

Linux 学习笔记

目 录

1	文件权限与目录配置	2
1.1	属性与权限	2
1.2	目录配置	2
2	文件与目录管理	5
2.1	管理	5
2.2	查阅文件内容	7
2.3	默认权限与隐藏权限	8
2.3.1	umask	8
2.3.2	隐藏属性	8
2.3.3	特殊权限	9
2.4	命令与文件查询	9
3	文件系统管理	11
3.1	简单操作	11
3.2	连接文件	12
3.3	磁盘操作	13
3.3.1	格式化	13
3.3.2	检查	13
3.3.3	挂载、卸载	14
3.3.4	修改参数	16
3.4	内存交换空间 (swap)	16

4 压缩与打包	17
4.1 压缩	17
4.2 备份	18
4.2.1 dump	18
4.2.2 restore	19
4.2.3 dd	19
4.2.4 cpio	20
5 bash	21
5.1 bash 的主要功能	21
5.2 变量	21
5.2.1 规则	21
5.2.2 环境变量	21
5.2.3 读入	23
5.2.4 修改	23
5.3 bash 环境	24
5.3.1 查找路径	24
5.3.2 欢迎信息	24
5.3.3 配置文件	24
5.3.4 环境设置	25
5.4 数据流	27
5.4.1 重定向	27
5.4.2 命令执行判断	27
5.5 管道命令	27
5.6 其他	30
5.6.1 命令别名	30
5.6.2 历史命令	31

目 录	3
6 正则表达式与文件格式化处理	32
6.1 正则表达式	32
6.2 文本处理工具	33
7 Linux 账号管理与 ACL 权限设置	35
7.1 用户账号	35
7.2 用户组	36
7.3 账号管理	36
7.3.1 创建账号	36
7.3.2 创建用户组	38
7.4 ACL	38
7.5 切换身份	39
8 vim	40
8.1 按键说明	40
8.2 块选择	41
8.3 有用的短命令	41
8.4 操作命令符	42
8.5 Ex 命令	42
8.6 分割窗口	42
8.6.1 切割窗口	42
8.6.2 切换窗口	43
8.6.3 关闭窗口	43
8.6.4 重排窗口	43
8.7 使用标签页	43
8.7.1 打开与关闭	43
8.7.2 标签页切换	44
8.8 文本对象	44
8.8.1 分隔符文本对象	44
8.8.2 范围文本对象	45

9 磁盘高级管理	46
9.1 磁盘配额 Quota	46
9.1.1 简介	46
9.1.2 使用流程	46
9.2 RAID	48
9.2.1 简介	48
9.2.2 软件 RAID 设置流程	48
9.3 LVM	49
9.3.1 基本概念	49
9.3.2 制作分区	49
9.3.3 PV 阶段	49
9.3.4 VG 阶段	50
9.3.5 LV 阶段	50
9.3.6 文件系统阶段	50
9.3.7 完整示例	51
9.3.8 制作快照	52
10 例行性工作 crontab	53
10.1 单一工作调度	53
10.1.1 atd 服务	53
10.1.2 相关命令	53
10.2 循环调度	54
10.3 唤醒停机期间的工作任务	54
11 程序管理	56
11.1 工作管理	56
11.2 进程管理	56
11.3 系统资源的查看	58

12 服务	61
13 虚拟机安装 Debian	62
13.1 分区设置	62
13.2 基本工具配置	62
13.2.1 更新源配置	62
13.2.2 安装虚拟机增强工具	62
13.2.3 建立共享文件夹	62
13.2.4 安装中文输入法	63
13.3 KDE 桌面环境	63
14 Samba 服务器	64
15 NFS 服务器	65
16 NIS 服务器	66
17 NIS,NFS 集群	67
17.1 配置	67
17.1.1 服务器	67
17.1.2 客户端	68
17.2 搭建	69
17.2.1 服务器	69
17.2.2 客户端	70

第 1 章 文件权限与目录配置

§ 1.1 属性与权限

- **chgrp**: 改变文件所属用户组

chgrp [-R] dirname/filename

参数 -R 表示进行递归修改.

- **chown**: 改变文件所有者

chown [-R] 账号名称 dirname/filename

chown [-R] 账号名称: 组名 dirname/filename

参数 -R 表示进行递归修改.

- **chmod**: 改变文件的权限

chmod [-R] xyz dirname/filename

或符号用法

chmode	u	+	r	
	g	-	x	文件或目录
	o	=	x	
	a			

§ 1.2 目录配置

1. FHS 标准

目录	应放置的文件内容
/bin	系统的执行文件. 单用户维护模式也可操作.
/boot	开机会使用到的文件, 包括 Linux 内核文件以及开机菜单与开机配置文件等.
/dev	任何设备与接口设备都以文件形式存在于这个目录中. 比较重要的文件有 /dev/null, /dev/zero, /dev/tty, /dev/hd* 等.
/etc	系统的主要配置文件, 例如系统账号密码文件、各种服务的起始文件等.
/home	默认的用户主文件夹
/lib	开机用到的函数库, 以及在 /bin 或 /sbin 下命令会调用的函数库.
/media	放置的是可删除设备, 包括光盘、DVD 等都暂挂在此. 常见的如 /media/cdrom 等.

目录	应放置的文件内容
/mnt	暂时挂载某些额外的设备, 例如 U 盘. 常见如 /mnt/flash 等.
/opt	第三方软件放置的目录, 如 KDE 等.
/root	系统管理员的主文件夹.
/sbin	包括开机、修复、还原系统所需的命令. 某些服务器软件放置到 /usr/sbin 中, 本机安装的软件所产生的系统执行文件放置到 /usr/local/sbin 中.
/srv	一些网络服务所需数据目录.
/tmp	一般用户正在执行的程序的临时目录.

2. 其余重要目录.

目录	应放置的文件内容
/lost+found	文件系统发生错误时, 会将丢失的片段放置到该目录.
/proc	虚拟文件系统, 该目录数据都在内存中. 较重要的文件如 /proc/cpuinfo, /proc/dma 等
/sys	虚拟文件系统, 记录与内核相关的信息, 包括已加载的内核模块和检测到的硬件设备信息等. 不占硬盘容量.

3. /usr 的内容与意义.

目录	应放置的文件内容
/usr/X11R6/	X Window 系统重要数据放置目录.
/usr/bin/	绝大部分用户可使用命令.
/usr/include/	头文件与包含文件.
/usr/lib/	包含应用软件的函数库、目标文件, 以及不被一般用户惯用的执行文件或脚本.
/usr/local/	系统管理员在本机安装自己下载的软件.
/usr/sbin/	非系统正常运行所需要的系统命令, 如某些网络服务器软件的服务命令.
/usr/share/	共享文件.
/usr/src/	一般源码. 内核源码建议放置在 /usr/src/linux/ 目录下.

4. /var 的内容与意义.

目录	应放置的文件内容
/var/cache/	应用程序的缓存文件.

目录	应放置的文件内容
/var/lib/	程序本身执行的过程中, 需要使用到的数据文件. 例如, MySQL 的数据库放置到 /var/lib/mysql/ 中.
/var/lock/	将设备上锁, 以确保文件只被单一软件使用.
/var/log/	登录文件.
/var/mail/	个人电子邮箱.
/var/run/	某些程序启动后, 会将他们的 PID 放置在该目录.
/var/spool/	通常放置一些队列数据.

第 2 章 文件与目录管理

§ 2.1 管理

1. 特殊目录:

.	: 当前目录
..	: 当前目录的上层目录
-	: 前一个工作目录
~	: 当前用户的主文件夹
~yong	: 用户名为 yong 的主文件夹

2. 相关命令:

- **cd** [相对路径或绝对路径 (不加该参数同 ~)]: 切换目录.
- **pwd** [-P]: 显示当前路径, 参数 -P 表示不显示连接路径而显示完整路径.
- **mkdir**: 新建目录

用法:	mkdir [-m] 目录名称
-m	: 配置文件夹权限. 直接设置, 不需要看默认权限 (umask)
-p	: 递归创建目录

- **rmdir**: 删除空目录

用法:	rmdir [-p] 目录名称
-p	: 连同上层空目录一并删除

- **ls**: 查看文件与目录

用法	: ls [-aAdFhilnrRSt] 目录名称
	ls [--color={never,auto,always}] 目录名称
	ls [--full-time] 目录名称
-a	: 列出全部文件 (含隐藏文件和 . 与 .. 两个目录)
-A	: 和 -a 同, 但无. 与 .. 两个目录
-d	: 仅目录本身
-f	: 不进行排序 (默认会以文件名排序)
-F	: 根据文件, 目录信息给予附加数据结构

<code>-h</code>	: 将文件容量以易读方式列出
<code>-i</code>	: 列出 inode 号码
<code>-l</code>	: 列出长数据串, 包含文件属性与权限等数据
<code>-n</code>	: 列出 UID 与 GID
<code>-r</code>	: 将排序结果反向输出
<code>-R</code>	: 连同子目录内容一起列出
<code>-S</code>	: 以文件容量大小排序
<code>-t</code>	: 以时间排序
<code>--color=never</code>	: 不依据文件特性给予颜色显示
<code>--color=always</code>	: 显示颜色
<code>--color=auto</code>	: 让系统自行判断
<code>--full-time</code>	: 以完整时间模式输出
<code>--time=atime,ctime</code>	: 输出 ctime

- `cp`: 复制
-

用法: `cp [-adfilprs] 源文件 目标文件`

`cp [options] 源文件1 源文件2 ... 目录`

<code>-a</code>	: 相当于 <code>-pdr</code>
<code>-d</code>	: 源文件为连接文件, 则复制连接文件属性而非文件本身
<code>-f</code>	: 强制
<code>-i</code>	: 覆盖时先询问
<code>-l</code>	: 进行硬连接, 而非复制文件本身
<code>-p</code>	: 连同文件的属性一起复制
<code>-r</code>	: 递归复制
<code>-s</code>	: 复制成 symbolic link
<code>-u</code>	: 当源文件更新时才复制

- `rm`: 移除文件或目录
-

用法: `rm [-fir] 文件或目录`

<code>-f</code>	: 强制
<code>-i</code>	: 互动模式
<code>-r</code>	: 递归删除

- `mv`: 移动文件与目录, 或更名

用法: `mv [-fiu] source destination`
`mv [options] source1 source2 ... destination`

`-f` : 强制
`-i` : 若覆盖先询问
`-u` : 若目标已存在, 且 source 更新, 才会更新

- `basename`: 取得路径的文件名
- `dirname`: 获取目录名

§ 2.2 查阅文件内容

- `cat`

用法: `cat [-AbEnTv] 文件`

`-A` : 相当于 `-vET`
`-b` : 列出行号, 空自行不标号
`-E` : 将结尾的断行符 \$ 显示出来
`-n` : 显示行号, 包括空自行
`-T` : 将 [Tab] 键以 ^I 显示
`-v` : 列出一些看不出来的特殊字符

- `tac`: 反向显示
- `nl`: 显示行号 (可设定行号格式和左右位置)
- `more`: 翻页查看
- `less`: 相比于 more, 功能更丰富
- `head [-n number] 文件`: 取出前面几行
- `tail [-n number] 文件`: 取出后面几行
- `od [-t TYPE(a,c,d,f,o,x)] 文件`: 读取非纯文本文件
- `touch [-acdmt] 文件`: 创建空文件或修改文件 mtime, atime
- `file 文件`: 查看文件类型

文件三个时间的意义:

- `mtime(modification time)`: 文件内容更改时, 会更新该时间.
- `ctime(status time)`: 文件的状态改变时 (如权限与属性等), 会更新该时间.
- `atime(access time)`: 文件内容被取用, 会更新该时间.

§ 2.3 默认权限与隐藏权限

§ 2.3.1 umask

`umask` 可指定当前用户在新建文件或目录时权限的默认值.

- `umask [分数]`: 设置默认权限. 不加参数时, 显示当前的分数, 分数表示的是, 创建文件或目录的默认值需要减掉的权限.
- `umask -S`: 符号显示当前的默认权限.

§ 2.3.2 隐藏属性

- `chattr`: 设置文件的隐藏属性

用法: `chattr [+--] [ASacdstu] 文件或目录`

- A : 访问该文件或目录时, atime 不会被修改
 - S : 对文件做任何修改都会同步写入磁盘
 - a : 只能增加数据, 不能删除和修改数据. root 权限才能设置.
 - c : 自动压缩, 读取时自动解压
 - d : 不会被 dump 备份
 - i : 不能删除, 改名, 设置连接文件, 写入或添加数据. root 权限才能设置.
 - s : 文件删除时, 将会完全从硬盘中删除
 - u : 文件删除时, 内容还存在于硬盘中, 可找回
-

- `lsattr`: 显示隐藏属性

用法: `chattr [-adR] 文件或目录`

- a : 显示隐藏属性
 - d : 仅列出目录本身的属性
 - R : 连同子目录的数据一并列出
-

§ 2.3.3 特殊权限

- Set UID: 数字为 4, 表现为 s 这个标志在文件所有者的 x 权限上.
 - (a). 仅对二进制程序有效
 - (b). 执行者对该程序具有 x 权限
 - (c). 本权限仅在执行过程中有效
 - (d). 执行者将获得该程序所有者的权限
- Set GID: 数字为 2, 表现为 s 这个标志在文件用户组的 x 权限上. 可针对目录设置.
 - (a). 对二进制程序有效
 - (b). 执行者对该程序具有 x 权限
 - (c). 在执行过程中将获得该程序用户组的支持

对目录设置时:

 - (a). 用户在此目录下的有效用户组将会变成该目录的用户组
 - (b). 在此目录下, 用户创建的文件用户组与该目录用户组相同
- Sticky Bit: 数字为 1, 只针对目录有效. 用户在此目录下创建的文件, 仅自己和 root 才能删除.

§ 2.4 命令与文件查询

1. 普通查找

命令	参数	意义
which	-a	将所有 PATH 目录中可找到的命令列出
whereis	-b	只找二进制格式文件
	-m	只找在说明文件 manual 路径下的文件
	-s	只找 source 源文件
	-u	查找不在上述三个选项中的其他特殊文件
locate	-i	忽略大小写差异
	-r	后可接正则表达式

updatedb 命令依据 /etc/updatedb.conf 的设置查找硬盘内的文件名, 并更新 /var/lib/mlocate 内的数据文件. locate 命令依据 /var/lib/mlocate 内的数据库进行关键字查找.

2. find

使用: `find [PATH] [option] [action]`

参数:

1. 与时间有关的参数: `-atime`, `-ctime`, `-mtime`, 以其中一个为例

`-mtime n` : n 天之前那天修改过的文件

`-mtime +n` : 改动时间超过 n 天 (不含第 n 天) 的文件

`-mtime -n` : n 天之内 (含第 n 天) 改动过的文件

`-newer file` : 比 file 还要新的文件, 其中 file 为已存在的文件

2. 与用户或用户组有关的参数

`-uid n` : UID

`-gid n` : GID

`-user name` : 用户账号

`-group name` : 用户组名

`-nouser` : 文件所有者不在 /etc/passwd 中

`-nogroup` : 文件所属组不在 /etc/group 中

2. 与权限有关的参数

`-name filename` : 文件名, 可用通配符

`-size [+-]SIZE` : 比 SIZE 大或小的文件

`-type TYPE` : f: 一般正规文件, b, c: 设备文件, d: 目录, l: 连接文件, s: socket, p: FIFO 等

`-perm mode` : 权限等于 mode 的文件

`-perm -mode` : 权限包含了 mode 的文件

`-perm +mode` : 权限包含了 mode 任意部分的文件

3. 其他

`-exec command` : 接其他命令来处理找到的结果. 不支持命令别名.

`-print` : 将结果输出到屏幕, 这是默认操作

第3章 文件系统管理

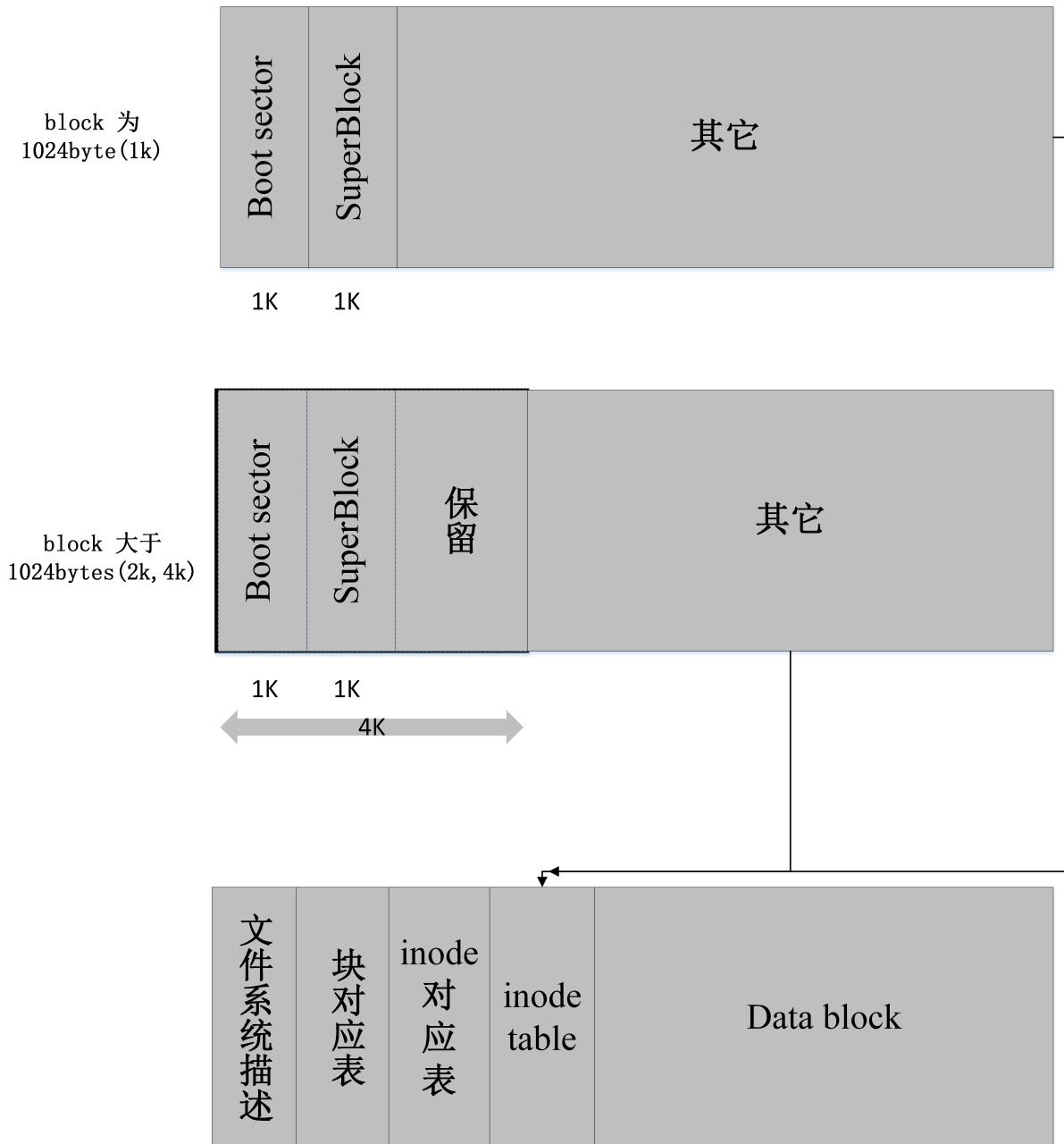


图 3-1 Ext2 文件系统

§3.1 简单操作

1. **df**: 列出文件系统的整体磁盘使用量

用法 : df [-ahikHTm] [目录或文件名]

-a : 列出所有文件系统

-k : 以 KB 为容量单位显示
-m : 以 MB 为容量单位显示
-h : 以容易阅读的方式为容量单位显示
-H : 以 M=1000K 替代 M=1024K 的进位方式
-T : 连同该分区的文件系统名称 (如 ext3) 也列出
-i : 不以硬盘容量, 而以 inode 数量来显示

2. du: 评估文件系统的磁盘使用量 (常用于评估目录所占容量)

用法 : du [-ahskm] 目录或文件名
-a : 列出所有文件和目录的容量, 因为默认仅统计目录下面的文件量而已
-k,-m,-h : 与 df 中的意义相同
-s : 列出总量而已, 而不列出单个目录占用容量
-S : 不包括子目录下的统计

§3.2 连接文件

1. **hard link**: 每个文件都会占用一个 inode, inode 会记录文件具体内容的位置 (block) 信息. 而目录的 block 中会记录文件名与文件占用的 inode 的对应关系, hard link 是在目录 block 中新建一条文件名连接到某 inode 的关联记录.

例如, ~/Desktop 目录下有 test.txt 的文件

创建 : ln test.txt test.ln ⇐ ln targetFile newFileName
查看 : ll -i test.txt test.ln
可以发现两个文件完全一样.

hard link 的限制是: 不能跨文件系统; 不能连接到目录.

2. **symbolic link**: 创建一个独立的文件, 这个文件会让数据的读取指向它连接的那个文件的文件名. 类似于 windows 下的快捷方式.

用法 : ln [-sf] 源文件 目标文件
-s : symbolic link, 不加该参数为 hard link
-f : 目标文件存在时, 就主动将目标文件删除后再创建

3. **连接数量**: 新建一个目录, 目录本身连接数为 2 (其本身和 .), 该目录的上层目录连接数增加 1 (该目录带来的 ..) .

§ 3.3 磁盘操作

§ 3.3.1 格式化

1. **fdisk**: 分区.

用法: `fdisk [-l] 设备名称`

`-l` : 输出后面接的设备的分区内容, 未加设备名称则列出整个系统的设备

不加 `-l` 参数, 则对设备进行分区, 具体操作会有提示. 操作完成以后, 使用 `partprobe*` 命令可以强制内核重新找一次分区表, 从而不需要重启.

限制: 无法处理大于 2TB 以上的磁盘分区.

2. **mkfs**: 格式化.

用法: `mkfs [-t 文件系统格式] 设备文件名`

`-t` : 可接的文件系统格式, 例如, ext3, ext2, vfat 等 (系统支持才生效)

3. **mke2fs**: 根据指定的详细信息格式化.

4. **parted**: 大硬盘分区.

用法: `parted [设备] [命令 [参数]]`

范例: “`parted /dev/hdc mkpart logical ext3 15GB 16GB`”

§ 3.3.2 检查

1. **fsck**: 格式化.

用法: `fsck [-t 文件系统格式] [-ACay] 设备名称`

`-t` : 和 `mkfs` 用法相同, 一般系统通过 super block 去分辨文件系统, 因此通常不需要这个参数

`-A` : 依据 `/etc/fstab` 的内容, 将需要的设备扫描一次

`-a` : 自动修复检查到的所有问题扇区, 加上该参数不用一直按 y

`-y` : 与 `-a` 类似, 但是某些文件系统仅支持 `-y` 这个参数

`-c` : 可以在检验的过程中使用一个直方图显示进度

EXT2/EXT3 的额外参数功能:

`-f` : 强制细化检查

-D : 针对文件系统进行优化配置

注意: 执行该命令可能会对系统造成危害, 被检查的分区必须是卸载状态.

2. **badblocks**: 不常用.

§ 3.3.3 挂载、卸载

挂载须知:

- 单一文件系统不应该被重复挂载在不同的挂载点 (目录) 中
- 单一目录不应该重复挂载多个文件系统
- 作为挂载点的目录理论上应该都是空目录. 若非空, 则原有内容会暂时隐藏.

1. **mount**: 挂载.

用法: `mount -a`

```
mount [-l]
mount [-t 文件系统类型] [-L Label 名] [-o 额外选项] \
[-n] 设备名挂载点
```

`-a` : 依照配置文件 `/etc/fstab` 的数据将所有未挂载的磁盘都挂载上来

`-l` : 单纯输入 `mount` 会显示目前挂载的信息, 加上该参数可增列 Label 名称

`-n` : 默认情况下, 系统将实际挂载情况写入 `/etc/mtab` 中, 加上该参数可以不写入

`-L` : 利用文件系统的卷名 (Label) 进行挂载

`-o` : 后接一些额外参数, 比如账号, 密码, 读写权限等

- 挂载 U 盘
 - (a). “`fdisk -l`”: 查看 U 盘的设备文件名
 - (b). 确定挂载点 (可以新建挂载目录)
 - (c). 挂载
 - (d). `df`: 查看挂载情况

- 挂载光盘/DVD 镜像文件: 属于特殊设备 loop 挂载

`(a). 确定挂载点 (可以新建挂载目录)`
`(b). 挂载: “mount -o loop 光盘文件名挂载点”`

- 新建大文件制作 loop 设备文件: 感觉上就像多了一个分区.

- (a). 新建大文件: “`dd if=/dev/zero of=/home/loopdev bs=1M count=1024`”
- (b). 格式化: “`mkfs -t ext4 /home/loopdev`”
- (c). 确定挂载点 (可以新建挂载目录)
- (c). 挂载
- (d). 查看: “`df`”

2. `umount`: 卸载.

用法: `umount [-fn] 设备文件名或挂载点`

`-f` : 强制卸载

`-n` : 不更新 `/etc/mtab` 的情况下卸载

3. `/etc/fstab`: 开机挂载*. 该文件的六个字段的意义是,

- 磁盘设备文件名或该设备的 Label
以设备名时, 不能随便更改插槽; 以 Label 名时, 不能随便更改 Label 的名称.
- 挂载点
- 磁盘分区的文件系统
手动挂载时, 可以让系统自动检测文件系统类型, 但在该文件当中, 必须手动写入, 包括 ext3 等.
- 文件系统参数
一般默认使用 defaults 即可.
- 能否被 dump 备份命令作用
0 代表不要做 dump 备份, 1 代表每天进行 dump 备份, 2 代表其他不定日期的 dump 备份操作. 通常不是 0 就是 1.
- 是否以 fsck 检验扇区
0 表示不要检验, 1 表示最早检验, 2 表示需要检验. 通常根目录设置 1, 其他要检验的目录设置为 2.

开机挂载设置步骤:

- (a). 在 `/etc/fstab` 中添加开机挂载设备, 例如,

“`/dev/hdc6 /mnt/hdc6 ext3 defaults 1 2`”

- (b). 测试挂载 (先查看 “`df`” 是否已经挂载, 若已挂在先卸载), 防止无法开机,

“`mount -a`”

- (c). 若真无法开机, 进入单用户维护模式, 此时根目录为 `readonly` 状态, 使用如下命令改为可写,

*在虚拟机中, 可以将共享文件开机挂载, 具体命令, 可以查看虚拟机软件的帮助文档.

```
“mount -n -o remount,rw /”
```

§ 3.3.4 修改参数

1. **mknod**: 修改设备的 major 和 minor 数值.
2. **e2label**: 修改设备的 Label.
3. **tune2fs**: 修改设备 Label, 将 ext2 转为 ext3 等.
4. **hdparm**: 可以针对 IDE 接口设置高级参数, 对 IDE, SATA 接口硬盘进行测试.

§ 3.4 内存交换空间 (swap)

- 使用物理分区
 - (a). 分区: **fdisk**
 - (b). 格式化构建: “**mkswap** 设备文件名”
 - (c). 查看: **free**
 - (d). 加载: “**swapon** 设备文件名”
 - (e). 确认: “**swapon -s**”
- 使用文件
 - (a). 新增文件: “**dd if=/dev/zero of=/tmp/swap bs=1M count=128**”
 - (b). 格式化构建: “**mkswap /tmp/swap** 名”
 - (c). 查看: **free**
 - (d). 加载: “**swapon /tmp/swap**”
 - (e). 确认: “**swapon -s**”
 - (f). 关闭: “**swapoff /tmp/swap**”

限制: 最多 32 个 swap, x86_64 最大为 64GB.

第 4 章 压缩与打包

§ 4.1 压缩

1. gzip, zcat

用法: `gzip [-cdktv#] 文件名`

`zcat 文件名.gz`

`-c` : 将压缩数据输出到屏幕上, 可通过重定向处理

`-d` : 解压缩的参数

`-k` : 保留原文件

`-t` : 检验文件的一致性, 看文件有无错误

`-v` : 显示出原文件/压缩文件的压缩比等信息

`-#` : # 是某个数字, 表示压缩等级, `-1` 最快 (压缩比最低), `-9` 最慢 (压缩比最高), 默认是 `-6`

默认会保存成 `.gz` 的文件, 并删除原文件.

2. bzip2, bzcat

用法 : `bzip2 [-cdkzv#] 文件名`

`bzcat 文件名.bz2`

`-c, -d, -k, -v, -#`: 与 `gzip` 意义相同

`-z` : 压缩的参数, 自动生成扩展名 `.bz2`

3. tar

用法: `tar [-j|-z] [cv] [-f defineName] targetFilename ...`

`tar [-j|-z] [tv] [-f 文件名]`

`tar [-j|-z] [xv] [-f 文件名] [-C 路径]`

`-c` : 新建打包

`-t` : 查看打包文件的内容

`-x` : 解包

`-j` : 通过 `bzip2` 的支持进行压缩/解压缩, 此时文件名最好是
`*.tar.bz2`

`-z` : 通过 `gzip` 的支持进行压缩/解压缩, 此时文件名最好是
`*.tar.gz`

`-v` : 在压缩/解压缩过程中, 将正在处理的文件名显示出来

-f : 后接文件名, 建议单独一个参数, 第一个文件名是要保持的文件名, 之后是要打包文件的文件名

-C : 后接路径, 解压到特定目录

-P : 保留打包文件的原本权限与属性, 常用于备份重要的配置文件

-p : 保留绝对路径, 允许备份数据中含有根目录 (解压时注意安全)

--exclude=otherFile : 不打包 otherFile 文件

--newer-mtime="date": 仅备份比这个时刻还新的文件, 例如,
“**--newer-mtime="2014/10/01"**”

通常只需记住它的常用用法即可,

压缩 : `tar -jcv -f filename.tar.bz2` 被压缩的文件或目录

`tar -zcv -f filename.tar.gz` 被压缩的文件或目录

查询 : `tar -jtv -f filename.tar.bz2`

`tar -ztv -f filename.tar.gz`

解压缩: `tar -jxv -f filename.tar.bz2 -C` 解压的路径

`tar -zxv -f filename.tar.gz -C` 解压的路径

§ 4.2 备份

§ 4.2.1 dump

用来备份单一文件系统, 可以指定备份等级对文件系统进行差异备份, 从而节省备份空间. 但是备份数据是目录而非单一文件系统时, 有如下限制:

- 所有备份数据都要在该目录下
- 仅支持完整备份, 即等级 0
- 不支持 **-u** 参数

用法 : `dump [-Suvj] [-level] [-f 备份文件] 待备份数据`
`dump -W`

-s : 列出后面的备份数据需要多少空间才能备份完毕

-u : 将这次 `dump` 时间记录到 `/etc/dumpdates` 文件中

-v : 将 `dump` 的文件过程显示出来

-
- j : 加入 bzip2 支持, 将数据进行压缩, 默认压缩等级 2
 - level : 备份等级, -0~9
 - f : 和 tar 用法类似
 - W : 列出在 /etc/fstab 里面的具有 dump 设置的分区是否有备份过
-

§ 4.2.2 restore

用于恢复 dump 备份的文件.

-
- 用法 : `restore -t [-f dumpFile] [-h]`
 - `restore -C [-f dumpFile] [-D 挂载点]`
 - `restore -i [-f dumpFile]`
 - `restore -r [-f dumpFile]`
 - t : 查看备份文件中的数据, 类似于 `tar -t`
 - C : 比较 dump 与实际文件, 列出 dump 有记录且与目前文件系统不同的文件
 - i : 互动模式, 可以仅还原部分文件
 - r : 将整个文件系统还原
 - h : 查看完整备份数据中 `inode` 和文件系统 `label` 等信息
 - f : 后接要处理的文件
 - D : 查出后面挂载点与 dump 内有不同的文件
-

§ 4.2.3 dd

可以读取磁盘设备的内容 (几乎是直接读取扇区), 然后将整个设备备份成一个文件.

-
- 用法 : `dd if=fileFrom of=outputFile bs=blockSize count=blockCount`
 - if : 输入位置, 可以是文件或设备
 - of : 输出位置, 也可是文件或设备
 - bs : block 大小, 未指定默认是 512byte
 - count : block 数量, 仅读取指定的区块数
 - 示例: `dd if=/dev/hdc1 of=/tmp/whole.disk`
-

§ 4.2.4 cpio

可以备份包括设备文件在内的任何东西, 但是要配合其他命令使用.

用法 : `cpio -ovcB > [file|device]`
`cpio -ivcd u < [file|device]`
`cpio -ivct < [file|device]`

`-o` : 将数据复制到文件或设备
`-B` : 将默认的 block 由 512byte 增加到 5120 byte
`-i` : 从文件或设备复制数据到系统
`-d` : 自动新建目录
`-u` : 自动将较新的文件覆盖较旧的文件
`-t` : 查看以 cpio 新建的文件或设备内容
`-v` : 让存储过程中文件名可以在屏幕上显示
`-c` : 一种较新的 portable format 方式存储

示例: `find /boot | cpio -ovcB > /temp/boot.cpio`

第 5 章 bash

§ 5.1 bash 的主要功能

- 命令记忆: `~.bash_history` 文件会记录上次登录执行过的命令, 当前登录的历史命令保存在内存中, 注销之后才会保存到该文件.
- 命令与文件补全: 按 [Tab] 键补全
- 命令别名设置功能: 使用 `alias` 命令设置
- 作业控制、前台、后台控制
- 程序脚本
- 通配符

§ 5.2 变量

§ 5.2.1 规则

格式	“ <code>varName=varValue</code> ”, 中间不含空格.
要求	变量名只能为字母或数字, 且开通不能为数字.
单、双引号	变量内容含空格用单或双引号括起来, 双引号可保留特殊字符的特殊含义, 而单引号则将特殊字符视为一般字符. 例如, “ <code>var="lang is \$LANG"</code> ”, 则 “ <code>echo \$var</code> ” 得到 “ <code>lang is en_US</code> ”; 而若改用单引号则得到的是元字符串.
转义	转义字符 “\” 可将特殊符号 (如 [Enter], \$, \, 空格符, ! 等) 变为一般字符.
反单引号	一串命令中, 可用反单引号 “`命令`” 或 “\$(命令)” 来获取信息.
累加内容	“ <code>"\$ 变量名称"</code> 累加内容” 或 “ <code>\$(变量)</code> 累加内容”.
子进程用	<code>export</code> 命令, 导出变量使之成为环境变量.
取消变量	“ <code>unset 变量名</code> ”.

§ 5.2.2 环境变量

1. `env` 命令查看环境变量

部分如下:

HOME	用户主文件夹
SHELL	使用哪个 shell 程序
HISTSIZE	保存的历史命令条数, (Ubuntu 在 set 中)
MAIL	邮箱
PATH	执行文件查找路径, 使用冒号 (:) 隔开
LANG	语系数据, 对程序执行很重要
RANDOM	对应/dev/random 这个文件, 可使用 “\$RANDOM” 来获取随机数, 不同版本可能有差异

2. set 命令查看所有变量 (环境变量与自定义变量)

部分如下:

HISTFILE	历史命令放置文件
MAILCHECK	每隔多少秒扫描有无新信
PS1	命令提示符, 例如, 我设置的是 “[\u @ \h : \w]”, 详细格式参数如下, \d: 显示出“星期月日”的日期格式, 如“Mon Feb 2” \H: 完整的主机名 \h: 仅取主机名在第一个小数点之前的名字 \t: 24 小时时间, “HH:MM:SS” \T: 12 小时时间, “HH:MM:SS” \A: 24 小时时间, “HH:MM” \@: 12 小时时间格式的上午下午, “am/pm” \u: 用户账号名称 \v: bash 版本信息 \w: 完整的工作目录 \W: 利用 basename 取得的工作目录名 \#: 执行的第几个命令 \\$: 提示符,root 时为 #, 否则是 \$

3. \$ 是变量, 表示关于本 shell 的 PID, 用 “echo \$\$” 即可查看.

4. ? 表示上一个命令的返回值, 一般执行成功返回 0, 用 “echo \$?” 即可查看.

5. “export 命令名称”, 可以将自定义变量变成环境变量. 子进程只能继承父进程的环境变量.

§ 5.2.3 读入

1. 读取来自键盘的输入: `read [-pt] variable`

参数:

`-p`: 后面接提示字符

`-t`: 后面接等待的秒数

示例: `read -p "Input name: " -t 30 name`

2. 声明变量类型: `declare/typeset`, 变量默认类型为字符串.

使用: `declare [-aixr] variable`

`-a`: 数组类型, 设置: “`var[index]=content`”, 读取建议: “`$var[index]`”

`-i`: 整数类型

`-x`: 变成环境变量, 与 `export` 一样

`-r`: `readonly` 类型

§ 5.2.4 修改

1. 变量的删除与替换. 假设定义了变量 “`myVar=abcdabcd`”.

设置方式	说明
<code> \${变量# 关键字}</code>	从头开始寻找, 将符合的最短的数据删除, 例如, <code>echo \${myVar#a*}</code> , 得到的是 <code>bcdabcd</code>
<code> \${变量% 关键字}</code>	从尾开始寻找, 将符合的最短的数据删除, 例如, <code>echo \${myVar%b*}</code> , 得到的是 <code>abcd</code>
<code> \${变量### 关键字}</code>	从头开始寻找, 将符合的最长的数据删除, 例如, <code>echo \${myVar##a*}</code> , 得到的是空字符串
<code> \${变量%%# 关键字}</code>	从尾开始寻找, 将符合的最长的数据删除, 例如, <code>echo \${myVar%%b*}</code> , 得到的是 <code>a</code>
<code> \${变量/oldStr/newStr}</code>	替换符合的第一个字符串
<code> \${变量//oldStr//newStr}</code>	替换所有符合的字符串

2. 变量的测试与替换

设置方式	str 未设置	str 为空	str 非空
<code>var=\${str-expr}</code>	<code>var=expr</code>	<code>var=</code>	<code>var=\$str</code>

var=\${str:-expr}	var=expr	var=expr	var=\$str
var=\${str+expr}	var=	var=expr	var=expr
var=\${str:+expr}	var=	var=	var=expr
var=\${str=expr}	str=expr, var=expr	str 不变, var=	str 不变, var=\$str
var=\${str:=expr}	str=expr, var=expr	str=expr, var=expr	str 不变, var=\$str
var=\${str?expr}	expr 输出至 stderr	var=	var=\$str
var=\${str:?expr}	expr 输出至 stderr	expr 输出至 stderr	var=\$str

§ 5.3 bash 环境

§ 5.3.1 查找路径

1. 以相对/绝对路径执行命令;
2. 由 alias 找到该命令来执行;
3. 由 bash 内置的 (builtin) 命令来执行;
4. 通过 \$PATH 这个变量的顺序找到的第一个命令来执行.

§ 5.3.2 欢迎信息

/etc/issue 和 /etc/issue.net (telnet 连接), 可在 /etc/motd 中加入自己想要显示的信息.

§ 5.3.3 配置文件

1. /etc/profile: 系统的整体设置, 一般不要修改, 该文件主要变量有,

变量	作用
PATH	会依据 UID 决定 PATH 变量要不要含有 sbin 的系统命令目录
MAIL	根据账号设置好用户 mailbox
USER	根据用户账号设置此变量内容
HOSTNAME	根据主机 hostname 命令决定此变量内容
HISTSIZE	历史命令记录条数

该文件还会调用外部的设置数据, 例如,

- `/etc/inputrc`: 该文件内容为 bash 热键、[Tab] 有无声音等的数据, 建议不修改.
- `/etc/profile.d/*sh`: 若需要帮所有用户设置一些共享的命令别名时, 可以在该目录创建.sh 的文件, 将数据写入即可.
- `/etc/sysconfig/i18n`: 语系配置.

2. 加载完`/etc/profile`文件之后, 接下来会读取个人配置文件, 顺序是,

- `~/.bash_profile`: 会调用 `~/.bashrc`
- `~/.bash_login`
- `~/.profile`

若寻找到其中一个文件, 则不再继续读取.

3. 读入环境配置文件命令: “`source 配置文件`”. 若配置文件发生了改变, 可用此命令将配置文件重新读入.

4. 其他相关的配置文件例如, `~/.bash_history` 记录了历史命令, `~/.bash_logout` 记录了注销 bash 之后系统的操作.

§ 5.3.4 环境设置

1. bash 默认的组合键

组合键	执行结果
<code>Ctrl+C</code>	终止目前的命令
<code>Ctrl+D</code>	输入结束 (EOF), 例如邮件结束的时候.*
<code>Ctrl+M</code>	就是 Enter
<code>Ctrl+S</code>	暂停屏幕的输出
<code>Ctrl+Q</code>	恢复屏幕的输出
<code>Ctrl+U</code>	在提示符下, 将整行命令删除
<code>Ctrl+Z</code>	暂停目前的命令

2. bash 的通配符

组合键	执行结果
<code>*</code>	代表 0 个到无穷多个任意字符
<code>?</code>	代表一定有一个任意字符

*在 Debian 中, 当无字符时, 相当于 `exit`.

[]	一定有在其中的字符
[-]	连续字符
[^]	一定有不在其中的字符

3. bash 中的特殊字符

符号	内容
#	注释
\	转义
	管道
;	连续命令分隔符
~	用户主文件夹
\$	变量前导符
&	作业控制, 后台运行
!	逻辑运算符 “非”
/	路径分隔符
>, >>	数据流重定向, 输出
<, <<	数据流重定向, 输入
' '	单引号, 不具有变量置换功能
'' ''	具有变量置换功能
`` ``	反单引号, 优先执行其中的命令
()	在中间为 shell 的起始与结束
{ }	在中间为命令块的组合

4. stty [-a]: 查看 stty 参数

符号	内容
eof	输入结束
erase	向后删除字符
intr	送出一个 interrupt 信号给目前正在运行的程序
kill	删除所有命令行文字
quit	送出一个 quit 信号给目前正在运行的程序
start	在某个进程停止之后, 重新启动它的输出
stop	送出一个 terminal stop 信号给正在运行的程序

5. set: 设置终端机值.

§ 5.4 数据流

§ 5.4.1 重定向

标准输出 (stdin)	代码为 0, < 或 <<(输入结束的意思)
标准输出 (stdout)	代码为 1, >(覆盖) 或 >>(追加)
标准错误输出 (stderr)	代码为 2, 2>(覆盖) 或 2>>(追加)

1. 忽略信息, 输出至 /dev/null 即可.
2. 正确与错误数据输出到同一个文件: >fileName 2>&1 或 &> .

§ 5.4.2 命令执行判断

- cmd1 && cmd2 : cmd1 执行完毕, 返回正确 (\$?=0), 则开始执行 cmd2, 否则不执行
- cmd1 || cmd2 : cmd1 执行完毕, 返回错误 (\$?≠0), 则开始执行 cmd2, 否则不执行

Linux 命令由左往右执行, 例如,

1. 测试 /tmp/yong 是否存在

```
ls /tmp/yong && echo "exist" || echo "not exist"
```

2. 创建 /tmp/yong/test.txt

```
ls /tmp/yong || mkdir /tmp/yong && touch /tmp/yong/test.txt
```

§ 5.5 管道命令

1. **cut**: 取出一行中想要的部分, 包含了其他语言中 **split** 函数的功能.

用法 1: `cut -d '分隔符' -f fields`

参数:

`-d` : 例如 ' ' 表示以空格分隔字段, ',' 表示以逗号分隔字段等

`-f` : 选取第几个字段, 例如, “`-f 1,2`” 表示选取第 1,2 两个字段

用法 2: `cut -c 字符范围`

参数:

`-c` : 例如, “`-c 12-`” 取出每行第 12 个字符以后的部分, “`-c 5-9`” 取出每行第 5 到第 9 个字符的部分

2. **grep**: 分析一行中信息, 取出我们想要的行.

用法 1 : grep [-acinv] [--color=auto] '查找字符串' filename

参数:

- a : 将 binary 文件以 text 文件的方式查找数据
- c : 计算找到 '查找字符串' 的次数
- i : 忽略大小写
- n : 输出行号

--color=auto: 将找到的关键字部分加上颜色显示

3. **sort**: 可以依据不同的数据类型来排序.

用法 1: sort [-fbMnrtuk] [file or stdin]

参数:

- f : 忽略大小写
 - b : 忽略最前面的空白
 - M : 以月份名字来排序, 例如, JAN, DEC
 - n : 使用“纯数字”进行排序
 - r : 反向排序
 - u : 就是 uniq, 相同数据仅显示一行
 - t : 分隔符, 默认是 [Tab] 键
 - k : 以哪个字段 (field) 排序
-

4. **uniq**: 排序完成以后, 重复数据仅显示一个.

用法 1: uniq [-ic]

参数:

- i : 忽略大小写
 - c : 计数
-

5. **wc**: 统计行数, 字数, 字符数.

用法 1: wc [-lwm]

参数:

- l : 列出行数
 - w : 列出字数 (英文单词)
 - m : 列出字符数
-

5. **tee**: 双重定向, 将数据流输至文件盒屏幕 (stdout).

用法 1: `tee [-a] file`

参数:

`-a` : 追加方式写入文件

6. **tr**: 删除一段信息当中的某些文字, 或者进行替换, 类似于其他语言的 replace 函数.

用法 1: `tr [-ds] SET1`

参数:

`-d` : 删除 SET1 这个字符串

`-s` : 替换掉重复的字符

示范:

`tr '[a-z]' '[A-Z]'`: 将小写字符变成大写字符

`tr -d '\r'`: 删除 DOS 断行符

7. **col**.

用法 1: `col [-xb]`

参数:

`-x` : 将 tab 键转换成对等的空格键

`-b` : 在文字有反斜杠 (/) 时, 仅保留反斜杠最后接的那个字符

8. **join**: 将两个文件中有相同数据的那一行加在一起. 需要注意的是, 使用之前, 应该对文件进行了排序.

用法 1: `join [-ti12] file1 file2`

参数:

`-t` : 默认以分隔符分隔数据, 并对比第一个字段

`-i` : 忽略大小写

`-1` : 后接数字, 表示第一个文件用哪个字段分析

`-2` : 后接数字, 表示第二个文件用哪个字段分析

示范:

`join -t ':' -1 2 -2 2 a.txt b.txt`

9. **paste**: 将两行粘贴一起, 以 [tab] 键隔开.

用法 1: `paste [-d] file1 file2`

参数:

- d : 后接分隔符, 默认以 [tab] 分隔
 - : file 部分写成 -, 表示数据来自 stdin
-

10. **expand**: 将 [tab] 键转成空格.

用法 1: expand [-t] file

参数:

- t : 后接数字, 表示一个 [tab] 转换为多少个空格键, 默认为 8 个
 - : file 部分写成 -, 表示数据来自 stdin
-

11. **split**: 根据大小或者行数将大文件切割成小文件.

用法 1: split [-bl] file preName

参数:

- b : 按大小切割, 后接预分隔的大小, 单位为 b,k,m 等
- l : 按行数切割, 后接数字
- : file 部分写成 -, 表示数据来自 stdin

preName: 分隔后的文件名的前缀

12. **xargs**: 读入 stdin 的数据, 以空格符或断行符进行分辨, 将其分割成参数, 传递给命令.

用法 1: xargs [-0epn] command

参数:

- 0 : 若 stdin 含有特殊字符, 该参数将其还原为一般字符
 - e : EOF 之意, 后接字符串, 当分析到这个字符串时, 停止分析工作
 - p : 执行每个命令的参数时, 都会询问用户
 - n : 后接数字, 每次 command 执行时, 使用几个参数
-

值得注意的是, 减号 - 的作用, 例如, “tar -cvf 0 /home/yong | tar -xvf -”.

§ 5.6 其他

§ 5.6.1 命令别名

- alias defineName=command: 设置别名规则与设置变量规则几乎相同
- alias: 查看已有别名

- unalias defineName: 取消别名

§ 5.6.2 历史命令

1. history

用法: `history [n]`
`history [-c]`
`history [-raw] histfiles`

参数:

- n : 数字, 列出最近的 n 条命令
 - c : 将目前 shell 中所有的 history 内容清除
 - a : 将目前新增的 history 命令追加写入 hisfiles 中
 - r : 将 hisfiles 的内容读到目前这个 shell 的 history 记忆中
 - w : 将目前的 history 记忆内容写入 hisfiles 中
-

注意, 若未加 hisfiles, 则默认写入 `~/.bash_history` 中, 该文件的记录条数与 `HISTSIZE` 这个变量的值有关.

2. 执行历史命令

- `!number` : 执行第几条命令
 - `!command`: 由最近的命令向前搜索, 寻找开头为 command 的命令并执行
 - `!!` : 执行上一个命令
-

第 6 章 正则表达式与文件格式化处理

§ 6.1 正则表达式

RE 字符	意义
<code>^word</code>	待查找的字符串在行首
<code>word\$</code>	待查找的字符串在行尾
<code>.</code>	一定有一个任意字符
<code>\</code>	转义字符, 将特殊符号的特殊意义去除
<code>*</code>	重复零个或无穷多个前一个字符
<code>[list]</code>	从字符集合找出想要的字符
<code>[n1-n2]</code>	- 代表连续字符
<code>[^list]</code>	找出不属于字符集合的字符
<code>\{n,m\}</code>	连续 n 到 m 个前一个字符

表 6-1 基础正则表达式

特殊符号	代表意义
<code>[:alnum:]</code>	大小写字母以及数字, 0-9, A-Z, a-z
<code>[:alpha:]</code>	大小写字母, A-Z
<code>[:blank:]</code>	空格键与 [Tab] 键
<code>[:cntrl:]</code>	控制按键, CR, LF, Tab, Del 等
<code>[:digit:]</code>	数字, 0-9
<code>[:graph:]</code>	除空格符 (空格键与 [Tab] 键) 外的其他按键
<code>[:lower:]</code>	小写字符, a-z
<code>[:print:]</code>	可以被打印的字符
<code>[:punct:]</code>	标点符号,"'?!;:#\$
<code>[:upper:]</code>	大写字符, A-Z
<code>[:space:]</code>	任何产生空白的字符, 空格键、[Tab] 键、CR 等
<code>[:xdigit:]</code>	十六进制数字

表 6-2 特殊符号

RE 字符	意义
+	重复一个或多个前一个 RE 字符
?	零个或一个 RE 字符
	用或 (or) 的方式找出字符串
()	找出“组”字符串
() +	多个重复组

表 6-3 扩展正则表达式

§ 6.2 文本处理工具

- **sed [-nefr] [动作]**: 作用于一行

参数:

-n: 使用安静模式, 只有经过 sed 特殊处理的行才列出
 -e: 直接在命令行模式上进行 sed 的动作编辑, 若不加该参数, 则 sed 后接的动作要以'`两个单引号括住

-f: -f filename 可直接执行 filename 文件中的 sed 动作
 -r: 支持扩展正则表达式语法 (默认是基础正则表达式语法)
 -i: 直接修改文件内容, 而不是由屏幕输出

动作: [n1[,n2]]] function

a: 新增, 后接字符串, 出现在新的一行 (当前行的下一行), 例如, '2a new line', 第 2 行后加上字符串
 c: 替换, 后接字符串, 替换 n1, n2 之间的行
 d: 删除, 例如, '2,5d', 删除第 2 至 5 行
 i: 插入, 后接字符串, 出现在新的一行 (当前行的上一行)
 p: 打印, 通常与 sed -n 一起运行
 s: 替换, 可搭配正则表达式, 例如, sed 's/old/new/g'

需要注意的是, 正则表达式置于//之间, 例如, sed -r '/#\.\.*\\$/ d', 删除所有以 # 号开头或者空白的行. 这个命令可以获取文档中除去注释和空白行后的有效信息.

- 格式化打印: printf '打印格式' 实际内容

参数:

\a : 警告声音输出
 \b : 退格键 (backspace)

\f : 清除屏幕 (form feed)
\n : 输出新的一行
\r : Enter 按键
\t : 水平的 [tab] 按键
\xNN : NN 为两位数的数字, 转换数字为字符

C 语言变量格式:

%ns : 输出字符串

%ni : 输出整数

%N.nf: 输出浮点数

- awk: 将一行分段处理

使用:

```
awk '条件类型 1 {动作 1} 条件类型 2 {动作 2} ...' filename
```

内置变量:

NF: 每一行 (\$0) 拥有的字段总数

NR: 目前 awk 所处的是“第几行”数据

FS: 目前的分隔字符, 默认是空格键

逻辑运算符: >, <, >=, <=, ==, !=

awk 以行为一次处理单位, 将一行分成各个字段填入 \$0,\$1,\$2 等变量中.

- diff: 以行为单位比较两个文件之间的区别
- cmp: 以字节为单位比较两个文件
- patch: 制作升级补丁

第 7 章 Linux 账号管理与 ACL 权限设置

§7.1 用户账号

系统处理流程,

- 进入 /etc/passwd 中寻找输入的账号, 读取 UID 以及对应的 GID(/etc/group 中), 若未找到则跳出;
- 进入 /etc/shadow 找出对应的账号和 UID, 核对密码;
- 以上都通过进入则进入 shell.

相关文件结构,

- /etc/passwd 文件结构:

例如, yong:x:1000:1000:Yong:/home/yong:/bin/bash

1. 账号名称: yong;
2. 密码: 该字段经过加密保存于 /etc/shadow 中;
3. UID: User ID;
4. GID: Group ID;
5. 用户信息说明列;
6. 主文件夹: /home/yong;
7. Shell: bash.

- /etc/shadow 文件结构:

例如, yong:\$6\$0... (省略):16310:0:99999:7:::

1. 账号名称: yong;
2. 密码: 经过加密的字符串;
3. 最近更改密码的日期: 从 1970 年 1 月 1 日作为 1 开始累加;
4. 密码不可被更改的天数: 0, 表示可以随意改动;
5. 密码需要重新更改的天数: 99999, 不强制你更改之意;
6. 密码需要更改期限前的警告天数: 7 天;
7. 密码过期后的账号宽限时间 (密码失效日): 空;
8. 账号失效日期: 空;
9. 保留: 空.

§ 7.2 用户组

相关文件结构,

- /etc/group 文件结构:

记录 GID 与组名的对应关系, 例如, family:x:1002:yong,luo

1. 用户组名称: family;
2. 密码: 该字段经过加密保存于 /etc/gshadow 中;
3. GID: Group ID;
4. 此用户组支持的账号名称: 逗号隔开, 不含空格.

- /etc/gshadow 文件结构:

例如, family!:!:yong:yong,luo

1. 用户组名称: family;
2. 密码: “!” 表示无密码;
3. 管理员账户: yong;
4. 用户组所属账户: 与 /etc/group 内容相同.

有效用户组: 创建文件默认所属的用户组 (groups 命令查看第一个即是, 使用命令 newgrp 更改).

§ 7.3 账号管理

§ 7.3.1 创建账号

- useradd [参数选项] 用户账户名 *

参数:

- u : 直接指定 UID;
- g : 初始用户组;
- G : 次要用户组;
- M : 强制! 不创建主文件夹 (系统账号默认) ;
- m : 强制! 创建主文件夹;
- c : /etc/passwd 第 5 列说明内容, 可以随便设置;
- d : 直接指定主文件夹 (使用绝对路径), 而不使用默认值;

* 在 Debian 中, 默认不会创建主文件夹, 且 shell 默认为 /bin/dash, 因此最好手动指定这些参数.

- r : 创建系统账号;
 - s : shell, 若不指定, 默认为/bin/bash;
 - e : 日期, “YYYY-MM-DD”, 账号失效日期, 写入 /etc/shadow;
 - f : 密码是否失效,0 为立即失效, -1 为永远不失效,shadow 第 7 个字段.
-

- useradd 参考文件数据是 /etc/default/useradd*, 例如,

HOME=/home	: 用户主文件夹基目录;
INACTIVE=-1	: 密码过期后是否失效;
EXPIRE=	: 账号失效日期;
SHELL=/bin/bash	: 默认使用的 shell 程序文件名;
CREATE_MAIL_SPOOL=yes	: 创建用户的 mailbox;
SKEL=/etc/skel	: 用户主文件夹参考基准目录, 新建用户主文件各项数据都是有该目录赋值过去的.

- UID/GID 以及密码参数参考文件是 /etc/login.defs, 可以使用命令

```
cat /etc/login.defs | sed -r '/#.*$|^$/ d' | sort
```

查看该文件默认设置.

总结: 使用 useradd 程序创建账号时, 会参考 /etc/default/useradd、/etc/login.defs、/etc/skel/* 并向 /etc/passwd、/etc/shadow、/etc/group、/etc/gshadow 写入数据, 以及创建用户主文件夹. 系统账号 shell 使用/sbin/nologin, 故无法登录 shell, 可以新建 /etc/nologin.txt 文件, 试图登录系统账号时, 屏幕出现该文件内容.

- 与创建账号相关的命令

passwd : 设置密码的相关参数

chage : 显示以及修改/etc/shadow 文件密码的相关字段, 例如可要求账号首次登陆修改密码

usermod: /etc/passwd 和/etc/shadow 与账号相关数据的微调

userdel: 删除用户的相关数据. 若只是暂停该账号使用, 可以将/etc/shadow 账号失效日期 (第 8 列) 设置为 0 即可.

- 用户功能的相关命令

finger: 查阅很多用户的相关信息

chfn : 有点像 change finger, 修改一些可查阅的信息, 例如办公室号码

*Debian 中, 默认的许多项处于注释状态, 且 shell 使用的是/bin/sh, 指向/bin/dash.

chsh : change shell

id : 查阅账号的 UID/GID 等信息.

§ 7.3.2 创建用户组

- groupadd [-g GID] [-r] 用户组名
- groupmod [-g GID] [-n group_name] 用户组名: 修改 GID 或组名;
- groupdel: 删除用户组;
- gpasswd: 用户组管理员功能, 设置账号为管理员, 将用户添加进用户组或从用户组中删除用户.
 - gpasswd groupname: 设置密码;
 - gpasswd [-A user1,...] [-M userA,...] groupname: A 为设置管理员,M 为添加用户进组;
 - gpasswd [-rR] groupname: r 为删除密码,R 为密码栏失效;
 - gpasswd [-ad] user groupname: 管理员添加与删除用户.

§ 7.4 ACL

ACL 可以针对单一用户或用户组、单一文件或目录进行权限设置.

- setfacl [-bkRd] [-m|-x acl 参数] 目标文件名
-

参数:

-m: 设置 ACL 参数;

-x: 删除 ACL 参数;

-b: 删除所有的 ACL 参数;

-k: 删除默认的 ACL 参数;

-R: 递归设置 ACL 参数;

-d: 对目录设置默认 ACL 参数, 在该目录新建数据将引用此值.

1. u:[用户账号列表]:[rwx]: 对特定用户设置

```
setfacl -m u:yong:rwx aclDir
```

2. g:[用户组列表]:[rwx]: 对用户组设置

```
setfacl -m g:family:rx aclDir
```

3. m: [rwx]: 设置 mask 有效权限, 用户与用户组实际权限不超过此设置

```
setfacl -m m:rwx aclDir
```

4. d: [ug]: 用户列表: [rwx]: 设置默认权限, 用户创建的文件都默认该 ACL 设置

```
setfacl -m d:u:yong:rx aclDir
```

- `getfacl filename`: 一个文件设置 ACL 参数以后, 它的权限部分多出一个 + 号, 使用该命令可以查询详细的 ACL 参数.

§ 7.5 切换身份

- `su [-lm] [-c 命令] [username]`

参数:

- : 使用 login-shell 的变量文件读取方式登录系统, 不加用户名表示切换为 root;

-l: 与上类似, 但需要加用户名;

-m: 与 -p 一样, 使用目前的环境设置, 而不读取新用户的配置文件;

-c: 仅进行一次命令.

- `sudo [-b] [-u 新用户账号]`

该命令会在 /etc/sudoers 文件中查找用户是否具有执行 sudo 的权限, 该文件使用 visudo 来编辑. 通过修改该文件, 可以限定用户或用户组可执行的命令, 可免除密码等.

第 8 章 vim

一般模式，可以进行复制、移动与删除

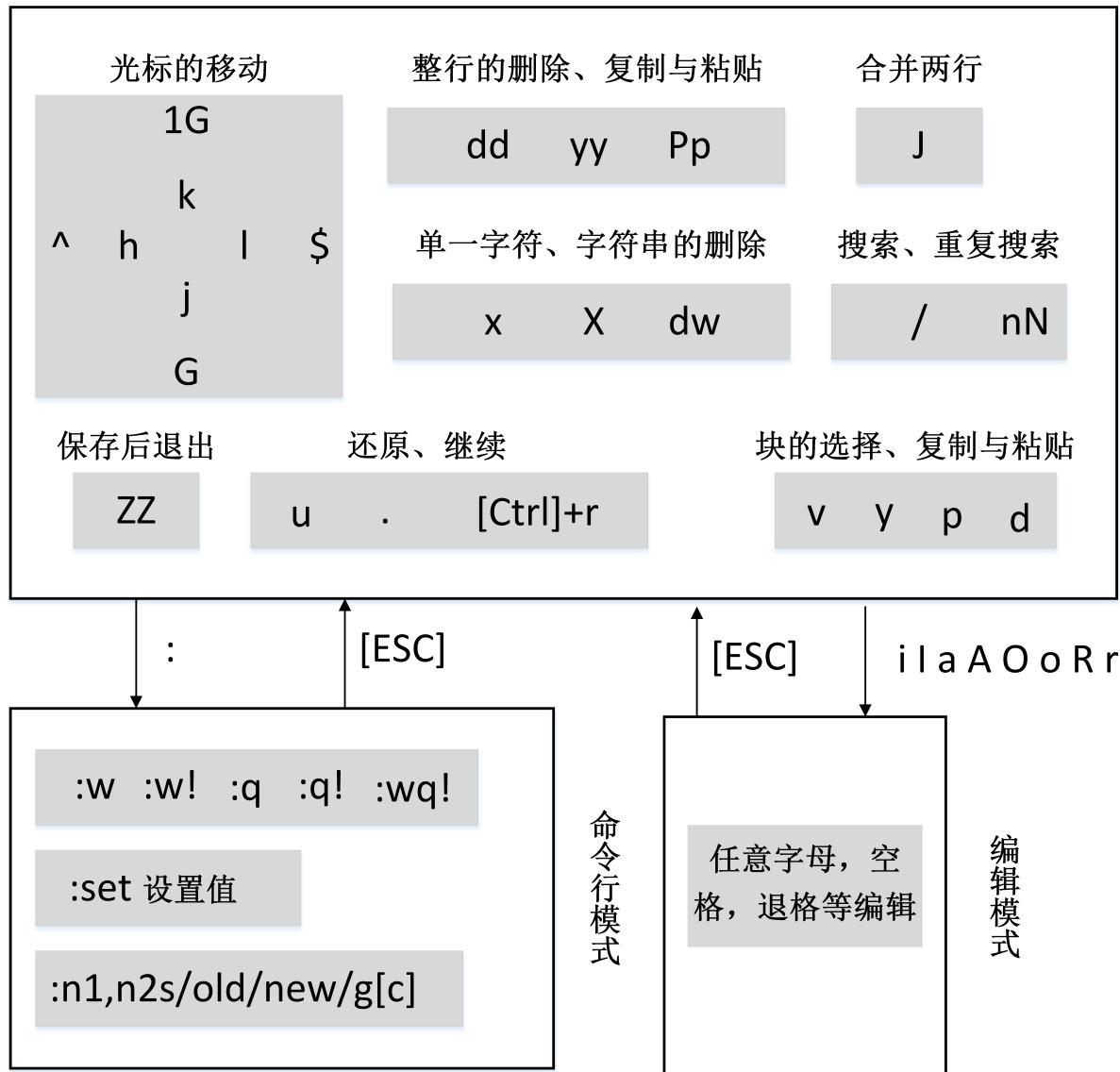


图 8-1 vim 常用命令查询

§ 8.1 按键说明

vim 按键说明

<code>[Ctrl]+f</code>	下一页，相当于 <code>[Page Down]</code> *
<code>[Ctrl]+b</code>	上一页，相当于 <code>[Page Up]</code>
0 或 <code>[Home]</code>	移动到这一行最前面的字符

*Debian 和 Ubuntu 预装的是 Vim tiny，需要安装完整版。

vim 按键说明

\$ 或 [End]	移动到这一行最后一个字符
G	移动到这个文件的最后一行
nG	移动到第 n 行
gg	移动到文件的第 1 行
/word	向下搜索字符串
:n1,n2s/old/new/g[c]	将第 n1 至 n2 行之间的字符串替换, c 表示要经过用户确认. \$ 表示最后一行.
x,X	分别为向后、向前删除一个字符
dd	删除光标所在行
ndd	删除 n 行
yy	复制光标所在行
nyy	复制 n 行
p,P	分别为向下和向上粘贴
J	合并光标所在行以及下一行
u	还原
[Ctrl]+r	重复, 与 u 对应
.	重复前一个操作

§ 8.2 块选择

块选择按键

V	字符与行选择
[Ctrl]+v	块选择
yy	复制选中部分
dd	删除选择部分
o	切换高亮选区的活动端

§ 8.3 有用的短命令

复合命令 等效的长命令 意义

C	c\$	替换该行光标之后部分 (含光标)
s	cl	替换光标所在处字符
S	^c	替换整行 (插入光标在原该行第一个非空白字符处)
I	^i	行首插入

复合命令	等效的长命令	意义
A	\$a	行尾插入
o	A<CR>	开启空白行
O	ko	在上方开启空白行

§ 8.4 操作命令符

命令	意义
c	修改
d	删除
y	复制到寄存器
g~	反转大小写
gu	转换为小写
gU	转换为大写
>	增加缩进
<	减小缩进
=	自动缩进
!	使用外部程序过滤 motion 所跨越的行

§ 8.5 Ex 命令

命令	意义
: [range]delete [x]	删除指定范围内的行 [到寄存器 x 中]
: [range]yank [x]	复制指定范围的行 [到寄存器 x 中]
: [range]put [x]	在指定行后粘贴寄存器 x 中的内容
: [range]copy {address}	在指定范围内的行拷贝到 {address}
: [range]move {address}	在指定范围内的行移动到 {address}
: [range]join	连接指定范围内的行
: [range]normal {commands}	对指定范围内的每一行执行普通模式命令
: [range]s/{pattern}/{string}/[flags]	按模式替换
: [range]global/{pattern}/{cmd}	对匹配行执行 Ex 命令

§ 8.6 分割窗口

§ 8.6.1 切割窗口

命令	意义
<C-w>s	水平切分当前窗口, 新窗口仍然显示当前缓冲区
<C-w>v	垂直切分当前窗口, 新窗口仍然显示当前缓冲区
:sp[lit] {file}	水平切分当前窗口, 并在新窗口载入文件
:vsp[lit] {file}	垂直切分当前窗口, 并在新窗口载入文件

§ 8.6.2 切换窗口

命令	意义
<C-w>w	在窗口间循环切换
<C-w>h	切换到左边的窗口
<C-w>j	切换到下边的窗口
<C-w>k	切换到上边的窗口
<C-w>l	切换到右边的窗口

§ 8.6.3 关闭窗口

Ex 命令	普通模式命令	意义
:clo[se]	<C-w>c	关闭活动窗口
:on[ly]	<C-w>o	保留活动活动窗口, 关闭其他窗口

§ 8.6.4 重排窗口

命令	意义
<C-w>=	使所有窗口等宽、等高
<C-w>_	最大化活动窗口的高度
<C-w>	最大化活动窗口的宽度
[N]<C-w>_	把活动窗口的高度设为 [N] 行
[N]<C-w>	把活动窗口的宽度设为 [N] 列

§ 8.7 使用标签页

§ 8.7.1 打开与关闭

命令	意义
:tabe [dit] {filename}	在新标签打开文件
<C-w>T	把当前窗口移动到一个新的标签
:tabc [lose]	关闭当前标签页及其包含的所有窗口
:tabo [nly]	只保留活动标签页, 关闭所以其他标签页

§ 8.7.2 标签页切换

命令	普通模式命令	意义
:tabn [ext] {N}	{N}gt	切换到编号为 N 的标签页
:tabn [ext]	gt	切换下一个标签页
:tabn [revious]	gT	切换上一个标签页

§ 8.8 文本对象

§ 8.8.1 分隔符文本对象

文本对象	选择区域
a)	或 ab 一对圆括号
i)	或 ib 圆括内部
a}	或 aB 一对花括号
i}	或 iB 花括号内部
a]	一对方括号
i]	方括号内部
a>	一对尖括号
i>	尖括号内部
a'	一对单引号
i'	单引号内部
a"	一对双引号
i"	双引号内部
a`	一对反单引号
i`	反单引号内部
at	一对 XML 标签
it	XML 标签内部

§ 8.8.2 范围文本对象

文本对象	选择范围
iw	当前单词
aw	当前单词及一个空格
iW	当前字串
aW	当前字串及一个空格
is	当前句子
as	当前句子及一个空格
ip	当前段落
ap	当前段落及一个空行
iW	当前字串及一个空格

第 9 章 磁盘高级管理

§ 9.1 磁盘配额 Quota

§ 9.1.1 简介

- 用途: 限制用户组和普通用户的磁盘使用量.
- 限制: 仅能针对整个文件系统, 内核需支持.

§ 9.1.2 使用流程

1. 文件系统支持. 开机挂载时, 对目标文件系统加入 Quota 支持. 修改 /etc/fstab 文件:

```
/sda5 /home ext4 defaults,usrquota,grpquota 1 2
```

2. 建立配置文件.

用法 : quotacheck [-avugfm] [挂载点]

-a : 扫描所在 /etc/mtab 内还有 Quota 支持的文件系统. 加上此参数可以不写挂载点.

-v : 显示扫描过程信息

-u : 针对用户扫描文件与目录的使用情况, 会新建 aquota.user

-g : 针对用户组扫描文件与目录的使用情况, 会新建 aquota.group

-f : 强制扫描文件系统, 并写入新的 Quota 配置文件 (危险).

-m : 强制以读写的方式扫描文件系统, 只有特殊情况下使用.

- 一般使用如下命令即可在目标目录下创建 aquota.group 和 aquota.user 两个配置文件:

```
quotacheck -avug
```

若确定无人使用 Quota 时, 可以强制操作:

```
quotacheck -avug -mf
```

3. 启动与关闭.

用法 : quotaon [-avug]

quotaon [-vug] [挂载点]

开启 Quota, 参数意义同前面.

用法 : quotaoff [-a]
 quotaoff [-ug] [挂载点]
 关闭 Quota, 参数意义同前面.

4. 设置限制信息.

用法 : edquota [-u 用户名] [-g 用户组名]
 edquota -t
 edquota -p 范本账号 -u 新账号

参数

-u : 进入 Quota 编辑界面设置针对该用户的限制值.
 -g : 进入 Quota 编辑界面设置针对该用户组的限制值.
 -t : 修改宽限时间, 默认是 7 天.
 -p : 将已设置账号的设置信息复制给新的账户.

针对账号进行限制 7 个字段的意义

- 文件系统: 表明针对哪一个文件系统进行设置.
- 磁盘容量 (blocks): Quota 自动计算, 单位为 KB, 不要修改.
- soft: 容量 soft 限制, 单位为 KB.
- hard: 容量 hard 限制, 单位为 KB.
- 文件数量 (inodes): Quota 自动计算, 单位为个数, 不要修改.
- soft: inode 的 soft 限制, 单位为 KB.
- hard: inode 的 hard 限制, 单位为 KB.

soft 和 hard 值为 0 时, 表示没有限制.

5. 限制报表.

用法 : quota [-uvs 用户名]
 quota [-gvs 用户组名]
 requota -a [-uvgs] ← 针对整个文件系统的限额报表

参数

-s : 使用 1024 为倍数来指定单位.

6. 其他. 直接设置命令如下:

```
setquota [-u|-g] 名称 block(soft) block(hard) \
inode(soft) inode(hard) 文件系统
```

§ 9.2 RAID

§ 9.2.1 简介

- RAID 0(等量模式,stripe): 性能最佳.
- RAID 1(映像模式,mirror): 完整备份.
- RAID 5: 需要三块以上磁盘, 可以允许损坏一块磁盘.
- RAID 6: 可以允许损坏两块磁盘.
- Spare Disk: 预备磁盘. 当有磁盘损坏时, 可加入该磁盘, 系统会自动重建数据系统.

§ 9.2.2 软件 RAID 设置流程

1. 建立*.

用法	: mdadm --detail /dev/md[0-9] mdadm --create --auto=yes /dev/md[0-9] \ --level=[015] --raid-devices=N --spare-devices=N \ /dev/sdx /dev/sdy ...
----	--

参数

--create	: 新建 RAID.
--auto=yes	: 新建后面接的软件磁盘阵列设备.
--level=[015]	: 磁盘阵列等级.
--raid-devices=N	: 使用几个磁盘作为磁盘阵列设备.
--spare-devices=N	: 使用几个备用设备.
--detail	: 列出磁盘阵列设备的详细信息. 也可以使用 cat /proc/mdstat 查看.

2. 救援.

用法	: mdadm --manage /dev/md[0-9] [--add 设备] \ [-remove 设备] [--fail 设备]
----	--

*Debian、Ubuntu 需要安装 mdadm

参数

- add : 将后面的设备加入该 md 中.
 - remove : 将后面的设备从该 md 中删除.
 - fail : 将后面的设备设置为出错状态.
-

3. 开机自动启动并挂载.

- 设置 /etc/mdadm.conf 文件*: ARRAY /dev/md0 UUID=...
- 修改 /etc/fstab 设置开机挂载.

4. 关闭. 直接关闭: mdadm --stop /dev/md0. 或者先卸载, 然后在 /etc/fstab 中信息删除.

§ 9.3 LVM

§ 9.3.1 基本概念

- PV: Physical Volum, 物理卷.
- VG: Volume Group, PV 组成的大磁盘.
- PE: Physical Extend, 物理扩展块, LVM 的单元, 默认为 4MB, 最多包含 65534 个, 故 LVM 最多只有 256G, 可通过修改 PE 大小改变 LVM 最大容量.
- LV: Logical Volume, 逻辑卷, 最终 VG 会被分成一个个的 LV.

§ 9.3.2 制作分区

制作物理分区, 并设置 systemID 为 8e.

§ 9.3.3 PV 阶段

-
- pvccreate : 将物理分区新建为 PV, 示范:
“pvccreate /dev/sda{6,7,8,9}”
 - pvscan : 查询当前系统里具有 PV 属性的磁盘.
 - pvdisplay : 显示当前系统 PV 状态.
 - pvremove : 将分区 PV 属性删除, 示范:
“pvremove /dev/sda6”
-

*查询 UUID: 1. blkid; 2. ls -l /dev/disk/by-uuid; 3. mdadm --detail

§ 9.3.4 VG 阶段

`vgcreate` : 新建 VG, 示范:
“`vgcreate [-s PE大小 [mgt]] VG名称 PV名称`”

`vgscan` : 查询当前系统里的 VG.

`vgdisplay` : 显示当前系统 VG 状态.

`vgextend` : 在 VG 内增加额外的 PV, 示范:
“`vgextend VG名称 磁盘分区`”

`vgreduce` : 在 VG 内删除 PV, 示范:
“`vgreduce VG名称 磁盘分区`”

`vgchange` : 设置 VG 是否启动 (active), 示范:
“`vgchange VG名称`”

`vgremove` : 删除一个 VG, 示范:
“`vgremove VG名称`”

§ 9.3.5 LV 阶段

`lvcreate` : 新建 LV. 示范:
“`lvcreate [-L N[mgt]] [-n LV名称] VG名称`”
“`lvcreate [-l PE个数] [-n LV名称] VG名称`”

`lvscan` : 查询当前系统里的 LV.

`lvdisplay` : 显示当前系统 LV 状态.

`lvextend` : 在 LV 内增加容量.

`lvreduce` : 在 LV 内减少容量

`lvremove` : 删除一个 LV, 示范:
“`lvremove LV设备名`”

`lvresize` : 对 LV 进行容量大小调整.

§ 9.3.6 文件系统阶段

格式化建立的 LV 设备并挂载使用.

§ 9.3.7 完整示例

1. 制作分区. 假设得到 4 个分区: `/dev/sda{6,7,8,9}`, 并将 `systemID` 设置为 8e (注: 不设置也没关系, 只是某些 LVM 检测命令可能检测不到该分区)[†], 每个分区 1G.

2. VG 阶段.

```
vgcreate -s 16M yongvg /dev/sda{6,7,8}
```

3. LV 阶段. 查看 VG 状态可知有 189 个 PE.

```
lvcreate -l 189 -n yonglv yongvg
```

4. 文件系统阶段.

```
mkfs.ext4 /dev/yongvg/yonglv
mkdir /mnt/lvm
mount /dev/yongvg/yonglv /mnt/lvm
```

5. 扩容.

```
# 将 PV 分区加入 VG
vgextend yongvg /dev/sda9

# 查看可知 VG 多了 63 个 FreePE
vgdisplay
...
      Total PE          315
      Alloc PE / Size    252 / 3.94 GiB
      Free  PE / Size    63 / 1008.00 MiB
...
# 将这些 FreePE 加入 LV
lvresize -l +63 /dev/yongvg/yonglv

# 文件系统扩充
resize2fs /dev/yongvg/yonglv
```

6. 缩减.

```
# 可以发现每个 PV 为 1008M, 欲抽离一个 PV, 剩余 4032M
pvscan
      PV /dev/sda6    VG yongvg    lvm2 [1008.00 MiB / 0      free]
      PV /dev/sda7    VG yongvg    lvm2 [1008.00 MiB / 0      free]
```

[†]本人开始忘记设置, 在最终制作好 LVM 以后, 再直接将 `systemID` 改为 8e, 导致重启系统遇到错误, Debian 经过一段时间的自我排错完成了开机, 真是惊喜.

```

PV /dev/sda8   VG yongvg   lvm2 [1008.00 MiB / 0   free]
PV /dev/sda9   VG yongvg   lvm2 [1008.00 MiB / 0   free]

#卸载，增加可以在线处理，减小不能
umount /mnt/lvm

#磁盘检查，缩减前的必须步骤
e2fsck -f /dev/yongvg/yonglv

#调整到目标容量
resize2fs /dev/yongvg/yonglv 4032M

#挂载使用
mount /dev/yongvg/yonglv /mnt/lvm

#降低LV的PE数量
lvresize -l -63 /dev/yongvg/yonglv

#查看PV情况，看看FreePE所在位置
pvdisplay

#假设欲抽离 /dev/sda9，而FreePE在 /dev/sda8 中
#此时需要转移PE
pvmove /dev/sda9 /dev/sda8

#再次查看可发现FreePE在 /dev/sda9 中
pvdisplay

#从VG中移出
vgremove /dev/sda9

```

§ 9.3.8 制作快照

```

#-s 表示 snapshot 快照功能之意
lvcreate -l 60 -s -n yongss /dev/yongvg/yonglv

```

第 10 章 例行性工作 crontab

§ 10.1 单一工作调度

§ 10.1.1 atd 服务

- 启动

```
/etc/init.d/atd restart
chkconfig atd on <== 开机启动
```

- at 运行方式

- 1). /etc/at.allow 文件若存在, 则只有写在该文件中*的用户才能使用 at;
- 2). 若上述文件不存在, 则寻找 /etc/at.deny, 若该文件存在, 则写在该文件中的用户不能使用 at, 其余用户可以使用;
- 3). 若上述两个文件都不存在, 则只有 root 可以使用.

§ 10.1.2 相关命令

1. at: 单一调度.

用法	: at [-mldv] TIME at -c 工作号码
-m	: 工作完成以后, 即使没有输出信息, 以 email 通知用户该工作已完成
-l	: 相当于 atq, 列出目前系统上面的所有该用户的 at 调度
-d	: 相当于 atrm, 取消一个在 at 调度中的工作
-v	: 使用较明显的时间格式列出调度任务列表
-c	: 列出后面接的该项工作的实际命令内容

TIME 格式:

HH:MM : 今天的 HH:MM 时刻进行, 若超过了则明天进行

HH:MM YYYY-MM-DD : 指定详细的执行时间

HH:MM [am|pm] [Month] [Date]: 指定在某月某日详细时间执行

HH:MM [am|pm] + number [minutes|hours|days|weeks]: 在指定时间点之后再加指定时间数才进行. 常用例如 “now + 5 minutes” .

*一个账号写一行

2. `atq`: 查询主机上所有的 `at` 工作调度.
3. `atrm [jobnumber]`: 取消指定工作号的调度.
4. `batch`: 系统空闲时才进行后台任务. 本质是利用 `at` 来调度, 用法同 `at`.

§ 10.2 循环调度

1. `crontab` 的设置限制文件是 `/etc/cron.allow` 和 `/etc/cron.deny`, 用法同 `at`.

用法: `crontab [-u username] [-l|-e|-r]`

`-u` : root 帮助其他用户管理 `crontab` 调度

`-e` : 编辑 `crontab` 的工作内容

`-l` : 查阅 `crontab` 的工作内容

`-r` : 删除所有 `crontab` 的工作内容, 单项删除用 `-e` 编辑

编辑界面的 7 个字段意义如下*,

代表意义	分钟	小时	日期	月份	周	命令
数字范围	0~59	0~23	1~31	1~12	0~7	命令内容

辅助字符代表意义如下,

* : 任何时刻都接受

, : 分隔时段. 例如, 第 1 个字段 “1,3” 表示第 1 分钟和第 3 分钟.

- : 代表一段时间范围内. 例如, 第 1 个字段 “1-3” 表示第 1 分钟和第 3 分钟.

/n : 每隔 n 个单位时间的意思. 例如, 第 1 个字段 “*/5” 表示每隔 5 分钟.

2. 上述命令针对用户, 而系统的工作内容实际上在 `/etc/crontab` 中, `/usr/bin/crontab` 每分钟读取该文件与 `/etc/spool/cron` 的数据内容.
3. 注意事项: 大量使用造成资源分配不均; 取消不必要输出; 注意检查 `/var/log/cron`[†]; 周与日、月不可并存.

§ 10.3 唤醒停机期间的工作任务

`anacron` 可以开机运行或写入 `crontab` 的调度中, 从而分析系统未进行的 `crontab` 工作.

*需要注意区域时间的设置.

[†]Debian 中, `/etc/rsyslog.conf` 文件中, 默认注释了 `cron` 的日志输出, 编辑该文件取消注释, 重启 `rsyslog` 服务, 即可启用 `cron` 日志.

用法: `anacrontab [-sfn] [job]..`

`anacrontab -u [job]..`

`-s` : 连续执行各项工作, 依据时间记录文件判断是否执行

`-f` : 强制执行, 而不判断时间记录文件的时间戳

`-n` : 立刻进行未进行的任务, 而不延迟

`-u` : 仅更新时间记录文件的时间戳, 不进行任何工作

`job` : 由 `/etc/anacrontab` 定义的各项任务名称

确定 `anacron` 是否会开机启动, 执行以下命令:

```
chkconfig --list anacron
```

注意第 3 和第 5 项是否开启.

第 11 章 程序管理

§ 11.1 工作管理

- **&**: 加在命令后表示直接将命令放入后台执行.
- **[Ctrl]+z**: 将当前的工作放入后台暂停.
- **jobs**: 查看当前后台工作状态.
- **fg %jobnumber**: 将后台工作放到前台来处理.
- **bg %jobnumber**: 让后台工作的状态变成运行中.
- **kill**: 管理后台的工作.

~ **kill -signal %jobnumber**
-1: 重新读取一次参数配置文件.
-2: 与键盘 [Ctrl]+c 操作相同.
-9: 立刻强制删除一个工作.
-15: 以正常的程序方式终止一项工作.

~ **kill -l**: 查看有哪些 signal 可以使用.

- **nohup**: 脱机管理. 需要注意的是不支持 bash 内置命令.

~ **nohup [命令与参数]**: 终端前台.
~ **nohup [命令与参数] &**: 终端后台.

§ 11.2 进程管理

1. **ps**: 查看进程运行情况.

用法 : **ps aux** ← 查看系统所有的进程的状态

ps -lA

ps axjf ← 连同部分进程树状态

-A : 所有进程状态均显示出来, 同 **-e**

-a : 与 terminal 无关的所有进程

-u : 有效用户相关的进程

x : 列出较完整的信息, 通常与 **a** 这个参数一起用

l : 较长、较详细地将该 PID 的信息列出

j : 工作的格式

-f : 做一个更为完整的输出

2. **top**: 动态查看进程的变化.

用法 : `top [-d 数字]`

`top [-bnp]`

`-d` : 后面接秒数, 默认为 5 秒, 表示整个界面更新的间隔.

`-b` : 以批次的方式执行, 通常搭配数据流将结果输出到文件.

`-n` : 与 `-b` 搭配, 表示需要进行几次 `top` 的输出结果.

`-p` : 指定某些 PID 来监控.

执行中的参数:

`?` : 显示可以输入的按键命令.

`P` : 以 CPU 的使用资源排序.

`M` : 以内存的使用资源排序.

`N` : 以 PID 排序.

`T` : 由该进程使用的 CPU 时间累积排序.

`k` : 给予某个 PID 一个信号.

`r` : 给予某个 PID 重新制定一个 nice 值.

`q` : 离开 `top` 软件的按键.

3. **pstree**: 进程树.

用法 : `pstree [-A|U] [up]`

`-A` : 各进程树之间以 ASCII 字符来连接.

`-U` : 各进程树之间以 utf8 字符来连接.

`-p` : 同时列出每个进程的 PID.

`-u` : 同时列出每个进程的所属账号名称.

4. **kill**: 管理进程.

- `kill -signal PID`: 可用信号如下,

代号	名称	内容
1	SIGHUP	重新读取一次参数配置文件.
2	SIGINT	与键盘 [Ctrl]+c 操作相同.
9	SIGKILL	立刻强制删除一个工作.
15	SIGTERM	以正常的程序方式终止一项工作.
17	SIGSTOP	与键盘 [Ctrl]+z 来暂停进程操作相同.

5. **killall**: 通过命令管理进程.

用法 : killall [-iIe] 命令名称

-i : 交互操作.

-I : 忽略大小写.

-e : exact, 完整命令相同.

6. **nice**: 新执行的命令给予新的 nice 值.

用法 : nice [-n 数字] 命令名称

-n : 后接数字, 范围 -20~19.

7. **renice**: 重新调整已存在进程的 nice 值, “renice [数字] PID”.

§ 11.3 系统资源的查看

1. **free**: 查看内存使用情况.

用法 : free [-b|-k|-m|-g] [-t]

-b,-k,-m,-g : 单位.

-t : 在输出结果中显示物理内存与 swap 总量.

2. **uname**: 查看系统内核相关信息.

用法 : uname [-asrmpi]

-a : 所有系统的相关信息.

-s : 系统内核名称.

-r : 内核的版本.

-m : 本系统的硬件名称, 如 x86_64.

-p : CPU 的类型.

-i : 硬件的平台.

3. **netstat**: 跟踪网络.

用法 : netstat -[atunlp]

-a : 将当前系统上的所有连接、监听、Socket 数据都列出.

-t : 列出 tcp 网络数据包的数据.

-u : 列出 udp 网络数据包的数据.

-n : 不列出进程的服务名称, 以端口号来显示.

-l : 列出当前正在网络监听的服务.

-p : 列出该网络服务的 PID.

4. **dmesg**: 分析内核产生的信息.

5. **vmstat**: 检测系统资源变化.

用法 : **vmstat** [-a] [延迟 [总计检测次数]] \Leftarrow CPU/内存等信息

vmstat [-fs] \Leftarrow 内存相关

vmstat [-S 单位] \Leftarrow 设置显示数据的单位

vmstat [d]

vmstat [-p 分区]

-a : 使用 inactive/active 替代 buffer/cache 的内存.

-f : 开机到目前为止系统复制的进程数.

-s : 将一些事件导致的内存变化情况列表说明.

-S : 后面可以接单位, 让显示数据有单位.

-d : 列出磁盘的读写总量统计表.

-p : 后面列出分区, 可显示该分区的读写总量统计表.

6. **fuser**: 通过文件 (或文件系统) 找出正在使用该文件的程序.

用法 : **fuser** [-umv] [-k [i] [-signal]] file/dir

-u : 除了进程 PID 之外, 同时列出该进程的所有者.

-m : 后面接的文件名会主动上提到该文件系统的顶层.

-v : 可以列出每个文件与程序还有命令的完整相关性.

-k : 找出使用该文件/目录的 PID, 并试图以 SIGKILL 这个信号给予该 PID.

-i : 必须与 -k 配合, 交互模式.

-signal : 默认为 -9.

6. **lsof**: 列出被进程所打开的文件名.

用法 : **lsof** [-aUu] [+d]

-a : 多项数据需要“同时成立”才显示出结果.

-U : 列出 Unix like 系统的 socket 文件类型.

-u : 后面接 username, 列出该用户相关进程所打开的文件.

+d : 后面接目录, 即找出某个目录下面已经被打开的文件.

7. **pidof**: 找出某个正在执行的进程的 PID.

用法 : pidof [-sx] 进程名

-s : 仅列出一个 PID 而不列出所有的 PID.

-x : 重复命中, 返回指定程序的 shell 脚本的 pid.

第 12 章 服务

1. `service`

用法 : `service [服务名] (start|stop|restart|...)`

`service --status-all`

2. `chkconfig`: 管理系统服务默认开机与否.

用法 : `chkconfig --list [服务名称]`

`chkconfig [--level [0123456]] [服务名称] [on|off]`

`--list` : 仅将目前的各项服务状态显示出来.

`--level` : 设置某个服务在该 level 下启动 (on) 或关闭 (off).

第 13 章 虚拟机安装 Debian

§ 13.1 分区设置

安装过程按照提示步骤完成即可, 值得注意的是分区的大小,

/boot	100M	启动分区, 放在分区最前面
/	2G ~ 6G	根分区
/usr	10G ~ 20G	用来安装各类软件, 可能需要一些容量
/var	3G ~ 8G	
/swap	500M ~ 2G	交换分区, 根据需求适当设置值
/home	10G ~ ...	可以使用 LVM, 方便以后增加容量
/temp		临时分区, 看个人需要设置

另外注意留几个 G 的未挂载分区, 方便以后有额外空间作为其他用途.

§ 13.2 基本工具配置

§ 13.2.1 更新源配置

1. 安装完成以后, 继续挂载安装光盘, 以便安装必须的软件. 此时, 可以查看/etc/fstab 文件, 可以发现安装光盘是开机挂载的, 查看/etc/apt/sources.list 也可以知道, 更新源中包含了该光盘.

2. 也可以在安装过程中, 根据提示配置网络更新源. 或者在安装完成以后, 手动修改/etc/apt/sources.list 文件配置, 然后运行 “`apt-get update`” 更新软件列表即可.

§ 13.2.2 安装虚拟机增强工具

1. 为了安装虚拟机提供的增强工具, 需要基本的开发工具包, 包括 make, gcc 等, 运行 “`apt-get install build-essential`” .

2. 增强工具可以使得主机和虚拟机之间更好的共享文件等. 按照虚拟机软件的帮助提示操作即可. 安装过程中, 可能遇到错误, 需要安装 linux-headers, “`apt-get install linux-headers-`uname -r``”.

§ 13.2.3 建立共享文件夹

1. 安装好了增强工具以后, 就可以建立主机和虚拟机之间的共享文件夹了. 例如 VMware, 查询帮助知, 在菜单中创建了共享文件夹将 E:\win_linux_share 以共享文

件名 win_linux_share 共享给虚拟机. 修改 /etc/fstab 文件将该文件夹设置为开机挂载

```
.host:/ /mnt/win_share vmhgfs defaults 0 0
```

此时, 虚拟机中即可找到 /mnt/win_share/win_linux_share 这个主机共享的文件夹.

2. 通常我将下载好的几张 Debian 的 DVD 放在该共享文件夹, 进一步将这几张光盘设置为开机挂载, 同时配置更新源为这几张 DVD, 这样可以不使用网络本地安装很多需要的软件. 修改 /etc/fstab, 同时注释 cdrom 的开机挂载,

```
/mnt/win_share/win_linux_share/xxxDVD1.iso /mnt/mirror/dvd1 \
iso9660 defaults,ro,loop 0 0
/mnt/win_share/win_linux_share/xxxDVD2.iso /mnt/mirror/dvd2 \
iso9660 defaults,ro,loop 0 0
/mnt/win_share/win_linux_share/xxxDVD3.iso /mnt/mirror/dvd3 \
iso9660 defaults,ro,loop 0 0
```

修改/etc/apt/sources.list, 同时注释 cdrom 的源,

```
deb file:/mnt/mirror/dvd1 wheezy contrib main
deb file:/mnt/mirror/dvd2 wheezy contrib main
deb file:/mnt/mirror/dvd3 wheezy contrib main
```

这里 wheezy 是 Debian7 系列的版本号. 不要忘了 “apt-get update”.

§ 13.2.4 安装中文输入法

若 Ctrl+Space 不能呼出中文输入法, 可能是默认的安装不完整. 打开新立得软件管理, 搜索 fcitx 检查安装情况, 可以从网上查询哪些部分需要安装, 然后一一对比自己的安装情况. 例如本人未安装 fcitx-ui-classic, fcitx-m17n 等包导致不能正常使用.

§ 13.3 KDE 桌面环境

新立得安装 kde-full 即可, 需要注意的是, KDE 中文环境不在该软件包的关联中. 另外搜索 kde, 并使用 zh 过滤找到相应的简体中文包安装即可. KDE 对应的服务为 kdm, GNOME 对应的服务为 gdm, Ubuntu 的 Unity 对应的服务为 lightdm.

第 14 章 Samba 服务器

实现 Linux 和 Windows 之间的文件共享. 快速实现方法见《Ubuntu Server 最佳方案》: 安装, 配置 smb.conf, 重启服务, Windows 映射驱动器. 需要注意之处是:

- 用户认证后台管理的三种方式.
- 控制用户和用户组的访问.

常用参数说明见“有道笔记”. 关于 Samba 原理等细节见《Linux 私房菜——服务器》. 更多细节见官方文档“HOW-TO”以及“ByExample”.

第 15 章 NFS 服务器

Unix Like 系统之间文件共享, 比 Samba 更快且方便. 一般和 NIS 服务器一起使用, 快速实现方法见《Ubuntu Server 最佳方案》: 安装, 配置/etc(exports, 设置共享文件夹的权限. 需要注意的是:

- NFS 文件访问权限, 分为几种情况.
- /etc(exports 的配置.

关于 NFS 原理讲解见《Linux 私房菜——服务器》. 关于/etc(exports 详细配置参数见“有道笔记”.

第 16 章 NIS 服务器

NIS 主要提供用户账号、密码、用户主目录文件名、UID 等信息, 但不提供文件系统. 重点:

- NIS Server 和 NIS Client 的运作流程.
- 配置文件.

关于 NIS 讲解见《Linux 私房菜——服务器》. 配置过程见“有道笔记”.

第 17 章 NIS,NFS 集群

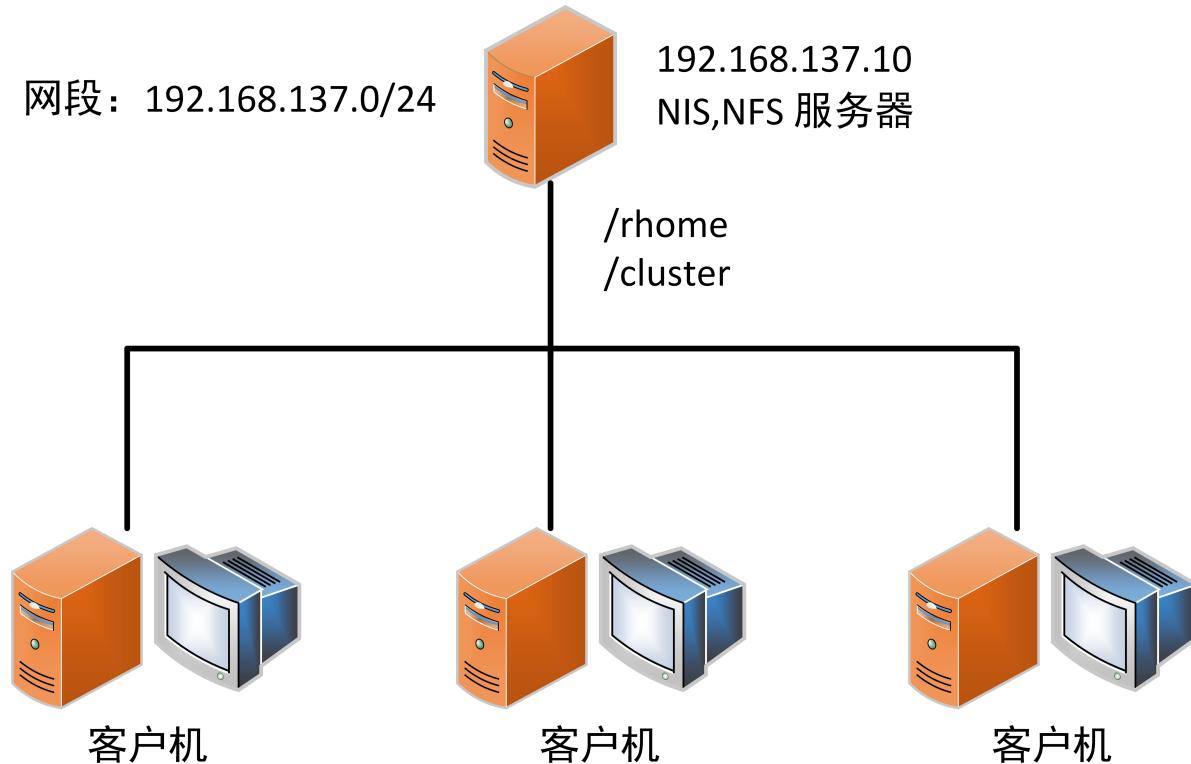


图 17-2 NIS,NFS 集群系统

建立 UID 大于 2000 的账号, 账号名为 cluser1,cluser2,cluser3, 主目录在/rhome 下, 新建组 clusergroup, 将新建的用户加入该组, 该组共享文件夹为/cluster. 目标: 该组人员在任何一台客户机上都能进行自己的工作.

§ 17.1 配置

§ 17.1.1 服务器

安装:

```
$: sudo apt-get install nfs-kernel-server nfs-common rpcbind nis -y
```

1. NIS

```
$: sudo vim /etc/hosts.deny
```

增加一行: “portmap: ALL”.

```
$: sudo vim /etc/hosts.allow
```

增加一行: “portmap: 192.168.137.”.

查看本机的主机名并添加到 hosts 中.

```
$: hostname
```

或

```
$: cat /etc/hostname
```

```
$: sudo vim /etc/hosts
```

增加一行: “192.168.137.10 主机名”.

```
$: sudo service rpcbind restart
```

```
$: sudo vim /etc/default/nis
```

修改两处: “NISSERVER=master” 和 “NISCLIENT=false”.

```
$: sudo vim /etc/yp.conf
```

添加服务器信息: “domain domainName server 主机名”.

domainName 就是安装 nis 时输入的信息, 可以直接在 /etc/default/domain 中修改, 或者运行以下命令修改:

```
$: sudo dpkg-reconfigure nis
```

```
$: sudo vim /etc/ypserv.securenets
```

添加客户端所在网段: “255.255.255.0 192.168.137.0”.

```
$: sudo vim /var/yp/Makefile
```

修改一行, 添加斜体字段: “ALL = passwd *shadow* group ...”.

```
$: sudo service ypserv start
```

```
$: sudo /usr/lib/yp/ypinit -m
```

```
$: sudo service ypserv restart
```

2. NFS

```
$: sudo vim /etc/idmapd.conf
```

修改一行: “Domain = domainName”.

```
$: sudo vim /etc/default/nfs-common
```

修改一行: “NEED_IDMAPD = YES”.

```
$: sudo service nfs-kernel-server restart
```

```
$: sudo service rpcbind restart
```

§ 17.1.2 客户端

安装:

```
$: sudo apt-get install nfs-common rpcbind nis -y
```

1. NIS

\$: sudo vim /etc/yp.conf

加入一行: “domain *domainName* server 服务器主机名”. 这里的 *domainName* 和服务器设置相同.

\$: sudo vim /etc/nsswitch.conf

修改几行, 添加斜体字段:

“passwd: compat *nis*”

“group: compat *nis*”

“shadow: compat *nis*”

“hosts: files dns *nis*”

2. NFS

\$: sudo vim /etc/hosts.deny

增加一行: “portmap: ALL”.

\$: sudo vim /etc/hosts.allow

增加一行: “portmap: 192.168.137.10”.

\$: sudo vim /etc/hosts

增加一行: “192.168.137.10 主机名”.

\$: sudo vim /etc/idmapd.conf

修改一行: “Domain = *domainName*”.

\$: sudo service idmapd restart

\$: sudo reboot

§ 17.2 搭建

§ 17.2.1 服务器

新建用户并指定主文件夹:

\$: sudo mkdir /rhome

\$: sudo useradd -u 3001 -d /rhome/cluser1 -m cluser1

\$: sudo useradd -u 3002 -d /rhome/cluser2 -m cluser2

\$: sudo useradd -u 3003 -d /rhome/cluser3 -m cluser3

添加到同一个组:

\$: sudo groupadd clusergroup

```
$: sudo usermod -a -G clusergroup cluser1
$: sudo usermod -a -G clusergroup cluser2
$: sudo usermod -a -G clusergroup cluser3
```

设置该组的公共文件夹, 这里要注意的是设置 SGID 权限:

```
$: sudo mkdir /cluster
$: sudo chgrp clusergroup /cluster
$: sudo chmod 2770 /cluster
```

为新建的用户设置密码, 并重建 YP 数据库:

```
$: sudo make -C /var/yp
```

通过 NFS 共享到客户端:

```
$: sudo vim /etc/exports
```

添加两行:

```
"/rhome 192.168.137.0/24(rw,no_root_squash)"
"/cluster 192.168.137.0/24(rw,no_root_squash)"
```

重启 NFS 或者重新挂载 exports:

```
$: sudo vim exportfs -arv
```

验证是否挂载:

```
$: showmount -e localhost
```

§ 17.2.2 客户端

查看 NFS 服务器共享情况:

```
$: sudo showmount -e 192.168.137.10
```

设置开机挂载:

```
$: sudo mkdir /rhome /cluster
$: sudo vim /etc/fstab
```

添加两行:

```
"192.168.137.10:/rhome /rhome nfs defaults 0 0"
"192.168.137.10:/cluster /cluster nfs defaults 0 0"
```



```
$: sudo reboot
```

接下来, 可在客户机上用 cluser1 等登陆.