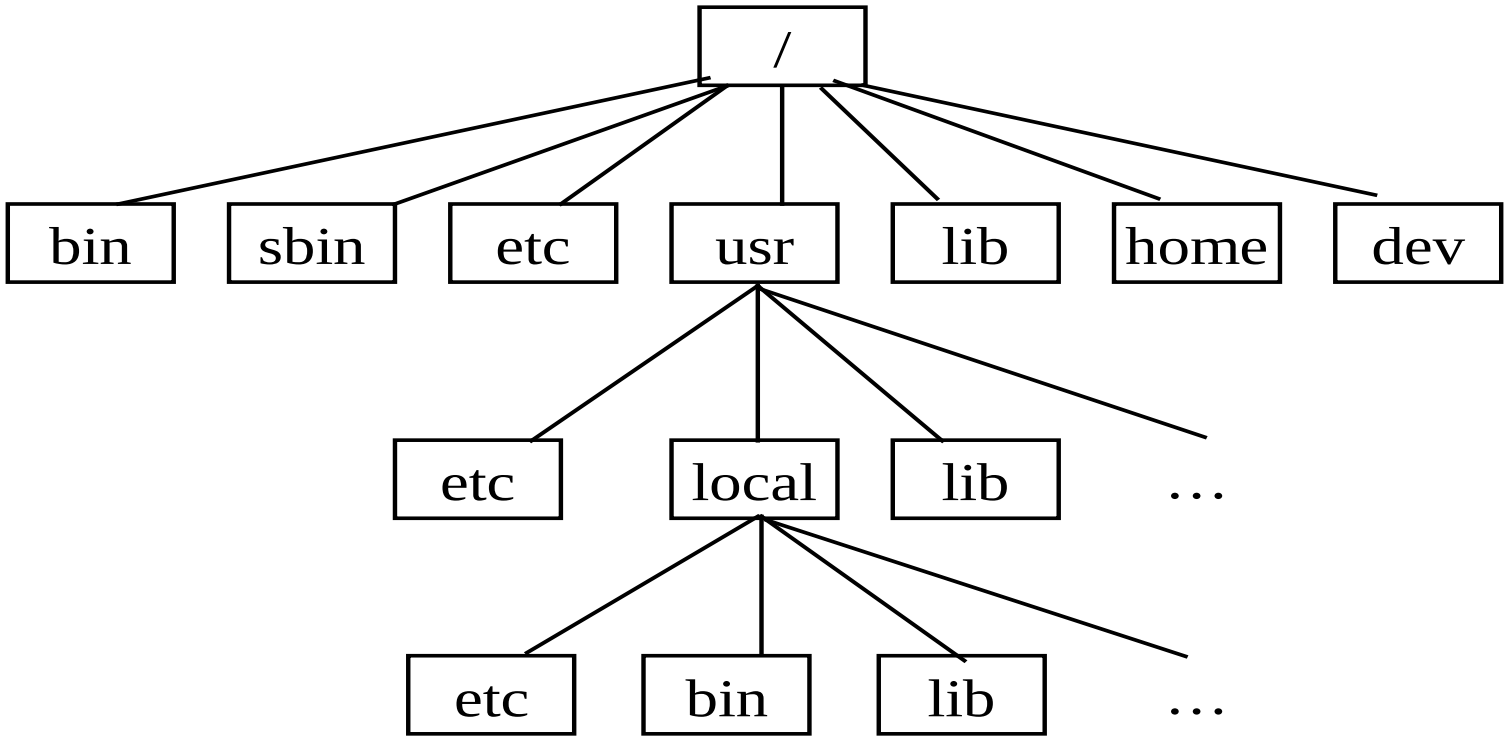
（2）根目录：是Linux系统中最特殊的目录。根目录所在的起点，操作系统本身的驻留程序存放在以根目录开始的专用目录中。

（3）用户主目录：是系统管理员增加用户时建立的用户目录。不同的用户主目录一般互不相同。用户刚登录到系统中时，其工作目录就是该用户的主目录。

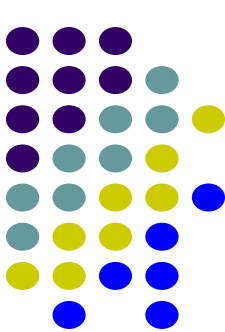
（4）工作目录：用户登录系统之后工作所处的当前目录。工作目录可以随时改变。

Linux树型目录结构

根目录下有一些非常重要的子目录，如/bin、/etc、/dev、/home、/usr等。



Linux系统主要目录说明

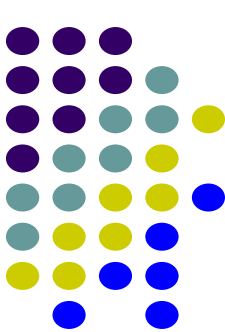


/bin ：bin是二进制（binary）英文缩写。 通常存放用户最常用的一些基本命令，包括对目录和文件操作的一些实用程序、系统实用程序、压缩工具、RPM包管理程序等，如login,date, ping, netstat, mount,unmount、su、vi、rpm等。

/sbin ：这个目录是用来存放系统管理员的系统管理程序。如

fdisk,mkfs,ext3,vfat,shutdnwn,dump,route,iptables等。

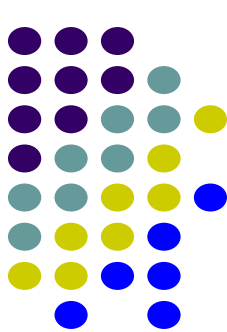
Linux系统主要目录说明



/boot ：在这个目录下存放的都是系统启动时要用到的各种文件。包括系统的引导程序和系统内核程序，不要轻易对该目录进行操作。

/etc ：etc这个目录是linux系统中最重要的目录之一。存放系统管理时要用到的各种配置文件。包括网络配置文件，设备配置信息、X-Windows系统配置、用户信息等，如securetty,passwd、inittab, fstab。

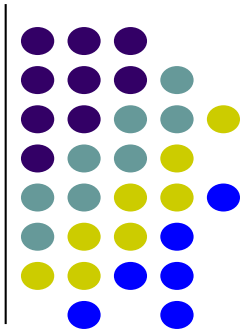
Linux系统主要目录说明



/dev：dev 是设备（device）的英文缩写。这个目录对所有的用户都十分重要。因为在这个目录中包含了所有linux系统中使用的外部设备。但是这里并不是放的外部设备的驱动程序。

- Linux将每一个I/O设备都看成一个文件，与普通文件一样处理，这样可以使文件与设备的操作尽可能统一。从用户的角度来看，对I/O设备的使用和普通文件的使用一样，不必了解I/O设备的细节。

Linux系统主要目录说明

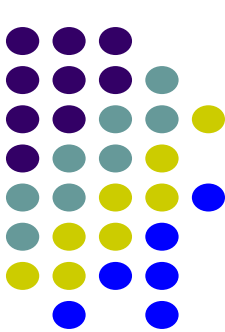


/home：系统中所有**普通用户的宿主(家)目录**，新建用户账户后，系统就会自动在/home中创建一个与账户同名的子目录，作为该用户的宿主目录。普通用户只能访问自己的宿主目录，无权访问其他用户的宿主目录。root用户的宿主目录为/root。

/lib：lib是库（library）英文缩写。这个目录是用来存放系统动态连接共享库的，包含和C、C++和FORTRAN 等语言的库文件。

/mnt：这个目录在一般情况下也是空的。可以临时将别的文件系统挂在这个目录下。

Linux系统主要目录说明

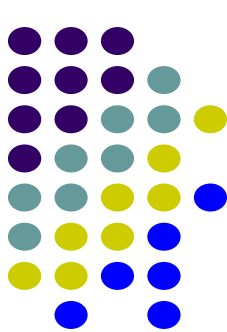


/proc：该目录中文件是**当前内存中的一个映像，**不占用磁盘空间，存放进程的运行信息，由内核在内存中产生。

/tmp：用来存放不同程序执行时产生的临时文件。

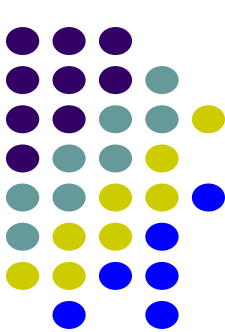
/usr：这是linux系统中占用硬盘空间最大的目录，一般用来存放与用户直接相关的程序或文件。用户安装的程序或要自行建立的目录，一般应放在该目录下面。

7.2 文件系统的挂载与卸载



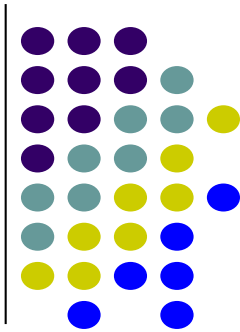
- 在整个树型结构中、只有一个根目录位于根分区，其他目录、文件以及外部设备(包括硬盘、软驱、光驱、调制解调器等)文件都是以根目录为起点，挂接在根目录下面的，即整个Linux的文件系统，都是以根目录为起点的，其他所有分区都被挂载到目录树的某个目录中。通过访问挂载点目录，即可实现对这些分区的访问。

文件系统的挂载与卸载



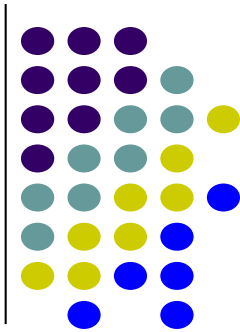
- 如果要使用USB存储设备、光盘或软盘等存储设备，必须将这些设备中的“小”目录像嫁接一样挂载（Mount）到Linux系统的“大”目录树中。当存储设备挂载成功后，就可以将其作为“大”目录树中的一个目录进行访问。
- 微软的DOS和windows也是采用树型结构，但是在DOS和windows中这样的树型结构的根是磁盘分区的盘符，有几个分区就有几个树型结构，他们之间的关系是并列的。

文件系统的挂载与卸载



- 挂载就是将存储介质的内容映射到指定的目录中，此目录即为该设备的挂载点。Linux系统中有一个/mnt目录，专门用作挂载点（mount Point）目录。
  - 硬盘上的各个磁盘分区都会在启动过程中自动挂载到指定的目录，并在关机时自动卸载。
  - 移动存储介质既可以在/etc/fstab中指定开机自动加载，也可以在需要时手动挂载/卸载。
  - 在挂载设备时首先查看挂载点目录是否存在，如果不存在必须首先创建该目录，否则mount命令无法正常执行。

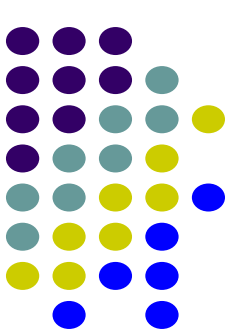
文件系统的挂载与卸载



- 文件系统的挂载记载在/etc/fstab文件中：

```
[root@Linux root]# cat /etc/fstab
LABEL=/                 /                       ext3    defaults        1 1
LABEL=/boot             /boot                   ext3    defaults        1 2
none                    /dev/pts                devpts  gid=5,mode=620  0 0
none                    /proc                   proc    defaults        0 0
none                    /dev/shm                tmpfs   defaults        0 0
/dev/sda3               swap                    swap    defaults        0 0
/dev/cdrom              /mnt/cdrom              udf,iso9660 noauto,owner,kudzu,ro 0 0
/dev/fd0                /mnt/floppy             auto    noauto,owner,kudzu 0 0
[root@Linux root]#
```

fstab文件内容



- 由上图显示的内容可以看出fstab文件是由一条条的记录所组成，其中每一行代表一个自动挂载项。每条记录由6个字段组成：

第1个字段是设备名；

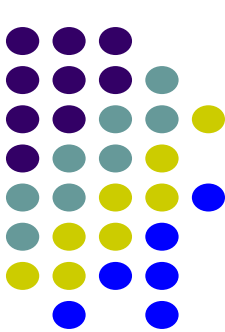
第2个字段设置挂载点；

第3个字段显示文件系统的类型；

第4个字段是挂载选项；

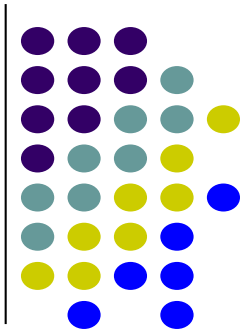
使用defaults表示系统自动识别文件系统进行挂载；

fstab文件内容



第5个字段设置是否备份，0表示不备份，1表示要备份；

第6个字段设置自检顺序，该字段被fsck命令用来决定在系统启动时需要被扫描的文件系统的顺序，根文件系统“/”对应该字段的值为1，其他文件系统为2，如果某文件系统在启动时不需要扫描，则该字段的值设置为0。



对设备文件进行操作，实际上是在操作该文件对应的物理设备。

Linux中常用的外部设备文件名：

软盘 /dev/fdN (N=0,1 ...)

硬盘(IDE) /dev/hdX (X=a,b,c,d) hda1,hda2…

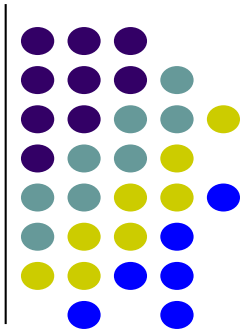
硬盘(SCSI) /dev/sdX (X=a,b,c ...p)

sda1,sda2… sdb1,sdb2…

硬盘(SATA)/dev/sd(a-p) sda1,sda2… sdb1,sdb2…

U盘 /dev/sdX (X=a,b,c ...)

挂载命令mount



mount 命令

语法：mount [选项] [设备名] [目录]

功能：查看文件系统挂载情况或将磁盘设备挂载到指定的目录。

无选项和参数时，查看当前文件系统的挂载情况，相当于查看/etc/mtab文件的内容。

有选项和参数时，进行磁盘挂载操作。此时，目录参数为设备的挂载点。挂载目录可以不为空，但必须存在。磁盘设备挂载后，该挂载点目录的原文件暂时不能显示且不能访问，取代它的是挂载设备上的文件。原目录上文件待挂载设备卸载后才能重新访问。

主要选项说明：

-t 文件系统类型 挂载指定的文件系统类型

-r 以只读方式挂载文件系统，默认为读/写方式

挂载设备的过程

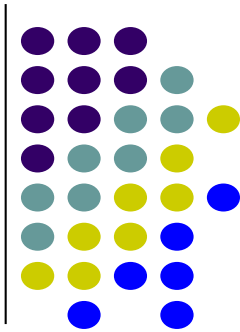
1 查看设备：使用命令“fdisk -l”可以查看系统的存储设备。

2 挂载设备：首先使用mkdir命令建立挂载点目录，然后再使用mount命令挂载相关设备。

3 访问设备 cd 挂载点目录

4 卸载设备：用户在使用完挂载设备后，不能直接将挂载设备从系统拔出，否则会出现问题，严重的会导致系统崩溃。用户必须先执行卸载命令然后再该设备拔出！

使用U盘



1　创建挂载点目录

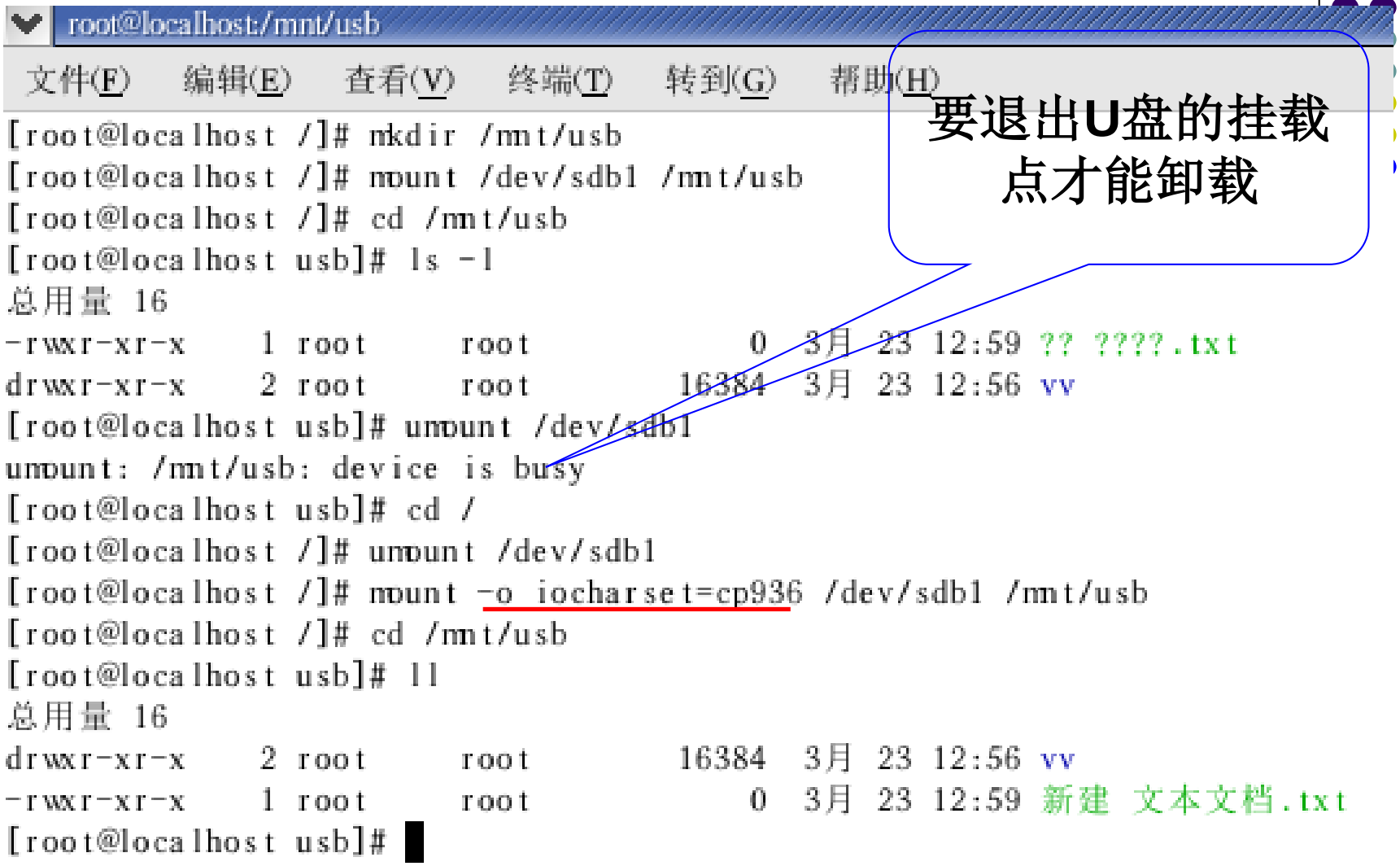
```
# mkdir /mnt/usb
```

2 挂载和使用U盘

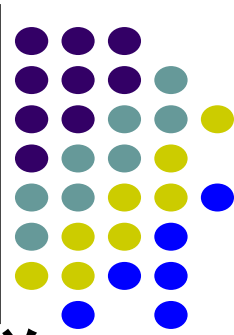
```
#mount      /dev/sda1    /mnt/usb
#cd /mnt/usb
```

3 卸载U盘

```
#umount /mnt/usb
```



要退出U盘的挂载点才能卸载

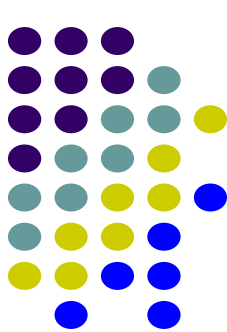


- umount 命令的功能是卸下已挂上的文件系统，在关闭系统前应该把所有挂载上的文件系统卸载。这个命令和mount是相对的。
- 语法：umount [设备名|加载点]
  - 卸下已挂上的光盘：

    umount /dev/cdrom
  - 卸下已挂上的某个分区：

    umount /dev/hdb1

7.3 管理磁盘的Shell命令



- df 命令用来检查各硬盘分区和已挂上来的文件系统的磁盘空间，显示文件系统的相关信息。

  语法：df [选项]

  主要各选项说明：

  -a 显示全部的文件系统和各分区的磁盘使用情况。

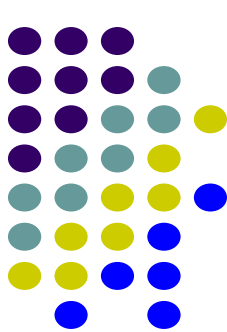
  -k 以k字节为单位显示。

  -t 文件系统类型 仅显示指定文件系统的使用情况。

  -x 文件系统类型 显示指定文件系统以外的其他文件系统的使用情况。

  -T 列出每个分区所属文件系统的名称。

df 命令

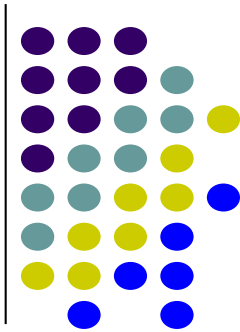


- df命令不加任何可选参数,依次显示的是文件系统,空间大小,以用,未用,百分数表示的已用空间,挂载位置。

$ df

文件系统	1K-块	已用	可用	已用%	挂载点
/dev/sda2	3834496	2580104	1059604	71%	/
/dev/sda1	101089	16079	79791	17%	/boot
none	22596	0	22596	0%	/dev/shm

du 命令



- du命令是查询档案或目录的磁盘使用空间。主要功能是显示文件或目录的大小。
- 语法：

  du [-选项] [文件名或目录名]

主要各选项说明：

a：显示全部目录及其次目录下的每个文件所占的磁盘空间

b：显示目录或文件的大小，以byte为单位。

c：最后再加上一个总计

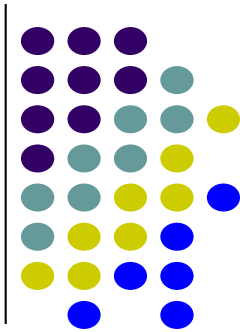
h：以k(kb)、M(MB)、G(GB)为单位，提高信息的可读性。

s：只列出各文件大小的总和

x： 计算指定文件系统的文件

例如：列出所有文件和目录所占的空间： du -ab

fsck 命令



- fsck命令的功能主要是检查和修复linux文件系统，这个命令最好在没有分区挂上来时使用。

  格式：fsck [-选项] 分区名称

主要各选项说明：

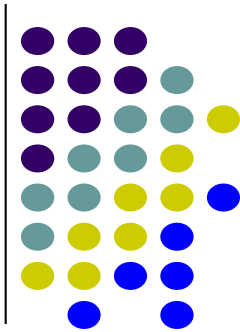
a :自动修复文件系统，不询问任何问题。

A :对/etc/fstab 中所有列出来的 分区做检查

r :采用交互式操作，在修复时询问问题，让用户确认并决定处理方式。

s :依次检查作业而不是同时执行。当依次指定多个文件系统且采用互动的方式进行检查时，使用此参数以序执行。否则fsck可能同时询问数个问题，让人不知所措。

fsck 命令



V :详细显示模式，显示命令执行的过程。

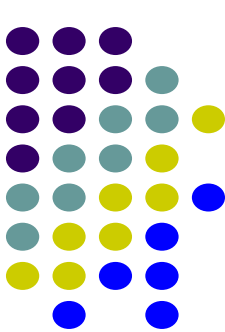
t :指定要检查的文件系统的类型

例如: 检查 msdos 档案系统的 /dev/hda5 是否正常，

如果有异常便自动修复：

fsck -t msdos -a /dev/hda5

文件的备份和压缩

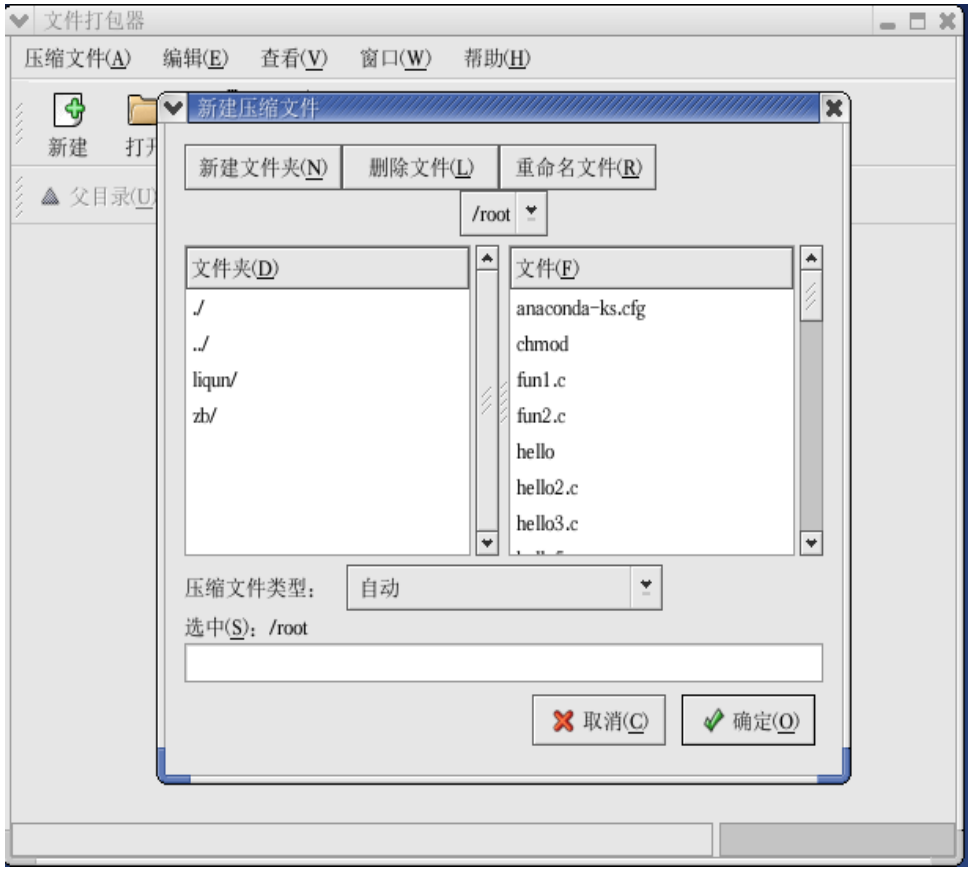


- 有时候，我们需要把一组文件存储成另一个文件，以便备份或传输到另一个目录甚至另一个计算机上或者需要把文件压缩成一个文件，而使它们仅使用少量磁盘空间并能更快地通过互联网下载。
- 可以通过两种方式来压缩、解压、归档文件和目录：图形化的压缩工具“文件打包器”和shell界面。

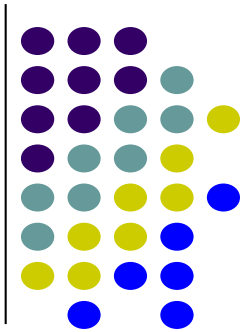
文件压缩

- 图形化的压缩工具：

主菜单 →“附件” →“文件打包器”



tar 命令



tar 命令

语法：tar 选项 归档/压缩文件 [文件或者目录]

功能：将多个文件或目录归档为tar文件，使用相关选项还可以压缩归档文件。

- 使用该命令时，主选项是必须要有的，它告诉tar要做什么事情，辅选项可以选用。

主要选项说明：

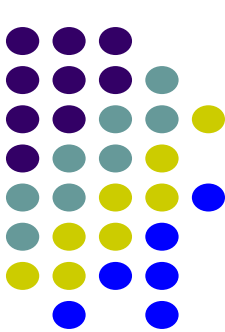
-c： 创建归档/压缩文件。

-f： 当与 -c 选项一起使用时，创建的 tar 文件使用该选项指定的文件名； 当与 -x 选项一起使用时，则解除该选项指定的归档。

-x： 还原归档/压缩文件中的文件和目录

-t： 显示归档/压缩文件的内容，查看已经备份了哪些文件。

tar 命令



-u： 更新文件。就是用新增的文件取代原备份文件，如果在备份文件中找不到要更新的文件，则把它追加到备份文件的最后。

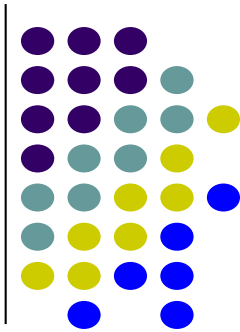
-v： 详细报告tar处理的文件信息。如无此选项，tar不报告文件信息。

-w： 每一步都要求确认。

-z： 使用 gzip 方式压缩/解压缩归档文件。

-j： 使用 bzip2 来压缩/解压缩归档文件。

tar 命令



例1：把/root/abc目录包括它的子目录全部做备份文件，备份文件名为abc.tar

tar -cvf abc.tar /abc

例2：查看abc.tar文件的内容

tar -tvf abc.tar

例3：将打包文件abc.tar解包出来

tar -xvf abc.tar

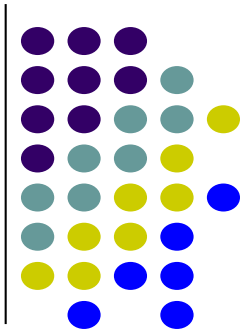
练习1：将文件root/abc目录包括子目录全部做备份文件，并进行压缩，压缩文件名为abc.tar.gz

tar -zcvf abc.tar.gz /root/abc

练习2：将压缩文件bc.tar.gz解压

tar -zxvf abc.tar.gz

gzip命令



gzip 命令

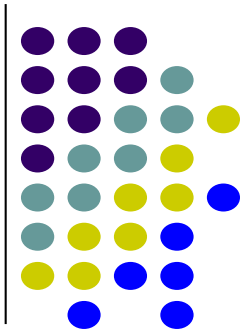
格式：gzip [选项] 文件|目录

功能：压缩/解压缩文件。无选项参数时执行压缩文件。压缩后产生扩展名为.gz的压缩文件，并删除源文件。

主要选项说明：

-c 将输出写到标准输出上，并保留原有文件。
-d：将压缩文件解压。
-l：以长格式列出压缩文件的信息，包括压缩文件大小，未压缩 文件大小，压缩比例，未压缩文件名字。
-r：递归的查找指定目录并压缩或解压缩其中的所有文件。
-v：对每一个压缩和解压文件显示文件名和压缩比例。

gzip命令



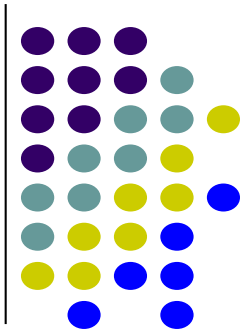
例4：假设一个目录/home下有文件mm.txt、sort.txt、xx.com。把/home目录下的每个文件压缩成.gz文件。

```
$ cd /home
$ gzip *
$ ls
m.txt.gz sort.txt.gz xx.com.gz
```

练习3：把例4中每个压缩的文件解压，并列出详细的信息。

```
$ gzip -dv *
mm.txt.gz 43.1%-----replaced with mm.txt
sort.txt.gz 43.1%-----replaced with sort.txt
xx.com.gz 43.1%-----replaced with xx.com
$ ls
mm.txt sort.txt xx.com
```

gunzip命令



gunzip命令

格式：gunzip 选项 文件列表

功能：解压缩用gzip命令（以及zip命令）压缩过的文件

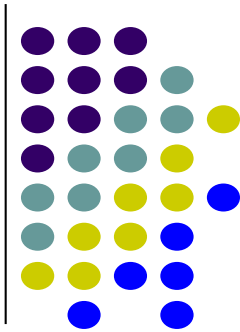
选项：

-c　将解压后的文件输出到标准输出。

-l　列出压缩文件中的文件而不解压缩。

-r　递归解压缩，解压缩命令行所指定目录中的所有子目录内的文件。

zip命令



zip命令

zip [选项] 压缩文件 文件列表

功能：将多个文件归档后压缩为zip文件

主要选项说明：

-m： 压缩完成后删除源文件

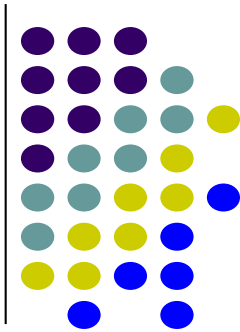
-r： 按目录结构递归压缩目录中的所有文件

-v ： 显示版本资讯或详细讯息。

例5：将当前目录下的文件a压缩成zip文件

zip a.zip a

unzip命令



unzip 命令

格式：unzip [选项] 压缩文件名

功能：解压缩扩展名为.zip的压缩文件

主要选项说明：

-l： 查看压缩所包含的文件

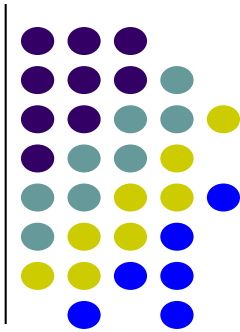
-t：测试压缩文件是否已损坏

-d：目录名，把压缩文件解压缩到指定目录

-n：不覆盖已经存在的文件

-o： 强制覆盖同名文件

unzip命令



练习4：

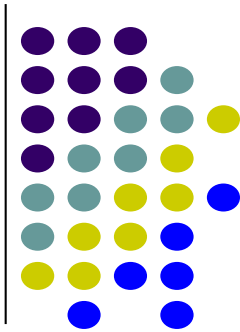
将压缩文件a.zip在当前目录下解压缩

unzip a.zip

将压缩文件a.zip在指定目录/root下解压，并不覆盖原先的文件

unzip –n a.zip –d /root

bzip2命令



bzip2命令

格式：bzip2 [选项] 文件|目录

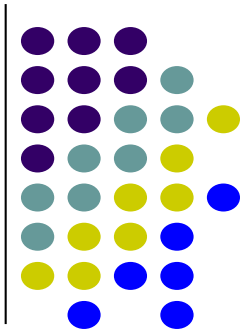
功能：压缩/解压缩文件。无选项参数时执行压缩操作，压缩后产生扩展名为.bz2的压缩文件，并删除源文件。

主要选项说明：

-d 解压缩文件，相当于使用bunzip2命令。

-v 显示文件的压缩比例等信息。

文件的备份和压缩



- 常用几种压缩文件格式：

tar.gz：用gzip压缩的tar文件

tar.bz2：用bzip2压缩的tar文件

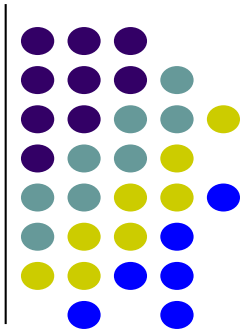
tar：归档但未压缩的文件

zip：zip压缩文件

gz：gzip压缩文件

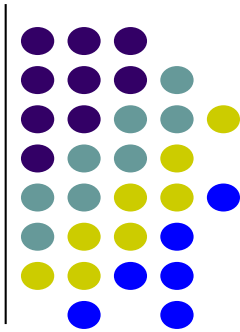
bz2：bzip2压缩文件

7.4 文件管理



- 文件可以简单地理解为一段程序或数据的集合。在操作系统中，文件被定义为一个命名的相关字符流的集合，或者一个具有符号名的相关记录的集合。符号名用来惟一地标识一个文件，也就是文件名。
- 文件定义中所指出的不同基本组成单位表示了两种形式的文件。相关字符流组成的文件是一种无结构文件或流式文件。相关记录组成的文件称为记录式文件。记录式文件通常主要用于信息管理。

文件结构

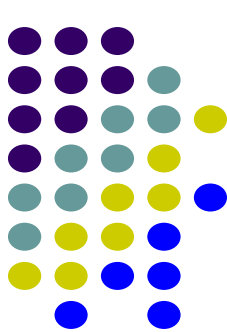


- 系统使用者关心的只是如何方便地组织和使用

文件，即文件的逻辑结构，而系统的设计者则更关心文件具体的实现方式，也即文件的物理结构。

- 文件的逻辑结构是用户可见结构，即从用户角度观察到的文件系统。从逻辑结构来看，Linux系统的文件采用的是字符流式的无结构文件。
- 用户通过对文件的存取访问来完成对文件的各种操作，常用的访问方式包括顺序方式和随机方式。Linux系统同时支持顺序和随机两种访问方式。

文件结构



- 文件的物理结构

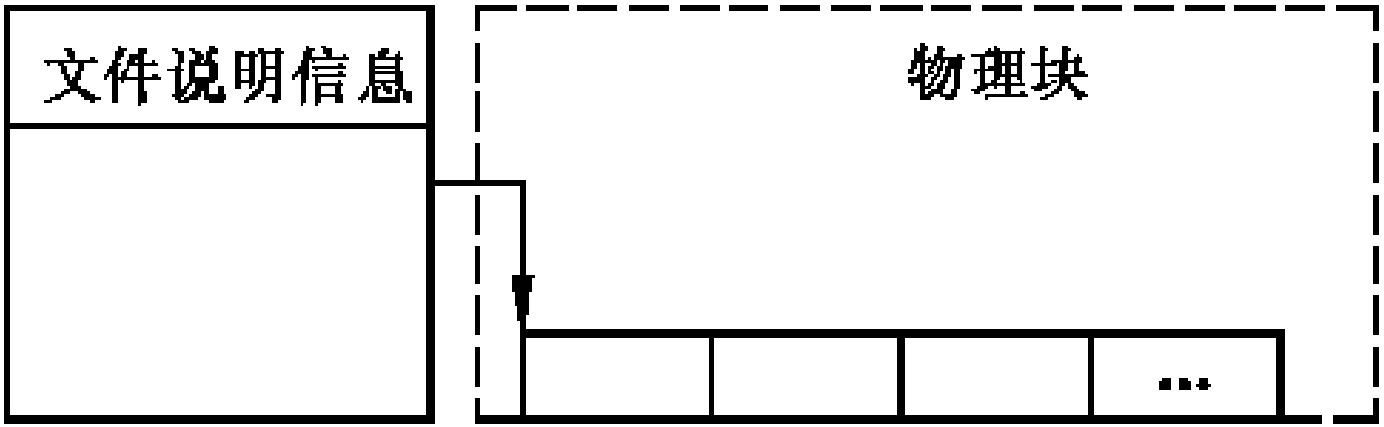
指的是文件在存储设备上的组织方式，即一个文件在具体设备上的实现方法。外部存储器按照一定的规则划分为小的物理单元，称为块，每个块大小相等，相应地，文件信息也划分为和外存物理块相等大小的逻辑块。对于无结构文件来讲，文件本身没有特定的逻辑结构，最小的单位是字节，可以很方便地划分为逻辑块。

- 文件的物理结构通常解决文件的逻辑块与物理块之间的映射关系，也即一个文件对应的物理块的具体组织方式。常用的文件物理结构有顺序、串联、索引和多重索引等方式。

文件结构

1. 顺序方式

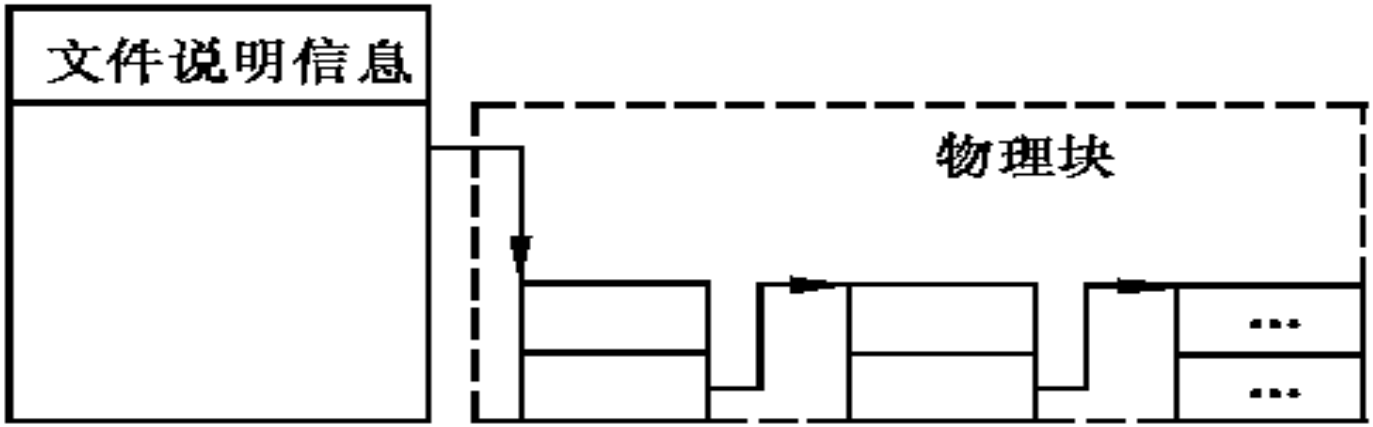
顺序方式按照逻辑上的连续关系把文件依次存放在连续的物理块中。如下图所示，文件逻辑块和物理块之间是简单的线性关系，很容易实现，也可以方便地实现文件的顺序和随机访问，只要确定了文件头的位置和文件长度，一切操作都可以快速实现。但这种方式也容易造成磁盘碎片。



文件结构

2. 串联方式

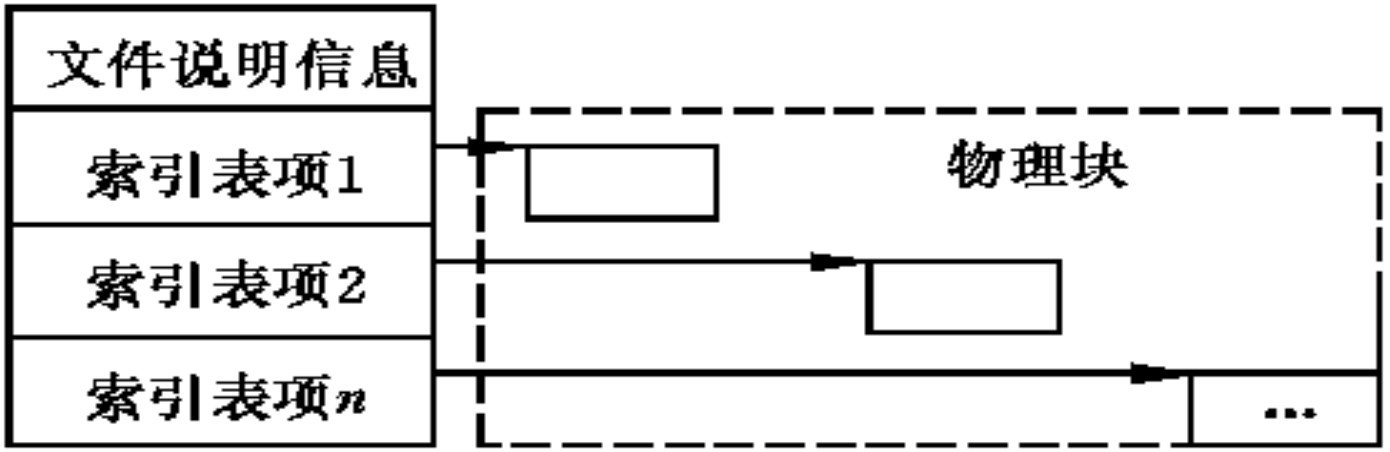
串联方式的结构采用链表来实现，每一个物理块分为两个部分，一部分用于存放数据，另一部分存放指针，指向后续连接的下一个物理块。存放同一个文件的物理块不需要连续。如下图所示。这种方法没有连续方式那样的碎片问题，可以实现文件的动态增长，也可以在文件中任何一个部位增加和删除内容。文件的顺序访问也可以方便地实现，但是随机访问的实现变得很复杂。



文件结构

3. 索引方式

索引方式为每一个文件建立一张索引表，表中存放文件的逻辑块与物理块之间的对应关系。也就是把串联方式中每一个接点所记录的链表指针信息统一保存在一个索引表中，如下图所示。这使得文件的非连续存放、文件动态增长、方便的文件内部修改等串联方式的优点都保留了下来，同时，也能够方便地实现文件的随机访问。



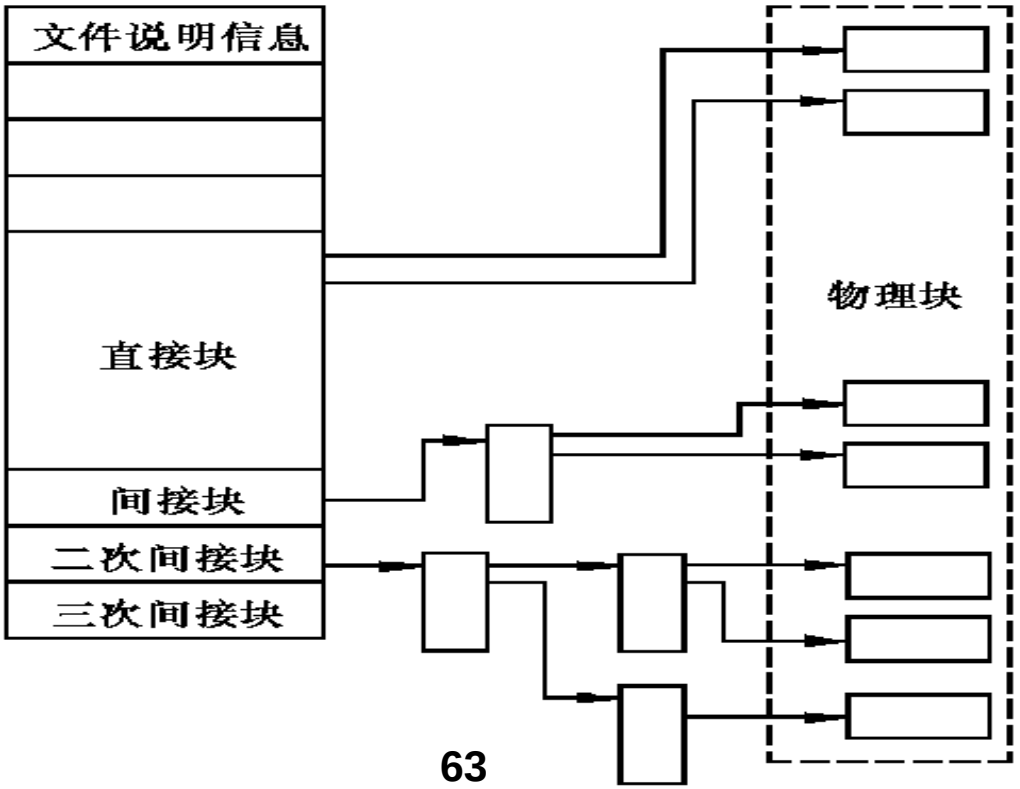
文件结构

多重索引

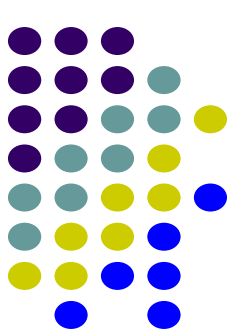
- 实际中，很多操作系统采用一种称为多重索引的物理结构，同时使用单级索引和多级索引。每个文件由一张索引表描述，表中一部分内容直接指向物理块，称为直接块。索引表中同时还包含一部分二级索引指针，指向新的索引表。
- 类似的，文件索引表中还可以包括三级甚至更高级的索引指针，提供更多的间接块。对于比较小的文件，可以直接存放在一次间接块提供的空间中，而对于较大的文件，才使用多次间接块。
- 这种方式的缺点一是索引表本身占用存储空间，二是每次访问数据都需要先访问索引表，需要更多的系统时间开销。

文件结构

- 只要索引表设计合理，能够保证正常使用的大多数都只利用直接块，还可以把索引表放在内存中，保证系统的访问效率。这种方式基本上能保证了文件的动态可变，实现了文件高效率地顺序和随机访问，见下图。

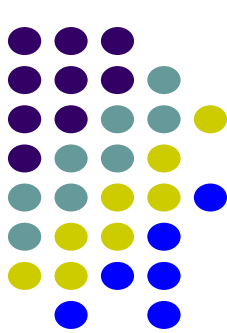


Linux文件



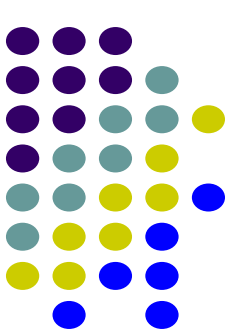
- Linux系统的文件是无结构的流式文件，基于字节流的概念，Linux系统把目录、设备等都当作文件来统一对待。文件的物理结构采用易于扩展的多重索引方式，便于文件动态增长，同时也可以方便地实现顺序和随机访问。Linux默认的是Ext2类型逻辑文件系统。
- Ext2文件系统中的所有文件都采用i节点来描述，每一个文件、目录或者设备都对应于一个且只能对应于一个i节点。
- 每一个i节点包含两个部分的基本参数：文件说明信息和索引表。

Linux文件



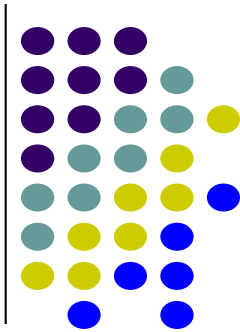
- 根据所存放的位置，i节点又可以分为磁盘i节点和内存i节点，其中磁盘i节点静态存放在外部存储器，包含了整个文件的完整说明信息（包括文件名称、类型、所占物理块数等信息）和物理块分布情况。
- 而内存i节点是为了减少设备存取次数、提高文件访问效率，在内存中特定区域中建立的磁盘i节点的映像。内存i节点除了包含磁盘i节点的说明信息之外，还增加了当前文件打开状态信息。

Linux文件



- 每一个i节点的索引表前12个项记录文件直接物理块的位置，后面3项分别是间接、二级间接和三级间接物理块索引，这种方式既可以保证小文件能够快速存取，又可以适应大文件存储扩展的需要。
- Ext2文件系统物理块的大小可以是1024、2048或者4096字节，默认值是1KB，块地址占4字节，因此每个物理块可以存放256个块的地址，这样，如果把一个i节点所有的物理块放满，得到最大的文件应该是：

Linux文件



直接块+间接块+二次间接块+三次间接块，即：

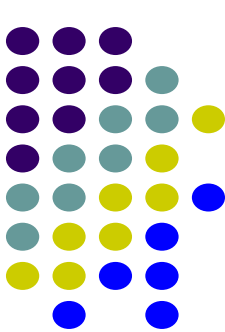
12KB+256KB+64MB+16GB

实际上，在32位PC上的Linux系统中，寻址范围32位

文件最大最多只能达到4GB。

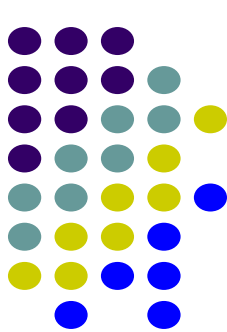
- 如果一个文件的内容小于等于12个块，物理块采用默认值时是12KB，就可以全部使用直接块来存放，能够保证相当高的存取效率。

文件系统布局



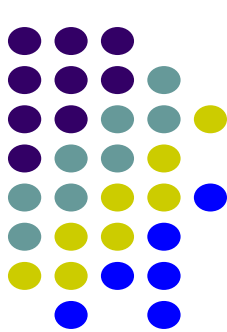
- 一个Linux系统可以支持多个物理磁盘设备，这主要取决于计算机硬件的支持能力，每一个物理磁盘又可以定义多个分区，每个分区是存放逻辑文件系统的最小单位。
- 在Linux系统中，这些分区可以安装不同的逻辑文件系统，系统可以同时使用这些文件系统，而且在用户的眼里，它们表现出基本相同的性能，互相之间可以进行数据传输，这就是Linux虚拟文件系统的优势。
- 每一个逻辑文件系统都按照不同的规则，把自己所控制的磁盘分区定义为一组逻辑块的序列，通常包括引导块、超级块、i节点区和数据区。

文件系统布局



- 引导块在磁盘分区的头部，一般占用一个扇区，用来存放引导程序。在系统启动过程中，根据系统引导设置，该分区引导块的内容被读入内存执行，负责启动操作系统，启动完成之后，引导块不再使用，因此引导块在系统实际运行过程中并不属于文件系统管理。
- 超级块用来记录整个逻辑文件系统的基本管理信息，不同的文件系统中定义为不同的数据结构，主要用来记录文件系统的状态，比如文件系统的大小、i节点和物理块的使用情况等等。

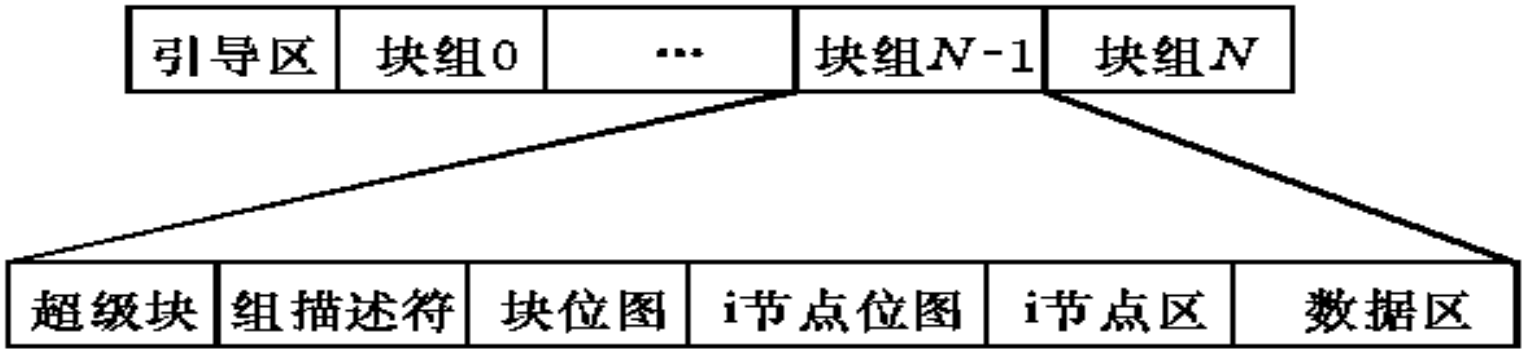
文件系统布局



- i节点区存放这个逻辑文件系统中所有的i节点，第一个i节点是这个系统的根节点，其他所有文件和目录都占用一个i节点。利用根节点，可以把存放在这个磁盘分区的整个文件系统安装到另一个文件系统的枝节点上，作为另外文件系统完整的一棵独立子树。
- 数据区中存放文件系统中的各种物理块，可以存放直接块和各种间接块等文件内容，也可以用来存放各级间接块的地址等管理数据。

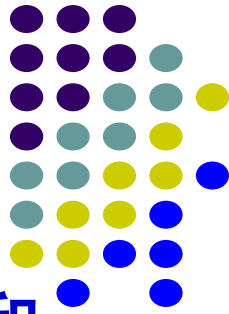
文件系统布局

- Ext2文件系统把它所占用的磁盘分区首先分为引导区和块组。
- 每一个块组中都由超级块、记录所有块组信息的组描述符表、块位图、i节点位图、i节点区和数据区组成，如图所示：



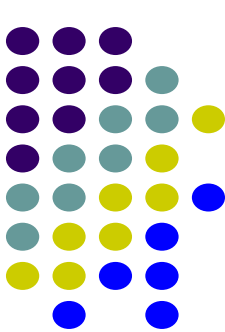
文件系统的布局

文件系统布局



- 在系统启动的过程中，只有第一个块组中的超级块和组描述符被读取到内存空间中，其他块组中的数据作为冗余备份。此外，所有块组中的超级块和组描述符必须保持一致，当系统意外关机遭到部分破坏后，系统启动之后要进行文件系统一致性检查，首先要检查的就是这些数据，冗余数据虽然占用了一部分存储空间，但是提高了文件系统的可靠性。
- Ext2系统采用位示图来描述空闲物理块和i节点，分别称为块位图和i节点位图，记录本块组数据区中物理块和i节点的使用情况。

基本数据结构

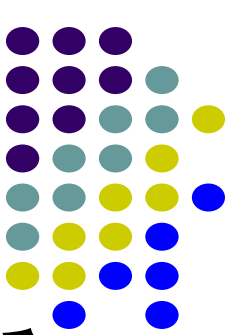


- 超级块和i节点是其中最关键的两个结构，另为组描述符，是描述文件系统块组分布和使用情况的主要数据。

超级块

- 文件系统的超级块用来描述目录和文件在物理设备上的静态分布情况，随着系统文件操作，超级块的内容也在不停地改变。每一个磁盘分区都格式化为一个独立的文件系统，具有自己的超级块。
- 超级块包含物理块和i节点的分配情况等基本参数，同时也用于空闲i节点和物理块的回收，

超级块



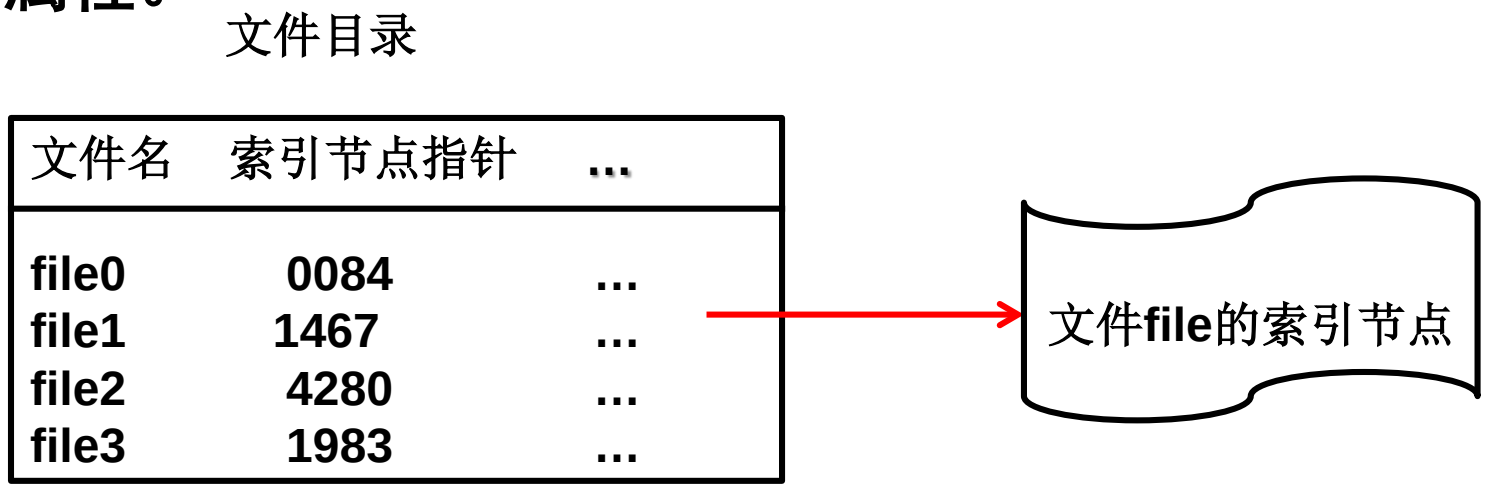
- 根据所存放的位置，超级块也可以分为磁盘超级块和内存超级块。磁盘超级块静态存放在外部存储器。
- 而内存超级块是文件系统启动时由外部超级块初始化形成，除了包含外部超级块所记录的信息之外，还增加了描述文件系统当前状态的动态信息和指向文件系统缓冲区的指针。
- 内存超级块是系统内核其他部分进行文件操作的接口，内存超级块在文件操作发生之后通常会被修改，系统定期用内存超级块的数据更新磁盘超级块的内容。

组描述符

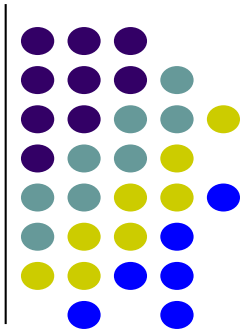
- 块组是文件系统划分和管理文件系统的单位，每个块组都由一个组描述符来具体描述，组描述符主要记录了该块组中空闲物理块和i节点的数目和描述它们的位图表的位置等信息，通过它可以管理整个块组内存储空间的分配和回收，它在块组中的地位就像超级块在整个文件系统中的地位。
- 文件系统的所有块组的描述符集中在一起，形成组描述符表，这张表在每一个块组中都保存一个相同的副本，以便得到有效的保护。组描述符表位于该块组中超级块之后，可能占用多个数据块。

i节点

- 为了加快对文件目录的检索，Linux将文件属性从目录中抽出，放入索引节点（i节点）中，并将索引节点指针或索引节点号放在文件目录中。目录就是将文件的名称和它的索引节点号结合在一起的一张表，用文件名首先在文件目录中查找文件索引节点指针，通过索引节点指针得到索引节点，在索引节点中得到文件的属性。



i节点

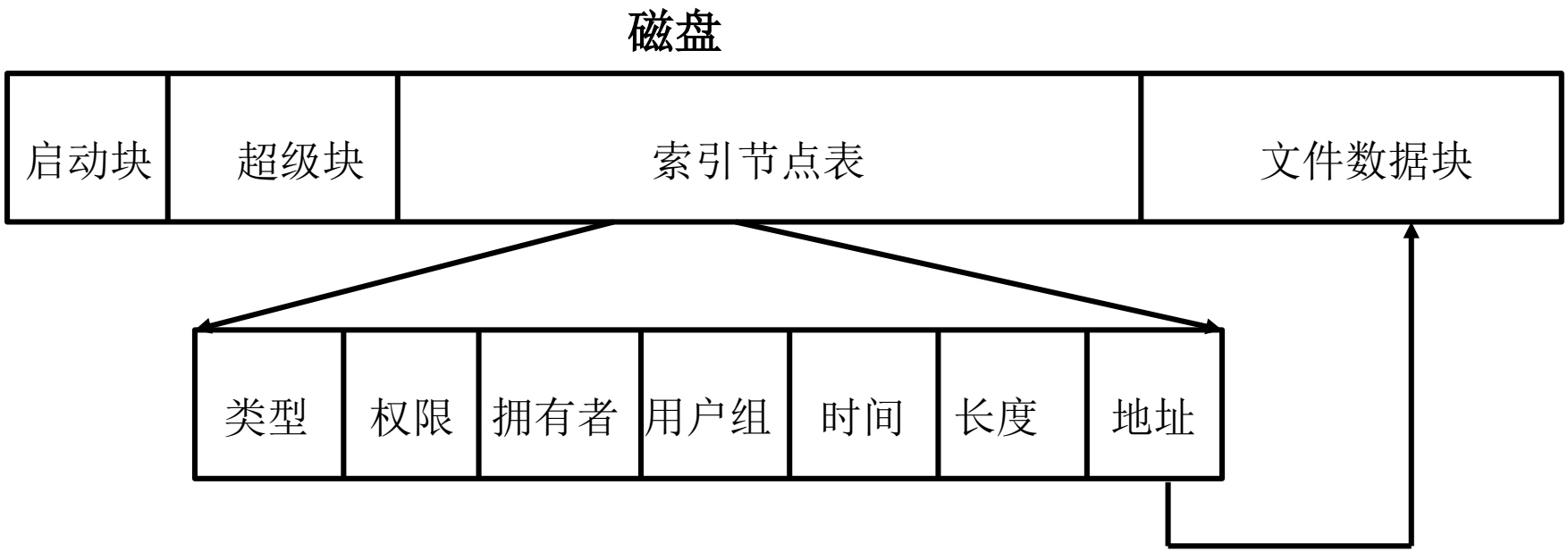


- Linux的索引结点，类似Windows操作系统文件分配表。
- 文件存储在磁盘上，索引节点也存储在磁盘上。它包含以下信息：
  - 文件类型：普通文件、目录文件、设备文件等
  - 文件访问权：文件所有者权限、文件所有者同组权
  - 限和其他用户权限。
  - 文件链接数目：在文件目录树中指向同一文件名的文件个数。
  - 文件所有者标识符：通常为创建文件的用户标识。
  - 文件所有组标识符：与文件同组的用户组标识符。

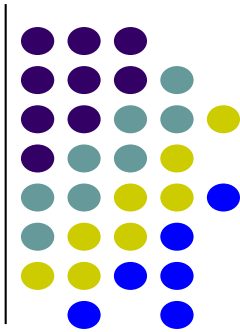
i节点

- 文件创建时间：创建文件的日期和时间
- 文件存取时间：文件最后修改的时间、最后访问的时间。
- 文件的长度：　文件所占的字节数。
- 文件数据的磁盘地址：文件在磁盘中的位置信息等。

- Linux磁盘索引节点与文件数据及磁盘块的关系如图：



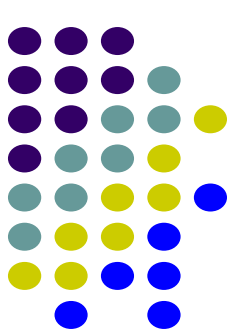
i节点



- 目录中每一对文件名称和索引节点号称为一个链接。一个文件有唯一的索引节点号与之对应，一个索引节点号，却可以有多个文件名与之对应。因此，在磁盘上的同一个文件可以通过不同的路径去访问它，这就是链接的概念。链接又分软链接和硬链接。
- 软链接将会生成一个很小的链接文件，该文件是要链接到的文件的路径，可指向位于任意一个文件系统的任意文件，也可指向一个不存在的文件。原文件删除软链接文件就失去作用，删除软链接对原文件无任何影响。
- 创建软链接，使用带-s(symbolic link)选项的ln命令：

  ln –s 原文件或目录名　要链接的文件或目录名

i节点



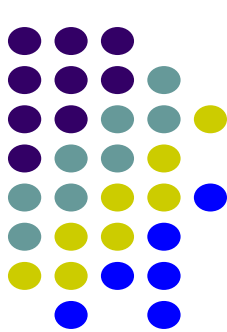
硬链接

- 文件被写到硬盘上的某个物理位置，该物理位置就是通过索引结点获得的文件内容的一个入口地址。创建硬链接，就是创建了另外一个指向同一索引节点的文件，也就是多个逻辑文件同时指向一个物理文件，删除任何一个逻辑文件，只是将索引节点中的文件链接数减一，直到减为零，才会删除物理文件。由于删除文件要在同一个索引节点属于唯一的连接时才能成功，因此可以防止不必要的误删除。
- 硬链接其实就是为物理文件增加一个逻辑文件名。创建硬链接使用不带-s参数的ln命令来创建:

  ln 原文件 要链接的文件名

  硬链接不能跨文件系统，不能跨越不同的分区建立硬链接；不能对目录建立硬链接。

存储空间管理

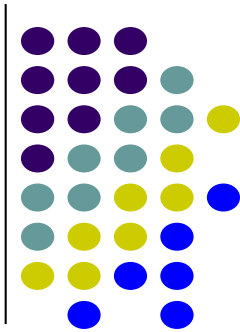


- Linux文件系统采用分块的组织方式，利用多重索引的物理文件结构来实现外部存储空间的管理。

块的大小

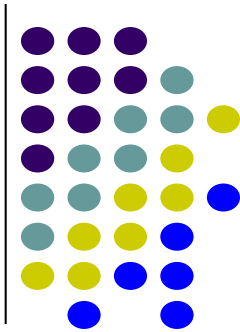
- 逻辑文件系统中，块的大小可以在一定范围内调整，但是在格式化磁盘分区生成文件系统时必须指定块的大小。因此一个实际文件系统的块的大小都是固定的，块的大小对文件系统性能和存储空间利用率有相当大的影响。
- 如果块的基本单位选取比较大的话，存储空间的利用率就可能比较低。反过来，如果存储单元选取很小，文件系统的时间效率就很低。

存储空间管理



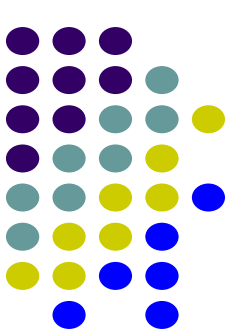
- 通常文件系统块的大小可选值为512、1K、2K或者是4K字节等。
- 块的大小根据文件系统中文件的平均长度来选取比较合适。Linux系统中采用多重索引方式来组织文件，块大小选取的标准是：能够保证大部分文件都可以用12个直接块存放，这样，每次寻址只要访问i节点就可以得到物理块的地址，能够保证相当高的时间效率。

空闲空间管理



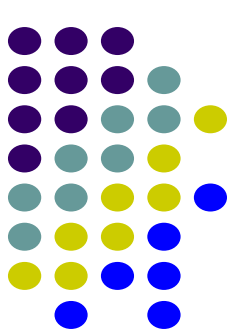
- 块的使用情况，具体记录在每一个文件的i节点中，而i节点的使用情况，记录在目录项中，这些都可以通过文件系统实现快速搜索和管理。
- 文件系统记录空闲存储空间，包括记录空闲块和空闲i节点，广泛采用的两种基本方式是空闲链表和位示图。
- 空闲链表把存储设备上的所有空闲块装入一个链表中，每一个空闲块中都记录着下一个空闲块的地址指针。

空闲空间管理



- 通常以多个相邻空闲块组成的空闲区作为链表的节点单元，分配和释放空闲块也尽可能以空闲区为单位，可以按照空闲区大小顺序或者释放顺序来组织链表。当申请使用硬盘空间时，从链表头开始搜索并分配合适的空闲区，然后相应地调整链表指针，当回收空闲块时，把释放的存储空间按照空闲区为单位加入到链表中。
- 空闲链表具有较好的空间效率，但是时间效率比较低。

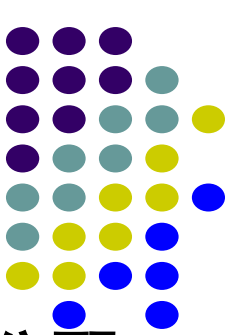
空闲空间管理



- 位示图方法是在存储区中划分一个单独的区域建立位图，位图中的每一个比特位数据都对应于存储区中的一个块，使用“0”来表示空闲，使用“1”来表示已经使用（或者相反）。

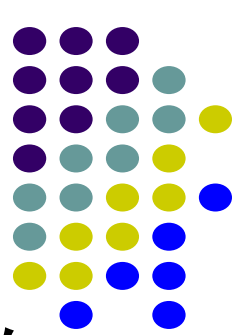
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	0	0	0	1	1	1	0	0	1	0	1	1	1	0
1	0	0	0	1	1	1	1	1	1	0	0	0	0	1	1	1
2	1	1	1	0	0	0	1	1	1	1	1	1	0	0	0	0
3																
⋮																
15																

空闲空间管理



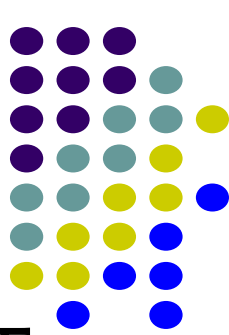
- 位图可以反映整个存储区域中所有块的使用情况。分配和释放的过程中，只要通过查找位图中相应位的数据，就可以确定指定块的使用情况。位图方法需要占用固定的存储空间，但是具有很好的时间效率。
- Ext2文件系统的空闲空间管理采用位图方式，在每个块组中都划分了特定的位图区来记录块和i节点的使用情况，分别称为块位图和i节点位图。每一种位图分别占用一个块的空间。

分配策略



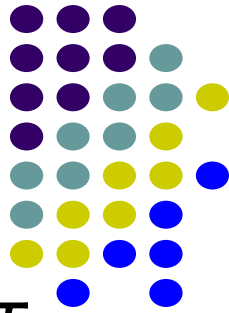
- Linux系统中的每个文件都是通过分配文件物理块的方式把数据存储在存储设备中。不同文件系统有着不同的块分配方法。通常有两种分配策略：块分配和扩展分配。
- 块分配是每当文件大小改变的时候重新为文件分配空间，磁盘上的文件块根据需要分配给文件，这样可以减少存储空间的浪费。
- 但块分配容易造成文件中的文件块不连续，增加了磁盘寻道时间，使得读取一个文件效率低。

分配策略



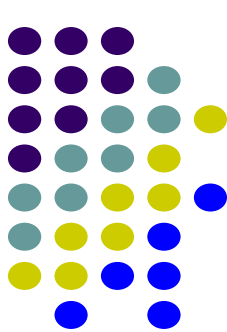
- 每一次当文件扩展时，块分配算法就要写入一些关于新分配的块所在位置的信息，那么就需要很多额外的磁盘I/O用来写入文件块的结构信息metadata。metadata总是一起同时写入存储设备的，使得改变文件大小的操作要等到所有的meta-data的操作都完成之后才能进行。

分配策略



- 扩展分配是预先给文件分配好空间，只有当文件超出预分配的空间时候一次性为文件分配连续的块。当创建一个文件时，很多文件块同时被分配，当文件扩展的时候，也一次分配很多块。文件系统的metadata在文件创建的时候被写入，当文件的大小没有超过所有已分配的文件块的大小时，就不用写入metadata，直到需要再分配文件块的时候。这样可以优化磁盘寻道的方式，也可以成组地分配块，有利于一次写入大批数据到存储设备中，减少设备写数据的时间。

7.5 虚拟文件系统



- 为保证Linux的开放性，设计人员必须考虑如何使Linux支持其他不同的文件系统。为此，就必须将不同文件系统能够的操作和管理纳入到一个统一的框架中能够，使得用户程序可以通过同一个文件系统界面，也就是同一组系统调用，对不同的文件系统以及文件进行操作。这个同一的框架就是虚拟文件系统（Virtual File System)。
- Linux支持的所有文件系统称为逻辑文件系统，而在传统逻辑文件系统的基础上增加了一个的虚拟文件系统的接口层。

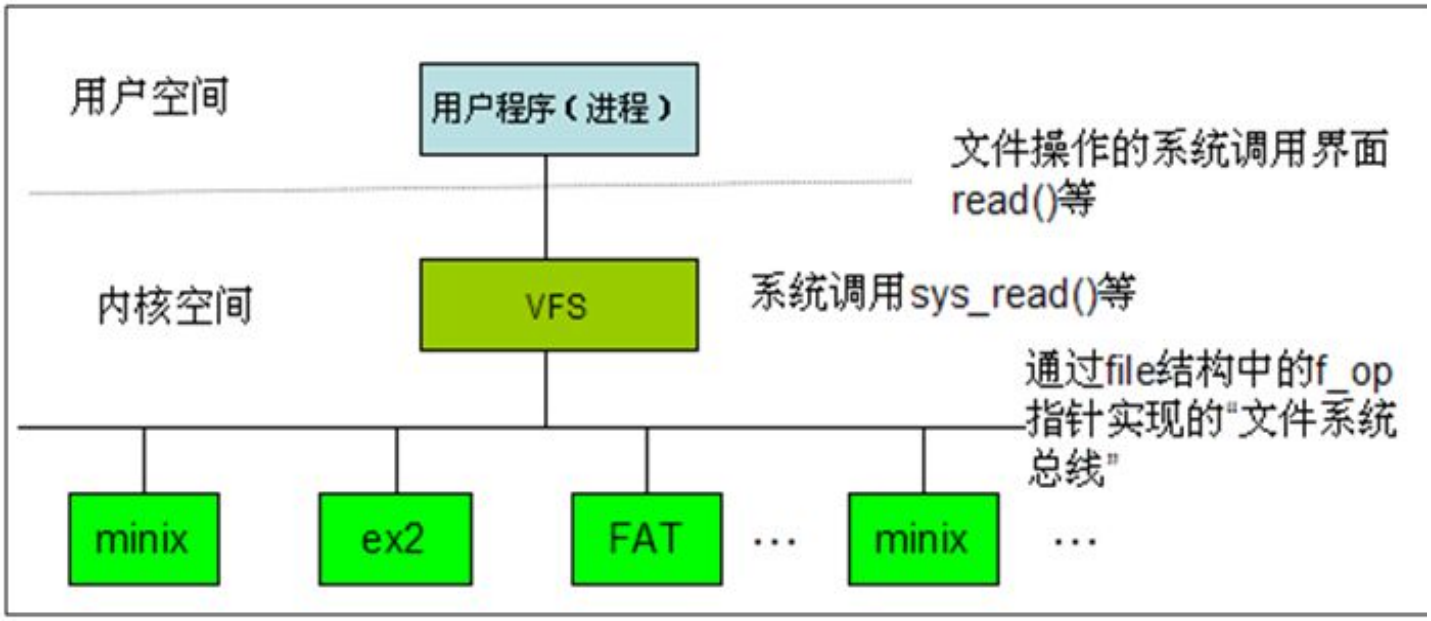
虚拟文件系统

VFS的作用

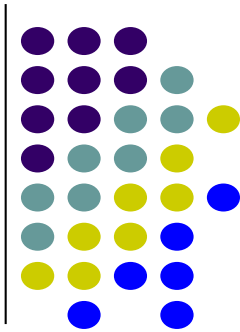
VFS是用户的应用程序与具体文件系统之间的抽象层。

VFS提供的抽象界面主要由一组标准、抽象的操作构成，如read()、open()、write()等。这些函数以系统调用的方式供用户程序调用。

VFS与具体文件系统之间的关系如下图：

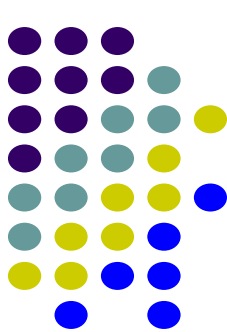


VFS所处理的系统调用



- mount、umount：挂载/卸载文件系统
- sysfs ： 获取文件系统信息
- statfs、fstatfs、ustat ： 获取文件系统统计信息
- chroot ： 更改根目录
- chdir、fchdir、getcwd ： 操纵当前工作目录
- mkdir、rmdir ： 创建/删除目录
- getdents、readdir 、link 、unlink 、rename ： 对目录项进行操作
- readlink 、symlink ： 对符号链接进行操作
- chown 、fchown 、lchown ： 更改文件所有者
- chmod 、fchmod 、utime ： 更改文件属性
- open、close、create …

虚拟文件系统



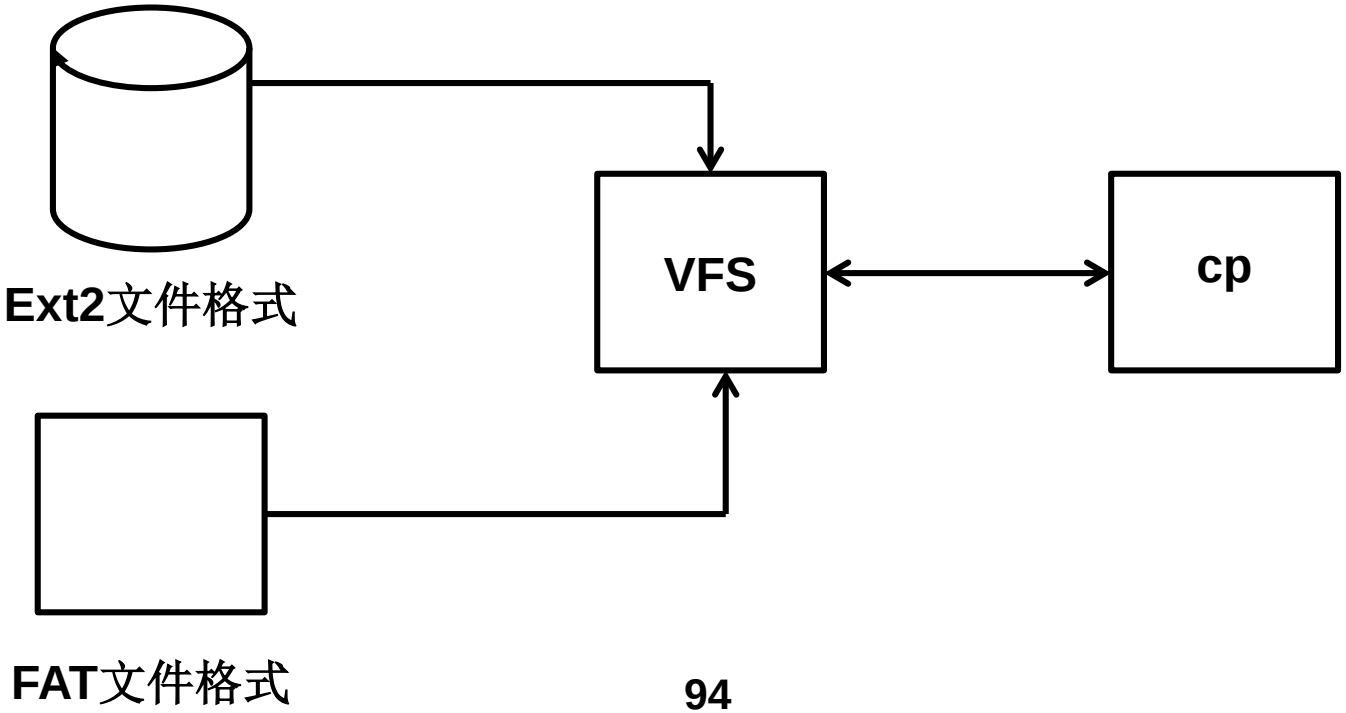
- Linux的目录建立了一棵根目录为“/”的树。根目录包含在根文件系统中，在Linux中，这个根文件通常就是Ext2类型的。其他所有文件系统都可以被安装在根文件系统的子目录中。
- 例如，用户可以通过mount命令，将DOS格式的磁盘分区（即FAT文件系统）安装到Linux系统中，然后，用户就可以像访问Ext2类型的文件一样访问DOS的文件。这时，假设用户输入如下命令：

```
cp   /user/test     /work/test
```

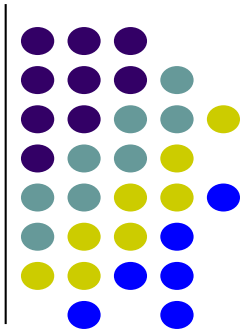
虚拟文件系统

由于VFS是应用程序与具体文件系统之间的抽象层，

因此cp命令并不需要知道/user/test和/work/test各属于什么类型的文件系统，而是通过函数直接与VFS交互。



虚拟文件系统



cp所执行的代码片段：

```
infile=open(“/user/test”,O_RDONLY,0);
outfile=open(”/work/test”,O_WRONLY｜O_CREAT|O_TRUNC,0600);
while ((charnum＝read（infile,buf,4096）)>0)
write（outfile,buf,charnum ）
close(infile);
close(outfile);
```

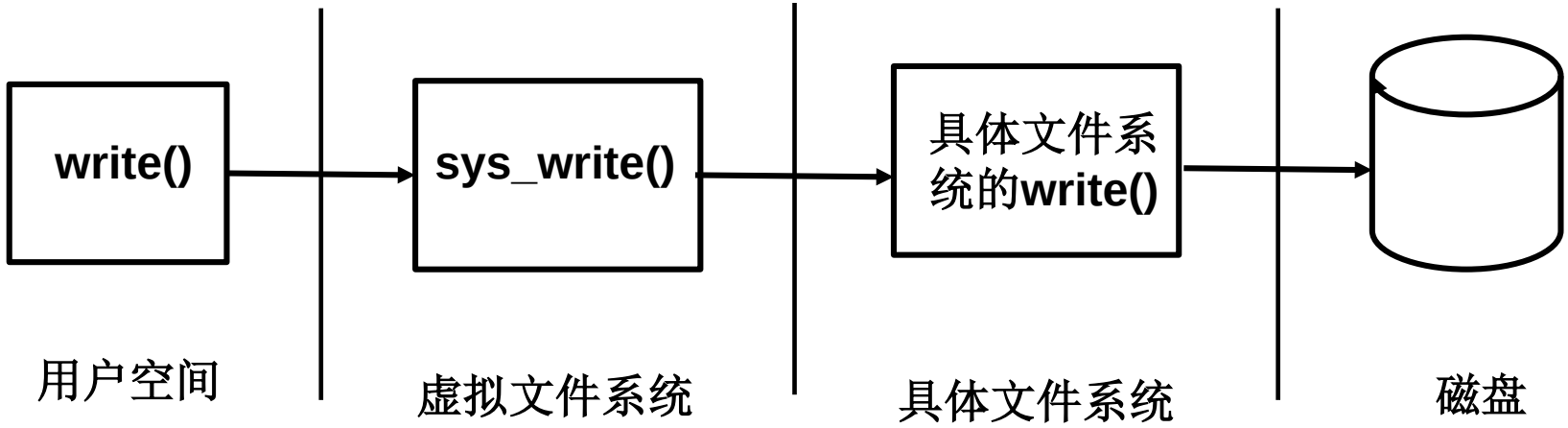
从上述cp代码片段可知，cp操作调用open()、read()、write()、和close()等函数。以write()为例，在用户空间执行的操作：

write(outfile、buf、charnum);

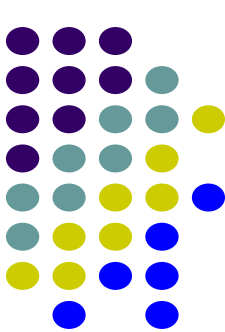
将buf指针指向的、长度为charnum字节的数据写入文件描述符outfile对应的文件的当前位置。

虚拟文件系统

- 该用户操作首先被一个通用内核函数sys_write()处理，sys_write()会找出outfile所在文件系统实际给出的是哪个操作，然后再执行该操作。实际的写操作是具体文件系统的一部分，数据最终通过该写操作被写入磁盘介质中去。一方面函数是VFS提供给用户空间的接口，另一方面调用文件系统的具体实现方法来实现这个操作。

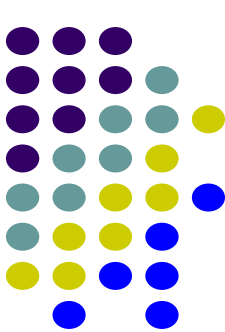


虚拟文件系统



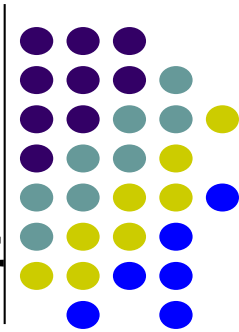
- 打开的文件用系统打开文件表file数据结构来表示，它有一 个成员是指向文件操作表的指针struct file_operation *f_op，其类型为file_operation结构。该文件操作表包含指向各种函数如read( )、write( )、open( )、close( )等 的入口，每种具体文件系统都有自己的file operation结构， 也就是有自己的操作函数。
- 当进程使用open( )打开文件时，将与具体文件建立连接， 并以一个file结构作为纽带，将其中的f_op设置成指向某个具体的file_operation结构，也就指定了这个文件所属的文件系统。

虚拟文件系统



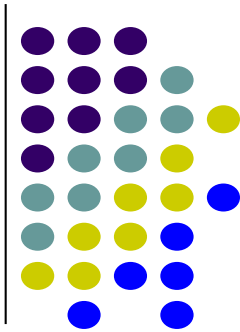
- 当应用程序调用read( )函数时，就会陷入内核而调用sys_read( )内核函数，而sys_read( )就会通过file结构中的指针f_op去调用DOS文件系统的读函数，即：file→f_op→read( )函数。同样道理，write( )操作也会 引发一个与输出文件相关的Ext2的file→f_op→write( )写函数的执行。
- 可见，为使多个文件系统能够共存一个操作系统中，并向用户屏蔽差异，VFS需要从不同的具体文件系统中抽象出共性，并进行定义，以明确统一的界面（如file_operation及其他接口)。这样，用户看到的是相同的文件操作方式。

虚拟文件系统



- 当然，这也需要具体文件系统将自身的诸如“如何打开文件”等概念在形式上与VFS的定义保持一致。
- VFS是一个软件层，位于文件系统的最上层，虚拟文件系统只存在于内存中，不真正存在于磁盘分区中，所有虚拟文件系统的数据结构都是在系统启动之后才建立完成，并在系统关闭时撤销。
- VFS对逻辑文件系统进行抽象，采用统一的数据结构在内存中描述这些文件系统。这样，VFS只负责处理设备无关的操作，主要进行具体操作的映射关系。通过VFS操作函数，按照一定的映射关系，把这些访问重新定向到逻辑文件系统中相应的函数调用，来完成具体操作。

关键流程步骤



应用程序中open(“/d1/f1”)

↓

通过系统调用进入内核

↓

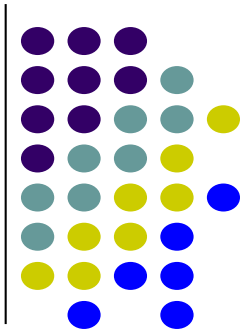
VFS判断/d1/f1所属文件系统类型

↓

根据文件操作表跳转至相应具体类型文件系统的open操作子程序

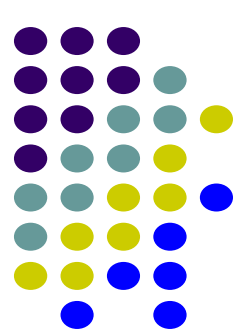
- 上述大部分操作只需要与通用文件模型中的一些对象打交道，而不需要真正操作具体的文件系统和文件，因此可以把VFS看成是一个“通用”的文件系统，在必要时依赖某种具体的文件系统。
- 为何不是1次转换那么简单，更多工作有哪些?

虚拟文件系统



- 安装点和链接
  - /d1/f1路径上可能存在两个不同类型的文件系统
  - 需逐个分量逐次调用具体类型文件系统，而不能一次性的转交整个路径名。
  - 因此,与具体类型文件系统的交互(即调用次数)会增多。
- 内存缓冲数据结构进入VFS
  - 内存活动索引结点，目录项，超级块等
- 为统一管理和减少代码重复，文件系统类型无关的操作（即大部分内存操作，进入VFS，文件系统类型相关的操作（即所有外存操作和少量内存操作，留在具体类型文件系统中。
- 以上这因素决定VFS实际结构和过程。

VFS中对象的演绎



- VFS的基本思想：引入一个通用文件模型，这个模型能够表示所有支持的文件系统
  - 对于一个具体实现的文件系统，在处理时，需要将其进行概念上的转换，类似面向对象的概念。
- 通用文件模型主要有下列对象类型组成：

（1）超级块对象（superblock object）

存放文件系统相关信息：例如文件系统控制块

（2）索引节点对象（inode object）

存放具体文件的一般信息：文件控制块/inode

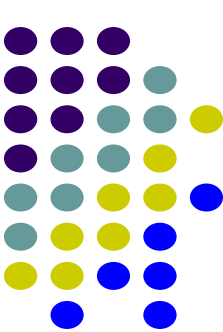
（3）文件对象（file object）

存放已打开的文件和进程之间交互的信息

（4）目录项对象（dentry object）

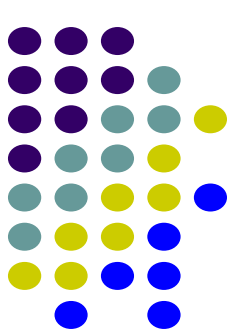
存放目录项与文件的链接信息

VFS超级块

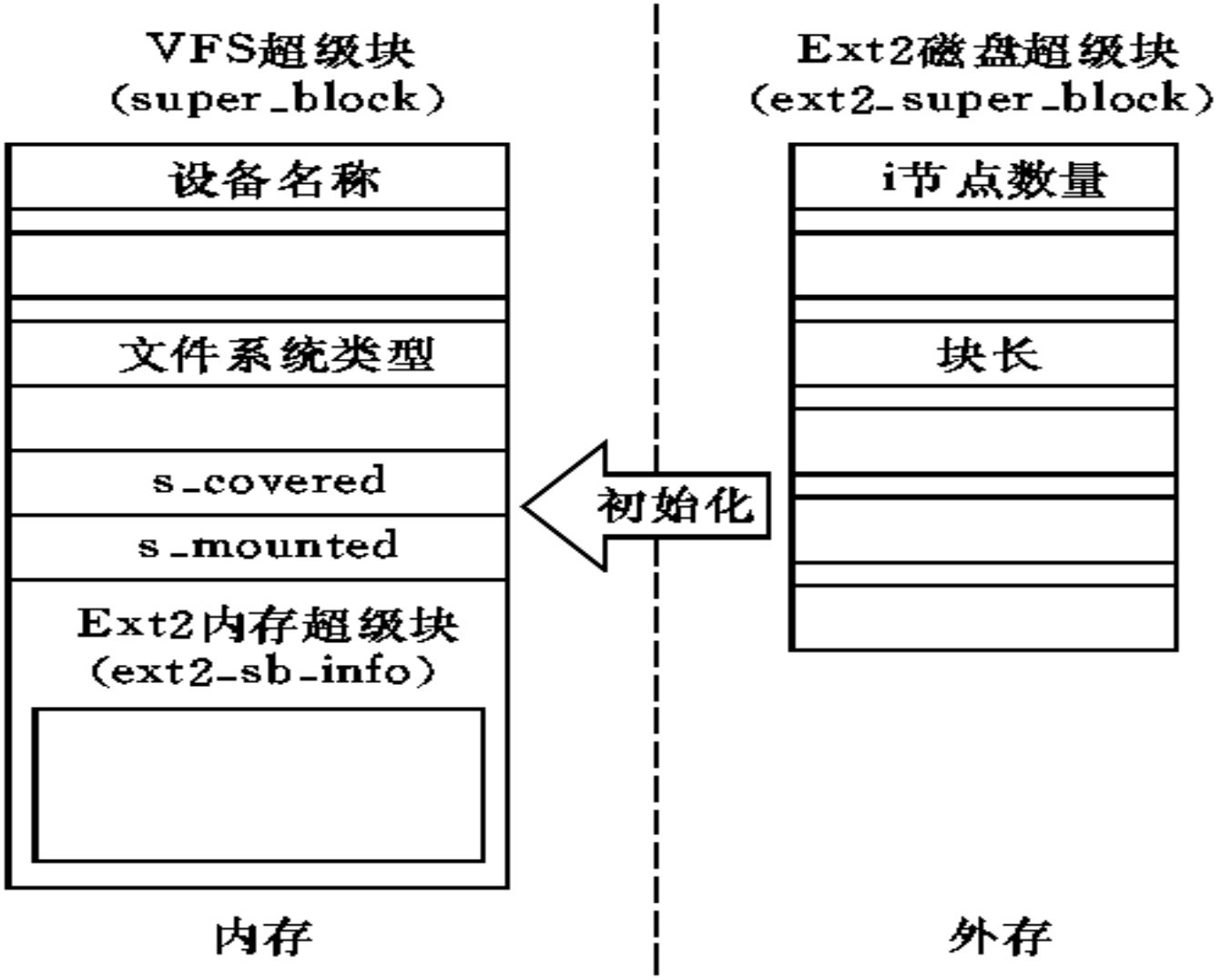


- 超级块用来描述整个文件系统的信息。对每个具体的文件系统来说，都有各自的超级块，如Ext2超级块和Ext3超级块，它们存放在于磁盘上。
- 当内核在对一个文件系统进行初始化和注册时在内存为其分配一个超级块，这就是VFS超级块。VFS超级块是各种具体文件系统在安装时建立的，并在这些文件系统卸载时被自动删除。

VFS超级块

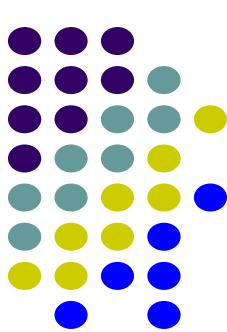


- VFS超级块使用数据结构super-block来描述。在VFS超级块中，记录具体逻辑文件系统所在的设备号、类型、第一个i节点号等基本信息，包含了对该文件系统文件操作和存储空间限额管理函数的入口指针和所对应的具体逻辑文件系统的内存超级块。
- 虚拟文件系统就是通过VFS超级块中记录的内存超级块来访问并管理具体的逻辑文件系统，文件操作通过VFS中记录的函数入口指针映射得到对应的函数。下图以Ext2文件系统为例描述了逻辑文件系统磁盘、内存超级块和VFS超级块之间的关系。



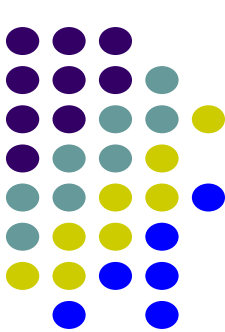
Ext2磁盘超级块、内存超级块和VFS超级块关系示意图

VFS超级块



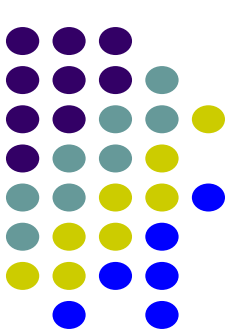
- 整个Linux文件系统形成一个完整的目录树，每一个逻辑文件系统都可以作为一个独立的子树安装到目录树的某个节点上。因此，VFS超级块中采用指针s-covered记录着该文件系统在另外一个文件系统中安装点的信息，同时VFS超级块中还使用指针s-mounted记录指向该逻辑文件系统的第一个节点的位置。对于根文件系统，s-mounted就是根目录的i节点，它没有安装到其他文件系统，s-covered指针无效。对于安装在根文件系统下的文件系统，访问该文件系统管理的文件首先要通过根文件系统，根文件系统中的安装点指针s-covered就是该文件系统的访问入口，指向该文件系统的根节点。

VFS索引节点



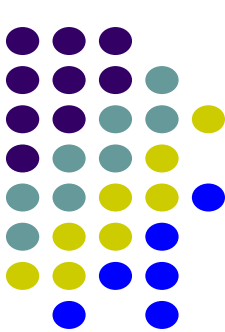
- 文件系统如何对文件的属性（文件名、访问控制权限、大小、拥有者、创建时间等信息）进行描述，这就是索引节点。索引节点号可以唯一标识文件。
- VFS i节点用来描述虚拟文件系统中的文件和目录，同样的，VFS i节点只存在于内存中，是所有逻辑文件系统i节点的抽象描述，和具体逻辑文件系统的i节点一一对应，并根据该文件系统的i节点信息建立。

VFS索引节点



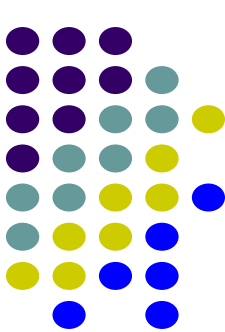
- VFS i节点包含具体i节点所在设备的设备号和磁盘i节点号、文件说明信息、文件在内存中的分布和使用信息、锁定和修改标志、一组VFS i节点操作函数指针和一个具体逻辑文件系统内存i节点。
- 设备号和磁盘i节点号在整个文件系统中惟一确定一个VFS i节点。在内存中，总是同时存在多个VFS i节点，每一个已经使用的VFS i节点表示某设备上一个具体存在的文件或者目录的i节点，所有VFS i节点组织成一个双向链表，存放在系统内核空间中，链表的头节点之前存放空闲节点，而链表头之后依次是分配出去的VFS i节点。

VFS目录项



- 文件系统通过目录来组织文件。目录也可以包含子目录，所以目录可以层层嵌套，形成文件路径。路径中的每一部分被称为目录项。例如/home/test/myfile是文件路径，其中根目录是/，home，test和文件myfile都是目录项。在Linux系统中，目录属于普通文件，所以对目录和文件可以实施同样的操作。
- VFS将目录作为一个文件来处理，目录项不同于目录，但目录和文件相同。

VFS文件对象



- 文件是最终要被进程访问的，一个进程可以打开多个文件，而一个文件可以被多个进程同时访问。在这里是通过文件描述符来抽象所打开的文件。file结构中有指向该文件索引节点的指针。
- VFS文件对象代表由进程打开的一个文件，存放已打开的文件与进程的交互信息，这些信息仅当进程访问文件期间才存在于主存。