# linux的文件系统

在linux中，一切皆文件

linux的目录以根目录为根，不停地向下扩展。而windows的目录结构则是森林结构。另外，linux的目录是逻辑目录，可以在目录树中的任意位置挂载硬盘分区或者硬件设备，也就是说根目录之下的所有文件以及文件夹并非一定要与根目录所在的硬盘分区保持一致，只是一个逻辑上的目录结构，在物理存储上并没有限制。这一点和windows是完全不同的。

## 挂载点

挂载点可以在根目录之下任意地选择，挂载的目的是为了将逻辑目录与对应的硬件设备（硬盘分区）联系到一起。该目录下的所有内容都会写到该硬盘分区之中。

一般是选取新建目录作为挂载点，如果选取一个已有的文件夹进行挂载，那么该文件夹下原先的内容（挂载之前的内容）将被隐藏，直到对该目录解除挂载。

## 根目录下各个文件夹的作用

bin：软链接，指向usr/bin, 存放的是一些命令（非root命令）（可以创建文件夹的软链接，但不允许创建文件夹的硬链接）

sbin：软链接，指向usr/sbin, 存放的是一些系统级命令(root命令)

lib：库目录（想想C语言的动态库）

lib64：64位库目录

usr：存放用户的所有应用程序和所需要的数据，

boot：存放引导分区中的文件

dev：device，存放所有硬件设备的文件映射。

sys：存放系统硬件信息的相关文件

etc：存放配置文件（比如配置软件的一些参数）

home：其下的每个目录就是每个用户的主目录

root：root用户的主目录

opt：可选目录，给第三方软件包留下的位置，推荐将第三方软件安装在此

media和mnt：两者都是移动设备的挂载目录。

proc：进程目录，存放现有进程的一些映射，是系统的重要目录，不推荐用户随意更改。

run：存放系统开机运行以来的一些临时信息。重启之后就失效。用户删除没有大碍，但是最好不要更改，因为记录了系统当前的一些信息。

srv：系统服务目录

tmp：临时目录，临时存放的东西，可以随意地删除。

var：可变目录，存放经常变化的文件，比如日志。

# vmware虚拟机的三种网络连接方式

内网、外网和网关的概念，就不用解释了

1.桥接

主机提供一个网桥，虚拟机通过这个网桥能与主机所对应的那个路由器连接，于是该路由器通过DHCP协议给虚拟机分配一个动态ip，虚拟机与该局域网内的其它主机是对等的关系，能够互相ping通，另外虚拟机也能够同外网交换信息，外网能够知道虚拟机的存在

2.NAT

主机虚拟出一个路由器，该网关向各个虚拟机分配动态ip地址以及做网络地址转化，换句话说，虚拟机的网关的网关，才是主机的网关，因此，外网无法知道虚拟机的存在，虚拟机也不占用实际网关的动态ip分配名额，但是虚拟机仍然能够访问外网，并且虚拟机之间也能相互访问。

仅凭次，主机还不能和虚拟机进行通讯，因为主机和虚拟机不再同一个网段，为了解决这个问题，主机虚拟出一张网卡（VM\_Net8），这张网卡连接的是主机虚拟出来的路由器，这样一来主机既在它自己原本的网段上，又在虚拟机所在的网段上，于是主机和虚拟机之间就能够相互之间进行访问。

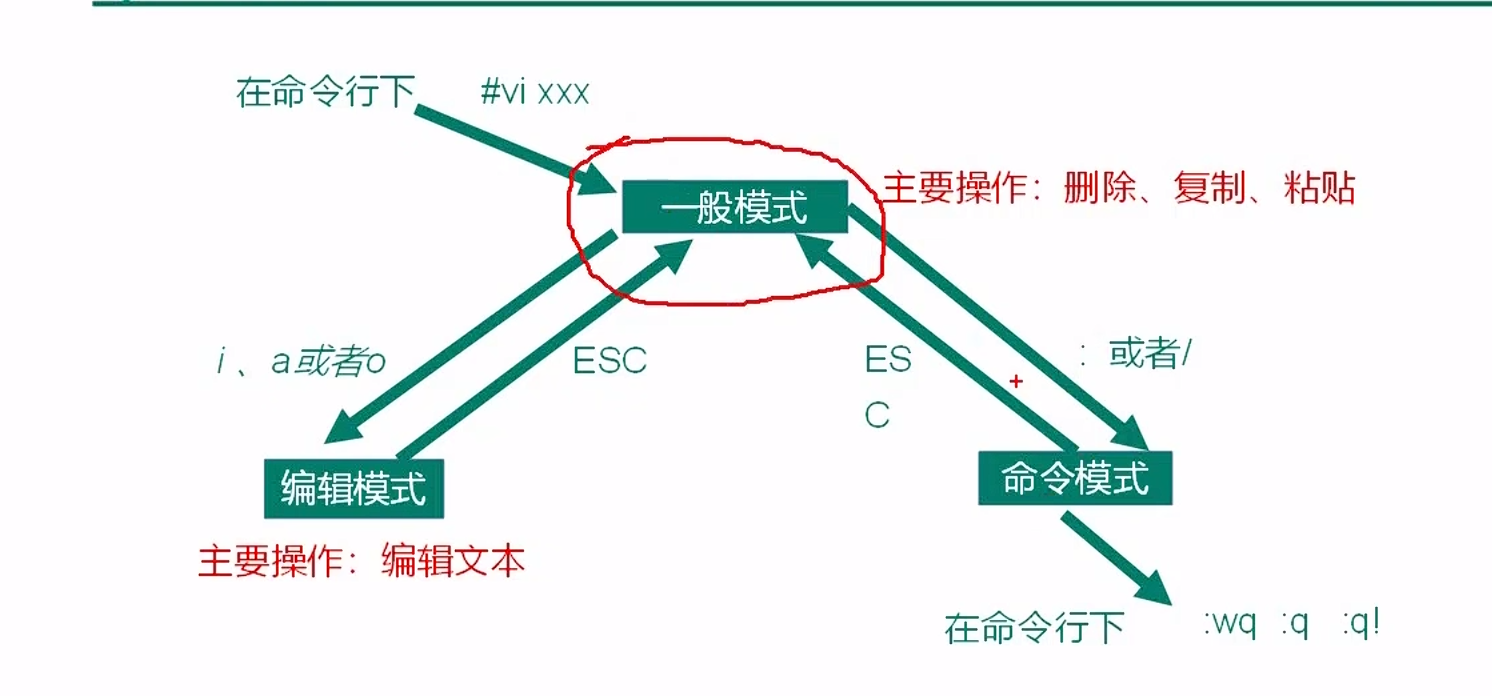
3.主机直连

主机虚拟出一个交换机，但是没有路由功能，因此虚拟机无法访问外网，但是虚拟机之间能够相互访问。

类似地，主机虚拟出一张网卡(VM\_NET1),这张网卡连接的是主机虚拟出来的交换机，这样一来主机既在它自己原本的网段上，又在虚拟机所在的网段上，于是主机和虚拟机之间就能够相互之间进行访问。

# vim的用法

vim的三种模式



普通模式

1.复制

复制一行：yy

复制多行：y5y, 5yy

复制当前位置到行尾的所有内容：y$

复制当前位置到行首的所有内容：y^

2.粘贴

粘贴一次：p

粘贴多次：5p

3.删除

删除一行:dd

删除多行:d5d, 5dd

删除当前位置到行尾的所有内容:d$

删除当前位置到行首的所有内容:d^

4.回滚：u

5.移动

移动到下一个单词的词头:w

移动到上一个单词的词头:b

移动到当前单词的词尾:e

移动到行首:^

移动到行尾:$

移动到文档开头:gg

移动到文档结尾：G

编辑模式

一般是按i进入插入模式

其它用的比较频繁的是

o:在光标位置的下面插入一行，并移动到行首

O: 在光标位置的上面插入一行，并移动到行首

命令模式

退出: :q；:w；:wq；:q!；:wq!

查找： /要查找的词 n查找下一个，N查找上一个

取消高亮显示：:noh

设置或者取消行号：set nu，set nonu

替换：

1.替换当前行 :s/old/new

2.替换文档中的每个匹配到的old :%s/old/new/g

3.替换文档中的每个匹配到的old，并且进行询问 :%s/old/new/gc

4.:2,$s/vivian/sky/g 替换第 2 行开始到最后一行中每一行所有 vivian 为 sky

5. :2,$s/vivian/sky/g 替换第 2 行开始到最后一行中每一行所有 vivian 为 sky, 并且进行询问

wq!是强制执行，比如通过vim打开该文件的用户是该文件的所属用户，但是该文件有没有写权限。

那么wq！会先把该文件的写权限（因为当前用户就是该文件的所属用户，他可以更改访问权限，相当于执行chmod u + x 某某文件）打开，然后再将更改写入该文件（从内存缓冲区持久化到磁盘）。做完以后再重新将写权限恢复为关闭状态。

# 系统管理类

## systemctl命令

### 服务和守护进程

常驻在内存中，一直运行着的进程，不受任何终端控制的，在后台运行着的进程，叫做守护进程(daemon process)，也叫做服务

服务常以d作为结尾，对应了单词daemon

以防火墙为例，

启动服务：systemctl start firewalld

终止服务：systemctl stop firewalld

重启服务：systemctl restart firewalld

查看服务状态：systemctl status firewalld

查看操作系统有哪些服务：ls /usr/lib/systemd/system/

打开开机自启:systemctl enable firewalld

关闭开机自启：systemctl disable firewalld

## linux的七种运行级别

级别0：系统停机状态

级别1：单用户工作状态，root权限

级别2.多用户状态，不支持网络，无gui

级别3：级别2拥有的基础之上，支持网络

级别4：未使用的级别，被保留

级别5：在级别3的基础之上，有gui

级别6：系统正常关闭并重启

## 关机重启命令

shutdown 关机命令，空参默认一分钟以后关机（系统会自动执行sync，事先将内存中的数据写入硬盘）

shutdown n n分钟后关机

shutdown hh:mm 规定在某个时刻关机

shutdown -h 关机但是不断电，以致于内存中的数据还可以保存下来

shutdown -r 重启

# 帮助命令

## man命令

man xxx

查看xxx命令的相关信息

## help 命令

help xxx

查看内嵌命令xxx的相关信息

## type命令

### 内嵌命令与非内嵌命令

内嵌命令是常驻在内存中的，而非内嵌命令才有存储的位置

比如ls是非内嵌命令，在/bin/目录下能够找到，而cd是内嵌命令，没有存储位置

type xxx

查看xxx命令的类型，比如是内嵌命令还是非内嵌。

## history

history

查看所有历史命令

## 常用快捷键

### 清屏命令

1.ctrl + l 或者输入clear，可以清屏

2.reset命令， 不仅终端内容清除掉了，并且历史记录也删去了

### ctrl+c

终止进程，等效于传递信号SIGINT给当前终端所运行的进程

### 键盘上下键

翻动之前或者之后的命令

### TAB键

命令补全

### 移动终端光标

ctrl + a：移动到终端正在输入内容的头部

ctrl + e：移动到终端正在输入内容的头部

ctrl + 右键：移动到上一个单词

ctrl + 左键：移动到上一个单词

# 文件目录类

## pwd显示当前目录

查看当前目录，显示的是绝对路径

## cd进入目录

进入某个目录，后跟相对路径或者绝对路径

cd ~或者cd，进入当前用户的主目录

另一种常见的用法：cd -，跳转到上一次所在的目录(好用！)

## ls列出文件

ls [] xxx

列出xxx目录中的所有文件

-a 列出所有文件，包括隐藏文件（隐藏文件以.开头）

-l详细列出各文件的信息，每个文件成一行

-h搭配文件使用，将大小部分改成kb，mb等易于识别的单位(不加-h，则显示以B为单位的详细数字)

-i 查看文件的inode

## mkdir创建目录

创建目录命令

-p 如果父目录不存在，则一并创建

递归创建

mkdir -p /temp1/temp2

temp1和temp2一并目录被创建

多次创建 mkdir

mkdir temp3 temp4

使用了相对路径，创建了temp3和temp4两个目录

## rmdir删除空目录

删除目录命令，只能删除空目录

若需删除非空目录，应使用 rm -rf

## touch创建空文件

创建空文件

## cp复制

cp [选项] source dest

复制source到dest去

source和dest可以是文件或者目录

-r：递归复制，用于source是非空文件夹的情况

## rm删除

删除指令

rm [选项] target

target是目录或者文件

-r 递归删除，用于非空文件夹

-f 删除时不需要询问

-v 删除时显示命令的详细执行过程

mv

移动或者重命名指令

mv source target

## 查看文件

### cat查看文件

cat [] xxx

将文件的所有内容，输出到控制台

不做分页，一次性输出，因此一般适用于文本内容少的文件

cat xxx | grep yyy

查看xxx文件中含有关键字yyy的每一行内容

-n 显示行号

### more分屏查看文件

基于vi的分屏查看器

f 向下翻页

b 向上翻页

### less分屏查看大文件

也是分屏查看器

比more更加强大，推荐使用这个。它不是一次性加载文件内容，而是根据需要动态加载，适合于大文本文件的浏览。

f 向下翻页

b 向上翻页

gg 移动到最开头

G 移动到最结尾

/xxx, 搜索xxx，按n向下继续搜索，按N向上继续搜索

## echo输出内容

echo [] xxx

输出内容xxx到控制台

-e 表示支持转义字符

1.输出一般文本：echo ”hello world”

2.输出支持转移字符的文本：echo -e ”hello\tworld\n”

3.输出变量名所对应的值：echo $PATH

## 重定向

### >覆盖

输出重定向，首先清除掉原来的内容，然后正式开始输出

### >> 追加

输出重定向，追加输出

echo hello >> text.txt, 将文本内容hello添加到text.txt中

## head查看文件头部

查看文件头部的前几行，默认前10行

head -n xxx,查看xxx文件的前n行

## tail查看文件尾部

查看文件尾部的最后几行，默认后10行

tail -n xxx,查看xxx文件的后n行

tail -f xxx, 实时查看xxx文件的更新情况（常用！）

## ln创建软硬链接

ln [参数] [原文件或者目录] [链接名]

-s 创建软链接

ln创建硬链接

ln a b

硬链接等同于别名，与C++中的引用类似，创建以后就与原文件处于平等的地位，其底层的实现主要是对inode进行引用计数，删除一个硬链接或者删除原文件之后引用计数减1，只有当引用计数为0时，才会真正删除那个文件，也就是将inode结点所对应的磁盘数据块进行删除。

硬链接的限制：

1.不能跨分区

2.不能

ln创建软连接

软链接等同于传送门，与C++中的指针类似，作用等同于windows中的快捷方式，软链接可以指向一个不存在的文件（这称为死链接），

软链接与硬链接的区别：

1. 不能给目录创建硬链接，但是能创建软链接（为什么不能创建硬链接？假如在文件夹A中，创建指向文件夹A的硬链接B，那么递归删除文件夹A就成了一个无限递归，这是不被允许的）

2. 硬链接不能跨分区，软链接可以

3. 硬链接不占用实际的存储空间,软链接占用（因为硬链接实际上仅仅多了一个引用计数，而软连接需要真实地创建一个文件，软链接有自己的inode，inode指向一个数据块，然后将该文件指向原文件）

4. 软链接能够指向一个不存在的文件，硬链接做不到

ln -s a b

history 10

查看过去执行的10条命令

history -c

清除所有命令相关的历史记录

# 时间日期类

## date显示当前时间

date [“+%\*”]

%\*可以被写成%Y,%m,%d,%H,%M,%S，分别表示年、月、日、时、分、秒

1.date 以默认形式显示当前时间

2.date +%Y 显示当前年

3.date +”%Y-%m-%d %H:%M:%S” 按照自己规定的格式显示当前时间

## date显示非当前时间

date -d “1 days ago” 显示昨天这个时候的时间

date -d “-1 days ago” 显示明天这个时候的时间

## date设置系统时间

date -s “2017-06-19 20:52:18”

## cal查看日历

略

用户管理类

## useradd添加新用户

useradd tony添加新用户tony

useradd -g bigdata tony, 创建用户tony，其主组为bigdata

## passwd设置用户密码

passwd tony

然后按照提示设置密码

## id查看用户及其所属组的id

id tony

终端将会显示tony的uid，所在主组的gid，以及所有附加组的gid

## cat /etc/passwd查看创建了哪些用户

/etc/passwd这个配置文件保存了所用用户的相关信息，可以通过cat、more、less来查看

## su切换用户

su tony切换到tony用户

## userdel删除用户

userdel tony删除tony用户，不删除对应的主目录

userdel -r tony删除tony用户，并删除对应的主目录

## who查看当前用户

whoami

返回当前用户，当前层的那个用户

who am i

返回创建会话的用户名称，最外层那个用户

## sudo赋予root权限

在需要执行的命令前加上sudo，以赋予该用户超级管理员（root）的权限

需要root用户事先将其加入到/etc/sudoers

## usermod修改用户

usermod -g meifa tony，将tony的主组修改为meifa

usermod -aG meifa tony,以附加组的形式将tony加入meifa

# 用户组管理命令

## 用户组的概念

inux有用户和用户组的概念，在创建一个用户的时候，不仅会创建用户id，还会创建与用户名相同的组id。

此外，一个用户是可以加入多个组的，而一个组内的所用用户拥有一些特定的、相同的性质，比如对于文件权限而言，分为所属用户权限、所属用户组权限，其它权限。该文件所属用户组中的所有用户至少拥有对该文件相同程度的权限，之所以说至少是可能该组中的某些用户可能就是该文件的所属用户。

## 主组和附加组

每个用户有且仅有一个主组，如果创建用户的时候没有给它分配主组，那么linux将会自动生成一个与用户名同名的组，并且将该组作为用户的主组。

每个用户可以有多个附加组

useradd -g bigdata Jack, 添加用户Jack，设置主组为bigdata

usermod -aG testing Jack 以附加组的形式将Jack加入testing组，-a表示追加，-G表示附加组

usermod -g deeplearning Jack，将Jack的主组改为deeplearning

## groupadd增加用户组

groupadd meifa，增加meifa用户组

## groupmod修改用户组

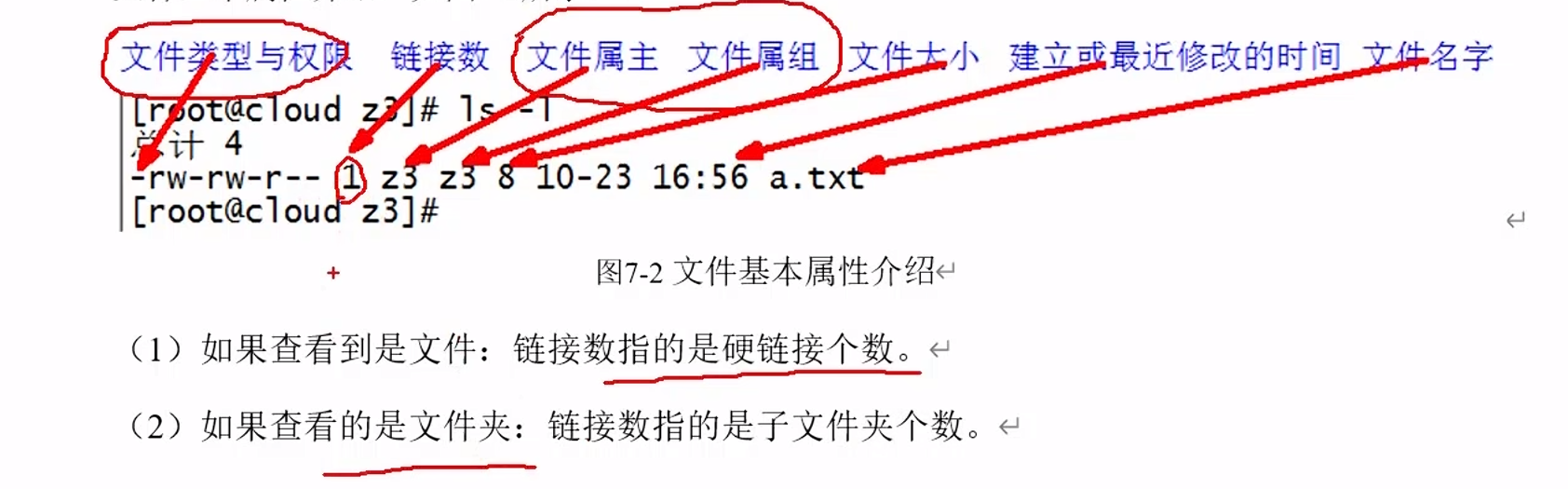
groupmod -n haircut meifa，修改meifa组的名称为haircut

## groupdel删除组

groupdel meifa，删除美发组

# 文件权限类

通过ls -l可以查看到文件权限



文件权限分为三类，所属用户权限，所属用户组权限，其它用户的权限

例如drwxr-xr--，表示该文件是目录类型，所属用户有读写执行权限，所属用户组有读和执行权限，其他用户有读权限。

文件权限也可以用八进制来表示， 0777代表所有权限对全部的用户都开放

## 目录读、写、执行权限的解释

目录读：是指是否具有ls该目录的权限

目录写：是指是否具有删除该目录下的文件，以及在该目录下新建文件的权限

目录执行：是指是否具有cd该目录的权限

## chmod更改用户权限

第一种更改权限的方式

chmod [{ugoa}{+-=}{rwx}] 文件或目录

比如chmod u = rx，代表让所属用户拥有读和执行的权限，没有写的权限

第二种更改权限的方式

2.chmod 0754 文件或目录

还有一种需要注意的

chmod -R 777 文件夹

表示递归的执行777命令，将该文件夹下的所有文件都设置成权限全开放

## chown更改所属用户

默认情况下，文件的所属用户是该文件的创建者，文件的所属用户组是该文件的创建者的主组。

此外执行chown需要超级管理员权限。

chown tony tempf， tempf的所属用户该为tony

chown tony:sgz tempf tempf的所属用户和所属用户组分别改为tony和sgz

-R，递归操作

## chgrp更改所属用户组

chgrp sgz tempf tempf的所属用户组该为sgz

-R，递归操作

# 搜索查找类

## find查找文件或目录

find 起始路径 [] xxx

### -name 根据目标文件名进行查找

1.find /home -name hello.cpp，以home目录为起点，查找文件” hello.cpp”

2. find /home -name \*.cpp，以home目录为起点，查找所有”.cpp” 文件

### -size 凭借文件大小作为筛选条件

find /root -size +10M 查找/root/下大于10M的所有文件

find /root -size -100 查找/root/下小于100B的所有文件

size默认单位为块（b），也可以后加c,k,M,G，分别表示字节，kB，MB，GB

### -user 查找特定用户的文件

## locate快速定位文件或目录

find直接在文件系统中遍历地查找，而locate则事先建立好所有文件的位置和关系的数据库（事先建档），无需遍历整个文件系统，因此速度更快。但是这种做法需要提前更新好locate

输入指令updatedb，以更新locate（并非每次locate之前都需要更新，只是为了定期存档）

正式用法：

locate 目标文件

## which查找非内嵌命令所在位置

which cd，查看cd命令所存储的位置

## grep过滤查找以及’|’管道符

-n 显示行号

grep [-n] 要查找的内容 原内容

其中原内容可以是文件，也可以是管道符’|’的输出

比如 ps -ef | grep -n bash, 首先查看所有进程的相关信息，接着查找其中包含bash关键字的所有行，附带行号， ‘|’前面的输出是后面的输入

-：忽略大小写i

-m：后跟数字，表示最多读取的行数，超过则略过

# 压缩和解压类

## gzip和gunzip

单纯的压缩和解压

## zip和unzip

归档就是将某个目录下的所有文件合成一个文件。

zip：归档压缩，一气呵成

unzip：解压解档，一气呵成

tar

归档和压缩是具有可选择的

-c：产生tar打包文件

-v：显示详细信息

-f：指定压缩后的文件名

-z：调用zip进行压缩或者解压

-x：解包.tar文件

-C：解压到指定目录

简而言之，tar的用法主要有如下两种。

压缩：tar -zcvf new.tar.gz /root，将root下的所有文件压缩成new.tar.gz

解压：tar -zxvf new.tar.gz /root\_bck/, 将压缩的打包文件解压和解包，存放到/root\_bck/下

# 磁盘查看和分区类

## du查看文件和目录所占用的磁盘空间

我们希望有一种方法能够统计某个目录下所有文件占用磁盘的总空间大小。

ls指令不能满足这一条件，它展现的仅仅只是目录本身（目录文件本身）的大小。

而du指令能够满足这样的需求。

du 目录或文件 查看目录或文件的磁盘使用情况

-h 以人类容易阅读的格式来显示

-a 不仅查看子目录大小，还要包含文件

-c 显示所有字目录之后，显示总和

-s 只显示总和

--max-depth=n 指定统计字目录的最大深度为n层

**常用写法: du -sh 文件或目录**

## df查看磁盘空间使用情况

df的全称是disk free，即空余磁盘，换句话说是用户可以使用的存储空间，该命令会列举出来

**常用写法：df -h**

## free查看内存使用情况

常用写法: free-h

这里的内存包括实际内存和虚拟内存（swap分区）

## lsblk查看设备挂载情况（次要知识）

常用写法： lsblk -f

## mount和unmount挂载/卸载（次要知识）

## fdisk硬盘分区（次要知识）

fdisk -l 查看硬盘分区详情

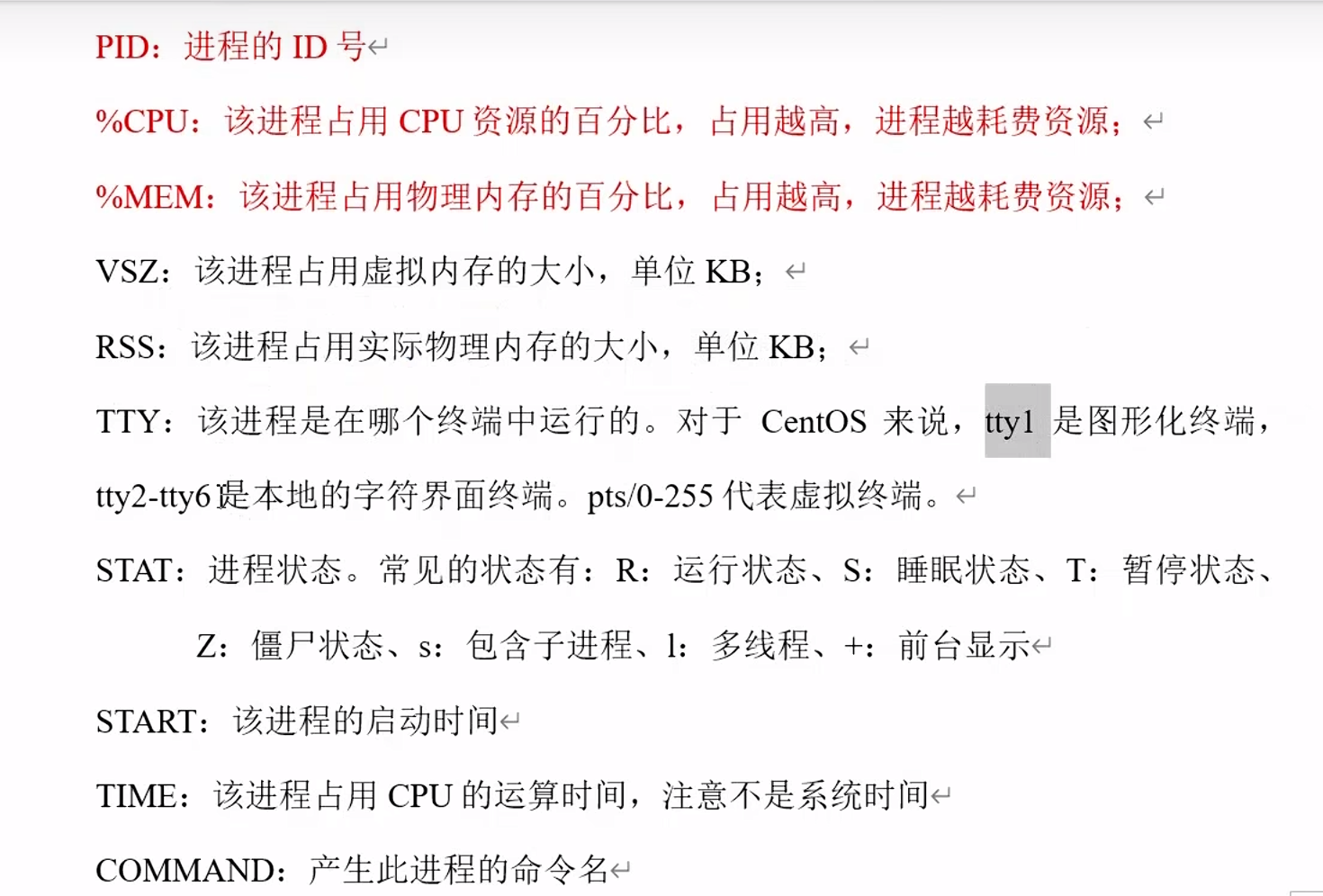
# 进程管理类

## ps查看当前系统进程状态

主要有两种用法，两种都是列出所有进程，但是显示的信息有不同

### 1.ps aux

显示CPU占用和内存占用



STAT补充 <代表高优先级，N代表低优先级

### 2.ps -ef

显示父进程的pid，即自身的ppid

## kill终止进程

#### 1.kill

kill [] 进程号

通过进程号杀死进程

**常用用法：kill -9 进程号**

#### 2.killall

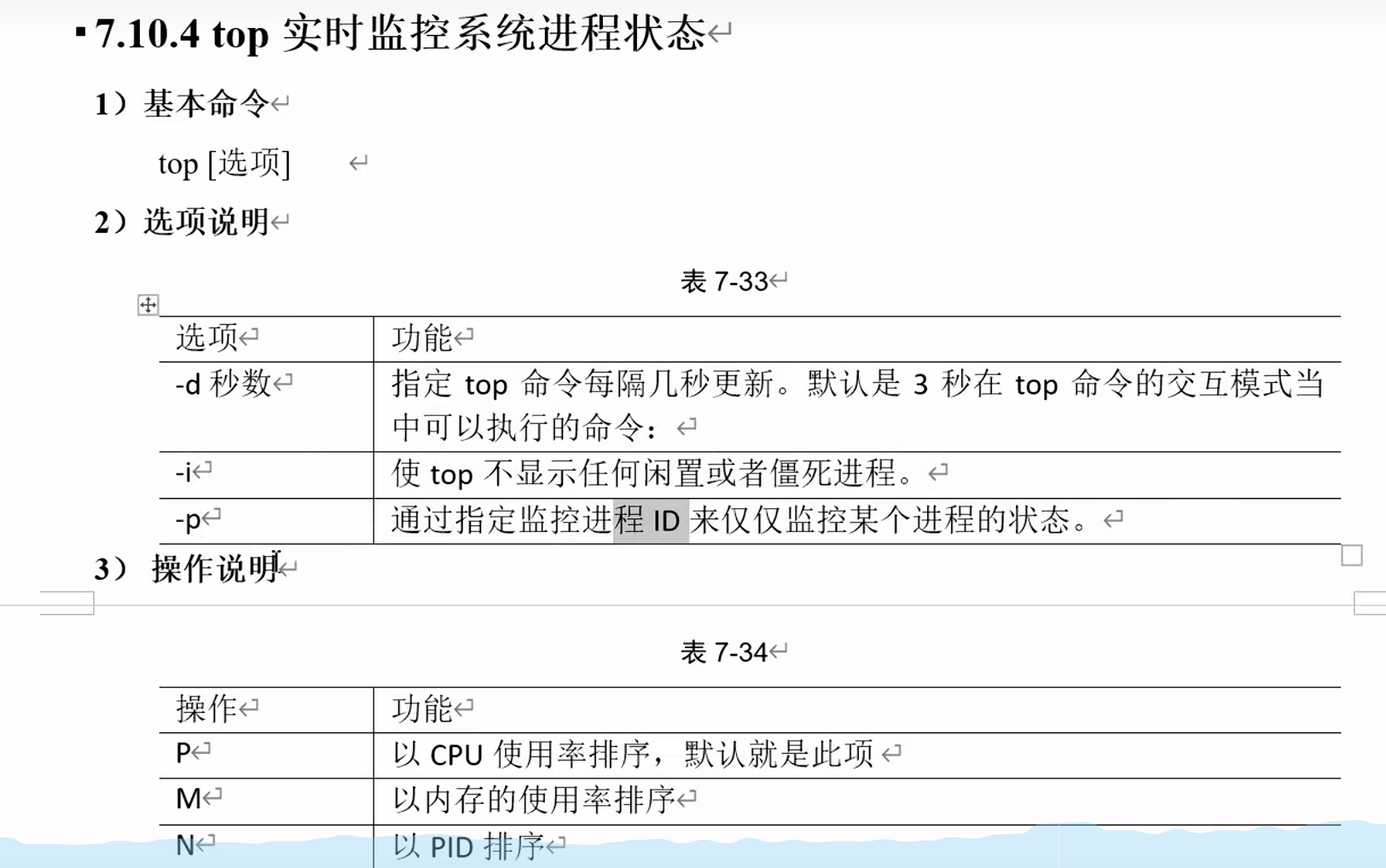
killall 进程名称

通过进程名称杀死进程,支持通配符

## pstree查看进程树

常用用法：pstree -up

## top查看系统健康状态



## netstat显示网络统计信息和端口占用

**常用用法：netstat -pan**

# 系统定时任务

## crontab 服务管理

crontab [参数]

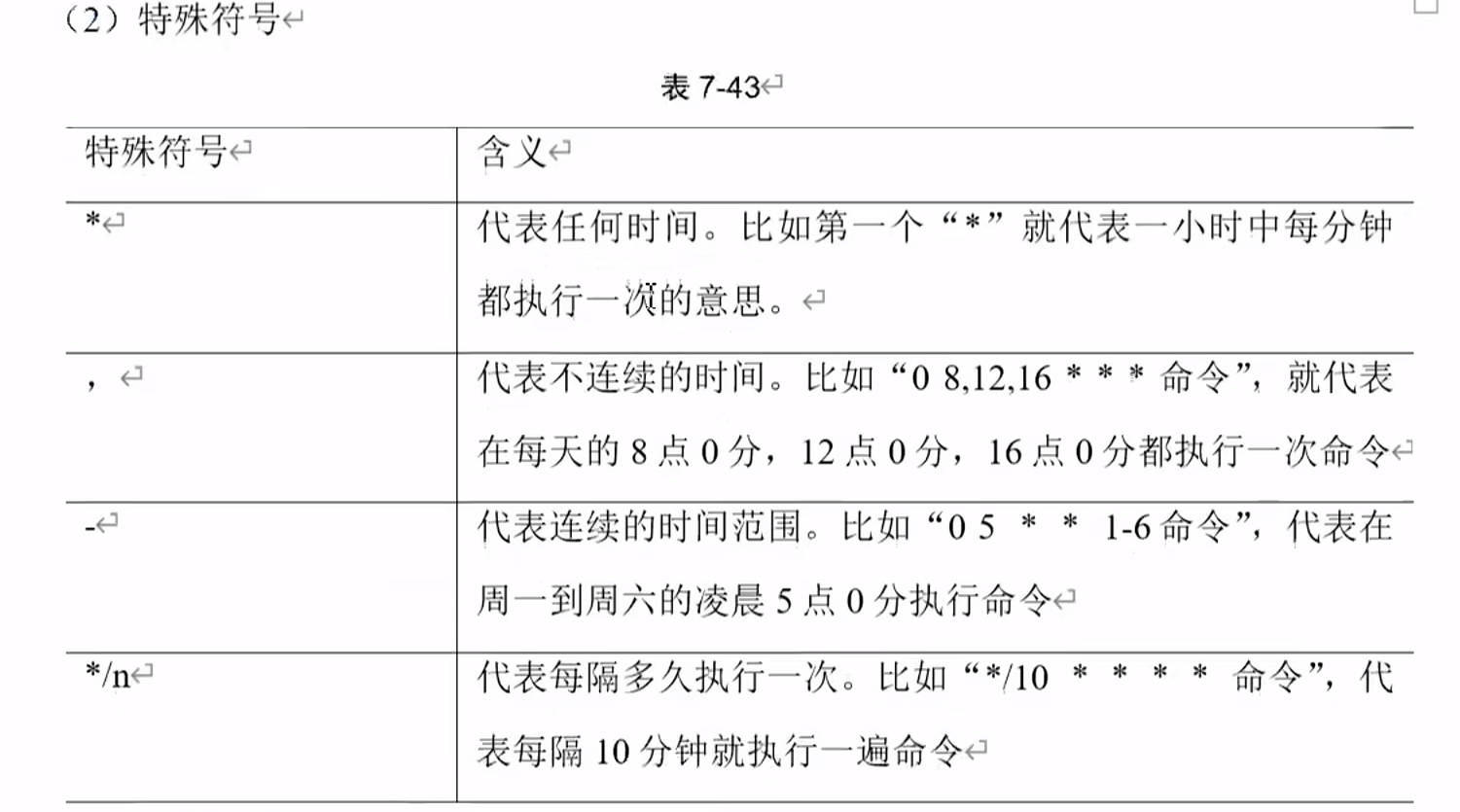
参数选项:

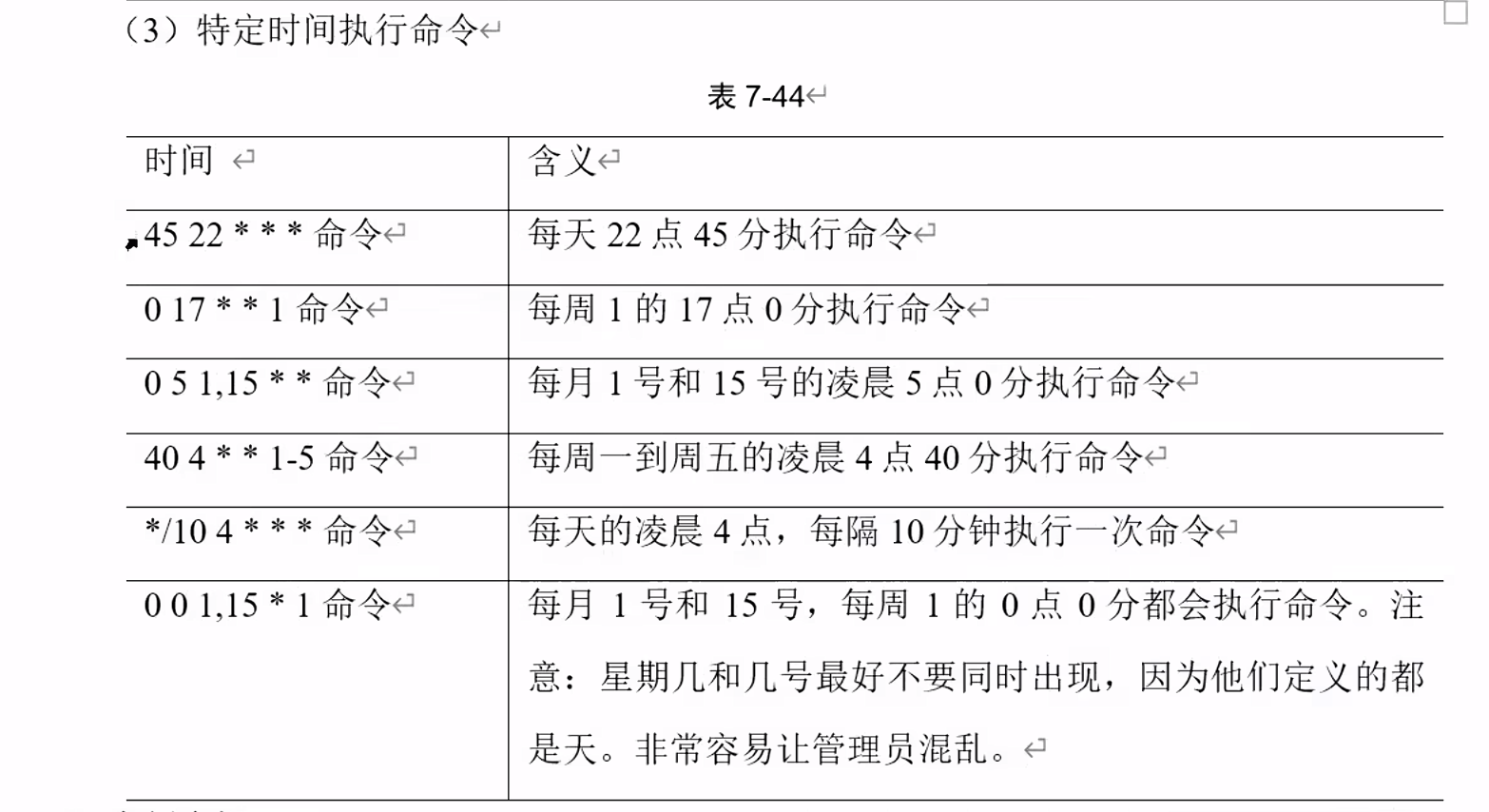
-e，编辑crontab定时任务

-l，查询crontab定时任务

-r，删除当前用户所有的crontab定时任务







## crontab定时任务设置

1.执行命令 crontab -e，进入文本编辑

2.添加一行：\*/1 \* \* \* \* echo “hello world” >> ~/regular\_hello.txt

所做的事情是每隔一分钟向该文件输入一句话

# shell脚本

## 执行shell脚本的方式

### 执行shell脚本的三种方式

执行shell脚本，有三种方式

1 ./hello.sh，需要有该文件的可执行权限, 作用域与当前的shell不同

2 bash hello/sh，另起一个子shell来执行，作用域与当前的shell不同

3 . hello.sh 或者source hello.sh，就在当前的shell执行，作用域与当前的shell相同

### 直接输入shell脚本的文件名来执行该脚本

有两种方法

1.将该脚本复制到/bin/。

2.更改PATH环境变量，将该shell脚本所在的目录添加到PATH环境变量之中。

## 变量

### 全局变量与局部变量

全局变量：子shell可见，但是子shell对该全局变量的修改不会影响到父shell中的同名全局变量

局部变量：子shell不可见

### 系统变量

即操作系统事先定义好的那些变量

查看系统全局变量的命令：env

查看所有系统变量和用户变量的命令：set

### 用户自定义变量

即用户自己定义的那些变量

#### 变量的定义

在shell脚本中，变量只有一种类型，那就是字符串，因此定义的时候无需声明类型。

myvar=”hello” #定义一个用户局部变量

export myvar #提升为用户全局变量

或者也可以直接定义全局变量：export aab=520

readonly a = 5 #定义只读变量

#### 变量的撤销

unset myvar 撤销变量myvar

另外，只读变量不能撤销

### 特殊变量

#### $n（类似于argv的单个成员）

主要用来捕获shell脚本文件的 命令行参数

n为某个特定的数字，$0为脚本名称，$1至$9为第1到9个参数，10或者10以上的变量需要写成${10}的形式

#### $#(类似于argc)

获取所有输入参数的个数，（不包括脚本文件名本身）

#### $\*和$@（类似于argv）

均为获取所有输入参数，前者是一个整体，后者是数组，建议只使用后者就行

#### $?

最后一次执行命令的返回状态，为0代表正确执行，非0代表错误

## 运算符

### 基本语法

“$((运算式))”或者”$[运算式]”

### 代码示例

计算式子4 + 3 \* 5：

my\_var=$[ 4 + 3 \* 5 ]

计算式子5^2（按位异或）

my\_var=$[ 5 ^ 2 ]

## 条件判断

### 基本语法

1. test condition

2. [ condition ]，注意condition前后要有空格，另外运算数和运算符之间也需要有空格

如果构成条件判断语句，条件成立返回值为0，否则为非0

如果不构成条件判断语句，条件为非空字符串则返回值为0，条件为空字符串则返回值为非0

返回值通过$?来获取

### 常用判断条件

#### 两个整数之间比较

-eq 等于 （equal） -ne不等于（not equal）

-lt 小于 （less than） -le小于等于（less equal）

-gt 大于 （greater than） -ge大于等于（greater equal）

如果是字符串之间的比较，用”=”判断相等，”!=”判读不相等

-z判断字符串长度是否为0，长度为0返回true

#### 按照文件权限进行判断

-r 有读的权限

-w 有写的权限

-x 有执行的权限

如 [ -x hello.sh ]，判断对该文件是否有写的权限

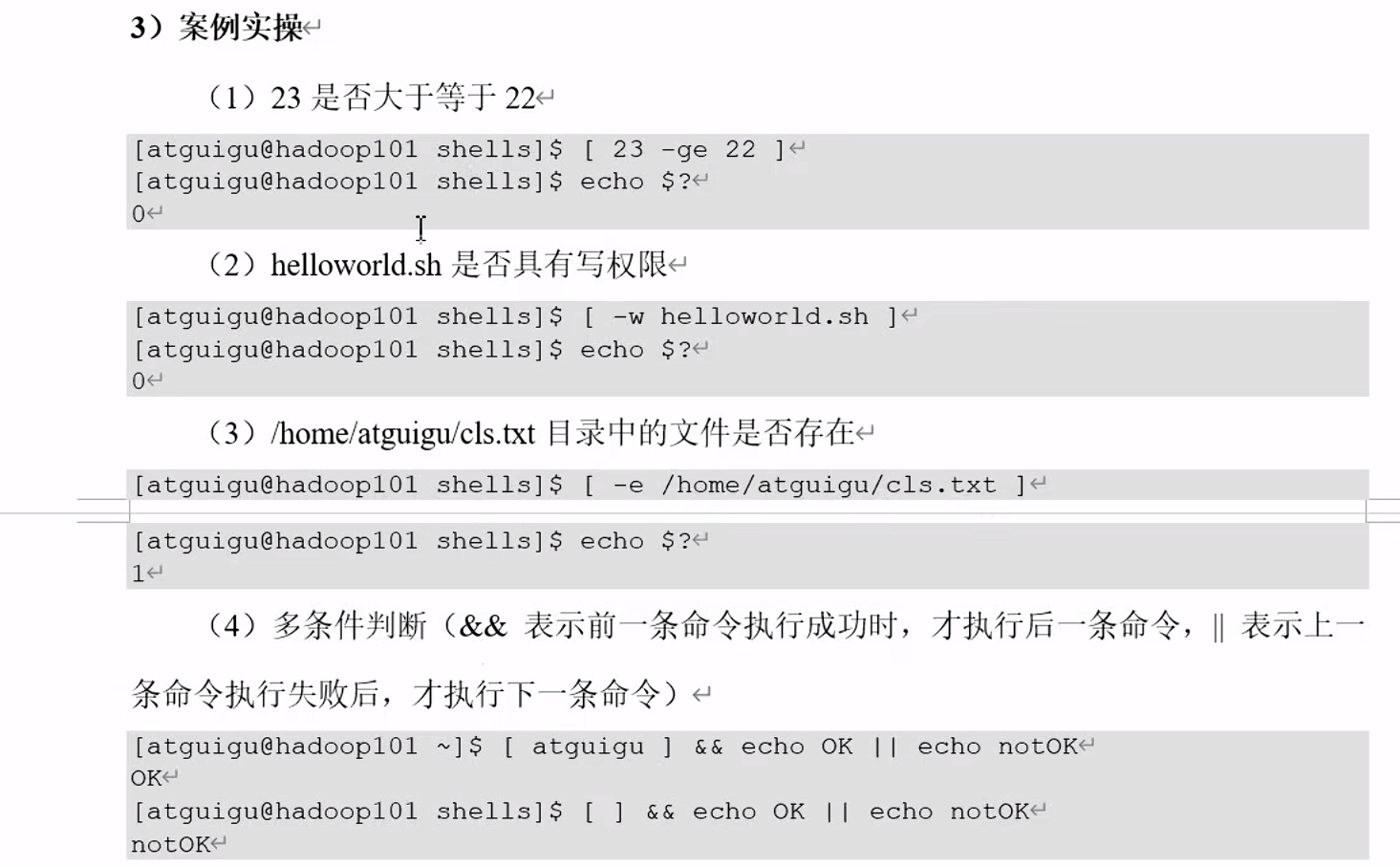
#### 按照文件类型进行判断

-e 文件存在

-f 文件存在并且使常规文件

-d 文件存在并且是一个目录

#### 代码示例



&&表示前一条命令执行成功，才执行后一条命令

||表示前一条命令执行失败，才执行后一条命令（短路匹配）

这样就能起到条件表达式的效果，如上图中的最后示例。

# 流程控制

## if判断

#### 单分支

if [条件判断式];then

程序

fi

或者写成下面的（推荐）

if [条件判断式]

then

程序

fi

#### 多分支

if [条件判断式]

then

程序

elif [条件判断式]

then

程序

else

程序

fi

## case语句

case $变量名 in

“值1”)

程序段1

“值2”)

程序段2

…

\*)

保底程序段

;;

esac

## for循环

### 用法1

for((初始值;循环控制条件;变量变化))

do

程序段

done

### 用法2（for each）

for 变量 in 值1 值2 值3…

do

程序段

done

搭配{1..100}，这个意思是生成1到100的序列

## while循环

while [ 条件判断式 ]

do

程序

done

# read读取控制台输入

read [选项] (参数)

选项：

-p：指定读取时的提示符

-t：指定读取时等待的时间（秒）如果-t不加，表示一直等待

参数：

指定读取值的变量名，即接收输入的变量

示例：

read -t 10 -p “请输入您的名字” name

echo “您好，$name”

# 函数

## 常用系统函数

这里需要有个概念，命令替换

比如获取当前时间戳，将其作为日志名的一部分，可以这么写

filename=”$1\_log\_$(date +%s)”

其中，$(date +%s)就是一个命令替换操作，相等于执行了date +%s

### basename

用法：basename pathname suffix，其中suffix为可选

比如basename /home/sgz/hello.txt .txt

将去掉前缀目录以及后缀.txt，输出hello

### dirname

用法：dirname pathname，获取文件的目录路径前缀（不包含文件名）

比如 dirname/home/sgz/hello.txt

输出dirname/home/sgz

## 自定义函数

### 语法

[ function ] funname[()]

{

程序段

[return n]

}

注意，上面的[]代表其中的内容可选择写或者不写

函数的返回值，只能通过$?系统变量获得，可以显示地加 return 返回值，如果不加返回值，将以最后一条命令的运行结果作为函数的返回值，return 后跟数值n（0-255）

# 正则表达式入门

grep、cut、awk均支持正则表达式

## 常规匹配

一串不包含特殊字符的正则表达式匹配它自己，比如：

cat /etc/passwd | grep atguigu

## 常规特殊字符

(1)特殊字符：^

匹配一行的开头，例如^a表示以a开头的所有行

(2)特殊字符：$

匹配一行的结束，例如b$表示以b结尾的所有行

^和$的混用

$aab^匹配为aab的所有行，这一行有且只能有aab三个字符

(3)特殊字符：.

匹配一个任意的字符

(4)特殊字符：\*

\*不单独使用，它和上一个字符连用，表示匹配上一个字符0次或多次

(5)字符区间（中括号）：[]

[]表示匹配某个范围内的一个字符，例如

[6,8] 匹配6或者8

[0-9] 匹配一个0-9的数字

[0-9]\* 匹配任意多个0-9的数字

[a-z] 匹配一个a-z的字符

[a-z]\* 匹配任意多个a-z的字符

[a-c,e-f] 匹配a-c或者e-f范围的任意字符

[^]表示不匹配这其中的字符，如[^0-9]表示不匹配0-9

(6)特殊字符：\

\表示转义，比如’grep \$’表示匹配$，因为$在正则表达式中具有特殊意义，不能单独出现。注意，应使用单引号，单引号表示不额外对其中的内容做处理，而双引号则相反。

(7)其它的一些规则（并不全）

?表示匹配0次或1次，+表示匹配1次或多次，a{2}表示匹配两次a字符

这些属于扩展规则，如grep要使用，需要加-E参数。

详细可查看[Linux-正则表达式 - 哔哩哔哩 (bilibili.com)](https://www.bilibili.com/read/cv14581803/)

# 文本处理工具

## cut

cut的工作就是“剪”，具体的说就是在文件中负责剪切数据用的，cut命令从文件的每一行剪切字节、字符和字段并将这些字节、字符和字段输出。

（1）基本用法

cut [选项参数] filename，也可以被用作管道

说明：默认分隔符是制表符

(2)选项参数说明

-f：后跟列号，提取第几列

-d：分隔符，按照指定分隔符分割列，默认是制表符”\t”

-c：按字符进行切割 后加n表示取第几列 比如 -c 1

例子：cat /etc/passwd | grep bash$ | cut -d : -f 1,6-7

以’:’为分隔符，截取第1、6、7行

## awk

一个强大的文本分析工具，把文件逐行的读入，使用时需要给出分隔符，以作为列分隔的依据。首先根据正则表达式对行进行筛选，然后再根据分隔符得出各个列，筛选过的行可以后跟具体的程序段，来做相应的处理，其中就包括提取某几列

换言之，awk一个api就做了grep和cut两个api联手才能完成的事情（grep筛选行，cut筛选列）

并且在此基础之上，还能够做很多额外的事情

注意：awk之中不要再使用单引号，因为awk自身使用了单引号

 AWK 拥有自己的语言,因此AWK代码块里面的内容不是shell脚本！很像C语言，可以初步将其理解为没有类型的C语言

基本用法

awk [选项参数] ‘/pattern1/{action1} /pattern1/{action1}’ … filename，也可以承接管道操作

选项参数说明

-F：指定分隔符，awk的默认分隔符是任意连续的多个空白符（比如空格，制表符）

-v：赋值一个用户定义变量

案例实操

(1)数据准备

sudo cp /etc/passwd ./

passwd数据的含义

用户名：密码（加密过后的）：用户id：组id：注释：用户家目录：shell解析器

(2)搜索passwd文件以root关键字开头的所有行，并输出该行的第7列

awk -F ‘:’ ‘/^root/{print $7}’ passwd

(3)搜索passwd文件以root关键字开头的所有行，并输出该行的第1列和第7列，中间以’，’号分割

awk -F ‘:’ ‘/^root/{print $1’,’$7}’ passwd

(4)只显示/etc/passwd的第一列和第七列，以逗号分隔，且在所有行前面添加列名user，shell在最后一行添加”dahaige，/bin/zuishuai”。

awk -F ‘:’ ‘BEGIN{print “user，shell”} {print $1”,”$7} END{print “dahaige，/bin/zuishuai”}’ passwd

注意：BEGIN在所有数据读取之前执行，END在所有数据读取之后执行，这个是与代码顺序无关的。

(5)将passwd文件中的用户id增加1并输出

awk -F ‘:’ -v i=1 ‘{print $3 + i}’ passwd

awk的内置变量

FILENAME：文件名

NR：当前处理的行是第几行

NF：当前行有多少列数据（这个在每行都会根据设定的分割符重新计算，默认分割符是任意连续的多个空白符）

简而言之，awk先根据正则表达式，筛选出需要的行，然后根据代码块，筛选出需要的列，代码块的设定使得awk能够做更多的事情

### print的用法

print不加任何参数，代表打印这一整行

print $m, $n代表打印第m和第n列，中间以空格隔开

print $NF代表打印最后一列

### awk中定义数组



# 综合应用案例

who -T可以查看所用用户的终端情况，终端前面带+号，代表打开了mesg，带-号，说明关闭了mesg

可以手动关闭或者打开mesg， mesg n或者mesg y

# linux扩展内容

## inode

在linux的单个硬盘分区中，所有的文件在储存的时候都有一个对应的编号（index），这个index叫做inode，通过inode可以唯一确认硬盘分区中的一个数据块，这个数据块上存储的内容对应的就是文件的所有信息。

可以通过 ls -i来查看inode

## 文件类型

通过ls -l命令可以查看linux’系统的文件类型

linux中有7种文件类型，下面依次以标识符来介绍它们

-:普通文件类型，可以通过vim来打开，硬链接也算作普通文件

d:目录文件，本质上是一个目录，linux将目录也视为文件

l:软链接文件，

c:字符类型设备（键盘、鼠标）所对应的文件映射

b: 块类型（硬盘）设备所对应的文件映射

s:套接字文件

p:管道文件

# 单引号与双引号

单引号：

单引号将它中间的所有任意字符还原为字面意义，实现屏蔽shell元字符的功能，单引号必须成对出现

双引号：

双引号类似于单引号，但不会屏蔽` \ $ 这三个shell元字符的含义，如果需要屏蔽需要前面加一个\符号，其他字符的功能将被屏蔽（包括单引号），两个双引号之间的单引号不必成对出现

1、当pattern或者说匹配模式中，没有空格，可以不用引号，但是有空格一定要用引号。

2、考虑单引号和双引号的区别，考虑pattern是否有变量，是否需要求值

在正则表达式中，\有特殊的用法，$在末尾也有特殊的用法，单引号和双引号的区别仅在于双引号会对`` 进行命令解析，对$进行变量替换，而在正则表达式中一般不会出现这两种用法，所以正则表达式中通常使用单引号即可。

————————————————

原文链接：https://blog.csdn.net/qinglongqishi1/article/details/82716020/