# Python开发之运维基础

讲师：王晓春

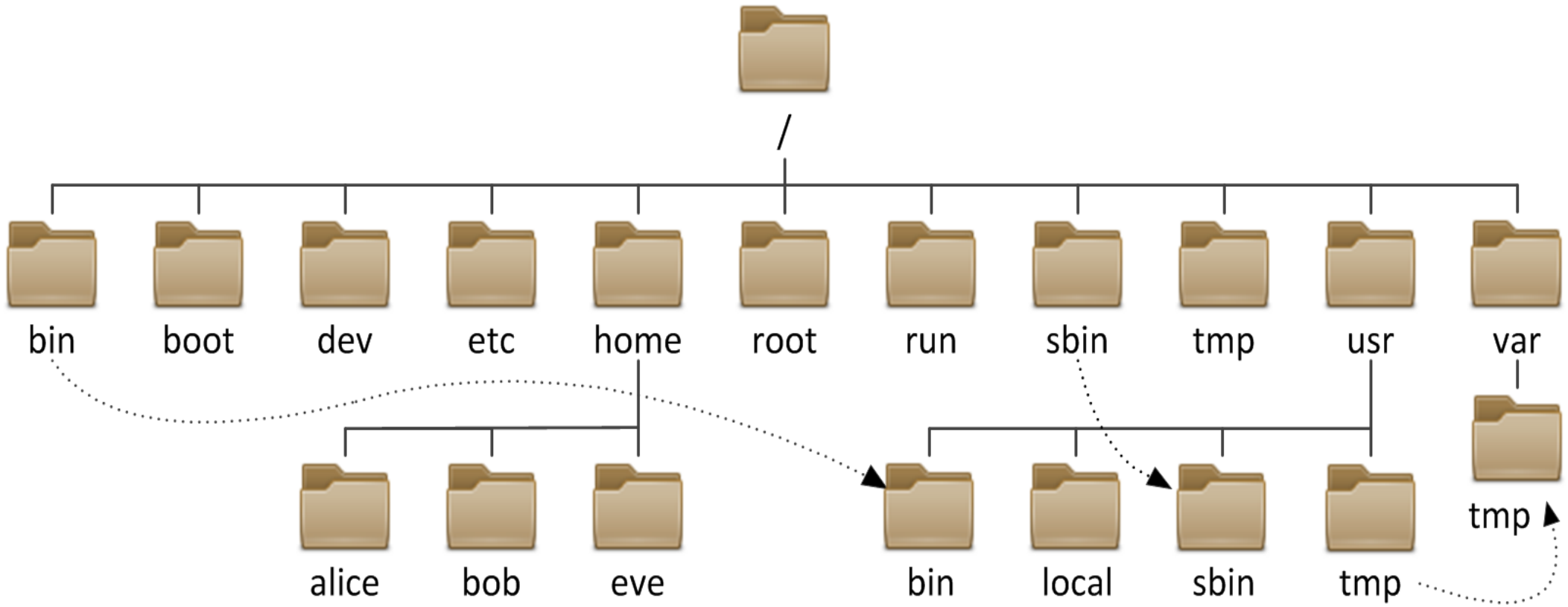
# 本章内容



- 文件系统结构元素
- 创建和查看文件
- 复制、转移和删除文件
- 软和硬链接
- 三种I/O设备
- 把I/O重定向至文件
- 使用管道

# 文件系统与目录结构





# 文件系统



- 文件和目录被组织成一个单根倒置树结构
- 文件系统从根目录下开始，用“/”表示
- 根文件系统(rootfs)：root filesystem
- 文件名称区分大小写
- 以.开头的文件为隐藏文件
- 路径分隔的 /
- 文件有两类数据：
  - 元数据：metadata
  - 数据：data
- 文件系统分层结构：LSB  Linux Standard Base
- FHS: (Filesystem Hierarchy Standard)
  - http://www.pathname.com/fhs/

# 文件名规则



- 文件名最长255个字节
- 包括路径在内文件名称最长4095个字节
- 蓝色-->目录　绿色-->可执行文件　红色-->压缩文件　浅蓝色-->链接文件　灰色-->其他文件
- 除了斜杠和NUL,所有字符都有效.但使用特殊字符的目录名和文件不推荐使用，有些字符需要用引号来引用它们。
- 标准Linux文件系统（如ext4），文件名称大小写敏感。例如：
  MAIL, Mail, mail, mAiL

# 文件系统结构



- /boot：引导文件存放目录，内核文件(vmlinuz)、引导加载器(bootloader, grub)都存放于此目录
- /bin：供所有用户使用的基本命令；不能关联至独立分区，OS启动即会用到的程序
- /sbin：管理类的基本命令；不能关联至独立分区，OS启动即会用到的程序
- /lib：启动时程序依赖的基本共享库文件以及内核模块文件(/lib/modules)
- /lib64：专用于x86_64系统上的辅助共享库文件存放位置
- /etc：配置文件目录
- /home/USERNAME：普通用户家目录
- /root：管理员的家目录
- /media：便携式移动设备挂载点

# 文件系统结构



- /mnt：临时文件系统挂载点
- /dev：设备文件及特殊文件存储位置
  - b: block device，随机访问
  - c: character device，线性访问
- /opt：第三方应用程序的安装位置
- /srv：系统上运行的服务用到的数据
- /tmp：临时文件存储位置
- /proc: 用于输出内核与进程信息相关的虚拟文件系统
- /sys：用于输出当前系统上硬件设备相关信息虚拟文件系统
- /selinux: security enhanced Linux，selinux相关的安全策略等信息的存储位置

# 文件系统结构



- /usr: universal shared, read-only data
  - bin: 保证系统拥有完整功能而提供的应用程序
  - sbin:
  - lib：32位使用
  - lib64：只存在64位系统
  - include: C程序的头文件(header files)
  - share：结构化独立的数据，例如doc, man等
- local：第三方应用程序的安装位置
  - bin, sbin, lib, lib64, etc, share

# 文件系统结构



- /var: variable data files
  - cache: 应用程序缓存数据目录
  - lib: 应用程序状态信息数据
  - local：专用于为/usr/local下的应用程序存储可变数据；
  - lock: 锁文件
  - log: 日志目录及文件
  - opt: 专用于为/opt下的应用程序存储可变数据；
  - run: 运行中的进程相关数据,通常用于存储进程pid文件
  - spool: 应用程序数据池
  - tmp: 保存系统两次重启之间产生的临时数据

# Linux上的应用程序的组成部分



- 二进制程序：/bin, /sbin, /usr/bin, /usr/sbin, /usr/local/bin, /usr/local/sbin
- 库文件：/lib, /lib64, /usr/lib, /usr/lib64, /usr/local/lib, /usr/local/lib64
- 配置文件：/etc, /etc/DIRECTORY, /usr/local/etc
- 帮助文件：/usr/share/man, /usr/share/doc, /usr/local/share/man, /usr/local/share/doc

# Linux下的文件类型



- -：普通文件
- d: 目录文件
- b: 块设备
- c: 字符设备
- l: 符号链接文件
- p: 管道文件pipe
- s: 套接字文件socket

# CentOS 7目录变化

- /bin 和 /usr/bin
- /sbin 和 /usr/sbin
- /lib 和/usr/lib
- /lib64 和 /usr/lib64

# 显示当前工作目录



- 每个shell和系统进程都有一个当前的工作目录
- CWD:current work directory
- 显示当前shell CWD的绝对路径

  pwd: printing working directory

  -P 显示真实物理路径

  -L 显示链接路径（默认）

# 绝对和相对路径



- 绝对路径
  - 以正斜杠开始
  - 完整的文件的位置路径
  - 可用于任何想指定一个文件名的时候
- 相对路径名
  - 不以斜线开始
  - 指定相对于当前工作目录或某目录的位置
  - 可以作为一个简短的形式指定一个文件名
- 基名：basename
- 目录名：dirname

# 更改目录



- cd 改变目录

  使用绝对或相对路径：

  cd /home/wang/

  cd home/wang

  切换至父目录： cd ..

  切换至当前用户主目录： cd

  切换至以前的工作目录： cd -
- 选项：-P
- 相关的环境变量：

  PWD：当前目录路径

  OLDPWD：上一次目录路径

# 列出目录内容



- 列出当前目录的内容或指定目录
- 用法：ls [options] [*files_or_dirs*]
- 示例:

  ls -a 包含隐藏文件

  ls -l 显示额外的信息

  ls -R 目录递归通过

  ls -ld 目录和符号链接信息

  ls -1 文件分行显示

  ls –S 按从大到小排序

  ls –t 按mtime排序

  ls –u 配合-t选项，显示并按atime从新到旧排序

  ls –U 按目录存放顺序显示

  ls –X 按文件后缀排序

# 查看文件状态



- stat
- 文件：metadata, data
- 三个时间戳：

  access time：访问时间，atime，读取文件内容

  modify time: 修改时间, mtime，改变文件内容（数据）

  change time: 改变时间, ctime，元数据发生改变

# 文件通配符



- * 匹配零个或多个字符
- ? 匹配任何单个字符
- ~ 当前用户家目录
- ~mage 用户mage家目录
- ~+ 当前工作目录
- ~- 前一个工作目录
- [0-9] 匹配数字范围
- [a-z]：字母
- [A-Z]：字母
- [wang] 匹配列表中的任何的一个字符
- [^wang] 匹配列表中的所有字符以外的字符

# 文件通配符



- 预定义的字符类：man 7 glob
  - [:digit:]：任意数字，相当于0-9
  - [:lower:]：任意小写字母
  - [:upper:]: 任意大写字母
  - [:alpha:]: 任意大小写字母
  - [:alnum:]：任意数字或字母
  - [:blank:]：水平空白字符
  - [:space:]：水平或垂直空白字符
  - [:punct:]：标点符号
  - [:print:]：可打印字符
  - [:cntrl:]：控制（非打印）字符
  - [:graph:]：图形字符
  - [:xdigit:]：十六进制字符

# 创建空文件和刷新时间



- touch命令：

  touch [OPTION]... FILE...

  -a　仅改变 atime和ctime

  -m　仅改变 mtime和ctime

  -t　[[CC]YY]MMDDhhmm[.ss]

  　　指定atime和mtime的时间戳

  -c　如果文件不存在，则不予创建

# 复制文件和目录cp



- cp [OPTION]... [-T] SOURCE DEST
- cp [OPTION]... SOURCE... DIRECTORY
- cp [OPTION]... -t DIRECTORY SOURCE...
- cp SRC DEST

SRC是文件：

如果目标不存在：新建DEST，并将SRC中内容填充至DEST中

如果目标存在：

如果DEST是文件：将SRC中的内容覆盖至DEST中

基于安全，建议为cp命令使用-i选项

如果DEST是目录：在DEST下新建与原文件同名的文件，并将SRC中内容填充至新文件中

# 复制文件和目录cp



- cp SRC... DEST

  SRC...：多个文件

  DEST必须存在，且为目录，其它情形均会出错；

- cp SRC DEST

  SRC是目录：此时使用选项：-r

  如果DEST不存在：则创建指定目录，复制SRC目录中所有文件至DEST中；

  如果DEST存在：

  如果DEST是文件：报错

  如果DEST是目录：

# 复制cp



| 目标 / 源 | 不存在 | 存在且为文件 | 存在且为目录 |
| --- | --- | --- | --- |
| **一个文件** | 新建DEST，并将SRC中内容填充至DEST中 | 将SRC中的内容覆盖至DEST中<br>**注意数据丢失风险！**<br>**建议用 –i 选项** | 在DEST下新建与原文件同名的文件，并将SRC中内容填充至新文件中 |
| **多个文件** | 提示错误 | 提示错误 | 在DEST下新建与原文件同名的文件，并将原文件内容复制进新文件中 |
| **目录**<br>**须使用-r选项** | 创建指定DEST同名目录，复制SRC目录中所有文件至DEST下 | 提示错误 | 在DEST下新建与原目录同名的目录，并将SRC中内容复制至新目录中 |

# cp常用选项



- -i：覆盖前提示　　–n:不覆盖，注意两者顺序
- -r, -R: 递归复制目录及内部的所有内容
- -a: 归档，相当于-dR --preserv=all
- -d：--no-dereference --preserv=links 不复制原文件，只复制链接名
- --preserv[=ATTR_LIST]
  - mode: 权限
  - ownership: 属主属组
  - timestamp:
  - links
  - xattr
  - context
  - all

# cp 选项



- -p: 等同--preserv=mode,ownership,timestamp
- -v: --verbose
- -f: --force
- -u:--update 只复制源比目标更新文件或目标不存在的文件
- --backup=numbered 目标存在，覆盖前先备份加数字后缀

# 移动和重命名文件



- mv [OPTION]... [-T] SOURCE DEST
- mv [OPTION]... SOURCE... DIRECTORY
- mv [OPTION]... -t DIRECTORY SOURCE...

常用选项：

-i: 交互式

-f: 强制

# 删除



- rm [OPTION]... FILE...
- 常用选项：

  -i: 交互式

  -f: 强制删除

  -r: 递归

  --no-preserve-root
- 示例：

  rm -rf　/

# 目录操作



- tree 显示目录树
  - -d: 只显示目录
  - -L level：指定显示的层级数目
  - -P pattern: 只显示由指定pattern匹配到的路径
- mkdir 创建目录
  - -p: 存在于不报错，且可自动创建所需的各目录
  - -v: 显示详细信息
  - -m MODE: 创建目录时直接指定权限
- rmdir 删除空目录
  - -p: 递归删除父空目录
  - -v: 显示详细信息
- rm -r 递归删除目录树

# 索引节点



- inode（index node）表中包含文件系统所有文件列表
- 一个节点（索引节点）是在一个表项，包含有关文件的信息（ 元数据 ），包括：

  文件类型，权限，UID，GID

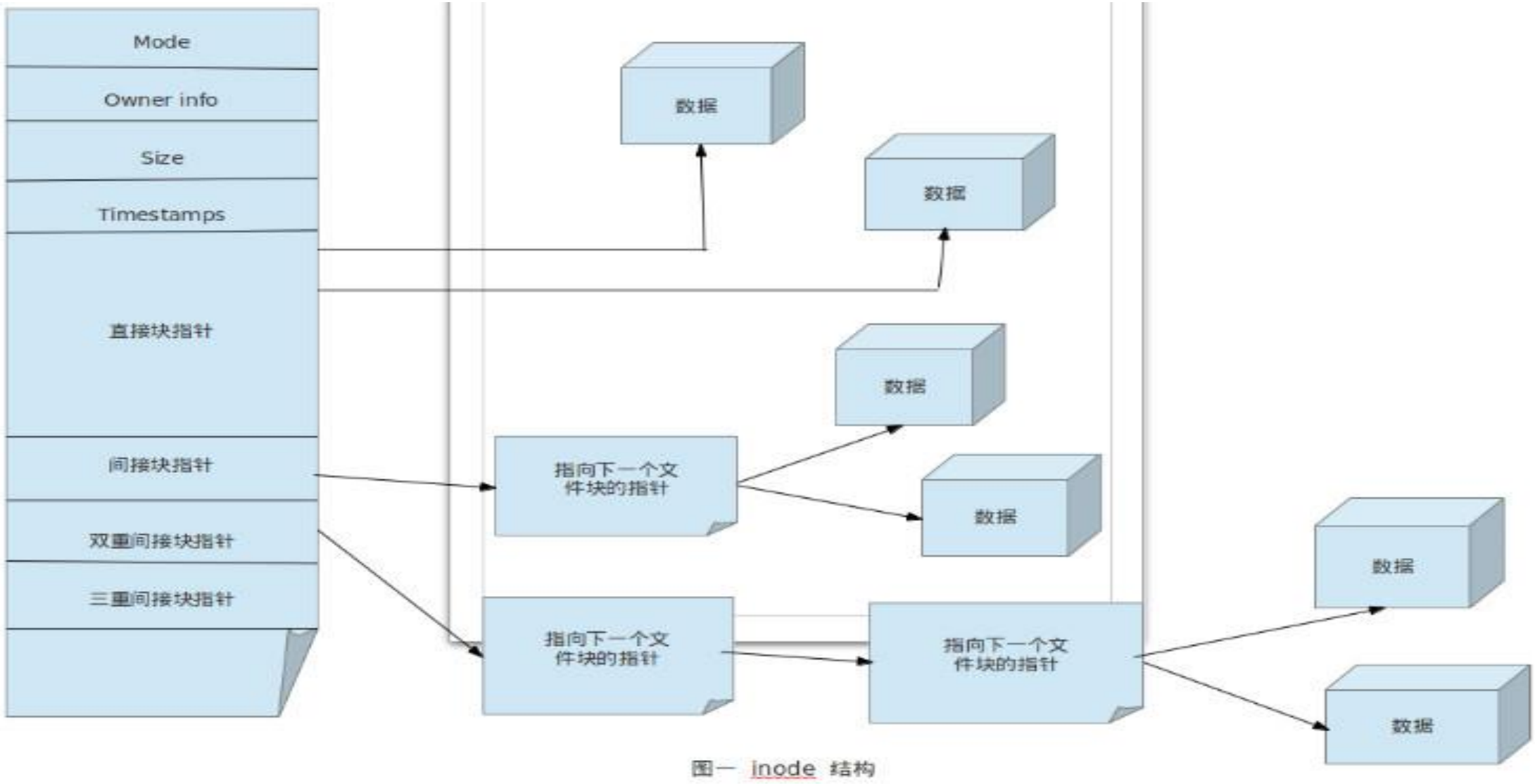
  链接数（指向这个文件名路径名称个数）

  该文件的大小和不同的时间戳

  指向磁盘上文件的数据块指针

  有关文件的其他数据

# inode表结构




图一 inode 结构

# inode表结构



**(1) 前12个直接指针，直接指向存储的数据区域**

*如Blocks大小为4096，则前12个直接指针就可以保存48KB文件。*

**(2) 一级指针可存储文件大小计算**

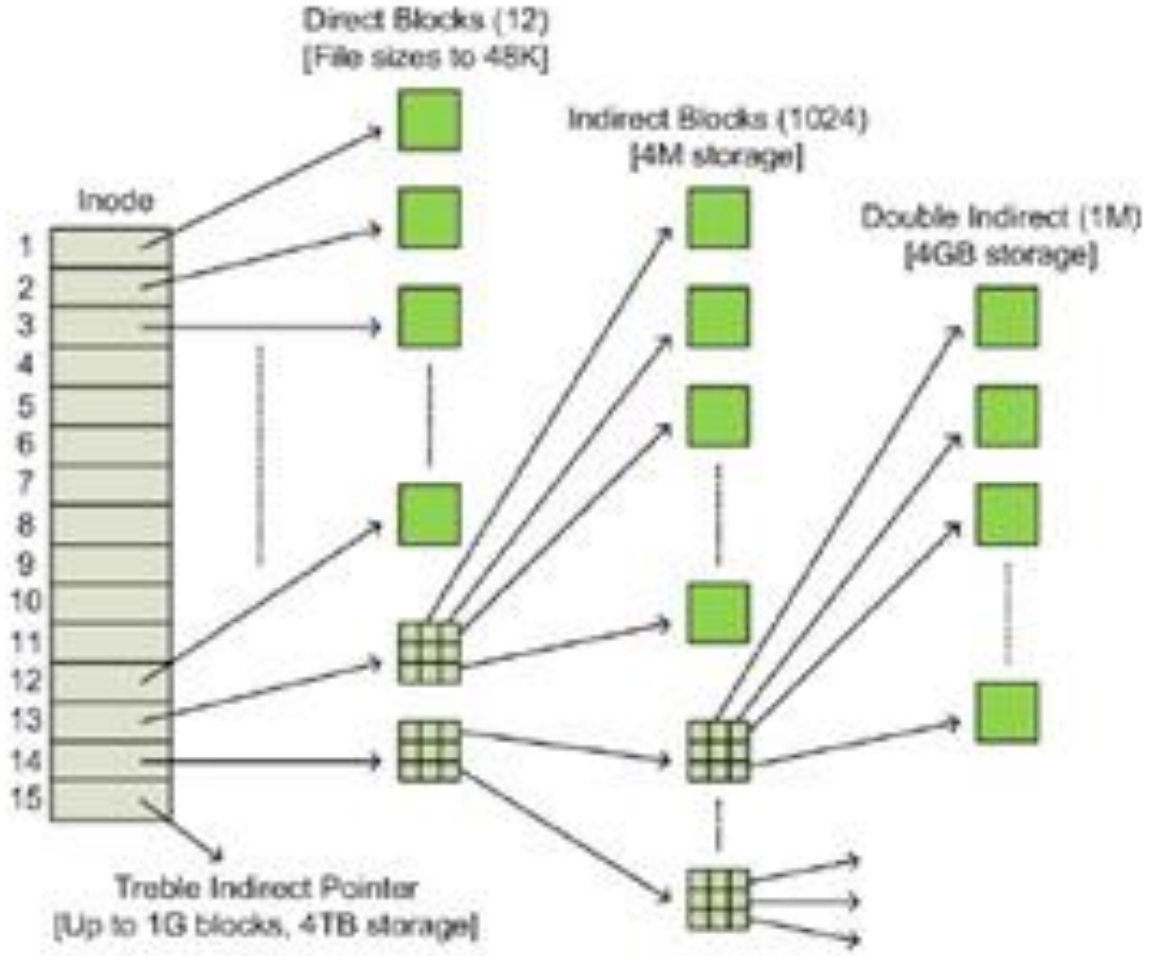
*假设每个指针占用4个字节，则一级指针指向的Block可保存4096/4个指针，可指向1024个Blocks。一级指针可存储文件数据大小为1024*4096 = 4MB。*

**(3)二级指针可存储文件大小计算**

*同样按照Blocks大小为4096，则二级指针可保存的Block指针数量为(4096/4) * (4096/4) = 1024*1024。则二级指针可保存的文件数量大小为(1024*1024)*4096 = 4GB。*

**(4)三级指针可存储文件大小计算**

*以一级、二级指针计算方法类推，三级指针可存储的文件数据大小为(1024*1024*1024)*4096 = 4TB。*



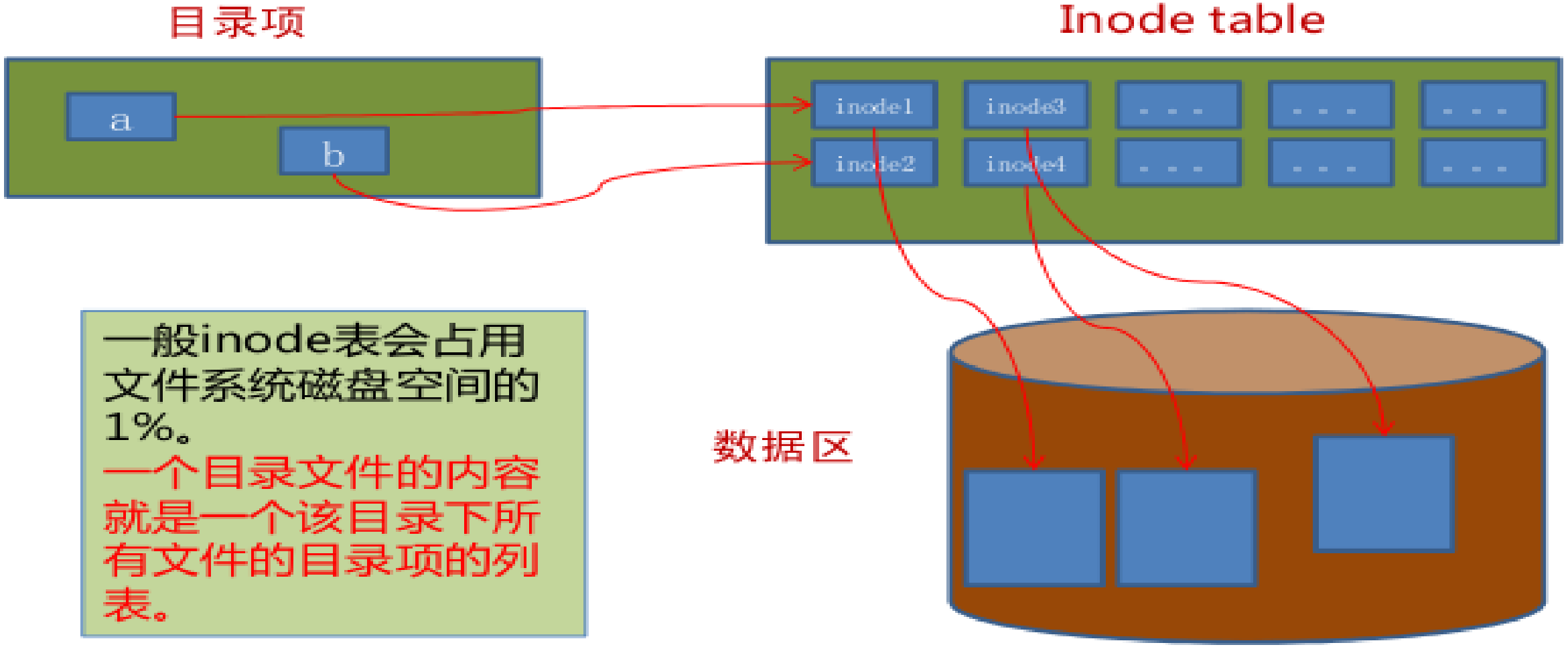
# 目录



- 文件引用一个是 inode号
- 人是通过文件名来引用一个文件
- 一个目录是目录下的文件名和文件inode号之间的映射

# inode表






一般inode表会占用文件系统磁盘空间的1%。

一个目录文件的内容就是一个该目录下所有文件的目录项的列表。

# cp和inode



- 在 CP的 命令：

  分配一个空闲的inode号，在inode表中生成新条目

  在目录中创建一个目录项，将名称与inode编号关联

  拷贝数据生成新的文件

# rm和inode



- rm 命令：

  链接数递减，从而释放的inode号可以被重用

  把数据块放在空闲列表中

  删除目录项

  数据实际上不会马上被删除，但当另一个文件使用数据块时将被覆盖。

# mv和inode



- 如果mv命令的目标和源在相同的文件系统，作为mv 命令
  - 用新的文件名创建对应新的目录项
  - 删除旧目录条目对应的旧的文件名
  - 不影响inode表（除时间戳）或磁盘上的数据位置：没有数据被移动！
- 如果目标和源在一个不同的文件系统， mv相当于cp和rm

# mv和inode



- 如果mv命令的目标和源在相同的文件系统，作为mv 命令
  - 用新的文件名创建对应新的目录项
  - 删除旧目录条目对应的旧的文件名
  - 不影响inode表（除时间戳）或磁盘上的数据位置：没有数据被移动！
- 如果目标和源在一个不同的文件系统， mv相当于cp和rm

# 硬链接



- 创建硬链接会增加额外的记录项以引用文件
- 对应于同一文件系统上一个物理文件
- 每个目录引用相同的inode号
- 创建时链接数递增
- 删除文件时：
  - rm命令递减计数的链接
  - 文件要存在，至少有一个链接数
  - 当链接数为零时，该文件被删除
- 不能跨越驱动器或分区
- 语法:

  ln *filename* [*linkname* ]

# 符号（或软）链接



- 一个符号链接指向另一个文件
- ls - l的 显示链接的名称和引用的文件
- 一个符号链接的内容是它引用文件的名称
- 可以对目录进行
- 可以跨分区
- 指向的是另一个文件的路径；其大小为指向的路径字符串的长度；不增加或减少目标文件inode的引用计数；
- 语法：

```
ln -s  filename  [linkname]
```

# 确定文件内容



- 文件可以包含多种类型的数据
- 检查文件的类型，然后确定适当的打开命令或应用程序使用
- file [options] <filename>...
- 常用选项:

  -b 列出文件辨识结果时，不显示文件名称

  -f filelist 列出文件filelist中文件名的文件类型

  -F 使用指定分隔符号替换输出文件名后默认的”:”分隔符

  -L 查看对应软链接对应文件的文件类型

  --help 显示命令在线帮助

# 标准输入和输出



- 程序：指令+数据
  - 读入数据：Input
  - 输出数据：Output
- 打开的文件都有一个fd: file descriptor (文件描述符)
- Linux给程序提供三种I/O设备
  - 标准输入（STDIN） - 0　默认接受来自键盘的输入
  - 标准输出（STDOUT） - 1　默认输出到终端窗口
  - 标准错误（STDERR） - 2　默认输出到终端窗口
- I/O重定向：改变默认位置

# 把输出和错误重新定向到文件



- STDOUT和STDERR可以被重定向到文件

  命令　操作符号　文件名

  支持的操作符号包括：

  > 把STDOUT重定向到文件

  2> 把STDERR重定向到文件

  &> 把所有输出重定向到文件

- > 文件内容会被覆盖
- >> 原有内容基础上，追加内容

# 把输出和错误重新定向到文件



- 2>  覆盖重定向错误输出数据流
- 2>>  追加重定向错误输出数据流
- 标准输出和错误输出各自定向至不同位置

  COMMAND > /path/to/file.out 2> /path/to/error.out
- 合并标准输出和错误输出为同一个数据流进行重定向
  - &>  覆盖重定向
  - &>>  追加重定向
  - COMMAND > /path/to/file.out 2>&1 （顺序很重要）
  - COMMAND >> /path/to/file.out 2>&1
- ()：合并多个程序的STDOUT

  ( cal 2007 ; cal 2008 ) > all.txt

# tr命令



- tr  转换和删除字符
- tr [OPTION]... SET1 [SET2]
- 选项：

  -c –C --complement：取字符集的补集

  -d --delete：删除所有属于第一字符集的字符

  -s --squeeze-repeats：把连续重复的字符以单独一个字符表示

  -t --truncate-set1：将第一个字符集对应字符转化为第二字符集对应的字符

- [:alnum:]：字母和数字 [:alpha:]：字母 [:cntrl:]：控制（非打印）字符 [:digit:]：数字 [:graph:]：图形字符 [:lower:]：小写字母 [:print:]：可打印字符 [:punct:]：标点符号 [:space:]：空白字符 [:upper:]：大写字母 [:xdigit:]：十六进制字符

# 从文件中导入STDIN



- 使用<来重定向标准输入
- 某些命令能够接受从文件中导入的STDIN

  tr ‘a-z’ ‘A-Z’ < /etc/issue

  该命令会把/etc/issue中的小写字符都转换成写写字符
- tr –d abc < /etc/fstab 删除fstab文件中的所有abc中任意字符
- cat > file

  mage

  wangxiaochun

  按ctrl+d离开，可以使用文件来代替键盘的输入
- Cat > filea < fileb

# 把多行发送给STDIN



- 使用“<<终止词”命令从键盘把多行重导向给STDIN
  - 直到　终止词　位置的所有文本都发送给STDIN
  - 有时被称为就地文本（heretext）

```
mail -s "Please Call"  admin@magedu.com <<END
> Hi Wang,
>
> Please give me a call when you get in. We may need
> to do some maintenance on server1.
>
> Details when you're on-site
> Zhang
> END
```

# 管道



- 管道（使用符号“|”表示）用来连接命令

  命令1 | 命令2 | 命令3 | ...
  - 将命令1的STDOUT发送给命令2的STDIN，命令2的STDOUT发送到命令3的STDIN
  - STDERR默认不能通过管道转发，可利用2>&1 或 |& 实现
  - 最后一个命令会在当前shell进程的子shell进程中执行用来
  - 组合多种工具的功能

    ls | tr 'a-z' 'A-Z'

# 管道



- less ：一页一页地查看输入

  ls -l /etc | less

- mail： 通过电子邮件发送输入

  echo "test email" | mail  -s "test" wang@example.com

- bc：算术运算

  echo  “2^3" | bc

# 重定向到多个目标（tee）



- 命令1 | tee　[-a ] 文件名 | 命令2

  把命令1的STDOUT保存在文件中，做为命令2的输入

  -a 追加

- 使用：
  - 保存不同阶段的输出
  - 复杂管道的故障排除
  - 同时查看和记录输出

# 祝大家学业有成

# 谢　谢