Linux

Linux介绍

- 介绍
  - Linux读法：【里纽克斯，里尼科斯，里纳克斯】
  - Linux是一款操作系统，免费，开源，安全，高效，稳定，处理高并发强悍
  - Linux创始人：linus
  - Linux吉祥物：企鹅，Tux
  - Linux主要的发行版：
    - RedHat
      - CentOS
      - RedHat
    - Ubuntu
    - Suse
    - 红旗Linux
    - 还有很多
  - 目前主要的操作系统有
    - window，android，车载系统，linux
- Linux与Unix的关系
  - Unix是由Ken Thompson和Dennis Ritchie共同发明的
    - Linux的开发过程呢是因为当时Linus使用的基于Unix的操作系统Minix太难用了，于是决定自己开发一个操作系统，采用了Unix的设计理念，于是Linux诞生了。
- Linux与Window比较
  - Linux
    - 免费或少许收费
    - 软件开源
    - 相对来说比window安全
    - 兼具图形界面操作和完整的命令行操作，可以只使用键盘完成一切操作
  - Window
    - 收费
    - 软件收费

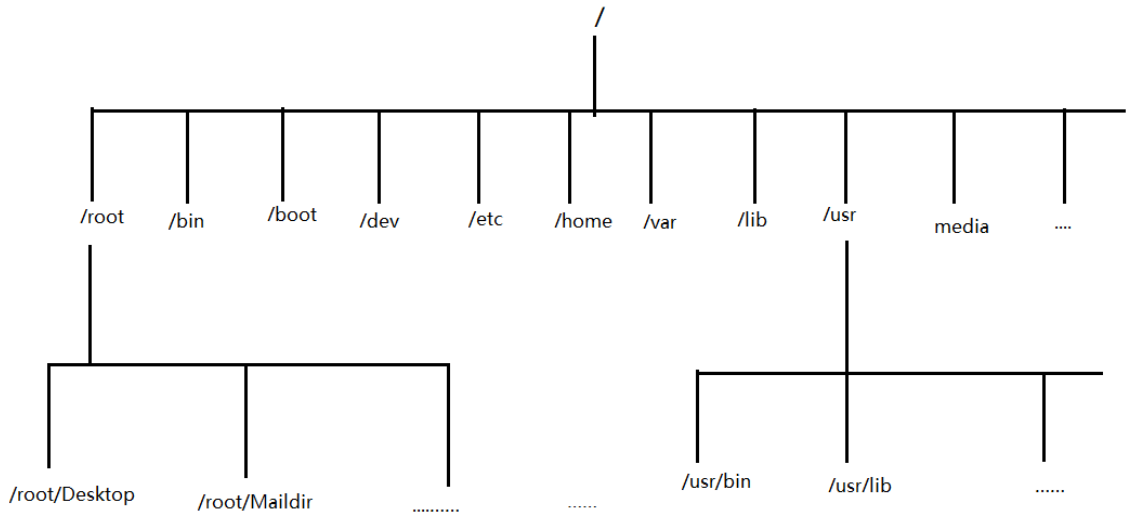
- 容易中病毒，三天两头打补丁
- 纯图形操作界面，容易入门

Linux基础

Linux文件目录系统

> 在Linux世界里，一切皆文件
>
> > Linux的目录中有且只有一个根目录
> >
> > linux的各个目录存放的内容是规划好，不用乱放文件
> >
> > linux是以文件的形式管理我们的设备，因此linux系统，一切皆文件

linux的文件系统是采用级层式的树状目录结构，在此结构中的最上层是根目录“/"，然后在此目录下再创建其他的目录。



- `/bin`
  - 存放着最经常使用的命令（Binary的缩写）
- `/sbin`
  - s就是Super User的意思，这里存放的是系统管理员使用的系统管理程序。
- `/home`
  - 存放普通用户的主目录，在Linux中每个用户都有一个自己的目录，一般该目录名是以用户的账号命名的。
- `/root`
  - 该目录为系统管理员，也称作超级权限者的用户主目录。

- `/lib`
  - 系统开机所需要最基本的动态连接共享库，其作用类似于windows里的DLL文件。几乎所有的应用程序都需要用到这些共享库。
- `/lost+found`
  - 这个目录一般情况下是空的，当系统非法关机后，这里就存放了一些文件。
- `/etc`
  - 所有的系统管理所需要的配置文件和子目录
- `/usr`
  - 这是一个非常重要的目录，用户的很多应用程序和文件都放在这个目录下，类似与windows下的program files目录。
- `/boot`
  - 存放的是启动Linux时使用的一些核心文件，包括一些连接文件以及镜像文件
- `/proc`
  - 这个目录是一个虚拟的目录，它是系统内存的映射，访问这个目录来获取系统信息.
- `/srv`
  - service缩写，该目录存放一些服务启动之后需要提取的数据。
- `/sys`
  - 这是linux2.6内核的一个很大的变化。该目录下安装了2.6内核中新出现的一个文件系统`ysfs`
- `/tmp`
  - 这个目录是用来存放一些临时文件的。
- `/dev`
  - 类似于windows的设备管理器，把所有的硬件用文件的形式存储。
- `/media`
  - linux系统会自动识别一些设备，例如u盘、光驱等等，当识别后，linux会把识别的设备挂载到这个目录下。
- `/mnt`
  - 系统提供该目录是为了让用户临时挂载别的文件系统的，我们可以将外部的存储挂载在`/mnt/`上，然后进入该目录就可以查看里的内容了。
- `/opt`
  - 这是给主机额外安装软件所摆放的目录。如安装ORACLE数据库就可放到该目录下。默认为空。
- `/usr/local`
  - 是另一个给主机额外安装软件所安装的目录，一般是通过编译源码方式安装的程序。
- `/var`

    - 这个目录中存放着在不断扩充着的东西，习惯将经常被修改的日录放在这个日录下。包括各种日志文件。
- `/selinux`
    - SELinux是一和安全子系统,它能控制程序只能方间特定文件。（*security-hanced*）

远程登陆Linux系统

说明:公司开发时候，具体的情况是这样

1. linux服务器是开发小组共享的
2. 正式上线的项目是运行在公网的
3. 因此程序员需要远程登录到centos进行项目管理或者开发

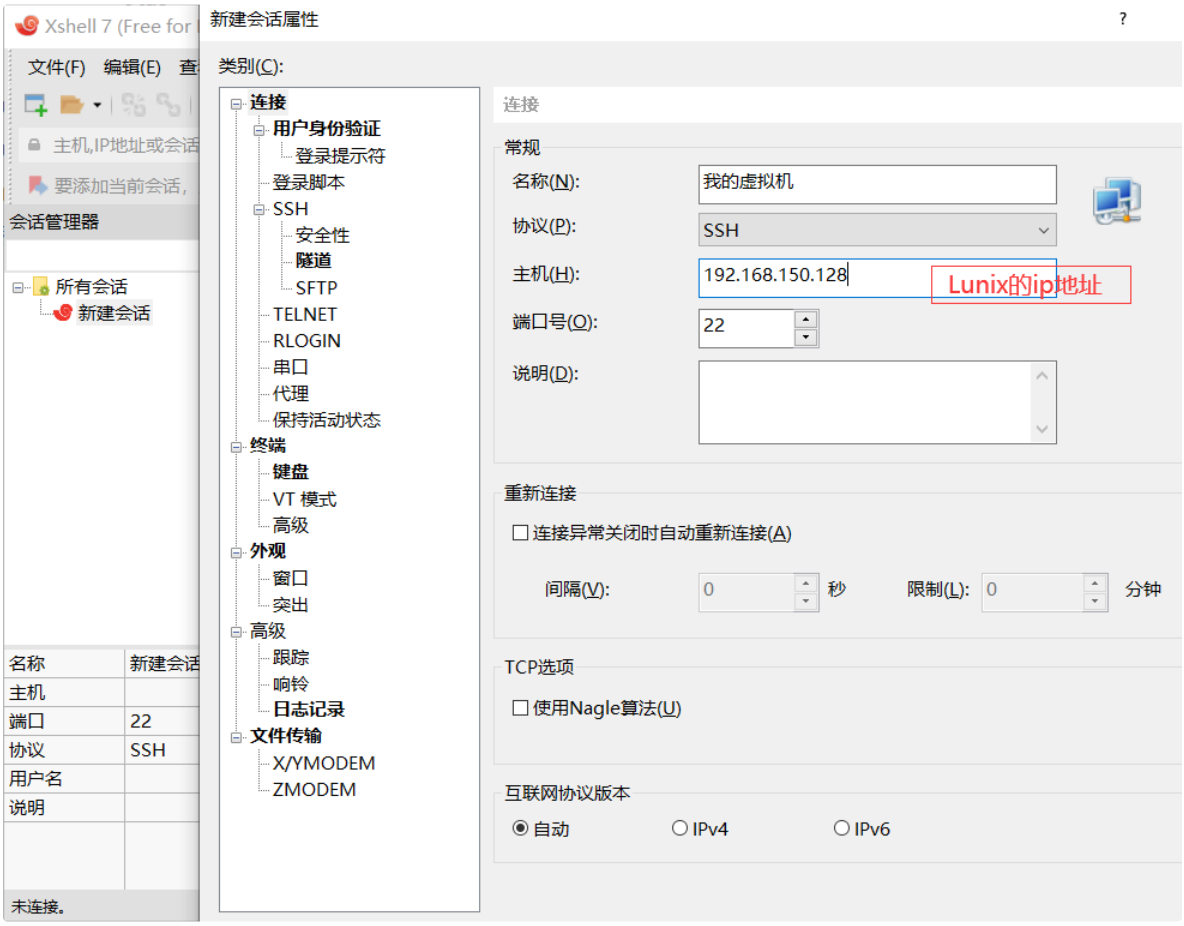
- Ubuntu系统查询ssh服务

```
// 安装ssh服务器
sudo apt-get install openssh-server

// 打开ssh服务
sudo service ssh start

// 查询ssh服务
sudo ps -e | grep ssh
```

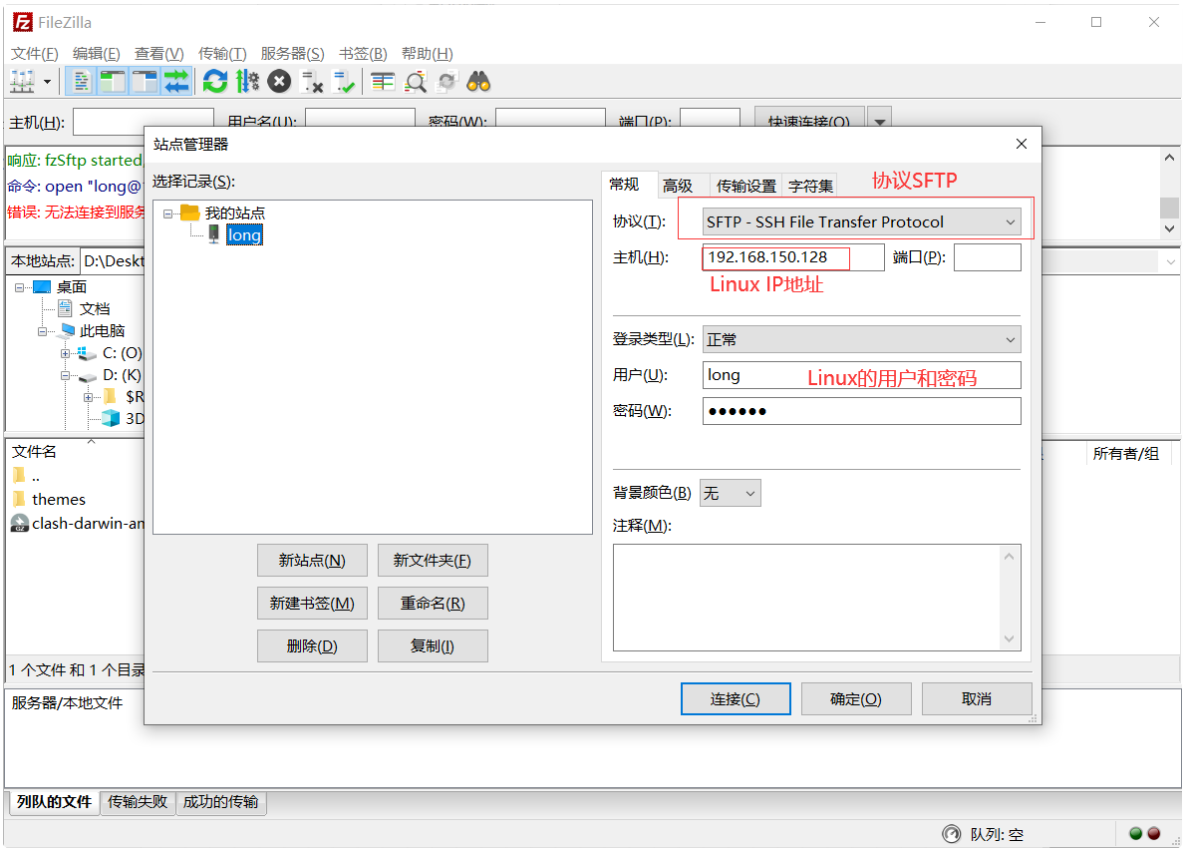
- CentOS系统开启ssh服务

```
// 安装ssh服务
yum install -y openssl openssh-server

// 启动ssh服务
systemctl start sshd.service
```

- 开启了ssh服务之后，使用Xhell工具进行远程登陆Linux系统
    - 安装好Xshell后，点击文件->新建



    - 然后点击确定，点击会话进行连接
    - 输入你要连接linux用户名和密码
    - 如果需要连接root用户，则需要在linux系统中开启ssh的root权限（百度一下）
- 使用FileZilla进行上传和下载文件
    - 安装好之后点击导航栏的文件——>站点管理器——>新建站点



- 然后点击连接就可以点击连接了，进行连接linux必须安装ssh服务

vi和vim使用

- 所有的Linux系统都会内建vi文本编辑器。
- Vim具有程序编辑的能力，可以看做是vi的增强版本，可以主动的以字体颜色辨别语法的正确性，方便程序设计。代码补完、编译及错误跳转等方便编程的功能特别丰富，在程序员中被广泛使用。
- vi和vim的三种常见模式
  - 正常模式:
    - 以vim打开一个档案就直接进入一般模式了(这是默认的模式)。在这个模式中，你可以使用『上下左右』按键来移动光标，你可以使用『删除字符』或『删除整行』来处理档案内容，也可以使用『复制、贴上』来处理你的文件数据。
  - 插入模式:
    - 在正常模式下按下i, I, o,O, a,A, r, R等任何一个字母之后才会进入编辑模式,一般来说按i即可.
  - 命令行模式
    - 在vim中按下esc键进入命令行模式，在这个模式当中，可以提供你相关指令，完成读取、存盘、替换、离开vim、显示行号等的动作则是在此模式中达成的!
- Vim 的常见指令案例
  - yy：拷贝当前行
  - 5yy：拷贝当前5行

- dd：删除当前行
- 5dd：删除当前行向下的5行
- 在文件中查找某个单词：命令行输入 /（查找内容），按n查找下一个
- 设置文件行号：set nu，取消文件行号：set nonu
- 编辑文件，正常模式下使用快捷键到达文档最末行：G，最首行：gg
- 撤销输入：在正常模式下输入u
- 编辑文件，光标移动到某行：shift+g
- 显示行号：set nu
  - 输入行号这个数
  - 输入shift+g

文件命令

指令	说明
vim [file1 file2 file3 ...]	打开单个或多个文件
:open file	在vim窗口中打开一个新文件
:split file	在新窗口中打开文件（split打开的窗口都是横向的，使用vsplit可以纵向打开窗口。）
Ctrl+ww	移动到下一个窗口
Ctrl+wj	移动到下方的窗口
Ctrl+wk	移动到上方的窗口
:close	最后一个窗口不能使用此命令，可以防止意外退出vim。
:only	关闭所有窗口，只保留当前窗口
:bn	切换到下一个文件
:bp	切换到上一个文件
:args	查看当前打开的文件列表，当前正在编辑的文件会用[]括起来
:e ftp://192.168.10.76/abc.txt	打开远程文件，比如ftp或者share folder

普通模式

- 插入命令

指令	说明
i	在当前位置生前插入
I	在当前行首插入
a	在当前位置后插入
A	在当前行尾插入
o	在当前行之后插入一行
O	在当前行之前插入一行

- 游标移动

指令	说明
gg	移动到文件头。 = [[
G（shift + g）	移动到文件尾。 =]]
行数 → Shift+G	移动到第 n 行
冒号+行号，回车	比如跳到240行就是 :240回车
h	左移一个字符
l	右移一个字符，这个命令很少用，一般用w代替。
k	上移一个字符
j	下移一个字符
w	向前移动一个单词（光标停在单词首部）
b	向后移动一个单词 2b 向后移动2个单词
e	同w，只不过是光标停在单词尾部
ge	同b，光标停在单词尾部。
^	移动到本行第一个非空白字符上。
0	移动到本行第一个字符上
HOME	移动到本行第一个字符。同0健。
$	移动到行尾 3$ 移动到下面3行的行尾
f（find）	fx将找到光标后第一个为x的字符，3fd将找到第三个为d的字符。
F	同f，反向查找

- 撤销和重做

指令	说明
u	撤销（Undo）
U	撤销对整行的操作
Ctrl + r	重做（Redo），即撤销的撤销。

- 删除命令

指令	说明
x	删除当前字符
3x	删除当前光标开始向后三个字符
X	删除当前字符的前一个字符。X=dh
dl	删除当前字符， dl=x
dh	删除前一个字符
dd	删除当前行
dj	删除上一行
dk	删除下一行
10d	删除当前行开始的10行。
D	删除当前字符至行尾。D=d$
d$	删除当前字符之后的所有字符（本行）
kdgg	删除当前行之前所有行（不包括当前行）
jdG（jd shift + g）	删除当前行之后所有行（不包括当前行）
:1,10d	删除1-10行
:11,$d	删除11行及以后所有的行
:1,$d	删除所有行
J(shift + j)	删除两行之间的空行，实际上是合并两行。

- 拷贝，剪贴和粘贴

指令	说明
yy	拷贝当前行
nyy	拷贝当前后开始的n行，比如2yy拷贝当前行及其下一行。
p	在当前光标后粘贴,如果之前使用了yy命令来复制一行，那么就在当前行的下一行粘贴。
shift+p	在当前行前粘贴
:1,10 co 20	将1-10行插入到第20行之后。
:1,$ co $	将整个文件复制一份并添加到文件尾部。
ddp	交换当前行和其下一行
xp	交换当前字符和其后一个字符
ndd	剪切当前行之后的n行。利用p命令可以对剪切的内容进行粘贴
:1,10d	将1-10行剪切。利用p命令可将剪切后的内容进行粘贴。
:1, 10 m 20	将第1-10行移动到第20行之后。

正常模式下按v（逐字）或V（逐行）进入可视模式，然后用jklh命令移动即可选择某些行或字符，再按y即可复制

- 退出命令

指令	说明
:wq	保存并退出
ZZ	保存并退出
:q!	强制退出并忽略所有更改
:e!	放弃所有修改，并打开原来文件。
:q	未修改直接退出

- 注释命令

perl程序中#开始的行为注释，所以要注释某些行，只需在行首加入#

指令	说明
3,5 s/^/#/g	注释第3-5行
3,5 s/^#//g	解除3-5行的注释
1,$ s/^/#/g	注释整个文档。
:%s/^/#/g	注释整个文档，此法更快。

执行shell命令

- :!command

指令	说明
:!ls	列出当前目录下文件
:!perl -c script.pl	检查perl脚本语法，可以不用退出vim，非常方便。
:!perl script.pl	执行perl脚本，可以不用退出vim，非常方便。
:suspend或Ctrl - Z	挂起vim，回到shell，按fg可以返回vim。

帮助命令

指令	说明
:help or F1	显示整个帮助
:help xxx	显示xxx的帮助，比如 :help i, :help CTRL-[（即Ctrl+[的帮助）。
:help 'number'	Vim选项的帮助用单引号括起
:help <Esc>	特殊键的帮助用<>扩起
:help -t	Vim启动参数的帮助用-
:help i_<Esc>	插入模式下Esc的帮助，某个模式下的帮助用模式_主题的模式

帮助文件中位于||之间的内容是超链接，可以用Ctrl+]进入链接，Ctrl+o（Ctrl + t）返回

其他非编辑命令

指令	说明
.	重复前一次命令
:set ruler?	查看是否设置了ruler，在.vimrc中，使用set命令设制的选项都可以通过这个命令查看
:scriptnames	查看vim脚本文件的位置，比如.vimrc文件，语法文件及plugin等。
:set list	显示非打印字符，如tab，空格，行尾等。如果tab无法显示，请确定用 set lcs=tab:>-命令设置了.vimrc文件，并确保你的文件中的确有tab，如果开启了 expandtab，那么tab将被扩展为空格。

- Vim教程
  - 在Unix系统上 $ vimtutor
  - 在Windows系统上 :help tutor
- 录制宏:
  - 按q键加任意字母开始录制，再按q键结束录制（这意味着vim中的宏不可嵌套），使用的时候@加宏名，比如qa。。。q录制名为a的宏，@a使用这个宏。
- :syntax 列出已经定义的语法项
- :syntax clear 清除已定义的语法规则
- :syntax case match 大小写敏感，int和Int将视为不同的语法元素
- :syntax case ignore 大小写无关，int和Int将视为相同的语法元素，并使用同样的配色方案

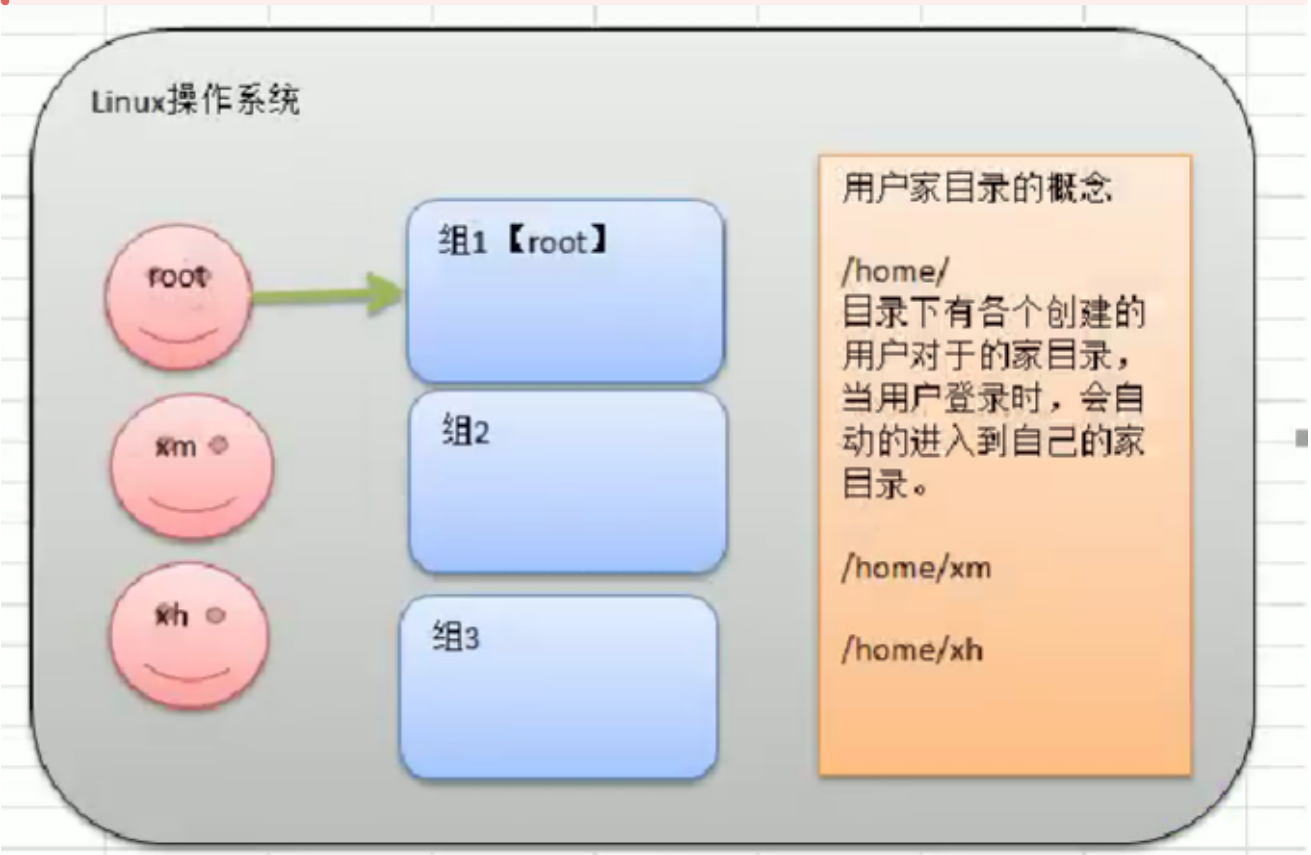
关机&重启&注销

- 关机
  - shutdown -h now：立即关机
  - shutdown -h 1：1分钟后关机
  - half：直接关机
- 重启
  - shutdown -r now：立即重启
  - reboot：重启系统
- sync
  - 把内存的内容同步到磁盘
  - 当我们关机或重启的时，都应该先执行以下sync指令，把内存的数据存入磁盘，防止数据丢失
- 登陆

    - 登陆时尽量少用root账号登陆，因为他是系统管理员，最大的权限，避免操作失误，可以使用普通用户登陆，登陆后在使用`su "用户名"`命令来切换成系统管理员身份
- 注销
    - logout：注销
    - logout在图形运行界面无效
    - 在运行级别3下有效

用户管理

> Linux系统是一个多用户多任务的操作系统，任何一个要使用系统资源的月户，都必须首先向系统管理员申请一个账号，然后以这个账号的身份进入系统。
>
> > Linux的用户至少属于一个组



- 添加用户
    - `adduser [选项] 用户名 [用户组]（常用）`
        - 当创建用户成功后，会自动的创建和用户同名的家目录
        - -h Dir指定家目录位置
        - ingroup：指定用户所属的组
        - 可以是使用`adduser -help`查看参数怎么写
- 指定密码
    - `passwd 用户名`

- 删除用户
  - `deluser 用户名（常用）`
  - 删除用户及家目录
    - `deluser -remove-home 用户名`
  - 删除用户组
    - `deluser -group 用户组名`
  - *删除用户时尽量不要删除家目录*
- 查询用户信息
  - `id 用户名`
  - 当该用户不存在时，则会报无此用户
- 切换用户
  - 在操作Linux中，如果当前用户的权限不够，可以通过su指令，切换到高权限用户，比如root
  - `su 用户名`
  - `exit` 返回原先用户
- 查看当前用户
  - `whoami`
- 用户组
  - 类似于角色，系统可以对有共性的多个用户进行统一的管理。
  - 增加组:
    - `addgroup [参数] GROUP`
  - 删除组
    - `delgroup [参数] GROUP`
  - 修改用户组
    - `usermod -g 用户组 用户`
- 用户和组相关的文件
  - /etc/passwd 文件
    - 用户(user!的配置文件，记录用户的名种信息
    - 每行的含义:用户名:口令:用户标识号:组标识号:注释性描述:主目录:登录shell
  - /etc/shadow文件
    - 口令的配置文件
    - 每行的含义。登录名:加密口令:最后一次修改时间:最小时间间隔:最大时间间隔:警告时间:不活动时间:失效时间:标志
  - /atc/group文件

    - 组(group)的配置文件，己录unux包含的组的信息
    - 每行含义:组名:口令:组标识号:组内用户列表

实用指令

指定运行级别

- 0，关机
- 1，单用户【找回丢失密码】
- 2，多用户状态没有网络服务
- 3，多用户状态有网络服务
- 4，系统未使用保留给用户
- 5，图形界面
- 6，系统重层
- 常用的运行级别是3和5，要更改默认的运行级别可修改文件`/etc/inittab`的`id:5:initdefault`这一行中的数字
- 命令`init[012356]`

帮助指令

- 当我们对某个指令不熟悉时，我们可以使用Linux提供的帮助指令来了解这个指令的使用方法。
- `man`
    - 获取帮助信息
    - `man [命令或配置文件]` (功能描述:获得帮助信息)
- `help`
    - 获取帮助信息
    - `help 指令` (功能描述。获得shell内置命令的帮助信息)

文件目录类

- **pwd**指令
    - `pwd`：显示当前工作目录的绝对路径
- **ls**指令
    - `ls [选项] 目录或是文件`：显示当前目录
    - -a：显示当前目录所有的文件和目录，包括隐藏的
    - -l：以列表的方式显示信息

- **cd**指令
  - `cd [参数]`：切换到指定目录目录
- **mkdir**指令
  - `mkdir [选项] 要创建的目录`：创建目录
  - -p：创建多级目录
- **rmdir**指令
  - `rmdir 空目录`：删除指定空目录
- **rm -rf**指令
  - `rm -rf 非空目录`：删除指定非空目录
  - -r：递归删除整个文件夹
  - -f：强制删除不提示
- **touch**指令
  - `touch 文件名称`：创建空文件
- **cp**指令
  - `cp [选项] source dest`：拷贝文件到指定目录
  - -r：递归复制整个文件夹
- **mv**指令
  - mv mvoldNameFile newNameFile (功能描述。重命名)
  - mv /temp/movefile /targetFolder (功能描述。移动文件)
- **cat**指令
  - `cat [选项] 要查看的文件常用选项`：查看文件内容
  - -n :显示行号
  - | more：分页显示（管道命令）
- **more**指令
  - more指令是一企基于vl编辑器的文本过滤器，它以全屏幕的方式按页显示文本文件的内容。more指令中内置了若干快捷键
  - `more 要查看的文件`

操作	功能说明
空格（space）	代表向下翻一页
Enter	代表向下翻【一行】
q	代表立即离开more，不在显示文件内容
Ctrl+F	向下滚动一屏
Ctrl+B	返回上一屏
=	输出当前行的行号
:f	输出文件名和当前行的行号
- **less**指令
  - less指令用来分屏查看文件内容，它的功能与more指令类似，但是比more指令更加强大，支持各种显示终端。less指令在显示文件内容时，并不是一次将整个文件加载之后才显示，而是根据显示需要加载内容，对于显示大型文件具有较高的效率。
  - `less 要查看的文件`
操作	功能说明
空白键	向下翻动一页
[pagedown]	向下翻动一页
[pageup]	向上翻动一页
/字串	向下搜索[字串]的功能，n：向下查找，N：向上查找
?字串	向上搜索[字串]的功能，n：向上查找，N：向下查找
q	离开less
- **>**指令和**>>**指令
  - `>`输出重定向
  - `>>`追加
  - `ls > 文件`：功能描述:列表的内容写入文件a.txt中（覆盖写）
  - `ls >> 文件`：功能描述:列表的内容追加到文件aa.txt的末尾
  - `cat文件1 > 文件2`：功能描述:将文件1的内容覆盖到文件2
- **echo**指令
  - 输出内容到控制到
  - `echo [选项] [输出内容]`
- **head**指令
  - head用于显示文件的开头部分内容，默认情况下head指令显示文件的前10行内容

    - `head 文件`：查看文件头10行内容
    - `head -n 5 文件`：查看文件头5行内容，5可以是任意行数
- **tail** 指令
    - tail 用于输出文件中尾部的内容，默认情况下tail指令显示文件的后10行内容。
    - `tail 文件`：查看文件后10行内容
    - `tail -n 5 文件`：查看文件后5行内容，5可以是任意行数
    - `tail -f 文件`：实时追踪该文档的所有更新
- **ln**指令
    - 软链接也叫符号链接，类似于windows里的快捷方式，主要存放了链接其他文件的路径
    - `ln -s [原文件或目录] [软连接名]`：给原文件创建一个软链接
- **history**指令
    - 查看已经执行过历史命令，也可以执行历史指令
    - `history`：查看已经执行过的历史命令
    - `history 10`：查看最近执行的10条历史命令
    - `!10`：执行历史编号为10的指令

时间日期类

- **date**指令
    - 显示当前日期
    - `date`：显示当前时间
    - `date +%Y`：显示当前年份
    - `date +%m`：显示当前月份
    - `date +%d`：显示当前是哪一天
    - `date "+%Y-%m-%d %H:%M:%S"`：显示年月日时分秒
    - 设置系统时间
    - `date -s 字符串时间`
- **cal**指令
    - 查看日历信息
    - `cal [选项]`

搜索查找类

- **find**指令
  - find指令将从指定目录向下递归地遍历其各个子目录，将满足条件的文件或者目录显示在终端。
  - `find [搜索范围] [选项]`
选项	功能
-name <查询方式>	按照指定的文件名查找模式查找文件，也可以使用通配符查找
-user <用户名>	查找属于指定用户所有的文件
-size <文件大小>	按照指定的文件大小查找文件（+n大于,-小于n，n等于）
- **locate**指令
  - locaate指令可以快速定位文件路径。locate指令利用事先建立的系统中所有文件名称及路径的locate数据库实现快速定位给定的文件。Locate指令无需遍历整个文件系统，查询速度较快。为了保证查询结果的准确度，管理员必须定期更新locate时刻。
  - `locate 搜索文件`
  - 由于locate指令基于数据库进行查询，所以第一次运行前，必须使用updatedb指令创建locate数据库。
- **grep**指令 和 |管道符号
  - grep过滤查找
    - `grep [选项] 查找内容 原文件`
选项	功能
-n	显示匹配行及行号
-i	忽略字母大小写
  - 管道符，“|"，表示将前一个命令的处理结果输出传递给后面的命令处理。

压缩和解压缩类

- **gzip/gunzip**指令
  - gzip用于压缩文件，gunzip用于解压文件
  - `gzip 文件`：压缩文件，只能将文件压缩为.gz文件
  - `gunzip文件.gz`：解压缩文件命令
- **zip/unzip**指令
  - zip用于压缩文件，unzip用于解压的，这个在项目打包发布中很有用的

    - `zip [选项] 名称.zip file(将要压缩的目录或内容)`：压缩文件和目录的命令
        - -r：递归压缩，及压缩目录
    - `unzip [选项] 名称.zip`：解压缩文件
        - -d<目录>：指定解压后文件的存放目录
- **tar**指令
    - tar指令是打包指令，最后打包后的文件是.tar.gz的文件。
    - `tar [选项] 名称.tar.gz 打包的内容` ：打包目录，压缩后的文件格式为.tar.gz
选项	功能
-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	打包同时压缩
-x	解包.tar文件

组管理

> 在linux中的每个用户必须属于一个组，不能独立于组外。在linux中每个文件有所有者、所在组、其它组的概念。
>
> - 所有者
>   - 一般为文件的创建者,谁创建了该文件，就自然的成为该文件的所有者。
> - 所在组
> - 其它组
>   - 除文件的所有者和所在组的用户外，系统的某它用户都是文件的其它组。
> - 改变用户所在的组

- 查看文件的所有者/目录所在组
    - 指令：`ls -ahl`
- 修改文件所有者
    - 指令：`chown 用户名 文件名`
    - 同时更改所有者以及所有组
        - `chown newowner:newgroup file`
    - -R 如果是目录，则使其下所有文件或目录递归生效
- 组的创建

    - `addgroup 组名`
    - `groupadd 组名`
- 修改文件所在组
    - 指令：`chgrp 组名 文件名`
    - -R 如果是目录，则使其下所有文件或目录递归生效
- 修改用户所在组
    - 在添加用户时，可以指定将该用户添加到哪个组中，同样的用root的管理权限可以改变某个用户所在的组。
    - `usermod -g 组名 用户名`
    - `usermod -d 目录名 用户名`：改变该用户登陆的初始目录

权限管理

- 权限基本介绍
    - ls -l中显示的内容如下

```
-rwxrw-r-- 1 root root 1213 七月 2 09:39 abc

按顺序来说：
第0位 - 代表文件的类型
    -：普通文件
    d：目录
    l：软链接
    c：字符设备（键盘，鼠标）
    b：块文件（硬盘）
第1-3位 rwx 代表文件所有者权限
    r：读权限
    w：写权限
    x：可被执行
    -：无权限
第4-6位 rw- 代表文件所在组的用户的权限
    r：读权限
    w：写权限
    -：无x权限
第7-9位个 r-- 代表文件其他组的用户的权限
    r：读权限
    -：无w权限
    -：无x权限
第10位 1 代表
    文件：1
    目录：目录中子目录的个数
第11个 root 代表用户名
```

```
第12个 root 代表用户所在组
第13个 1213 代表文件的大小，如果是目录则显示4096
第14个 七月 2 09:39 代表文件最后修改的时间
第15个 abc 代表文件名称
```

- 第0位确定文件类型(d,-,l,c,b)
- 第1-3位确定所有者（该文件的所有者)拥有该文件的权限。---User
- 第4-6位确定所属组（同用户组的)拥有该文件的权限，---Group
- 第7-9位确定其他用户拥有该文件的权限---Other
- rwx权限详解
  - rwx可用数字表示：r=4，w=2，x=1，rwx=7
  - rwx作用到文件
  - [r]代表可读(read)：可以读取,查看
  - [w]代表可写(write)：可以修改,但是不代表可以删除该文件,删除一个文件的前提条件是对该文件所在的目录有写权限，才能删除该文件.
  - [x]代表可执行(execute)：可以被执行
  - rwx作用到目录
    - [r]代表可读(read)：可以读取，ls查看目录内容
    - [w]代表可写(write)：可以修改,目录内创建+删除+重命名目录
    - [x]代表可执行(execute)：可以进入该目录
- 权限的管理
  - 通过chmod指令，可以修改文件或者目录的权限。
  - 第一种方式：+、-、=变更权限
    - u:所有者，g：所有组，o：其他人，a：所有人(u、g、o的总和)
    - `chmod u=rwx,g=rx,o=x 文件目录名`
    - `chmod o+w 文件目录名`
    - `chmod a-x 文件目录名`
  - 第二种方式：通过数字更改权限
    - r=4，w=2，x=1（rwx=4+2+1=7）
    - `chmod 751 文件目录名`

crond任务调度

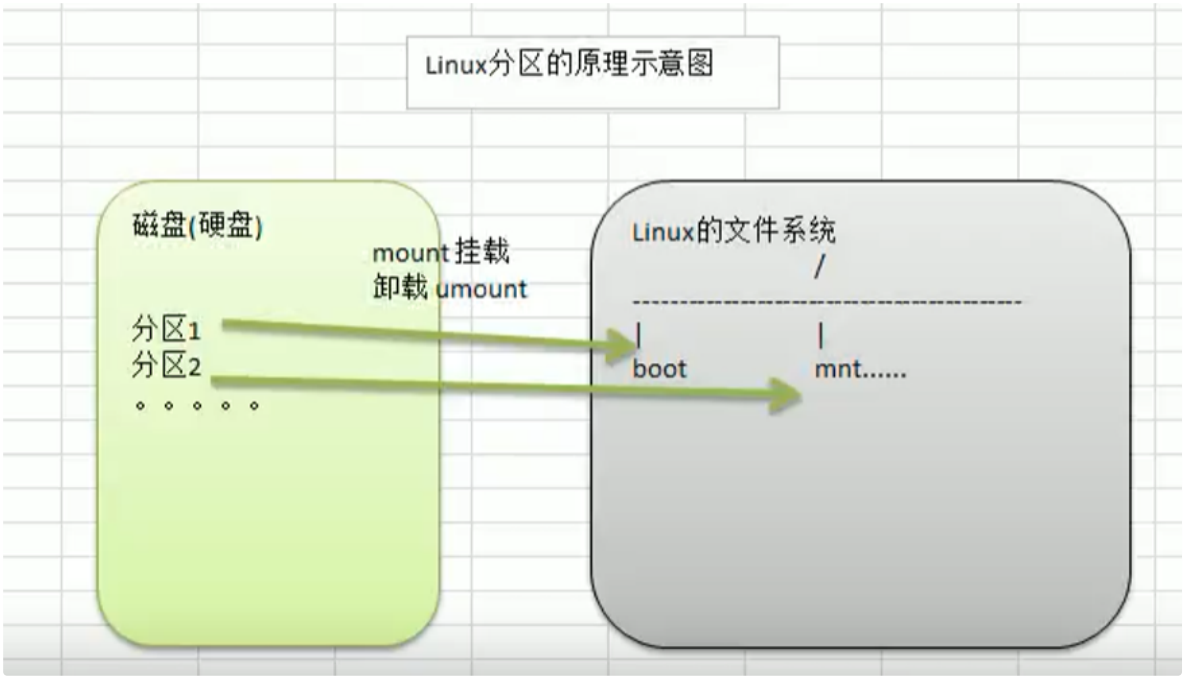
- 任务调度:是指系统在某个时间执行的特定的命令或程序。
- 任务调度分类:
  - 1.系统工作，有些重要的工作必须周而复始地执行。如病毒扫描等

    - 2.个别用户工作。个别用户可能希望执行某些程序，比如对mysql数据库定时的备份。
- crontab [选项]
选项	功能
-e	编辑crontab定时任务
-l	查询crontab任务
-r	删除当前用户所有的crontab任务
- 例如

```
*/1 * * * * ls -l /etc/ >> /tmp/to.txt
# 每1min执行一次将/etc/列表添加到/tmp/to.txt

# 格式如下:
*     *     *     *     *
-     -     -     -     -
|     |     |     |     |
|     |     |     |     +------ 星期中星期几 (0 - 7) (星期天 为0)
|     |     |     +------------ 月份 (1 - 12)
|     |     +------------------ 一个月中的第几天 (1 - 31)
|     +------------------------ 小时 (0 - 23)
+-------------------------------分钟 (0 - 59)

* * * * *     每分钟执行
*/1 * * * *   每1分钟执行
0 * * * *     每小时执行
0 0 * * *     每天执行
0 0 * * 0     每周执行
0 0 1 * *     每月执行
0 0 1 1 *     每年执行
1 * * * *     每小时的第1分钟执行
```

磁盘分区与挂载

- 分区基础知识
    - mbr份区:
        - 最多支持四个主分区
        - 系统只能安装在主分区
        - 扩展分区要占一个主分区
        - MBR最大只支持2TB，但拥有最好的兼容性

    - gtp分区:
        - 支持无限多个主分区（但操作系统可能限制，比如windows下最多128个分区）
        - 最大支持18EB的大容量(EB=1024 PB，PB=1024 TB)
        - windows7 64位以后支持gtp
- Linux分区
    - `lsblk -f | lsblk`：查看系统的分区和挂载的情况
    - Linux来说无论有几个分区，分给哪一目录使用，它归根结底就只有一个根目录，一个独立且唯一生文件结构，Linux中每个分区都是月来组成整个文件系统的一部分。
    - Linux采用了一和叫“载入”的处理方法，它的整个文件系统口包含了一整套的文件和目录，且将一个分区和一个目录联系起来。这时要载入的一个分区将使它的存诸空间在一个目录下获得。
    - 

- 硬盘说明
    - Linux硬盘分IDE硬盘和SCSI硬盘，目前基本上是SCSI硬盘
    - 对于IDE硬盘，驱动器标识符为“hdx~",其中“hd"表明分区所在设备的类型，这里是指IDE硬盘了。“x"为盘号（a为基本盘，b为基本从属盘，c为辅助主盘，d为辅助从属盘），“~代表分区，前四个分区用数字1到4表示，它们是主分区或扩展分区，从5开始就是逻辑分区。例，hda3表示为第一个IDE硬盘上的第三个主分区或扩展分区,hdb2表示为第三个IDE硬盘上的第二个主分区或扩展分区。
    - 对于SCSI硬盘则标识为“sdx~”，SCSI硬盘是用“sd"来表示分区所在设备的类型的，其余则和IDE硬盘的表示方法一样。

如何增加一块硬盘

- 虚拟机添加硬盘
- 分区：`fdisk /dev/sdb1(分区目录)`：对硬盘进行分区
- 格式化硬盘：`mkfs -t ext4 /dev/sdb1(分区目录)`
- 挂载：
  - 先创建一个目录 /home/newdisk
  - 挂载：`mount /dev/sdb1(分区目录) /home/newdisk(挂载目录)`
- 自动挂载
  - 打开一个文件：`vim /etc/fstab`
  -
```
打开一个文件
vim /etc/fstab

将需要挂载的硬盘添加到这里

UUID或者硬盘分区的目录如(/devl/slb1)              硬盘分区需要挂载的地址  文件类型，一般是ext4   defaults    0 0
/devl/slb1                                        /home/newdisk       ext4                defaults    0 0

UUID=278ce4a9-406c-4b4d-9d97-cf480e0d218f       /                   ext4                defaults    1 1
UUID=b3ece1f6-9547-4352-b6f5-a36486d09853       /boot               ext4                defaults    1 2
UUID=3320085d-1416-43f9-a9c5-a3758880e4d3       swap                swap                defaults    0 0
tmpfs       /dev/shm        tmpfs   defaults            0 0
devpts      /dev/pts        devpts  gid=5,mode=620      0 0
sysfs       /sys            sysfs   defaults            0 0
proc        /proc           proc    defaults            0 0
```
  - 完成之后，执行`mount -a`生效
- 如何卸载硬盘
  - `umount 设备名称或挂载目录`

磁盘情况查询

- 查询系统整体磁盘管理情况
  - `df -h`
- 查询指定目录的磁盘占用情况
  - `du -h /目录`
  - -s：指定目录占用大小汇总
  - -h：带计量单位
  - -a：带文件
  - -c：列出明细的同时，增加汇总值
  - --max-depth=1：子目录深度
- 统计目录下文件的个数
  - `ls -l /目录 | grep "^-" | wc -l`
- 统计目录下目录的个数

    - `ls -l /目录 | grep "^d" | wc -l`
- 统计目录下文件的个数，包括子目录下的文件
    - `ls -lR /目录 | grep "^-" | wc -l"`
- 统计目录下目录的个数，包括子目录下的目录
    - `ls -lR /目录 | grep "^d" | wc -l`
- 树状显示结构
    - `tree`

进程管理

- 介绍
    - 在Linux中，每个执行的程序（代码）都称为一个进程。每一个进程都分配一个ID号。
    - 每一个进程，都会对应一个父进程，而这个父进程可以复制多个子进程。例如www服务器。
    - 每个进程都可能以两种方式存在的。前台与后台，所谓前台进程就是用户目前的屏幕上可以进行操作的。后台进程则是实际在操作，但由于屏幕上无法看到的进程,通常使用后台方式执行。
    - 一般系统的服务都是以后台进程的方式存在，而且都会常驻在系统中。直到关机才才结束。

显示系统执行的进程

- 查看进程树
    - `pstree [选项]`
- `ps`：用来查看目前系统中，有哪些正在执行，以及它们执行的状况。可以不加任何参数
    -
```
// 显示的信息选项

进程识别号    终端机号      此进程所消CPU时间    正在执行的命令或者进程名
PID          TTY          TIME               CMD
6362         pts/1        00:00:00           bash
6454         pts/1        00:00:00           ps
```
    - -a：显示当前终端的所有进程信息
    - -u：以用户的格式显示进程信息
    - -x：显示后台进程运行的参数
    - -ef：查看父进程

```
ps指令详解：
```

```
指令：ps -aux | grep more

指令说明：
    System V展示风格
    USER：用户名称
    PID：进程号
    %CPU：进程占用CPU的百分比
    %MEM：进程占用的物理内存的百分比
    VSZ：进程占用的虚拟内存大小（单位KB）
    RSS：进程占用的物理内存大小（单位KB）
    TT：终端名称，缩写
    STAT：进程状态，
        S-：睡觉
        s-：表示该进程是会话的先导进程
        N-：表示进程拥有比普通优先级更低的优先级
        R-：正在运行
        D-：短期等待
        Z-：僵死进程
        T-：被跟踪或者被停止等等
    STARTED：进程的启动时间
    TIME：CPU时间，即进程使用CPU的总时间
    COMMAND：启动进程所用的命令和参数，如果时间过长会被截断显示
```

终止进程

- 若是某个进程执行一半需要停止时，或是已消了很大的系统资源时，此时可以考虑停止该进程。使用kill命令来完成此项任务。
- `kill [选项] 进程号`：通过进程号杀死进程
- `killall 进程名称`：通过进程名称杀死进程，也支持通配符，这在系统因负载过大而变得很慢时很有用
- -9：表示强迫进程立即停止

服务管理

- 服务(service)本质就是进程，但是是运行在后台的，通常都会监听某个端口，等待其它程序的请求，比如(mysql , sshd防火墙等)，因此我们又称为守护进程，是Linux中非常重要的知识点。【原理图】
- service管理指令
  - `service 服务名 start | stop | restart | reload | status`
  - 在centos7.0后使用`systemctl`
- 查看服务名
  - 方法1：service --status-all

    - 方法2：/etc/init.d/服务名称
- `chkconfig`指令
    - 通过 *chkconfig* 命令可以给各个运行级别设置自启动/关闭
    - `chkconfig --list | grep xxx`
    - `chkconfig 服务名 --list`
    - `chkconfig --level 5(级别) 服务名 on/off`

监控进程

- 动态监控进程
    - top与ps命令很相似。它们都用来显示正在执行的进程。Top与ps最大的不同之处，在于top在执行一段时间可以更新正在运行的的进程。
    - `top [选项]`
    - -d 秒数：指定top命令每隔几秒更新，默认是3秒在top命令的交互模式当中可以执行的命令
    - -i：使top不是显示任何闲置或者僵死进程
    - -p：通过指定监控进程ID来仅仅监控某个进程的状态
    - 交互操作
    - P：以CPU使用率排序，默认是此项
    - M：以内存的使用率排序
    - N：以PID排序
    - q：退出top
- 监控网络状态
    - `netstat [选项]`：查看系统网络情况
    - -an：按一定顺序排列输出
    - -p：显示哪个进程在调用

shell

第一节 Shell概述

Shell是一个命令行解释器，它接收应用程序/用户命令，然后调用操作系统内核。

Shell还是一个功能相当强大的编程语言，易编写、易调试、灵活性强。

1）Linux提供的Shell解析器有：

```
[atguigu@hadoop101 ~]$ cat /etc/shells

/bin/sh

/bin/bash

/sbin/nologin

/bin/dash

/bin/tcsh

/bin/csh
```

2）bash和sh的关系

```
[atguigu@hadoop101 bin]$ ll | grep bash

-rwxr-xr-x. 1 root root 941880 5月  11 2016 bash

lrwxrwxrwx. 1 root root      4 5月  27 2017 sh -> bash
```

3）Centos默认的解析器是bash

```
[atguigu@hadoop101 bin]$ echo $SHELL

/bin/bash
```

第二节 Shell脚本入门

（1）需求：创建一个Shell脚本，输出helloworld
（2）案例实操：

```
[atguigu@hadoop101 datas]$ touch helloworld.sh

[atguigu@hadoop101 datas]$ vim helloworld.sh



在helloworld.sh中输入如下内容

#!/bin/bash

echo "helloworld"
```

（3）脚本的常用执行方式
第一种：采用bash或sh+脚本的相对路径或绝对路径（不用赋予脚本+x权限）
sh+脚本的相对路径

```
[atguigu@hadoop101 datas]$ sh helloworld.sh

Helloworld
```

sh+脚本的绝对路径

```
[atguigu@hadoop101 datas]$ sh /home/atguigu/datas/helloworld.sh

helloworld
```

bash+脚本的相对路径

```
[atguigu@hadoop101 datas]$ bash helloworld.sh

Helloworld
```

bash+脚本的绝对路径

```
[atguigu@hadoop101 datas]$ bash /home/atguigu/datas/helloworld.sh

Helloworld
```

第二种：采用输入脚本的绝对路径或相对路径执行脚本（必须具有可执行权限+x）

（a）首先要赋予helloworld.sh 脚本的+x权限

```
[atguigu@hadoop101 datas]$ chmod +x helloworld.sh
```

（b）执行脚本

```
相对路径

[atguigu@hadoop101 datas]$ ./helloworld.sh

Helloworld

绝对路径

[atguigu@hadoop101 datas]$ /home/atguigu/datas/helloworld.sh

Helloworld
```

注意：第一种执行方法，本质是bash解析器帮你执行脚本，所以脚本本身不需要执行权限。
第二种执行方法，本质是脚本需要自己执行，所以需要执行权限。
【了解】第三种：在脚本的路径前加上“.”或者 source
（a）有以下脚本

```
[atguigu@hadoop101 datas]$ cat test.sh

#!/bin/bash

A=5

echo $A
```

(b) 分别使用sh，bash，./ 和 . 的方式来执行，结果如下：

```
[atguigu@hadoop101 datas]$ bash test.sh

[atguigu@hadoop101 datas]$ echo $A



[atguigu@hadoop101 datas]$ sh test.sh

[atguigu@hadoop101 datas]$ echo $A
```

```
[atguigu@hadoop101 datas]$ ./test.sh

[atguigu@hadoop101 datas]$ echo $A



[atguigu@hadoop101 datas]$ . test.sh

[atguigu@hadoop101 datas]$ echo $A

5
```

原因：
前三种方式都是在当前shell中打开一个子shell来执行脚本内容，当脚本内容结束，则子shell关闭，回到父shell中。
第四种，也就是使用在脚本路径前加“.”或者 source的方式，可以使脚本内容在当前shell里执行，而无需打开子shell！这也是为什么我们每次要修改完/etc/profile文件以后，需要source一下的原因。
开子shell与不开子shell的区别就在于，环境变量的继承关系，如在子shell中设置的当前变量，父shell是不可见的。

第三节 变量

3.1 系统预定义变量

1）常用系统变量
$HOME、$PWD、$SHELL、$USER等
2）**案例实操**
（1）查看系统变量的值

```
[atguigu@hadoop101 datas]$ echo $HOME

/home/atguigu
```

（2）显示当前Shell中所有变量：set

```
[atguigu@hadoop101 datas]$ set

BASH=/bin/bash

BASH_ALIASES=()

BASH_ARGC=()

BASH_ARGV=()
```

3.2 自定义变量

1）**基本语法**
（1）定义变量：变量名=变量值，注意=号前后不能有空格
（2）撤销变量：unset 变量名
（3）声明静态变量：readonly变量，注意：不能unset
2）**变量定义规则**
（1）变量名称可以由字母、数字和下划线组成，但是不能以数字开头，环境变量名建议大写。
（2）等号两侧不能有空格
（3）在bash中，变量默认类型都是字符串类型，无法直接进行数值运算。
（4）变量的值如果有空格，需要使用双引号或单引号括起来。
3）**案例实操**
（1）定义变量A

```
[atguigu@hadoop101 datas]$ A=5

[atguigu@hadoop101 datas]$ echo $A

5
```

（2）给变量A重新赋值

```
[atguigu@hadoop101 datas]$ A=8

[atguigu@hadoop101 datas]$ echo $A

8
```

（3）撤销变量A

```
[atguigu@hadoop101 datas]$ unset A

[atguigu@hadoop101 datas]$ echo $A
```

（4）声明静态的变量B=2，不能unset

```
[atguigu@hadoop101 datas]$ readonly B=2

[atguigu@hadoop101 datas]$ echo $B

2

[atguigu@hadoop101 datas]$ B=9

-bash: B: readonly variable
```

（5）在bash中，变量默认类型都是字符串类型，无法直接进行数值运算

```
[atguigu@hadoop102 ~]$ C=1+2

[atguigu@hadoop102 ~]$ echo $C

1+2
```

（6）变量的值如果有空格，需要使用双引号或单引号括起来

```
[atguigu@hadoop102 ~]$ D=I love banzhang

-bash: world: command not found

[atguigu@hadoop102 ~]$ D="I love banzhang"

[atguigu@hadoop102 ~]$ echo $D

I love banzhang
```

（7）可把变量提升为全局环境变量，可供其他Shell程序使用
export 变量名

```
[atguigu@hadoop101 datas]$ vim helloworld.sh
```

在helloworld.sh文件中增加echo $B

```
#!/bin/bash



echo "helloworld"

echo $B



[atguigu@hadoop101 datas]$ ./helloworld.sh

Helloworld
```

发现并没有打印输出变量B的值。

```
[atguigu@hadoop101 datas]$ export B

[atguigu@hadoop101 datas]$ ./helloworld.sh

helloworld

2
```

3.3 特殊变量

3.3.1 $n

1）**基本语法**
$n　（功能描述：n为数字，$0代表该脚本名称，$1-$9代表第一到第九个参数，十以上的参数，十以上的参数需要用大括号包含，如${10}）
2）**案例实操**

```
[atguigu@hadoop101 datas]$ touch parameter.sh

[atguigu@hadoop101 datas]$ vim parameter.sh

#!/bin/bash

echo '==========$n=========='

echo $0
```

```
echo $1

echo $2



[atguigu@hadoop101 datas]$ chmod 777 parameter.sh

[atguigu@hadoop101 datas]$ ./parameter.sh cls xz

==========$n==========

./parameter.sh

cls

xz
```

3.3.2 $#

1）**基本语法**
$#　（功能描述：获取所有输入参数个数，常用于循环）。
2）**案例实操**

```
[atguigu@hadoop101 datas]$ vim parameter.sh

#!/bin/bash

echo '==========$n==========='

echo $0

echo $1

echo $2

echo '==========$#==========='

echo $#



[atguigu@hadoop101 datas]$ chmod 777 parameter.sh
```

```
[atguigu@hadoop101 datas]$ ./parameter.sh cls xz

==========$n==========

./parameter.sh

cls

xz

==========$#==========

2
```

3.3.3 $*、$@

1）**基本语法**
$*（功能描述：这个变量代表命令行中所有的参数，$*把所有的参数看成一个整体）
$@ （功能描述：这个变量也代表命令行中所有的参数，不过$@把每个参数区分对待）
2）**案例实操**

```
[atguigu@hadoop101 datas]$ vim parameter.sh

#!/bin/bash

echo '==========$n=========='

echo $0

echo $1

echo $2

echo '==========$#=========='

echo $#

echo '==========$*=========='

echo $*

echo '==========$@=========='

echo $@
```

```
[atguigu@hadoop101 datas]$ ./parameter.sh a b c d e f g

==========$n==========

./parameter.sh

a

b

==========$#==========

7

==========$*==========

a b c d e f g

==========$@==========

a b c d e f g
```

3.3.4 $?

1）**基本语法**
$?　（功能描述：最后一次执行的命令的返回状态。如果这个变量的值为0，证明上一个命令正确执行；如果这个变量的值为非0（具体是哪个数，由命令自己来决定），则证明上一个命令执行不正确了。）
2）**案例实操**
判断helloworld.sh脚本是否正确执行

```
[atguigu@hadoop101 datas]$ ./helloworld.sh

hello world

[atguigu@hadoop101 datas]$ echo $?

0
```

第四节 运算符

- 如何在shell中进行各种运算操作
- 基本语法
  - `"$((运算式))"`或`"${运算式}"`
  - `expr m + n`(运算符之间要有空格)
  - `expr m +-*/% n`：运算符加减乘除，取余
- **案例实操：**

计算（2+3）* 4的值

```
[atguigu@hadoop101 datas]# S=$[(2+3)*4]

[atguigu@hadoop101 datas]# echo $S
```

第五节 条件判断

1）**基本语法**
（1）test condition
（2）[condition]（注意condition前后要有空格）
注意：条件非空即为true，[atguigu]返回true，[ ] 返回false。
2）**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）

- 按照文件权限进行判断

-r 有读的权限（read）
-w 有写的权限（write）
-x 有执行的权限（execute）

- 按照文件类型进行判断

-e 文件存在（existence）
-f 文件存在并且是一个常规的文件（file）
-d 文件存在并且是一个目录（directory）
3）**案例实操**
（1）23是否大于等于22

```
[atguigu@hadoop101 datas]$ [ 23 -ge 22 ]

[atguigu@hadoop101 datas]$ echo $?

0
```

（2）helloworld.sh是否具有写权限

```
[atguigu@hadoop101 datas]$ [ -w helloworld.sh ]

[atguigu@hadoop101 datas]$ echo $?

0
```

（3）/home/atguigu/cls.txt目录中的文件是否存在

```
[atguigu@hadoop101 datas]$ [ -e /home/atguigu/cls.txt ]

[atguigu@hadoop101 datas]$ echo $?

1
```

（4）多条件判断（&& 表示前一条命令执行成功时，才执行后一条命令，|| 表示上一条命令执行失败后，才执行下一条命令）

```
[atguigu@hadoop101 ~]$ [ atguigu ] && echo OK || echo notOK

OK

[atguigu@hadoop101 datas]$ [ ] && echo OK || echo notOK

notOK
```

第六节 流程控制（重点）

6.1 if判断

1）**基本语法**
（1）单分支
if [条件判断式];then
　　程序
fi
或者

if [条件判断式]
then
    程序
fi

（2）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
注意事项：
（1）[条件判断式]，中括号和条件判断式之间必须有空格
（2）if后要有空格
2）**案例实操**
输入一个数字，如果是1，则输出banzhang zhen shuai，如果是2，则输出cls zhen mei，如果是其它，什么也不输出。

```
[atguigu@hadoop101 datas]$ touch if.sh

[atguigu@hadoop101 datas]$ vim if.sh



#!/bin/bash



if [ $1 -eq 1 ]

then

        echo "banzhang zhen shuai"

elif [ $1 -eq 2 ]

then

        echo "cls zhen mei"

fi
```

```
[atguigu@hadoop101 datas]$ chmod 777 if.sh

[atguigu@hadoop101 datas]$ ./if.sh 1

banzhang zhen shuai
```

6.2 case语句

1）**基本语法**
case $变量名 in
"值1"）
　如果变量的值等于值1，则执行程序1
;;
"值2"）
　如果变量的值等于值2，则执行程序2
;;
　…省略其他分支…
*）
　如果变量的值都不是以上的值，则执行此程序
;;
esac
注意事项：
（1）case行尾必须为单词“in”，每一个模式匹配必须以右括号“）”结束。
（2）双分号“;;”表示命令序列结束，相当于java中的break。
（3）最后的“*）”表示默认模式，相当于java中的default。
2）**案例实操**
输入一个数字，如果是1，则输出banzhang，如果是2，则输出cls，如果是其它，输出renyao。

```
[atguigu@hadoop101 datas]$ touch case.sh

[atguigu@hadoop101 datas]$ vim case.sh



!/bin/bash



case $1 in

```

```
"1")
        echo "banzhang"
;;

"2")
        echo "cls"
;;
*)
        echo "renyao"
;;
esac


[atguigu@hadoop101 datas]$ chmod 777 case.sh
[atguigu@hadoop101 datas]$ ./case.sh 1
1
```

6.3 for循环

1）**基本语法1**
for ((初始值;循环控制条件;变量变化))
do
　程序
done
2）**案例实操**
从1加到100

```
[atguigu@hadoop101 datas]$ touch for1.sh
[atguigu@hadoop101 datas]$ vim for1.sh

```

```
#!/bin/bash


sum=0

for((i=0;i<=100;i++))

do

        sum=$[$sum+$i]

done

echo $sum


[atguigu@hadoop101 datas]$ chmod 777 for1.sh

[atguigu@hadoop101 datas]$ ./for1.sh

5050
```

3）**基本语法2**
for 变量 in 值1 值2 值3…
do
　程序
done
4）**案例实操**
（1）打印所有输入参数

```
[atguigu@hadoop101 datas]$ touch for2.sh

[atguigu@hadoop101 datas]$ vim for2.sh


#!/bin/bash

#打印数字
```

```
for i in cls mly wls

do

     echo "ban zhang love $i"

done



[atguigu@hadoop101 datas]$ chmod 777 for2.sh

[atguigu@hadoop101 datas]$ ./for2.sh

ban zhang love cls

ban zhang love mly

ban zhang love wls
```

（2）比较$*和$@区别

$*和$@都表示传递给函数或脚本的所有参数，不被双引号“”包含时，都以$1 $2 …$n的形式输出所有参数。

```
[atguigu@hadoop101 datas]$ touch for3.sh

[atguigu@hadoop101 datas]$ vim for3.sh



#!/bin/bash

echo '============$*============'

for i in $*

do

      echo "ban zhang love $i"

done
```

```
echo '============$@============'

for j in $@

do

      echo "ban zhang love $j"

done


[atguigu@hadoop101 datas]$ chmod 777 for3.sh

[atguigu@hadoop101 datas]$ ./for3.sh cls mly wls

============$*============

banzhang love cls

banzhang love mly

banzhang love wls

============$@============

banzhang love cls

banzhang love mly

banzhang love wls
```

当它们被双引号“”包含时，$*会将所有的参数作为一个整体，以“$1 $2 …$n”的形式输出所有参数；$@会将各个参数分开，以“$1” “$2”…“$n”的形式输出所有参数。

```
[atguigu@hadoop101 datas]$ vim for4.sh


#!/bin/bash

echo '============$*============'
```

```
for i in "$*"

#$*中的所有参数看成是一个整体，所以这个for循环只会循环一次

do

        echo "ban zhang love $i"

done



echo '============$@============'

for j in "$@"

#$@中的每个参数都看成是独立的，所以“$@”中有几个参数，就会循环几次

do

        echo "ban zhang love $j"

done



[atguigu@hadoop101 datas]$ chmod 777 for4.sh

[atguigu@hadoop101 datas]$ ./for4.sh cls mly wls

============$*============

banzhang love cls mly wls

============$@============

banzhang love cls

banzhang love mly

banzhang love wls
```

6.4 while循环

1）**基本语法**
while [条件判断式]
do
　程序
done
2**）案例实操**
从1加到100

```
[atguigu@hadoop101 datas]$ touch while.sh

[atguigu@hadoop101 datas]$ vim while.sh



#!/bin/bash

sum=0

i=1

while [ $i -le 100 ]

do

      sum=$[$sum+$i]

      i=$[$i+1]

done



echo $sum



[atguigu@hadoop101 datas]$ chmod 777 while.sh

[atguigu@hadoop101 datas]$ ./while.sh

5050
```

第七节 read读取控制台输入

1）**基本语法**
read（选项）(参数)
选项：
-p：指定读取值时的提示符；
-t：指定读取值时等待的时间（秒）。
参数
　变量：指定读取值的变量名
2）**案例实操**
提示7秒内，读取控制台输入的名称

```
[atguigu@hadoop101 datas]$ touch read.sh

[atguigu@hadoop101 datas]$ vim read.sh



#!/bin/bash



read -t 7 -p "Enter your name in 7 seconds :" NAME

echo $NAME



[atguigu@hadoop101 datas]$ ./read.sh

Enter your name in 7 seconds : atguigu

atguigu
```

第八节 函数

8.1 系统函数

8.1.1 basename

1）**基本语法**
basename [string / pathname] [suffix]　（功能描述：basename命令会删掉所有的前缀包括最后一个（‘/’）字符，然后将字符串显示出来。
选项：
suffix为后缀，如果suffix被指定了，basename会将pathname或string中的suffix去掉。
2）**案例实操**
截取该/home/atguigu/banzhang.txt路径的文件名称

```
[atguigu@hadoop101 datas]$ basename /home/atguigu/banzhang.txt

banzhang.txt

[atguigu@hadoop101 datas]$ basename /home/atguigu/banzhang.txt
.txt

banzhang
```

8.1.2 dirname

1）**基本语法**
　dirname 文件绝对路径　　（功能描述：从给定的包含绝对路径的文件名中去除文件名（非目录的部分），然后返回剩下的路径（目录的部分））
2）**案例实操**
获取banzhang.txt文件的路径

```
[atguigu@hadoop101 ~]$ dirname /home/atguigu/banzhang.txt

/home/atguigu
```

8.2 自定义函数

1）**基本语法**
[function] funname[()]
{
　Action;
　[return int;]
}
2）**经验技巧**
（1）必须在调用函数地方之前，先声明函数，shell脚本是逐行运行。不会像其它语言一样先编译。
（2）函数返回值，只能通过$?系统变量获得，可以显示加：return返回，如果不加，将以最后一条命令运行结果，作为返回值。return后跟数值n(0-255)

3）**案例实操**

计算两个输入参数的和

```
[atguigu@hadoop101 datas]$ touch fun.sh

[atguigu@hadoop101 datas]$ vim fun.sh



#!/bin/bash

function sum()

{

    s=0

    s=$[$1+$2]

    echo "$s"

}



read -p "Please input the number1: " n1;

read -p "Please input the number2: " n2;

sum $n1 $n2;



[atguigu@hadoop101 datas]$ chmod 777 fun.sh

[atguigu@hadoop101 datas]$ ./fun.sh

Please input the number1: 2

Please input the number2: 5

7
```

第九节 Shell工具（重点）

9.1 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
说明：默认分隔符是制表符
2）选项参数说明

选项参数	功能
-f	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	指定具体的字符

3）案例实操
（1）数据准备

```
[atguigu@hadoop101 datas]$ touch cut.txt

[atguigu@hadoop101 datas]$ vim cut.txt

dong shen

guan zhen

wo  wo

lai  lai

le  le
```

（2）切割cut.txt第一列

```
[atguigu@hadoop101 datas]$ cut -d " " -f 1 cut.txt

dong

guan

wo

lai

le
```

（3）切割cut.txt第二、三列

```
[atguigu@hadoop101 datas]$ cut -d " " -f 2,3 cut.txt

shen

zhen

 wo

 lai

 le
```

（4）在cut.txt文件中切割出guan

```
[atguigu@hadoop101 datas]$ cat cut.txt | grep "guan" | cut -d " "
-f 1

guan
```

（5）选取系统PATH变量值，第2个“：”开始后的所有路径：

```
[atguigu@hadoop101 datas]$ echo $PATH

/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/atguigu/.
local/bin:/home/atguigu/bin

[atguigu@hadoop101 datas]$ echo $PATH | cut -d ":" -f 3-

/usr/local/sbin:/usr/sbin:/home/atguigu/.local/bin:/home/atguigu/
bin
```

（6）切割ifconfig 后打印的IP地址

```
[atguigu@hadoop101 datas]$ ifconfig ens33 | grep netmask | cut -d "i" -f 2 | cut -d " " -f 2

192.168.6.101
```

9.2 awk

一个强大的文本分析工具，把文件逐行的读入，以空格为默认分隔符将每行切片，切开的部分再进行分析处理。
1）基本用法
awk [选项参数] ‘/pattern1/{action1} /pattern2/{action2}...’ filename
pattern：表示awk在数据中查找的内容，就是匹配模式
action：在找到匹配内容时所执行的一系列命令
2）**选项参数说明**

选项参数	功能
-F	指定输入文件折分隔符
-v	赋值一个用户定义变量

3）案例实操
（1）数据准备

```
[atguigu@hadoop101 datas]$ sudo cp /etc/passwd ./
```

（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。

```
[atguigu@hadoop101 datas]$ awk -F : '/^root/{print $7}' passwd

/bin/bash
```

（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。

```
[atguigu@hadoop101 datas]$ awk -F : '/^root/{print $1","$7}' passwd

root,/bin/bash
```

注意：只有匹配了pattern的行才会执行action

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

```
[atguigu@hadoop101 datas]$ awk -F : 'BEGIN{print "user, shell"}
{print $1","$7} END{print "dahaige,/bin/zuishuai"}' passwd

user, shell

root,/bin/bash

bin,/sbin/nologin

。。。

atguigu,/bin/bash

dahaige,/bin/zuishuai
```

注意：BEGIN 在所有数据读取行之前执行；END 在所有数据执行之后执行。
（5）将passwd文件中的用户id增加数值1并输出

```
[atguigu@hadoop101 datas]$ awk -v i=1 -F : '{print $3+i}' passwd

1

2

3

4
```

4）awk的内置变量

变量	说明
FILENAME	文件名
NR	已读的记录数（行号）
NF	浏览记录的域的个数（切割后，列的个数）

5）案例实操
（1）统计passwd文件名，每行的行号，每行的列数

```
[atguigu@hadoop101 datas]$ awk -F : '{print "filename:" FILENAME
",linenum:" NR ",col:"NF}' passwd

filename:passwd,linenum:1,col:7

filename:passwd,linenum:2,col:7

filename:passwd,linenum:3,col:7

。。。
```

（2）查询ifconfig命令输出结果中的空行所在的行号

```
[atguigu@hadoop101 datas]$ ifconfig | awk '/^$/{print NR}'

9

18

26
```

（3）切割IP

```
[atguigu@hadoop101 datas]$ ifconfig ens33 | grep netmask | awk -F
"inet" '{print $2}' | awk -F " " '{print $1}'

192.168.6.101
```

9.3 sort

sort命令是在Linux里非常有用，它将文件进行排序，并将排序结果标准输出。
1）基本语法
Sort（选项）(参数)

选项	说明
-n	依照数值的大小排序
-r	以相反的顺序来排序
-t	设置排序时所用的分隔字符
-k	指定需要排序的列

参数：指定待排序的文件列表

2）案例实操
（1）数据准备

```
[atguigu@hadoop101 datas]$ touch sort.txt

[atguigu@hadoop101 datas]$ vim sort.txt

bb:40:5.4

bd:20:4.2

xz:50:2.3

cls:10:3.5

ss:30:1.6
```

（2）按照“：”分割后的第三列倒序排序。

```
[atguigu@hadoop101 datas]$ sort -t : -nrk 3  sort.txt

bb:40:5.4

bd:20:4.2

cls:10:3.5

xz:50:2.3

ss:30:1.6
```

9.4 wc

wc命令用来统计文件信息。利用wc指令我们可以计算文件的行数，字节数、字符数等。
1）基本语法
wc [选项参数] filename

选项参数	功能
-l	统计文件行数
-w	统计文件的单词数
-m	统计文件的字符数
-c	统计文件的字节数

2）案例实操
统计/etc/profile文件的行数、单词数、字节数!

```
[atguigu@hadoop101 datas]$ wc -l /etc/profile

[atguigu@hadoop101 datas]$ wc -w /etc/profile

[atguigu@hadoop101 datas]$ wc -m /etc/profile
```

第十节 正则表达式入门

正则表达式使用单个字符串来描述、匹配一系列符合某个语法规则的字符串。在很多文本编辑器里，正则表达式通常被用来检索、替换那些符合某个模式的文本。在Linux中，grep，sed，awk等命令都支持通过正则表达式进行模式匹配。

10.1 常规匹配

一串不包含特殊字符的正则表达式匹配它自己，例如：

```
[atguigu@hadoop101 datas]$ cat /etc/passwd | grep atguigu
```

就会匹配所有包含atguigu的行

10.2 常用特殊字符

1. 特殊字符：**^**

^ 匹配一行的开头，例如：

```
[atguigu@hadoop101 datas]$ cat /etc/passwd | grep ^a
```

会匹配出所有以a开头的行

2. 特殊字符：**$**

$ 匹配一行的结束，例如

```
[atguigu@hadoop101 datas]$ cat /etc/passwd | grep t$
```

会匹配出所有以t结尾的行
思考：^**$** 匹配什么?

3. 特殊字符：.

. 匹配一个任意的字符，例如

```
[atguigu@hadoop101 datas]$ cat /etc/passwd | grep r..t
```

会匹配包含rabt,rbbt,rxdt,root等的所有行

4. 特殊字符：*

- 不单独使用，他和上一个字符连用，表示匹配上一个字符0次或多次，例如

```
[atguigu@hadoop101 datas]$ cat /etc/passwd | grep ro*t
```

会匹配rt, rot, root, rooot, roooot等所有行
思考：.* 匹配什么?

5. 特殊字符：[ ]

[ ] 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]*------匹配任意长度的数字字符串
[a-z]------匹配一个a-z之间的字符
[a-z]* ------匹配任意长度的字母字符串
[a-c, e-f]-匹配a-c或者e-f之间的任意字符

```
[atguigu@hadoop101 datas]$ cat /etc/passwd | grep r[a,b,c]*t
```

会匹配rt,rat, rbt, rabt, rbact,rabccbaaacbt等等所有行

6. 特殊字符：\

\ 表示转义，并不会单独使用。由于所有特殊字符都有其特定匹配模式，当我们想匹配某一特殊字符本身时（例如，我想找出所有包含 '$' 的行），就会碰到困难。此时我们就要将转义字符和特殊字符连用，来表示特殊字符本身，例如

```
[atguigu@hadoop101 datas]$ cat /etc/passwd | grep a\$b
```

就会匹配所有包含 a$b 的行。