

# Linux 系统概论

天津医科大学  
生物医学工程与技术学院

2018-2019 学年下学期 (春)  
2017 级生信班

# 第三章 文件系统

伊现富 (Yi Xianfu)

天津医科大学 (TIJMU)  
生物医学工程与技术学院

2019 年 5 月



# 教学提纲

- 1 插曲
- 2 引言
- 3 文件系统基础
  - 文件系统和分区
  - 目录结构
  - 路径
- 4 文件系统导航
  - 目录操作
  - 文件操作
  - 文件系统管理
  - 命令详解
- 5 文件类型
  - 类型简介
  - 链接
- 6 文件和目录权限
  - 权限简介
  - 修改权限
  - 特殊权限
- 7 挂载文件系统
- 8 回顾与总结
  - 总结
  - 思考题

## 1 插曲

## 2 引言

## 3 文件系统基础

- 文件系统和分区
- 目录结构
- 路径

## 4 文件系统导航

- 目录操作
- 文件操作
- 文件系统管理
- 命令详解

## 5 文件类型

- 类型简介
- 链接

## 6 文件和目录权限

- 权限简介
- 修改权限
- 特殊权限

## 7 挂载文件系统

## 8 回顾与总结

- 总结
- 思考题



## 使用学生名单

- `shuf -n 10 students.list`
- `sort -R students.list | head`

## 通过学号

- 简单的学号数字（剔除没有的学号）：

```
seq 30 | grep -v "^9$" | grep -v "^21$" | shuf -n 1
```

- 添加学号前缀：

```
seq -w 30 | grep -v "^09$" | nl -s "20160521" | cut -d ' ' -f 2
```

- 添加学号前缀：`seq -w 30 | grep -v "^09$" | sed "s/^/20160521/" | shuf -n 10`



## 使用学生名单

- `shuf -n 10 students.list`
- `sort -R students.list | head`

## 通过学号

- 简单的学号数字（剔除没有的学号）：

```
seq 30 | grep -v "^9$" | grep -v "^21$" | shuf -n 10
```

- 添加学号前缀：

```
seq -w 30 | grep -v "^09$" | nl -s "20160521" | cut -d' ' -f2
```

- 添加学号前缀：`seq -w 30 | grep -v "^09$" | sed "s/^/20160521/" | shuf -n 10`



❶ 抽取三等奖：`shuf -n 10 all.list > level3.list`

❷ 删除中三等奖观众：

```
grep -f level3.list -v all.list > all-3.list
```

❸ 抽取二等奖：`shuf -n 3 all-3.list > level2.list`

❹ 删除中二等奖观众：

```
grep -f level2.list -v all-3.list > all-3-2.list
```

❺ 抽取一等奖：`shuf -n 1 all-3-2.list > level1.list`

❻ 未中奖观众：

```
grep -f level1.list -v all-3-2.list > all-3-2-1.list
```

### 【课外作业】编写 shell 脚本

- 随意指定中奖人数
- 自动删除中奖观众
- 处理缺席中奖观众
- .....

❶ 抽取三等奖：`shuf -n 10 all.list > level3.list`

❷ 删除中三等奖观众：

```
grep -f level3.list -v all.list > all-3.list
```

❸ 抽取二等奖：`shuf -n 3 all-3.list > level2.list`

❹ 删除中二等奖观众：

```
grep -f level2.list -v all-3.list > all-3-2.list
```

❺ 抽取一等奖：`shuf -n 1 all-3-2.list > level1.list`

❻ 未中奖观众：

```
grep -f level1.list -v all-3-2.list > all-3-2-1.list
```

### 【课外作业】编写 shell 脚本

- 随意指定中奖人数
- 自动删除中奖观众
- 处理缺席中奖观众
- .....



1 插曲

2 引言

3 文件系统基础

- 文件系统和分区
- 目录结构
- 路径

4 文件系统导航

- 目录操作
- 文件操作
- 文件系统管理
- 命令详解

5 文件类型

- 类型简介
- 链接

6 文件和目录权限

- 权限简介
- 修改权限
- 特殊权限

7 挂载文件系统

8 回顾与总结

- 总结
- 思考题



## 操作系统 vs. 文件系统

- 终端用户  $\longleftrightarrow$  操作系统  $\longleftrightarrow$  计算机硬件
- 终端用户  $\longleftrightarrow$  文件系统  $\longleftrightarrow$  硬盘等存储设备

计算机的**文件系统 (File system)** 是一种存储和组织计算机数据的方法，它使得对其访问和查找变得容易，文件系统使用**文件**和**树形目录**的抽象逻辑概念代替了硬盘和光盘等物理设备使用数据块的概念，用户使用文件系统来保存数据不必关心数据实际保存在硬盘（或者光盘）的地址为多少的数据块上，只需要记住这个文件的所属目录和文件名。在写入新数据之前，用户不必关心硬盘上的那个块地址有没有被使用，硬盘上的存储空间管理（分配和释放）功能由文件系统自动完成，用户只需要记住数据被写入了到了哪个文件中。

严格地说，文件系统是一套实现了数据的存储、分级组织、访问和获取等操作的抽象数据类型 (Abstract data type)。

文件系统通常使用硬盘和光盘这样的存储设备，并维护文件在设备中的物理位置。但是，实际上文件系统也可能仅仅是一种访问数据的界面而已，实际的数据是通过网络协议（如 NFS、SMB、9P 等）提供的或者存储在内存中，甚至可能根本没有对应的文件（如 proc 文件系统）。



## 操作系统 vs. 文件系统

- 终端用户  $\longleftrightarrow$  操作系统  $\longleftrightarrow$  计算机硬件
- 终端用户  $\longleftrightarrow$  文件系统  $\longleftrightarrow$  硬盘等存储设备

计算机的**文件系统 (File system)** 是一种存储和组织计算机数据的方法，它使得对其访问和查找变得容易，文件系统使用**文件**和**树形目录**的抽象逻辑概念代替了硬盘和光盘等物理设备使用数据块的概念，用户使用文件系统来保存数据不必关心数据实际保存在硬盘（或者光盘）的地址为多少的数据块上，只需要记住这个文件的所属目录和文件名。在写入新数据之前，用户不必关心硬盘上的那个块地址有没有被使用，硬盘上的存储空间管理（分配和释放）功能由文件系统自动完成，用户只需要记住数据被写入到了哪个文件中。

严格地说，文件系统是一套实现了数据的存储、分级组织、访问和获取等操作的抽象数据类型 (Abstract data type)。

文件系统通常使用硬盘和光盘这样的存储设备，并维护文件在设备中的物理位置。但是，实际上文件系统也可能仅仅是一种访问数据的界面而已，实际的数据是通过网络协议（如 NFS、SMB、9P 等）提供的或者存储在内存中，甚至可能根本没有对应的文件（如 proc 文件系统）。



- ❶ 面向磁盘的文件系统（本地的文件系统）：位于硬盘、移动硬盘、光盘、U 盘或其他设备上的实际可访问的文件系统。
  - FAT（File Allocation Table，文件分配表）：Windows
  - NTFS（New Technology File System，新技术文件系统）：Windows
  - **EXT4**（Extended Filesystem 4，扩展文件系统 4）：Linux
  - Btrfs（B-Tree File System）：Linux
  - XFS（X File System）：Linux
  - **ISO9660**：CD-ROM
  - UFS（Unix File System，Unix 文件系统）：Unix
- ❷ 面向网络的文件系统（基于网络的文件系统）：可以远程访问的文件系统。
- ❸ 专用的或虚拟的文件系统：没有实际驻留在磁盘上的文件系统。



- ❶ 面向磁盘的文件系统（本地的文件系统）：位于硬盘、移动硬盘、光盘、U 盘或其他设备上的实际可访问的文件系统。
  - FAT（File Allocation Table，文件分配表）：Windows
  - NTFS（New Technology File System，新技术文件系统）：Windows
  - **EXT4**（Extended Filesystem 4，扩展文件系统 4）：Linux
  - Btrfs（B-Tree File System）：Linux
  - XFS（X File System）：Linux
  - **ISO9660**：CD-ROM
  - UFS（Unix File System，Unix 文件系统）：Unix
- ❷ 面向网络的文件系统（基于网络的文件系统）：可以远程访问的文件系统。
  - NFS（Network File System，网络文件系统）
  - Samba（SMB/CIFS）
- ❸ 专用的或虚拟的文件系统：没有实际驻留在磁盘上的文件系统。



- ❶ 面向磁盘的文件系统（本地的文件系统）：位于硬盘、移动硬盘、光盘、U 盘或其他设备上的实际可访问的文件系统。
  - FAT（File Allocation Table，文件分配表）：Windows
  - NTFS（New Technology File System，新技术文件系统）：Windows
  - **EXT4**（Extended Filesystem 4，扩展文件系统 4）：Linux
  - Btrfs（B-Tree File System）：Linux
  - XFS（X File System）：Linux
  - **ISO9660**：CD-ROM
  - UFS（Unix File System，Unix 文件系统）：Unix
- ❷ 面向网络的文件系统（基于网络的文件系统）：可以远程访问的文件系统。
  - NFS（Network File System，网络文件系统）
  - Samba（SMB/CIFS）
- ❸ 专用的或虚拟的文件系统：没有实际驻留在磁盘上的文件系统。
  - TMPFS（临时文件系统）
  - PROCFS（Process File System，进程文件系统）



- ❶ 面向磁盘的文件系统（本地的文件系统）：位于硬盘、移动硬盘、光盘、U 盘或其他设备上的实际可访问的文件系统。
  - FAT（File Allocation Table, 文件分配表）：Windows
  - NTFS（New Technology File System, 新技术文件系统）：Windows
  - **EXT4**（Extended Filesystem 4, 扩展文件系统 4）：Linux
  - Btrfs（B-Tree File System）：Linux
  - XFS（X File System）：Linux
  - **ISO9660**：CD-ROM
  - UFS（Unix File System, Unix 文件系统）：Unix
- ❷ 面向网络的文件系统（基于网络的文件系统）：可以远程访问的文件系统。
  - NFS（Network File System, 网络文件系统）
  - Samba（SMB/CIFS）
- ❸ 专用的或虚拟的文件系统：没有实际驻留在磁盘上的文件系统。
  - TMPFS（临时文件系统）
  - PROCFS（Process File System, 进程文件系统）



- ① 面向磁盘的文件系统（本地的文件系统）：位于硬盘、移动硬盘、光盘、U 盘或其他设备上的实际可访问的文件系统。
  - FAT（File Allocation Table, 文件分配表）：Windows
  - NTFS（New Technology File System, 新技术文件系统）：Windows
  - **EXT4**（Extended Filesystem 4, 扩展文件系统 4）：Linux
  - Btrfs（B-Tree File System）：Linux
  - XFS（X File System）：Linux
  - **ISO9660**：CD-ROM
  - UFS（Unix File System, Unix 文件系统）：Unix
- ② 面向网络的文件系统（基于网络的文件系统）：可以远程访问的文件系统。
  - NFS（Network File System, 网络文件系统）
  - Samba（SMB/CIFS）
- ③ 专用的或虚拟的文件系统：没有实际驻留在磁盘上的文件系统。
  - TMPFS（临时文件系统）
  - PROCFS（Process File System, 进程文件系统）





- ❶ 面向磁盘的文件系统（本地的文件系统）：位于硬盘、移动硬盘、光盘、U 盘或其他设备上的实际可访问的文件系统。
  - FAT (File Allocation Table, 文件分配表)：Windows
  - NTFS (New Technology File System, 新技术文件系统)：Windows
  - **EXT4** (Extended Filesystem 4, 扩展文件系统 4)：Linux
  - Btrfs (B-Tree File System)：Linux
  - XFS (X File System)：Linux
  - **ISO9660**：CD-ROM
  - UFS (Unix File System, Unix 文件系统)：Unix
- ❷ 面向网络的文件系统（基于网络的文件系统）：可以远程访问的文件系统。
  - NFS (Network File System, 网络文件系统)
  - Samba (SMB/CIFS)
- ❸ 专用的或虚拟的文件系统：没有实际驻留在磁盘上的文件系统。
  - TMPFS (临时文件系统)
  - PROCFS (Process File System, 进程文件系统)



- 1 插曲
- 2 引言
- 3 文件系统基础
  - 文件系统和分区
  - 目录结构
  - 路径
- 4 文件系统导航
  - 目录操作
  - 文件操作
  - 文件系统管理
  - 命令详解

- 5 文件类型
  - 类型简介
  - 链接
- 6 文件和目录权限
  - 权限简介
  - 修改权限
  - 特殊权限
- 7 挂载文件系统
- 8 回顾与总结
  - 总结
  - 思考题



- 1 插曲
- 2 引言
- 3 文件系统基础
  - 文件系统和分区
  - 目录结构
  - 路径
- 4 文件系统导航
  - 目录操作
  - 文件操作
  - 文件系统管理
  - 命令详解

- 5 文件类型
  - 类型简介
  - 链接
- 6 文件和目录权限
  - 权限简介
  - 修改权限
  - 特殊权限
- 7 挂载文件系统
- 8 回顾与总结
  - 总结
  - 思考题



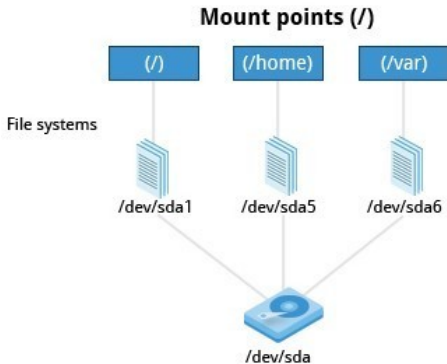
## 文件系统和分区

- 分区是信息的容器，包含整个硬盘或硬盘的一部分
- 文件系统是多个文件的逻辑集合，位于分区或磁盘上
- 一个分区通常只包含一个文件系统

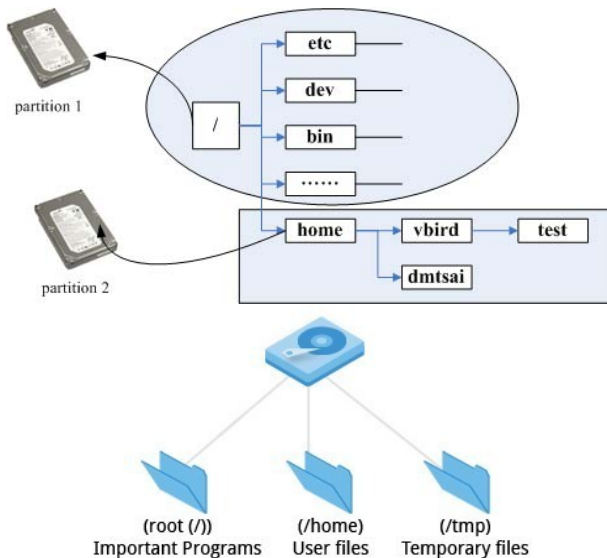


## 文件系统和分区

- 分区是信息的容器，包含整个硬盘或硬盘的一部分
- 文件系统是多个文件的逻辑集合，位于分区或磁盘上
- 一个分区通常只包含一个文件系统



# 文件系统 | 基础 | 文件系统和分区

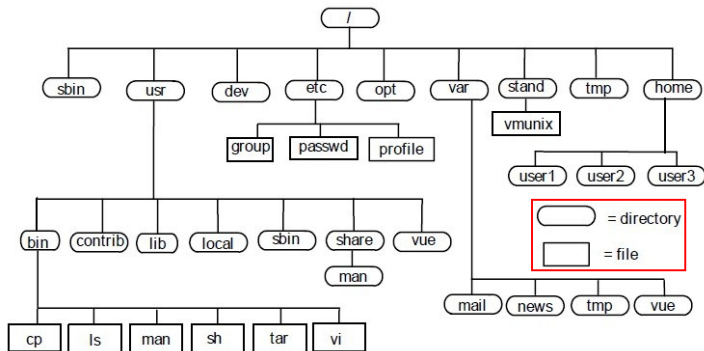


- 1 插曲
- 2 引言
- 3 文件系统基础
  - 文件系统和分区
  - 目录结构
  - 路径
- 4 文件系统导航
  - 目录操作
  - 文件操作
  - 文件系统管理
  - 命令详解

- 5 文件类型
  - 类型简介
  - 链接
- 6 文件和目录权限
  - 权限简介
  - 修改权限
  - 特殊权限
- 7 挂载文件系统
- 8 回顾与总结
  - 总结
  - 思考题



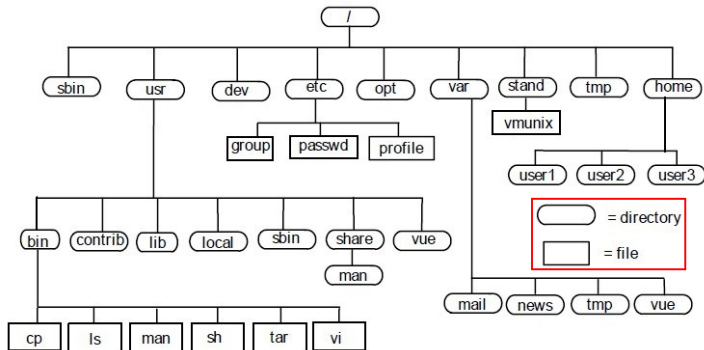
# 文件系统 | 基础 | 目录结构



- Everything is a file. (一切皆文件。)
- 使用自顶而下的分层结构来组织文件
- 每个文件和目录都是从根目录 (/) (Root Directory) 开始的

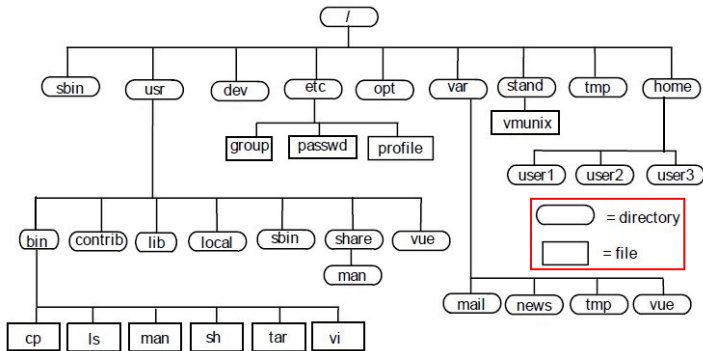






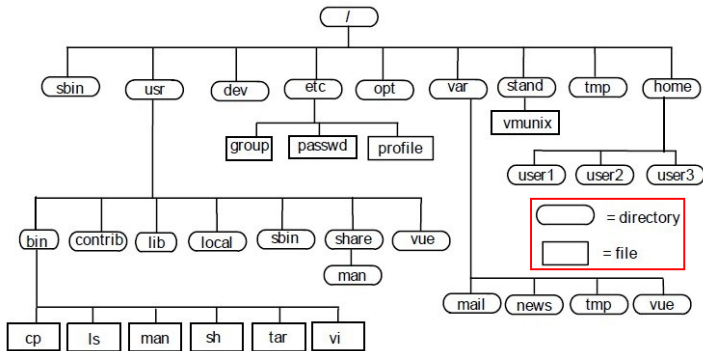
- Everything is a file. (一切皆文件。)
- 使用自顶而下的分层结构来组织文件
- 每个文件和目录都是从根目录 (/) (Root Directory) 开始的
- 文件和目录名的大小写是有区别的
- 定位文件：(根) 目录 ⇒ 子目录 ⇒ ... ⇒ 文件





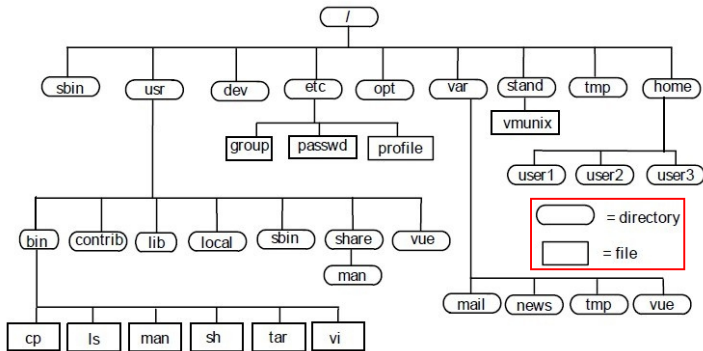
- Everything is a file. (一切皆文件。)
- 使用自顶而下的分层结构来组织文件
- 每个文件和目录都是从根目录 (/) (Root Directory) 开始的
- 文件和目录名的大小写是有区别的
- 定位文件：(根) 目录 ⇒ 子目录 ⇒ ... ⇒ 文件





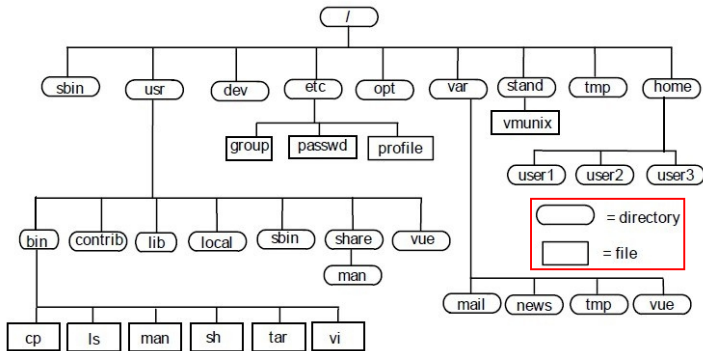
- Everything is a file. (一切皆文件。)
- 使用自顶而下的分层结构来组织文件
- 每个文件和目录都是从根目录 (/) (Root Directory) 开始的
- 文件和目录名的大小写是有区别的
- 定位文件：(根) 目录 ⇒ 子目录 ⇒ ... ⇒ 文件





- Everything is a file. (一切皆文件。)
- 使用自顶而下的分层结构来组织文件
- 每个文件和目录都是从根目录 (/) (Root Directory) 开始的
- 文件和目录名的大小写是有区别的
- 定位文件：(根) 目录 ⇒ 子目录 ⇒ ... ⇒ 文件

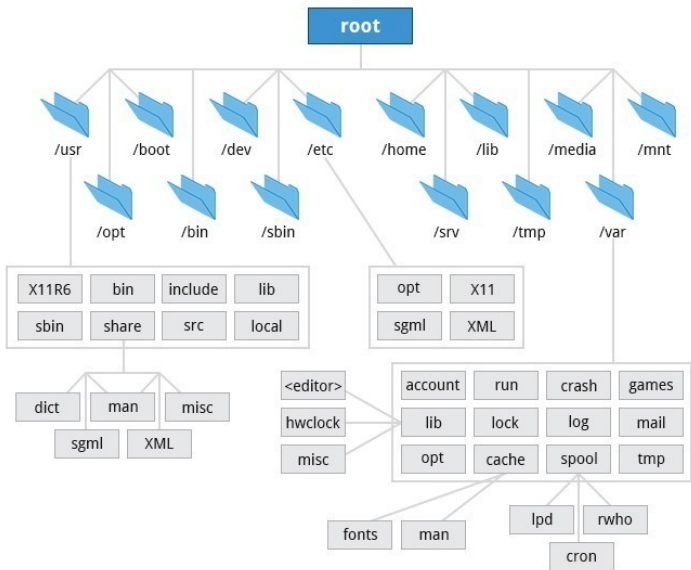




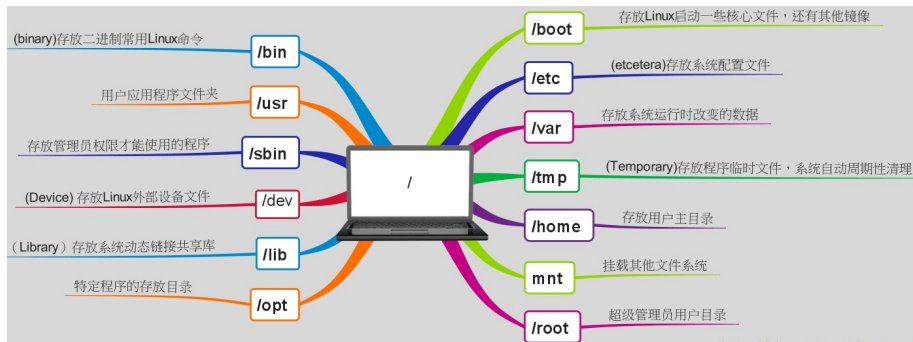
- Everything is a file. (一切皆文件。)
- 使用自顶而下的分层结构来组织文件
- 每个文件和目录都是从根目录 (/) (Root Directory) 开始的
- 文件和目录名的大小写是有区别的
- 定位文件：(根) 目录 ⇒ 子目录 ⇒ ... ⇒ 文件



# 文件系统 | 基础 | 目录结构



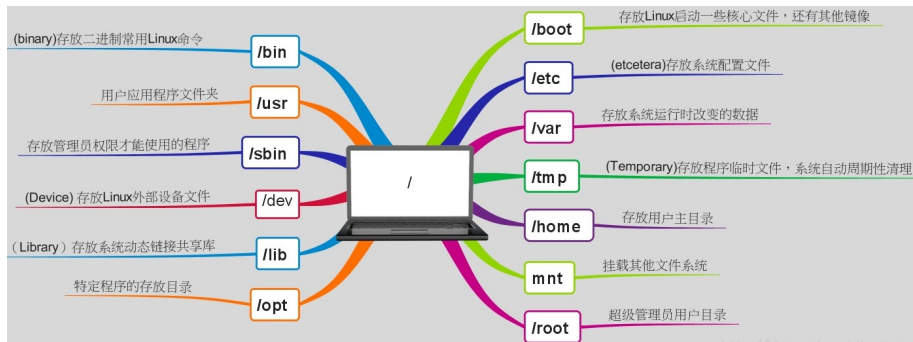
# 文件系统 | 基础 | 目录结构 | 基本目录



## 全称/助记

bin: binary; dev: device; lib: library; mnt: mount; proc: process;  
etc: etcetera => Extended Tool Chest, Editable Text Configuration;  
opt: optional; sbin: system binary; srv: service; tmp: temporary;  
usr: user => User System/Software Resources; var: variable.

# 文件系统 | 基础 | 目录结构 | 基本目录

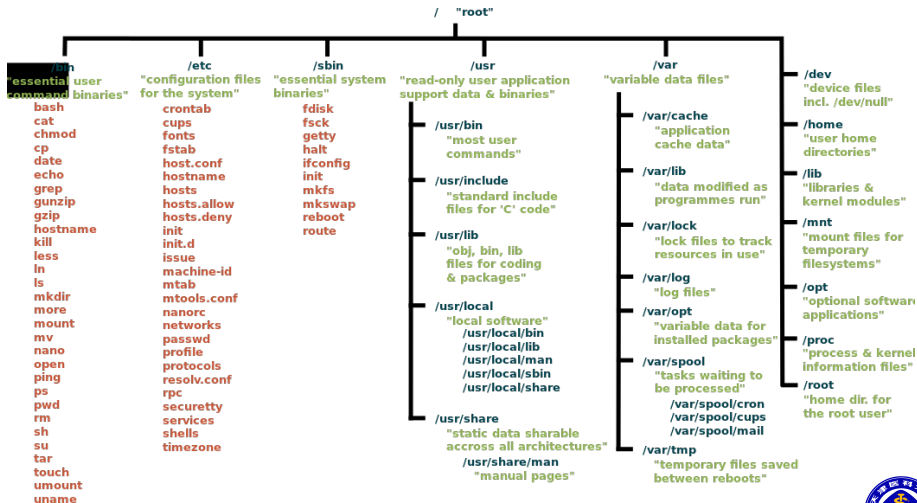


## 全称/助记

bin: binary; dev: device; lib: library; mnt: mount; proc: process;  
etc: etcetera => Extended Tool Chest, Editable Text Configuration;  
opt: optional; sbin: system binary; srv: service; tmp: temporary;  
usr: user => User System/Software Resources; var: variable.



# 文件系统 | 基础 | 目录结构 | 基本目录



目录	内容
/	根目录
/bin	基本程序
/boot	启动系统时所需的文件
/dev	设备文件
/etc	配置文件
/home	用户的 home 目录
/lib	基本共享库, 内核模块
/lost+found	由 fsck 恢复的受损文件
/media	可移动介质的挂载点
/mnt	不能挂载在其他位置上的固定介质的挂载点



目录	内容
/opt	第三方应用程序（“可选软件”）
/proc	proc 文件
/root	根用户（超级用户）的 home 目录
/sbin	由超级用户运行的基本系统管理程序
/srv	本地系统所提供服务的数据库
/tmp	临时文件
/usr	静态数据使用的辅助文件系统
/var	可变数据使用的辅助文件系统



**/boot** 内核文件及自举程序文件保存位置

**/dev** 存放设备文件

**/etc** 系统配置文件

**/home** 用户默认家目录

**/lib** 存放系统程序运行所需的共享库

**/lost+found** 存放一些系统出错的检查结果

**/mnt** 临时文件系统的安装点

**/proc** 虚拟文件系统，存放当前进程信息

**/tmp** 存放临时文件

**/usr** 存放所有命令、库、手册页等

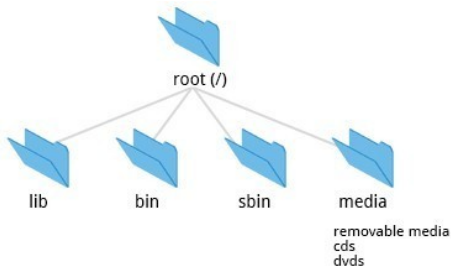
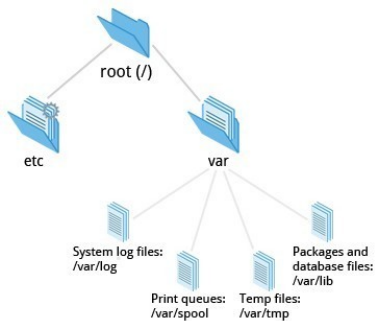
**/usr/bin, /bin** 存放所有用户可以执行的命令

**/usr/sbin, /sbin** 存放只有 root 可以执行的命令

**/var** 包含经常发生变动的文件，如邮件、日志文件、计划任务等



# 文件系统 | 基础 | 目录结构 | 基本目录



目录	内容
<b>/usr/bin</b>	非基本程序（大多数用户程序）
/usr/games	游戏等娱乐和教育程序
/usr/include	C 程序的头文件
/usr/lib	非基本共享库
<b>/usr/local</b>	本地安装程序
/usr/sbin	由超级用户运行的非基本系统管理程序
/usr/share	共享系统数据
/usr/src	源代码（只用于参考）



Directory name	Usage
<code>/usr/include</code>	Header files used to compile applications.
<code>/usr/lib</code>	Libraries for programs in <code>/usr/bin</code> and <code>/usr/sbin</code> .
<code>/usr/lib64</code>	64-bit libraries for 64-bit programs in <code>/usr/bin</code> and <code>/usr/sbin</code> .
<code>/usr/sbin</code>	Non-essential system binaries, such as system daemons.
<code>/usr/share</code>	Shared data used by applications, generally architecture-independent.
<code>/usr/src</code>	Source code, usually for the Linux kernel.
<code>/usr/X11R6</code>	<b>X Window</b> configuration files; generally obsolete.
<code>/usr/local</code>	Data and programs specific to the local machine. Subdirectories include <code>bin</code> , <code>sbin</code> , <code>lib</code> , <code>share</code> , <code>include</code> , etc.
<code>/usr/bin</code>	This is the primary directory of executable commands on the system.



`/` 存放系统程序，也就是 AT&T 开发的 Unix 程序

`/usr` 存放 Unix 系统商（比如 IBM 和 HP）开发的程序

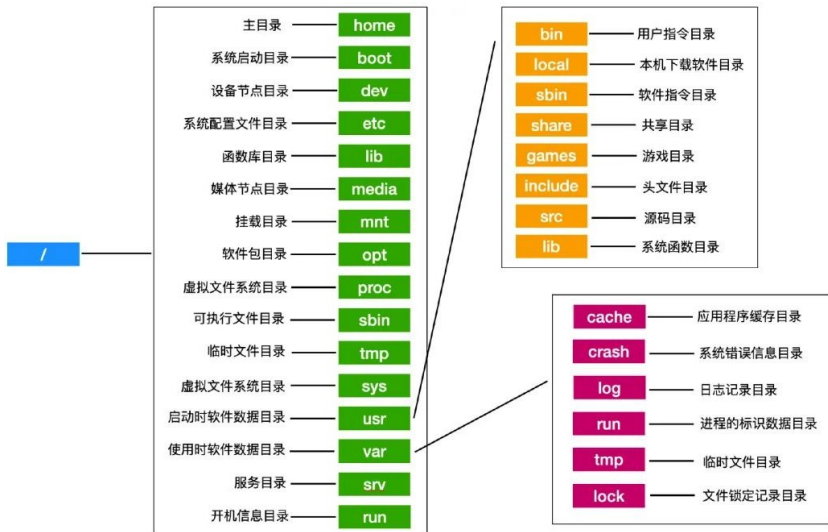
`/usr/local` 存放用户自己安装的程序

`/opt` 在某些系统，用于存放第三方厂商开发的程序，所以取名为 option，意为“选装”





# 文件系统 | 基础 | 目录结构 | 总结



- 1 插曲
- 2 引言
- 3 文件系统基础
  - 文件系统和分区
  - 目录结构
  - 路径
- 4 文件系统导航
  - 目录操作
  - 文件操作
  - 文件系统管理
  - 命令详解

- 5 文件类型
  - 类型简介
  - 链接
- 6 文件和目录权限
  - 权限简介
  - 修改权限
  - 特殊权限
- 7 挂载文件系统
- 8 回顾与总结
  - 总结
  - 思考题



## 绝对 vs. 相对

- 绝对路径 (Absolute Path) : 文件在文件系统中的精确位置, 总是起始于 root (/)
- 相对路径 (Relative Path) : 相对于用户当前位置的一个文件或目录的位置

## 相对路径

- . : 当前目录
- .. : 上一层目录
- ~ : 当前用户的家目录 (Home Directory)
- - : 上一个工作目录



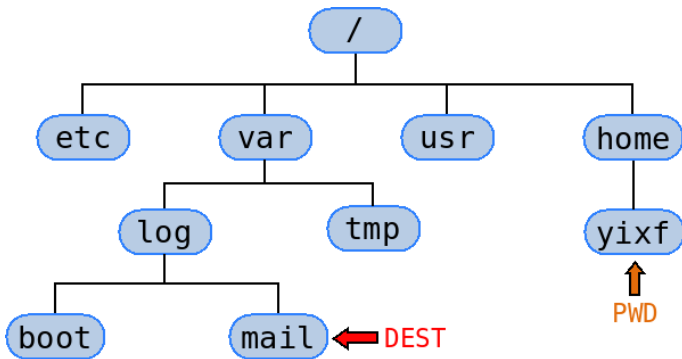
## 绝对 vs. 相对

- 绝对路径 (Absolute Path) : 文件在文件系统中的精确位置, 总是起始于 root (/)
- 相对路径 (Relative Path) : 相对于用户当前位置的一个文件或目录的位置

## 相对路径

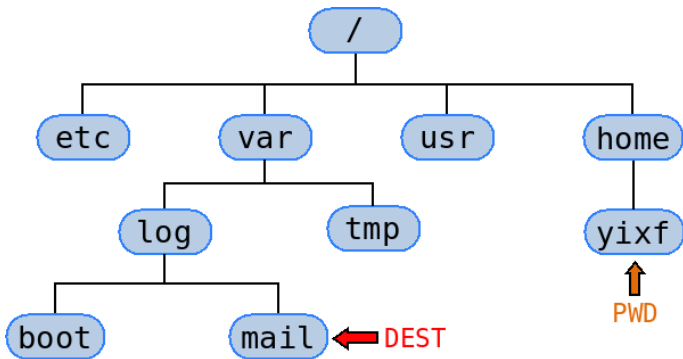
- . : 当前目录
- .. : 上一层目录
- ~ : 当前用户的家目录 (Home Directory)
- - : 上一个工作目录





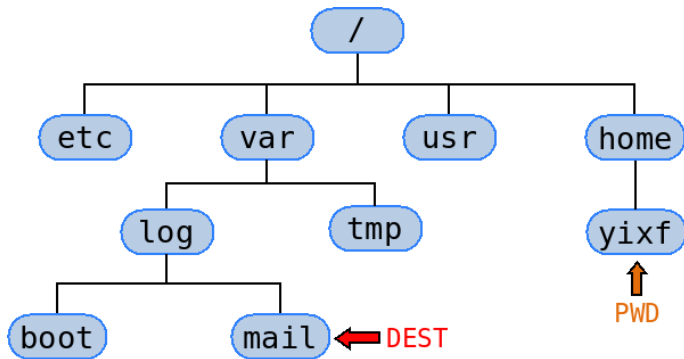
## 绝对 vs. 相对

- 绝对路径：`/var/log/mail`；精确 vs. 冗长
- 相对路径：`../../var/log/mail`；（多数时候）简短 vs. 隐患



## 绝对 vs. 相对

- 绝对路径：`/var/log/mail`；精确 vs. 冗长
- 相对路径：`../../var/log/mail`；（多数时候）简短 vs. 隐患



## 绝对 vs. 相对

- 绝对路径：`/var/log/mail`；精确 vs. 冗长
- 相对路径：`../../var/log/mail`；（多数时候）简短 vs. 隐患

- 一切都源于根目录 (/)
- 文件名除了 / 之外，所有的字符都合法
- 有些字符最好不用，如空格符、制表符、退格符和 @#\$%&() - 等字符
- 避免使用 . 作为普通文件名的第一个字符（隐藏文件）
- 大小写敏感，Linux 是区分大小写的操作系统
- real\_file、Real\_file、REAL\_FILE 是三个不同的文件名
- 按惯例文件名都是小写的





- 一切都源于根目录 (/)
- 文件名除了 / 之外，所有的字符都合法
- 有些字符最好不用，如空格符、制表符、退格符和 @#\$%&() - 等字符
- 避免使用 . 作为普通文件名的第一个字符（隐藏文件）
- 大小写敏感，Linux 是区分大小写的操作系统
- real\_file、Real\_file、REAL\_FILE 是三个不同的文件名
- 按惯例文件名都是小写的



- 一切都源于根目录 (/)
- 文件名除了 / 之外，所有的字符都合法
- 有些字符最好不用，如空格符、制表符、退格符和 @#\$%&() - 等字符
- 避免使用 . 作为普通文件名的第一个字符（隐藏文件）
- 大小写敏感，Linux 是区分大小写的操作系统
- real\_file、Real\_file、REAL\_FILE 是三个不同的文件名
- 按惯例文件名都是小写的



- 一切都源于根目录 (/)
- 文件名除了 / 之外，所有的字符都合法
- 有些字符最好不用，如空格符、制表符、退格符和 @#\$%&() - 等字符
- 避免使用 . 作为普通文件名的第一个字符（隐藏文件）
- 大小写敏感，Linux 是区分大小写的操作系统
- real\_file、Real\_file、REAL\_FILE 是三个不同的文件名
- 按惯例文件名都是小写的



- 一切都源于根目录 (/)
- 文件名除了 / 之外，所有的字符都合法
- 有些字符最好不用，如空格符、制表符、退格符和 @#\$%&() - 等字符
- 避免使用 . 作为普通文件名的第一个字符（隐藏文件）
- 大小写敏感，Linux 是区分大小写的操作系统
- real\_file、Real\_file、REAL\_FILE 是三个不同的文件名
- 按惯例文件名都是小写的



- 一切都源于根目录 (/)
- 文件名除了 / 之外，所有的字符都合法
- 有些字符最好不用，如空格符、制表符、退格符和 @#\$%&() - 等字符
- 避免使用 . 作为普通文件名的第一个字符（隐藏文件）
- 大小写敏感，Linux 是区分大小写的操作系统
- real\_file、Real\_file、REAL\_FILE 是三个不同的文件名
- 按惯例文件名都是小写的



- 一切都源于根目录 (/)
- 文件名除了 / 之外，所有的字符都合法
- 有些字符最好不用，如空格符、制表符、退格符和 @# \$ & ( ) - 等字符
- 避免使用 . 作为普通文件名的第一个字符（隐藏文件）
- 大小写敏感，Linux 是区分大小写的操作系统
- real\_file、Real\_file、REAL\_FILE 是三个不同的文件名
- 按惯例文件名都是小写的



- 1 插曲
- 2 引言
- 3 文件系统基础
  - 文件系统和分区
  - 目录结构
  - 路径
- 4 文件系统导航
  - 目录操作
  - 文件操作
  - 文件系统管理
  - 命令详解

- 5 文件类型
  - 类型简介
  - 链接
- 6 文件和目录权限
  - 权限简介
  - 修改权限
  - 特殊权限
- 7 挂载文件系统
- 8 回顾与总结
  - 总结
  - 思考题



## 目录

- 定位
- 切换
- 列出
- 创建
- 删除
- 树图
- ...

## 文件

- 查看
- 识别
- 状态
- 创建
- 复制
- 移动
- 重命名
- 删除
- ...

## 管理

- 查找
- 空间
- 大小
- ...





## 目录

- 定位
- 切换
- 列出
- 创建
- 删除
- 树图
- ...

## 文件

- 查看
- 识别
- 状态
- 创建
- 复制
- 移动
- 重命名
- 删除
- ...

## 管理

- 查找
- 空间
- 大小
- ...



## 目录

- 定位
- 切换
- 列出
- 创建
- 删除
- 树图
- ...

## 文件

- 查看
- 识别
- 状态
- 创建
- 复制
- 移动
- 重命名
- 删除
- ...

## 管理

- 查找
- 空间
- 大小
- ...



## 目录

- 定位
- 切换
- 列出
- 创建
- 删除
- 树图
- ...

## 文件

- 查看
- 识别
- 状态
- 创建
- 复制
- 移动
- 重命名
- 删除
- ...

## 管理

- 查找
- 空间
- 大小
- ...



- 1 插曲
- 2 引言
- 3 文件系统基础
  - 文件系统和分区
  - 目录结构
  - 路径
- 4 文件系统导航
  - 目录操作
  - 文件操作
  - 文件系统管理
  - 命令详解

- 5 文件类型
  - 类型简介
  - 链接
- 6 文件和目录权限
  - 权限简介
  - 修改权限
  - 特殊权限
- 7 挂载文件系统
- 8 回顾与总结
  - 总结
  - 思考题



命令	助记	说明
pwd	Print Work Directory	显示用户的当前目录
ls	LiSt	列出指定目录的内容
cd	Change Directory	转到指定的目录
mkdir	MaKe DIRectory	创建指定的目录
rmdir	ReMove DIRectory	删除空目录
tree	—	以树状图列出目录的内容结构



- 1 插曲
- 2 引言
- 3 文件系统基础
  - 文件系统和分区
  - 目录结构
  - 路径
- 4 文件系统导航
  - 目录操作
  - 文件操作
  - 文件系统管理
  - 命令详解

- 5 文件类型
  - 类型简介
  - 链接
- 6 文件和目录权限
  - 权限简介
  - 修改权限
  - 特殊权限
- 7 挂载文件系统
- 8 回顾与总结
  - 总结
  - 思考题



命令	助记	说明
file	—	识别文件类型（二进制、文本等）
cat	conCATenate	显示一个文件
touch	—	创建一个空文件或者修改一个现有文件的属性
stat	STATus	查看文件的详细属性/状态
cp	CoPy	把一个文件/目录复制到指定位置
mv	MoVe	移动文件/目录的位置或重命名一个文件/目录
rm	ReMove	删除文件
head	—	显示文件的开始部分
tail	—	显示文件的结尾部分
more	—	从头到尾浏览一个文件
less	—	从开头或结尾开始浏览整个文件



- 1 插曲
- 2 引言
- 3 文件系统基础
  - 文件系统和分区
  - 目录结构
  - 路径
- 4 文件系统导航
  - 目录操作
  - 文件操作
  - 文件系统管理
  - 命令详解

- 5 文件类型
  - 类型简介
  - 链接
- 6 文件和目录权限
  - 权限简介
  - 修改权限
  - 特殊权限
- 7 挂载文件系统
- 8 回顾与总结
  - 总结
  - 思考题





命令	助记	说明
which	—	如果文件位于用户的 PATH 内，则显示文件位置
whereis	—	显示程序名文件的位置
locate	—	在文件索引数据库中查找文件/目录
find	—	在文件系统中查找文件/目录
df	Disk Free	显示磁盘空间的使用情况
du	Disk Usage	显示目录空间占用情况



- 1 插曲
- 2 引言
- 3 文件系统基础
  - 文件系统和分区
  - 目录结构
  - 路径
- 4 文件系统导航
  - 目录操作
  - 文件操作
  - 文件系统管理
  - 命令详解

- 5 文件类型
  - 类型简介
  - 链接
- 6 文件和目录权限
  - 权限简介
  - 修改权限
  - 特殊权限
- 7 挂载文件系统
- 8 回顾与总结
  - 总结
  - 思考题



## 选项

- -p : Preserve, 保持目录和文件的属性
- -R : Recursive, 递归
- -u : Update, 增量备份

cp 妙用：备份目录

cp -Rpu 待备份目录 目标目录



## 选项

- -p : Preserve, 保持目录和文件的属性
- -R : Recursive, 递归
- -u : Update, 增量备份

## cp 妙用：备份目录

cp -Rpu 待备份目录 目标目录



## cd

- cd : 返回家目录
- cd ~ : 返回家目录
- cd .. : 返回上一层目录

## which vs. whereis

- which : 只在用户的 PATH 所指定的文件中查找
- whereis : 在系统的所有目录中定位要查找的命令

## find

```
find /usr/share -name lostfile -print
```



## cd

- cd : 返回家目录
- cd ~ : 返回家目录
- cd .. : 返回上一层目录

## which vs. whereis

- which : 只在用户的 PATH 所指定的文件中查找
- whereis : 在系统的所有目录中定位要查找的命令

## find

```
find /usr/share -name lostfile -print
```



## cd

- cd : 返回家目录
- cd ~ : 返回家目录
- cd .. : 返回上一层目录

## which vs. whereis

- which : 只在用户的 PATH 所指定的文件中查找
- whereis : 在系统的所有目录中定位要查找的命令

## find

```
find /usr/share -name lostfile -print
```



- `ls`：列出用户有权访问的任何目录的内容
- `ls -i` (Inode)：显示文件的 inode 信息
- `ls -a` (All)：显示所有的文件和目录，包括隐藏的文件和目录
- 在文件名的前面加一个 `.` (英文句号) 可以隐藏该文件或目录
- `ls -l` (Long)：显示目录内容的相关扩展信息





- `ls`：列出用户有权访问的任何目录的内容
- `ls -i` (Inode)：显示文件的 inode 信息
- `ls -a` (All)：显示所有的文件和目录，包括隐藏的文件和目录
- 在文件名的前面加一个 `.` (英文句号) 可以隐藏该文件或目录
- `ls -l` (Long)：显示目录内容的相关扩展信息



- `ls`：列出用户有权访问的任何目录的内容
- `ls -i` (Inode)：显示文件的 inode 信息
- `ls -a` (All)：显示所有的文件和目录，包括隐藏的文件和目录
- 在文件名的前面加一个 `.` (英文句号) 可以隐藏该文件或目录
- `ls -l` (Long)：显示目录内容的相关扩展信息



- `ls`：列出用户有权访问的任何目录的内容
- `ls -i` (Inode)：显示文件的 inode 信息
- `ls -a` (All)：显示所有的文件和目录，包括隐藏的文件和目录
- 在文件名的前面加一个 `.` (英文句号) 可以隐藏该文件或目录
- `ls -l` (Long)：显示目录内容的相关扩展信息



- `ls` : 列出用户有权访问的任何目录的内容
- `ls -i` (Inode) : 显示文件的 inode 信息
- `ls -a` (All) : 显示所有的文件和目录, 包括隐藏的文件和目录
- 在文件名的前面加一个 `.` (英文句号) 可以隐藏该文件或目录
- `ls -l` (Long) : 显示目录内容的相关扩展信息

Permissions (3 for owner, 3 for group, 3 for other)	Owner	Group		Date and time of last modification	
- rw-r--r--	1 mdw	users	2321	Mar 15 1994	Fontmap
- rw-r--r--	1 mdw	users	139836	Aug 11 09:11	Index.whole
d rwxr-xr-x	2 mdw	users	1024	Jan 25 1994	Xfonts
d rwxr-xr-x	3 mdw	users	1024	Sep 20 07:40	bin
- rw-r--r--	1 mdw	users	124408	Nov 2 10:53	bitgif.tar.gz
d rwxr-xr-x	2 mdw	users	2048	Jan 21 1994	bitmaps

Type of file  
("d" means  
"directory")

Number of  
hard links

Size in bytes  
(for a directory, bytes used  
to store directory information)

Name



## cat vs. more vs. less

友好性：cat < more < less

## head vs. tail

- 默认显示文件的前/后 10 行
- `-n x`：指定查看文件的前/后 `x` 行
- `tail -f` (Follow)：监视文件内容的变化



## cat vs. more vs. less

友好性：cat < more < less

## head vs. tail

- 默认显示文件的前/后 10 行
- `-n x`：指定查看文件的前/后 `x` 行
- `tail -f (Follow)`：监视文件内容的变化



- **rmmdir**：只能删除空目录
- **rm**：不能删除目录
- **-f** (Force)：强行删除文件
- **-r** (Recursive)：进入到目录中递归删除文件
- **-fr**：删除目录及其子目录，**谨慎使用**
- **切勿尝试**：`rm -rf /`, `rm -rf *`



- `rmdir`：只能删除空目录
- `rm`：不能删除目录
- `-f` (Force)：强行删除文件
- `-r` (Recursive)：进入到目录中递归删除文件
- `-fr`：删除目录及其子目录，**谨慎使用**
- **切勿尝试**：`rm -rf /`, `rm -rf *`





- `rmdir`：只能删除空目录
- `rm`：不能删除目录
- `-f` (Force)：强行删除文件
- `-r` (Recursive)：进入到目录中递归删除文件
- `-fr`：删除目录及其子目录，**谨慎使用**
- **切勿尝试**：`rm -rf /`, `rm -rf *`



- `rmdir`：只能删除空目录
- `rm`：不能删除目录
- `-f` (Force)：强行删除文件
- `-r` (Recursive)：进入到目录中递归删除文件
- `-fr`：删除目录及其子目录，**谨慎使用**
- **切勿尝试**：`rm -rf /`, `rm -rf *`



- `rmdir`：只能删除空目录
- `rm`：不能删除目录
- `-f` (Force)：强行删除文件
- `-r` (Recursive)：进入到目录中递归删除文件
- `-fr`：删除目录及其子目录，**谨慎使用**
- **切勿尝试**：`rm -rf /`, `rm -rf *`



- `rmdir`：只能删除空目录
- `rm`：不能删除目录
- `-f` (Force)：强行删除文件
- `-r` (Recursive)：进入到目录中递归删除文件
- `-fr`：删除目录及其子目录，**谨慎使用**
- **切勿尝试**：`rm -rf /`, `rm -rf *`



```
SUSE-LES-11:/usr/local/resin/conf # df -h
文件系统      容量  已用  可用  已用%  挂载点
/dev/xvda2      97G   7.3G   89G    8% /
devtmpfs        520M   92K   520M    1% /dev
tmpfs           520M  140K   520M    1% /dev/shm

root@bt:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1       19G   14G   4.1G   78% /
none            291M  240K   291M    1% /dev
none            298M   12K   298M    1% /dev/shm
none            298M   84K   298M    1% /var/run
none            298M    0   298M    0% /var/lock
none            298M    0   298M    0% /lib/init/rw
```

## 参数

- df -h, du -h (Human-readable) : K, M, G
- du -s (Summarize) : 目录的总大小



- 1 插曲
- 2 引言
- 3 文件系统基础
  - 文件系统和分区
  - 目录结构
  - 路径
- 4 文件系统导航
  - 目录操作
  - 文件操作
  - 文件系统管理
  - 命令详解

- 5 文件类型
  - 类型简介
  - 链接
- 6 文件和目录权限
  - 权限简介
  - 修改权限
  - 特殊权限
- 7 挂载文件系统
- 8 回顾与总结
  - 总结
  - 思考题



- 1 插曲
- 2 引言
- 3 文件系统基础
  - 文件系统和分区
  - 目录结构
  - 路径
- 4 文件系统导航
  - 目录操作
  - 文件操作
  - 文件系统管理
  - 命令详解

- 5 文件类型
  - 类型简介
  - 链接
- 6 文件和目录权限
  - 权限简介
  - 修改权限
  - 特殊权限
- 7 挂载文件系统
- 8 回顾与总结
  - 总结
  - 思考题



文件类型	说明
-	普通文件（文本文件、二进制可执行文件、硬链接）
d	目录文件
l (小写 L)	符号链接文件
b	块设备文件（块输入/输出设备文件，如存储设备）
c	字符设备文件（原始输入/输出设备文件，将数据作为字节流处理，如 Terminal）
p	命令管道（一种进程间通信的机制）
s	套接字（用于进程间通信）





- 1 插曲
- 2 引言
- 3 文件系统基础
  - 文件系统和分区
  - 目录结构
  - 路径
- 4 文件系统导航
  - 目录操作
  - 文件操作
  - 文件系统管理
  - 命令详解

- 5 文件类型
  - 类型简介
  - 链接
- 6 文件和目录权限
  - 权限简介
  - 修改权限
  - 特殊权限
- 7 挂载文件系统
- 8 回顾与总结
  - 总结
  - 思考题



- 在 Linux 中，每一个文件都有一个相关联的数字：inode
- Linux 使用 inode 而不是文件名来引用文件
- 在一个分区中，inode 是唯一的
- 不同分区内的文件可以有相同的 inode



- 在 Linux 中，每一个文件都有一个相关联的数字：inode
- Linux 使用 inode 而不是文件名来引用文件
- 在一个分区中，inode 是唯一的
- 不同分区内的文件可以有相同的 inode



- 在 Linux 中，每一个文件都有一个相关联的数字：inode
- Linux 使用 inode 而不是文件名来引用文件
- 在一个分区中，inode 是唯一的
- 不同分区内的文件可以有相同的 inode

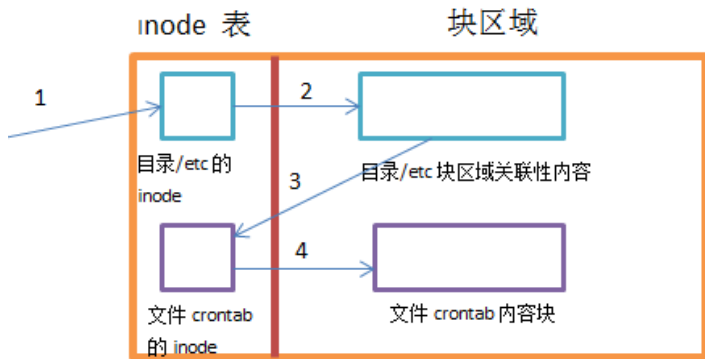


- 在 Linux 中，每一个文件都有一个相关联的数字：inode
- Linux 使用 inode 而不是文件名来引用文件
- 在一个分区中，inode 是唯一的
- 不同分区内的文件可以有相同的 inode



# 文件系统 | 文件类型 | 链接 | inode

- 在 Linux 中，每一个文件都有一个相关联的数字：inode
- Linux 使用 inode 而不是文件名来引用文件
- 在一个分区中，inode 是唯一的
- 不同分区内的文件可以有相同的 inode



## 硬链接 (Hard Link)

- 硬链接与原文件具有相同的 inode，两者本质上没有区别
- 对硬链接的修改会反映到原文件上，反之亦然
- 如果删除硬链接，原文件照样正常使用，反之亦然
- 不能跨越文件系统，“等同于”不占空间的复制 + 同步更新
- 只能对文件建立硬链接，而不能对目录建立硬链接

## 软链接 (Soft Link, 符号链接, Symbolic Link)

## 硬链接 (Hard Link)

- 硬链接与原文件具有相同的 inode，两者本质上没有区别
- 对硬链接的修改会反映到原文件上，反之亦然
- 如果删除硬链接，原文件照样正常使用，反之亦然
- 不能跨越文件系统，“等同于”不占空间的复制 + 同步更新
- 只能对文件建立硬链接，而不能对目录建立硬链接

## 软链接 (Soft Link, 符号链接, Symbolic Link)



## 硬链接 (Hard Link)

- 硬链接与原文件具有相同的 inode，两者本质上没有区别
- 对硬链接的修改会反映到原文件上，反之亦然
- 如果删除硬链接，原文件照样正常使用，反之亦然
- 不能跨越文件系统，“等同于”不占空间的复制 + 同步更新
- 只能对文件建立硬链接，而不能对目录建立硬链接

## 软链接 (Soft Link, 符号链接, Symbolic Link)

## 硬链接 (Hard Link)

- 硬链接与原文件具有相同的 inode，两者本质上没有区别
- 对硬链接的修改会反映到原文件上，反之亦然
- 如果删除硬链接，原文件照样正常使用，反之亦然
- 不能跨越文件系统，“等同于”不占空间的复制 + 同步更新
- 只能对文件建立硬链接，而不能对目录建立硬链接

## 软链接 (Soft Link, 符号链接, Symbolic Link)

## 硬链接 (Hard Link)

- 硬链接与原文件具有相同的 inode，两者本质上没有区别
- 对硬链接的修改会反映到原文件上，反之亦然
- 如果删除硬链接，原文件照样正常使用，反之亦然
- 不能跨越文件系统，“等同于”不占空间的复制 + 同步更新
- 只能对文件建立硬链接，而不能对目录建立硬链接

## 软链接 (Soft Link, 符号链接, Symbolic Link)

能够跨越文件系统，相当于 Windows 中的快捷方式

## 硬链接 (Hard Link)

- 硬链接与原文件具有相同的 inode，两者本质上没有区别
- 对硬链接的修改会反映到原文件上，反之亦然
- 如果删除硬链接，原文件照样正常使用，反之亦然
- 不能跨越文件系统，“等同于”不占空间的复制 + 同步更新
- 只能对文件建立硬链接，而不能对目录建立硬链接

## 软链接 (Soft Link, 符号链接, Symbolic Link)

- 能够跨越文件系统，相当于 Windows 中的快捷方式
- 软链接具有唯一的 inode，内部保存的是原文件的路径地址
- 删除原文件或更改原文件，则软链接也会随之改变

## 硬链接 (Hard Link)

- 硬链接与原文件具有相同的 inode，两者本质上没有区别
- 对硬链接的修改会反映到原文件上，反之亦然
- 如果删除硬链接，原文件照样正常使用，反之亦然
- 不能跨越文件系统，“等同于”不占空间的复制 + 同步更新
- 只能对文件建立硬链接，而不能对目录建立硬链接

## 软链接 (Soft Link, 符号链接, Symbolic Link)

- 能够跨越文件系统，相当于 Windows 中的快捷方式
- 软链接具有唯一的 inode，内部保存的是原文件的路径地址
- 如果“打开并修改”软链接，原文件也会随之改变
- 如果删除软链接，原文件并不会受到影响
- 如果删除原文件，软链接将失效

## 硬链接 (Hard Link)

- 硬链接与原文件具有相同的 inode，两者本质上没有区别
- 对硬链接的修改会反映到原文件上，反之亦然
- 如果删除硬链接，原文件照样正常使用，反之亦然
- 不能跨越文件系统，“等同于”不占空间的复制 + 同步更新
- 只能对文件建立硬链接，而不能对目录建立硬链接

## 软链接 (Soft Link, 符号链接, Symbolic Link)

- 能够跨越文件系统，相当于 Windows 中的快捷方式
- 软链接具有唯一的 inode，内部保存的是原文件的路径地址
- 如果“打开并修改”软链接，原文件也会随之改变
- 如果删除软链接，原文件并不会受到影响
- 如果删除原文件，软链接将失效

## 硬链接 (Hard Link)

- 硬链接与原文件具有相同的 inode，两者本质上没有区别
- 对硬链接的修改会反映到原文件上，反之亦然
- 如果删除硬链接，原文件照样正常使用，反之亦然
- 不能跨越文件系统，“等同于”不占空间的复制 + 同步更新
- 只能对文件建立硬链接，而不能对目录建立硬链接

## 软链接 (Soft Link, 符号链接, Symbolic Link)

- 能够跨越文件系统，相当于 Windows 中的快捷方式
- 软链接具有唯一的 inode，内部保存的是原文件的路径地址
- 如果“打开并修改”软链接，原文件也会随之改变
- 如果删除软链接，原文件并不会受到影响
- 如果删除原文件，软链接将失效

## 硬链接 (Hard Link)

- 硬链接与原文件具有相同的 inode，两者本质上没有区别
- 对硬链接的修改会反映到原文件上，反之亦然
- 如果删除硬链接，原文件照样正常使用，反之亦然
- 不能跨越文件系统，“等同于”不占空间的复制 + 同步更新
- 只能对文件建立硬链接，而不能对目录建立硬链接

## 软链接 (Soft Link, 符号链接, Symbolic Link)

- 能够跨越文件系统，相当于 Windows 中的快捷方式
- 软链接具有唯一的 inode，内部保存的是原文件的路径地址
- 如果“打开并修改”软链接，原文件也会随之改变
- 如果删除软链接，原文件并不会受到影响
- 如果删除原文件，软链接将失效



## 硬链接 (Hard Link)

- 硬链接与原文件具有相同的 inode，两者本质上没有区别
- 对硬链接的修改会反映到原文件上，反之亦然
- 如果删除硬链接，原文件照样正常使用，反之亦然
- 不能跨越文件系统，“等同于”不占空间的复制 + 同步更新
- 只能对文件建立硬链接，而不能对目录建立硬链接

## 软链接 (Soft Link, 符号链接, Symbolic Link)

- 能够跨越文件系统，相当于 Windows 中的快捷方式
- 软链接具有唯一的 inode，内部保存的是原文件的路径地址
- 如果“打开并修改”软链接，原文件也会随之改变
- 如果删除软链接，原文件并不会受到影响
- 如果删除原文件，软链接将失效

## Make a file:

```
$ touch name_A
```

name\_A  
(ordinary file)



## Hard link:

```
$ ln name_A name_B
```

name\_A

name\_B  
(Hard link)



## Symbolic link:

```
$ ln -s name_A name_B
```

name\_A

name\_B  
(Symbolic link)



项目	硬链接	软链接
语法 (LiNk)	In source hardlink	In -s source softlink
本质	与原文件没区别	保存原文件的路径
inode	与原文件相同	与原文件不同, 唯一
类比	不占空间的复制 + 同步更新	快捷方式
文件系统	不能跨越	能跨越
删除原文件	不受影响	失效
使用对象	文件	文件和目录
修改链接	原文件随之改变	
删除链接	原文件不受影响	

## 删除链接

- 删除硬链接或者链接到文件的软链接：`rm link_file`
- 删除链接到目录的软链接：`unlink link_dir`

项目	硬链接	软链接
语法 (LiNk)	In source hardlink	In -s source softlink
本质	与原文件没区别	保存原文件的路径
inode	与原文件相同	与原文件不同, 唯一
类比	不占空间的复制 + 同步更新	快捷方式
文件系统	不能跨越	能跨越
删除原文件	不受影响	失效
使用对象	文件	文件和目录
修改链接	原文件随之改变	
删除链接	原文件不受影响	

## 删除链接

- 删除硬链接或者链接到文件的软链接：rm link\_file
- 删除链接到目录的软链接：unlink link\_dir

## 可能的用途

- 为命令、程序或文件取别名
- 创建不占存储空间的文件副本
- 为文件创建方便的快捷方式
- 对文件进行分组

## 应用实例

(假设) 在 /home/share 下面有一个 2G 的 Linux 演示视频，服务器上的 50 个用户都要保存该视频到自己的家目录并进行观看。用户该如何保存该视频呢？哪种方式更加节省硬盘资源？哪种方式更加安全？

- `cp`：复制 2G 的视频到家目录——总共占据 100G！
- `ln -s`：把 2G 的视频软链接到家目录——总共占据 2G+！
- `ln`：把 2G 的视频硬链接到家目录——总共占据 2G！

## 可能的用途

- 为命令、程序或文件取别名
- 创建不占存储空间的文件副本
- 为文件创建方便的快捷方式
- 对文件进行分组

## 应用实例

(假设) 在 /home/share 下面有一个 2G 的 Linux 演示视频，服务器上的 50 个用户都要保存该视频到自己的家目录并进行观看。用户该如何保存该视频呢？哪种方式更加节省硬盘资源？哪种方式更加安全？

- cp：复制 2G 的视频到家目录——总共占据 100G！
- ln -s：把 2G 的视频软链接到家目录——总共占据 2G+！
- ln：把 2G 的视频硬链接到家目录——总共占据 2G！

## 扩展思考

cp、ln 和 mv 都可以“拷贝”出一个文件，它们有什么不同？

目录文件是一种特殊的文件，可以理解成存储的是 `dirent` 列表。`dirent` 只是名字到 `inode` 的映射，这个是树形结构的基础；

常说目录树在内存中确实是一个树的结构，每个节点由 `dentry` 结构体表示；

`ln -s` 创建软链接文件，软链接文件是一个独立的新文件，有一个新的 `inode`，有新的 `dentry`，文件类型为 link，文件内容就是一条指向源的路径，所以**软链的创建可以无视文件系统，跨越山河**；

`ln` 默认创建硬连接，硬链接文件只在目录文件里添加了一个新 `dirent` 项 <新 name: 原 inode>，文件 `inode` 还是和原文件同一个，所以**硬链接不能跨文件系统（因为不同的文件系统是独立的一套 inode 管理方式，不同的文件系统实例对 inode number 的解释各有不同）**；

`ln` 命令貌似创建出了新文件，但其实不然，`ln` 只跟元数据相关，涉及到 `dirent` 的变动，**不涉及到数据的拷贝**，起不到数据备份的目的；

`mv` 其实是调用 `rename` 调用，在**同一个文件系统中不涉及数据拷贝，只涉及到元数据变更（`dirent` 的增删）**，所以速度也很快。但如果 `mv` 的源和目的在不同的文件系统，那么就会退化成真正的 `copy`，会涉及到数据拷贝，这个时候速度相对慢一些，慢成什么样子？就跟 `cp` 命令一样；

`cp` 命令才是真正的数据拷贝命令，速度可能相对慢一些，但是 `cp` 命令有 `--sparse` 可以优化拷贝速度，针对空洞和全 0 数据，可以跳过，从而**针对稀疏文件可以节省大量磁盘 IO**；



- 1 插曲
- 2 引言
- 3 文件系统基础
  - 文件系统和分区
  - 目录结构
  - 路径
- 4 文件系统导航
  - 目录操作
  - 文件操作
  - 文件系统管理
  - 命令详解

- 5 文件类型
  - 类型简介
  - 链接
- 6 文件和目录权限
  - 权限简介
  - 修改权限
  - 特殊权限
- 7 挂载文件系统
- 8 回顾与总结
  - 总结
  - 思考题



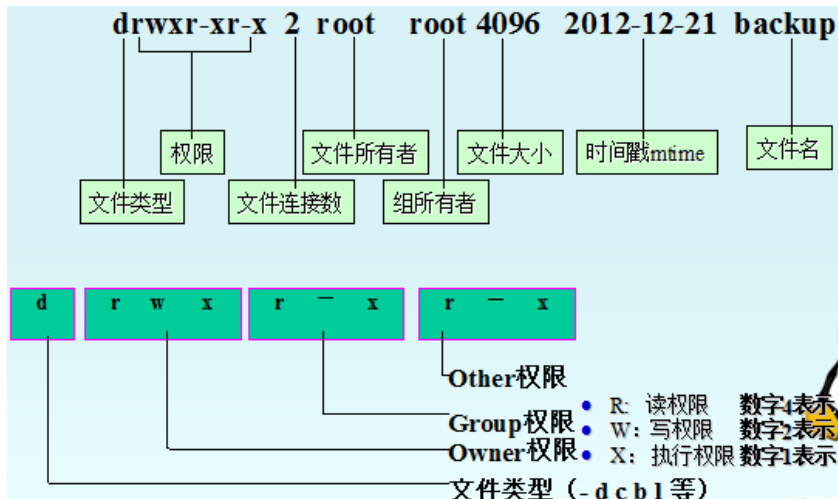


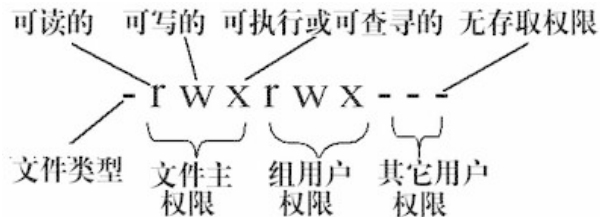
- 1 插曲
- 2 引言
- 3 文件系统基础
  - 文件系统和分区
  - 目录结构
  - 路径
- 4 文件系统导航
  - 目录操作
  - 文件操作
  - 文件系统管理
  - 命令详解

- 5 文件类型
  - 类型简介
  - 链接
- 6 文件和目录权限
  - 权限简介
  - 修改权限
  - 特殊权限
- 7 挂载文件系统
- 8 回顾与总结
  - 总结
  - 思考题



# 文件系统 | 权限 | 简介





字符位置	含义	助记
2~4	文件所有者	user
5~7	文件所属组	group
8~10	其他任何人	others



字符	助记	权限	对文件	对目录
r	Read	读	查看文件内容【cat】	读取/列出目录或子目录内容【ls】
w	Write	写	修改文件内容（添加文本或删除文件）【vim】	在目录中创建、修改、删除文件或子目录【touch】
x	eXecute	执行	执行/运行文件【sh】	进入目录搜索【cd】
-	-	-	无	无

注意：目录必须具有 x 权限，否则无法进入并查看其内容！



# 教学提纲

- 1 插曲
- 2 引言
- 3 文件系统基础
  - 文件系统和分区
  - 目录结构
  - 路径
- 4 文件系统导航
  - 目录操作
  - 文件操作
  - 文件系统管理
  - 命令详解

- 5 文件类型
  - 类型简介
  - 链接
- 6 文件和目录权限
  - 权限简介
  - 修改权限
  - 特殊权限
- 7 挂载文件系统
- 8 回顾与总结
  - 总结
  - 思考题



## 修改权限

chmod (CHange MODe)

### 两种方式

#### 1 符号模式：容易理解

- 用户：u, g, o, a
- 操作：+, -, =
- 权限：r, w, x

#### 2 绝对模式：更加高效

- 0, 1, 2, 4
- $3 = 1 + 2$
- $5 = 1 + 4$
- $6 = 2 + 4$
- $7 = 1 + 2 + 4$



## 修改权限

chmod (CHange MODe)

## 两种方式

### 1 符号模式：容易理解

- 用户：u, g, o, a
- 操作：+, -, =
- 权限：r, w, x

### 2 绝对模式：更加高效

- 0, 1, 2, 4
- $3 = 1 + 2$
- $5 = 1 + 4$
- $6 = 2 + 4$
- $7 = 1 + 2 + 4$



## 用户

- u : User, 用户
- g : Group, 组
- o : Other, 其他人
- a : All, 所有人

## 操作

- + : 添加
- - : 删除
- = : 指定

## 权限

- r : Read, 读
- w : Write, 写
- x : eXecute, 执行

## 实例

- `chmod u-x testfile`
- `chmod g=rx testfile`
- `chmod o+wx testfile`
- `chmod uo+x,g-w testfile`
- `chmod u-x,g=rx,o+wx testfile`



## 用户

- u : User, 用户
- g : Group, 组
- o : Other, 其他人
- a : All, 所有人

## 操作

- + : 添加
- - : 删除
- = : 指定

## 权限

- r : Read, 读
- w : Write, 写
- x : eXecute, 执行

## 实例

- `chmod u-x testfile`
- `chmod g=rx testfile`
- `chmod o+wx testfile`
- `chmod uo+x,g-w testfile`
- `chmod u-x,g=rx,o+wx testfile`

## 用户

- u : User, 用户
- g : Group, 组
- o : Other, 其他人
- a : All, 所有人

## 操作

- + : 添加
- - : 删除
- = : 指定

## 权限

- r : Read, 读
- w : Write, 写
- x : eXecute, 执行

## 实例

- `chmod u-x testfile`
- `chmod g=rx testfile`
- `chmod o+wx testfile`
- `chmod uo+x,g-w testfile`
- `chmod u-x,g=rx,o+wx testfile`

## 用户

- u : User, 用户
- g : Group, 组
- o : Other, 其他人
- a : All, 所有人

## 操作

- + : 添加
- - : 删除
- = : 指定

## 权限

- r : Read, 读
- w : Write, 写
- x : eXecute, 执行

## 实例

- `chmod u-x testfile`
- `chmod g=rx testfile`
- `chmod o+wx testfile`
- `chmod uo+x,g-w testfile`
- `chmod u-x,g=rx,o+wx testfile`

数字	符号	权限
0	---	无权限
1	--x	可执行
2	-w-	可写
3	-wx	可写、可执行 (2+1)
4	r--	可读
5	r-x	可读、可执行 (4+1)
6	rw-	可读、可写 (4+2)
7	rwx	可读、可写、可执行 (4+2+1)

## 实例

- `chmod 740 testfile`
- `chmod 755 testfile`



数字	符号	权限
0	---	无权限
1	--x	可执行
2	-w-	可写
3	-wx	可写、可执行 (2+1)
4	r--	可读
5	r-x	可读、可执行 (4+1)
6	rw-	可读、可写 (4+2)
7	rwX	可读、可写、可执行 (4+2+1)

## 实例

- `chmod 740 testfile`
- `chmod 755 testfile`



## 背景

有个商人雇用了一位手艺高超的工匠来为他做一个精致的产品，工作一星期七天的代价是一条金条。商人手头上有一条金条，刚好可以付工匠一星期的工钱。但工匠要求工钱要按每天来付。虽然他并不急着用钱，每天有钱进账，工匠心里总是踏实一些。但商人家中有个规矩，金条只能切二刀。

## 问题

商人应该怎么切这两刀，才能满足工匠的要求？

## 解释

1-2-4;  $3=1+2$ ;  $5=4+1$ ;  $6=2+4$ ;  $7=1+2+4$

1、2、4 这三个二进制数的组合能表示 0-7 中的任何一个。



## 背景

有个商人雇用了一位手艺高超的工匠来为他做一个精致的产品，工作一星期七天的代价是一条金条。商人手头上有一条金条，刚好可以付工匠一星期的工钱。但工匠要求工钱要按每天来付。虽然他并不急着用钱，每天有钱进账，工匠心里总是踏实一些。但商人家中有个规矩，金条只能切二刀。

## 问题

商人应该怎么切这两刀，才能满足工匠的要求？

## 解释

1-2-4;  $3=1+2$ ;  $5=4+1$ ;  $6=2+4$ ;  $7=1+2+4$

1、2、4 这三个二进制数的组合能表示 0-7 中的任何一个。



## 背景

有个商人雇用了一位手艺高超的工匠来为他做一个精致的产品，工作一星期七天的代价是一条金条。商人手头上有一条金条，刚好可以付工匠一星期的工钱。但工匠要求工钱要按每天来付。虽然他并不急着用钱，每天有钱进账，工匠心里总是踏实一些。但商人家中有个规矩，金条只能切二刀。

## 问题

商人应该怎么切这两刀，才能满足工匠的要求？

## 解释

1-2-4;  $3=1+2$ ;  $5=4+1$ ;  $6=2+4$ ;  $7=1+2+4$

1、2、4 这三个二进制数的组合能表示 0-7 中的任何一个。





- 1 插曲
- 2 引言
- 3 文件系统基础
  - 文件系统和分区
  - 目录结构
  - 路径
- 4 文件系统导航
  - 目录操作
  - 文件操作
  - 文件系统管理
  - 命令详解

- 5 文件类型
  - 类型简介
  - 链接
- 6 文件和目录权限
  - 权限简介
  - 修改权限
  - 特殊权限
- 7 挂载文件系统
- 8 回顾与总结
  - 总结
  - 思考题



## SetUID：Set User ID，设置用户标识

当一个可执行程序（一般为一个命令）具有 SetUID 权限，用户执行这个程序时，将以这个程序的所有者的身份执行。

## 设置 SetUID

- 设置：`chmod u+s FILE, chmod 4755 FILE (SetUID=4000)`
- 取消：`chmod u-s FILE, chmod 755 FILE`
- 查看：`-rwsr-xr-x 1 root root ... /usr/bin/passwd`
- 查找：`find / -perm -4000`

## 注意事项

将命令设置成 SetUID 是一件很危险的事情！它可以让一个用户瞬间变成超级用户、使系统不断重启、使用户不需要密码就可以登录……比如将 vi 设置成 SetUID，则它可以编辑并保存系统中所有的文件，甚至是系统配置文件！

## SetUID：Set User ID，设置用户标识

当一个可执行程序（一般为一个命令）具有 SetUID 权限，用户执行这个程序时，将以这个程序的所有者的身份执行。

## 设置 SetUID

- 设置：`chmod u+s FILE, chmod 4755 FILE (SetUID=4000)`
- 取消：`chmod u-s FILE, chmod 755 FILE`
- 查看：`-rwsr-xr-x 1 root root ... /usr/bin/passwd`
- 查找：`find / -perm -4000`

## 注意事项

将命令设置成 SetUID 是一件很危险的事情！它可以让一个用户瞬间变成超级用户、使系统不断重启、使用户不需要密码就可以登录……比如将 vi 设置成 SetUID，则它可以编辑并保存系统中所有的文件，甚至是系统配置文件！

## SetUID：Set User ID，设置用户标识

当一个可执行程序（一般为一个命令）具有 SetUID 权限，用户执行这个程序时，将以这个程序的所有者的身份执行。

## 设置 SetUID

- 设置：`chmod u+s FILE, chmod 4755 FILE (SetUID=4000)`
- 取消：`chmod u-s FILE, chmod 755 FILE`
- 查看：`-rwsr-xr-x 1 root root ... /usr/bin/passwd`
- 查找：`find / -perm -4000`

## 注意事项

将命令设置成 SetUID 是一件很危险的事情！它可以让一个用户瞬间变成超级用户、使系统不断重启、使用户不需要密码就可以登录……比如将 vi 设置成 SetUID，则它可以编辑并保存系统中所有的文件，甚至是系统配置文件！

## SetGID : Set Group ID

当一个可执行程序（一般为一个命令）具有 SetGID 权限，用户执行这个程序时，将以这个程序所属组的身份执行。

### 设置 SetGID

- 设置：`chmod g+s FILE, chmod 2755 FILE (SetGID=2000)`
- 取消：`chmod g-s FILE, chmod 755 FILE`
- 查看：`drwxr-s--- 2 root dip ... /etc/chatscripts/`
- 查找：`find / -perm -2000`
- 同时设置 SetUID 和 SetGID：`chmod 6755 FILE`



## SetGID : Set Group ID

当一个可执行程序（一般为一个命令）具有 SetGID 权限，用户执行这个程序时，将以这个程序所属组的身份执行。

## 设置 SetGID

- 设置：`chmod g+s FILE`, `chmod 2755 FILE` (SetGID=2000)
- 取消：`chmod g-s FILE`, `chmod 755 FILE`
- 查看：`drwxr-s--- 2 root dip ... /etc/chatscripts/`
- 查找：`find / -perm -2000`
- 同时设置 SetUID 和 SetGID：`chmod 6755 FILE`



## 粘着位 (Sticky Bit)

如果一个权限为 777 的目录，被设置了粘着位，每个用户都可以在这个目录里面创建文件，但是只可以删除所有者是自己的文件。

## 设置粘着位

- 设置：`chmod o+t DIR, chmod +t DIR, chmod 1777 DIR`  
(粘着位 = 1000)
- 取消：`chmod o-t DIR, chmod -t DIR, chmod 777 DIR`
- 查看：`drwxrwxrwt 15 root root ... /tmp`
- 查找：`find / -perm -1000`

## 注意事项

设定粘着位的条件是文件必须具有 777 的权限，否则没有意义。

## 粘着位 (Sticky Bit)

如果一个权限为 777 的目录，被设置了粘着位，每个用户都可以在这个目录里面创建文件，但是只可以删除所有者是自己的文件。

## 设置粘着位

- 设置：`chmod o+t DIR, chmod +t DIR, chmod 1777 DIR`  
(粘着位 = 1000)
- 取消：`chmod o-t DIR, chmod -t DIR, chmod 777 DIR`
- 查看：`drwxrwxrwt 15 root root ... /tmp`
- 查找：`find / -perm -1000`

## 注意事项

设定粘着位的条件是文件必须具有 777 的权限，否则没有意义。



## 粘着位 (Sticky Bit)

如果一个权限为 777 的目录，被设置了粘着位，每个用户都可以在这个目录里面创建文件，但是只可以删除所有者是自己的文件。

## 设置粘着位

- 设置：`chmod o+t DIR, chmod +t DIR, chmod 1777 DIR`  
(粘着位 = 1000)
- 取消：`chmod o-t DIR, chmod -t DIR, chmod 777 DIR`
- 查看：`drwxrwxrwt 15 root root ... /tmp`
- 查找：`find / -perm -1000`

## 注意事项

设定粘着位的条件是文件必须具有 777 的权限，否则没有意义。

The `chmod` command can be used to set or unset with the following values as a **prefix** to the normal three numeric privileges:

Value	Explanation
0	SetUID, SetGID, sticky bits are unset
1	sticky bit is in place
2	SetGID bit is in place
3	SetGID and sticky bits are in place
4	SetUID bit is in place
5	SetUID and sticky bits are in place
6	SetUID and SetGID bits are on
7	SetUID, SetGID, sticky bits are activated



Permissions	Meaning
--S-----	SetUID is set, but user (owner) execute is not set.
--s-----	SetUID and user execute are both set.
-----S---	SetGID is set, but group execute is not set.
-----s---	SetGID and group execute are both set.
-----T	Sticky bit is set, but other execute is not set.
-----t	Sticky bit and other execute are both set.



- 1 插曲
- 2 引言
- 3 文件系统基础
  - 文件系统和分区
  - 目录结构
  - 路径
- 4 文件系统导航
  - 目录操作
  - 文件操作
  - 文件系统管理
  - 命令详解

- 5 文件类型
  - 类型简介
  - 链接
- 6 文件和目录权限
  - 权限简介
  - 修改权限
  - 特殊权限
- 7 挂载文件系统
- 8 回顾与总结
  - 总结
  - 思考题



# 文件系统 | 挂载 | mount

```
[root@AY130424102830Z ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/xvda1       20G   5.8G   13G   32% /
tmpfs            498M     0   498M    0% /dev/shm
/dev/xvdb1       9.9G  151M   9.2G    2% /www
[root@AY130424102830Z ~]# mount
/dev/xvda1 on / type ext3 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
tmpfs on /dev/shm type tmpfs (rw)
xenfs on /proc/xen type xenfs (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
/dev/xvdb1 on /www type ext3 (rw)
```

## 语法

- `mount -t FILE.SYSTEM.TYPE DEVICE DIRECTORY`
- `mount -t iso9660 /dev/cdrom /mnt/cdrom`
- `umount DEVICE.TO.UNMOUNT`
- `umount /dev/cdrom`

# 文件系统 | 挂载 | mount

```
[root@AY130424102830Z ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/xvda1      20G   5.8G   13G   32% /
tmpfs           498M    0   498M    0% /dev/shm
/dev/xvdb1      9.9G  151M   9.2G    2% /www
[root@AY130424102830Z ~]# mount
/dev/xvda1 on / type ext3 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
tmpfs on /dev/shm type tmpfs (rw)
xenfs on /proc/xen type xenfs (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
/dev/xvdb1 on /www type ext3 (rw)
```

## 语法

- `mount -t FILE.SYSTEM.TYPE DEVICE DIRECTORY`
- `mount -t iso9660 /dev/cdrom /mnt/cdrom`
- `umount DEVICE.TO.UNMOUNT`
- `umount /dev/cdrom`

# 教学提纲

- 1 插曲
- 2 引言
- 3 文件系统基础
  - 文件系统和分区
  - 目录结构
  - 路径
- 4 文件系统导航
  - 目录操作
  - 文件操作
  - 文件系统管理
  - 命令详解

- 5 文件类型
  - 类型简介
  - 链接
- 6 文件和目录权限
  - 权限简介
  - 修改权限
  - 特殊权限
- 7 挂载文件系统
- 8 回顾与总结
  - 总结
  - 思考题



- 1 插曲
- 2 引言
- 3 文件系统基础
  - 文件系统和分区
  - 目录结构
  - 路径
- 4 文件系统导航
  - 目录操作
  - 文件操作
  - 文件系统管理
  - 命令详解

- 5 文件类型
  - 类型简介
  - 链接
- 6 文件和目录权限
  - 权限简介
  - 修改权限
  - 特殊权限
- 7 挂载文件系统
- 8 回顾与总结
  - 总结
  - 思考题





## 知识点

- Linux 的文件系统：目录结构，主要的基本目录
- Linux 中的路径：绝对路径和相对路径
- 文件系统导航的常见命令
- Linux 中的文件类型：常见类型，硬链接和软链接
- Linux 中的权限：文件和目录的权限，符号模式和绝对模式
- 文件系统的挂载与卸载

## 技能

- 在命令行中进行文件系统的导航
- 在命令行中创建硬链接和软链接
- 在命令行中修改文件的权限
- 在命令行中挂载、卸载文件系统

# 教学提纲

- 1 插曲
- 2 引言
- 3 文件系统基础
  - 文件系统和分区
  - 目录结构
  - 路径
- 4 文件系统导航
  - 目录操作
  - 文件操作
  - 文件系统管理
  - 命令详解

- 5 文件类型
  - 类型简介
  - 链接
- 6 文件和目录权限
  - 权限简介
  - 修改权限
  - 特殊权限
- 7 挂载文件系统
- 8 回顾与总结
  - 总结
  - 思考题



- 1 列举 Linux 中的基本目录并解释其功能。
- 2 举例说明绝对路径和相对路径的区别。
- 3 列举几个进行文件系统导航的命令。
- 4 解释 ls -l 输出结果中每一列的含义。
- 5 比较 Linux 中的硬链接和软链接。
- 6 Linux 中的权限包括几种，针对哪些用户？
- 7 文件和目录的 rwx 权限有何异同？
- 8 举例说明如何使用符号模式修改权限？
- 9 举例说明如何使用绝对模式修改权限？



# 下节预告

总结日常使用 Windows 过程中的基本操作：目录操作、文件操作、系统管理、压缩解压、关机重启、……





TEX

LATEX

X<sub>Y</sub>TEX

Beamer

