



Linux 系列汇总

Linux 系列汇总

Linux 系统简介

UNIX

Linux

UNIX 和 Linux 的区别

在 VMware 上安装 CentOS

安装 VMware

在 VMware 上安装 CentOS

Linux 入门

Linux 常用命令之文件和目录处理命令

Linux 命令的普遍语法格式

目录处理命令

一、显示目录文件命令：ls

二、创建目录命令：mkdir

三、切换目录命令：cd

四、shell内置命令和外部命令的区别

五、显示当前目录命令：pwd

六、删除空目录命令：rmdir

七、复制文件或目录命令：cp

八、剪切文件或目录命令：mv

九、删除文件或目录命令：rm

文件处理命令

一、创建空文件命令：touch

二、显示文件内容命令（适合内容较少的文件）：cat

三、反向显示文件内容命令（适合内容较少的文件）：tac

四、分页显示文件内容命令（不能向前翻页）：more

五、分页显示文件内容命令（可以前后翻页）：less

六、显示文件内容命令（指定行数）：head

七、反向文件内容命令（文件即时更新后也能动态显示，多用于日志文件显示）：tail

Linux 常用命令之链接命令和权限管理命令

链接命令

一、生成链接文件命令：ln

权限管理命令

一、更改文件或目录权限命令：chmod

二、改变文件或目录所有者命令：chown

三、改变文件或目录所属组命令：chgrp

四、显示、设置文件的缺省权限命令：umask

Linux 常用命令之文件搜索命令

最强大的搜索命令：find

一、根据 文件或目录名称 搜索

二、根据 文件大小 搜索

三、根据 所有者和所属组 搜索



程序员 cxuan



Java 建设者

四、根据时间属性 搜索

五、根据文件类型或i节点 搜索

六、组合条件 搜索

在文件资料库中查找文件命令: locate

搜索命令所在的目录及别名信息: which

搜索命令所在的目录及帮助文档路径: whereis

在文件中搜寻字符串匹配的行并输出: grep

Linux 常用命令之帮助和用户管理命令

帮助命令

一、获得命令或配置文件帮助信息: man

二、获得shell内置命令的帮助信息: help

三、获得命令的中文帮助信息: --help

用户管理命令

一、添加新用户: useradd

二、设置用户密码: passwd

三、查看登录用户简单信息: who

四、查看登录用户详细信息: w

Linux 常用命令之压缩和解压缩命令

压缩解压缩格式 .gz

一、将文件压缩为 .gz 格式,只能压缩文件: gzip

二、将 .gz 文件解压: gunzip

压缩解压缩格式 .tar.gz

一、将文件或目录压缩为 .tar.gz 格式: tar -zcf

二、将 .tar.gz 文件解压: tar -zxf

压缩解压缩格式 .zip

一、将文件或目录压缩为 .zip 格式: zip

二、将 .zip 文件解压: unzip

压缩解压缩格式 .bz2

一、将文件压缩为 .bz2 格式, 只能压缩文件: bzip2

二、将 .bz2 文件解压: bunzip2

Linux 常用命令之网络和关机重启命令

网络命令

一、给指定用户发送信息: write

二、给所有用户发送广播信息: wall

三、测试网络连通性: ping

四、查看和设置网卡信息: ifconfig

五、查看发送电子邮件: mail

六、列出所有登录系统的用户信息: last

七、显示数据包到主机间的路径: traceroute

八、显示网络相关信息: netstat

九、配置网络: setup

十、挂载命令: mount

关机重启命令

一、shutdown命令(推荐使用)

二、其他关机命令

三、其他重启命令

四、退出登录命令



Linux 软件包管理之RPM命令

Linux 软件包分类

一、源码包

二、二进制包

rpm 包命名规则

rpm 包安装

rpm 包升级

rpm 包卸载

查询 rpm 包是否安装

查询软件包的详细信息

查询软件包的安装位置

查询系统文件属于哪个RPM包

查询软件包的依赖性

Linux 软件包管理之yum在线管理

yum 在线管理

网络 yum 源

光盘 yum 源搭建步骤

一、挂载光盘

二、让网络 yum 源失效

三、修改光盘 yum 源文件

四、输入yum list 可以查看光盘yum源里面的软件包

常用的 yum 命令

一、查询所有可用软件包列表: yum list

二、查询服务器上和关键字相关的软件包： yum search 关键字

三、yum 安装软件包： yum -y install 包名

四、yum 升级软件包： yum -y update 包名

五、yum 卸载软件包： yum -y remove 包名

yum 软件组管理

Linux 用户和用户组管理之相关配置文件

用户信息文件： /etc/passwd

用户密码文件： /etc/shadow

用户组信息文件： /etc/group

用户组密码文件： /etc/gshadow

用户的家目录

用户的邮箱

用户的模板目录

Linux 用户和用户组管理之用户管理命令

添加用户命令： useradd

修改用户密码： passwd

修改用户信息： usermod

修改用户密码状态： chage

删除用户命令： userdel

查看用户 id

切换用户身份 su

添加用户组： groupadd

修改用户组： groupmod

删除用户组： groupdel

Linux 权限管理之 ACL 权限

什么是 ACL 权限?

查看分区 ACL 权限是否开启: dump2fs

一、查看当前系统有哪些分区: df -h

二、查看指定分区详细文件信息: dumpe2fs -h 分区路径

开启分区 ACL 权限

一、临时开启分区 ACL 权限

二、永久开启分区 ACL 权限

设定 ACL 权限: setfacl 选项 文件名

一、给用户设定 ACL 权限: setfacl -m u:用户名:权限 指定文件名

二、给用户组设定 ACL 权限: setfacl -m g:组名:权限 指定文件名

查看 ACL 权限: getfacl 文件名

最大有效权限 mask

删除 ACL 权限

递归 ACL 权限

默认 ACL 权限

Linux 权限管理之文件系统系统属性 chattr 权限和 sudo 命令

设定文件系统属性: chattr

查看文件的系统属性: lsattr

sudo 权限

一、超级用户赋予普通用户执行命令权限, 配置 /etc/sudoers 文件

二、授权用户可以重启服务器

三、查看可用的sudo 命令

四、普通用户执行 sudo 赋予的命令

Linux文件系统管理之文件系统常用命令

1、为什么要给硬盘分区?

- ①、易于管理和使用
- ②、有利于数据安全
- ③、节约寻找文件的时间

2、Linux系统分区类型

3、Linux 文件系统的格式

4、文件系统的常用命令

- ①、文件系统查看命令: df
- ②、统计目录或文件大小: du
- ③、文件系统修复命令: fsck
- ④、显示磁盘状态命令: dumpe2fs

5、挂载命令

- ①、查询系统中已经挂载的设备: mount
- ②、依据配置文件 /etc/fstab 的内容自动挂载: mount -a

6、挂载光盘与U盘

- ①、挂载光盘
- ②、挂载 U 盘
- ③、卸载命令

7、支持 NTFS 文件系统

Linux文件系统管理之手工分区

1、添加新硬盘

2、查看新硬盘: fdisk -l

3、使用 fdisk 命令分区

4、分区自动挂载

Linux的shell概述以及如何执行脚本

1、Shell 是什么？

2、Shell 的分类

3、查看Linux系统支持的 shell: /etc/shells

4、echo 输出命令

5、脚本执行方式

①、作为可执行程序

②、作为解释器参数

Linux的bash基本功能

1、历史命令

2、命令与文件补全: Tab

3、命令的别名: alias

4、命令的执行顺序

5、bash 常用快捷键

6、输入输出重定向

①、标准输入输出

②、输出重定向: 将命令执行结果本该显示在屏幕上的存储到别的地方

③、输入重定向: 本该由键盘输入的信息改为由文件进行输入

7、多命令顺序执行

8、管道符 命令1 | 命令2

9、通配符

10、bash 中的其他特殊符号

Linux 的 bash 变量

1、什么是变量

2、变量的声明规则

3、变量的分类

4、用户自定义变量的用法

①、变量定义

②、变量调用

③、变量查看

④、变量删除

5、环境变量的用法

①、通过 pstree 命令区分当前shell 的级别是父还是子

②、声明环境变量

③、查询所有环境变量

④、查看、删除指定环境变量

⑤、系统查找命令的路径环境变量 \$PATH

⑥、定义系统提示符的变量 \$PS1

6、位置参数变量的用法

7、预定义变量的用法

8、声明变量类型 declare

9、数值运算的三种方法

①、declare -i

②、expr 或 let 数值运算工具

③、((运算式))或((运算式))或[运算式]

10、运算符及其优先级顺序

11、变量测试与替换

12、环境变量配置文件

Linux的服务管理

Linux服务管理总览

2、启动和自启动

3、RPM包和源码包服务启动差别根本原因

4、独立服务

5、xinetd 服务

6、源码包服务

Linux 的系统管理

1、进程管理

2、查看系统中的所有进程:ps aux

3、查看系统健康状态: top

4、查看进程数: pstree

5、终止进程:kill

6、将进程放入后台运行

7、查看后台的工作

8、将后台暂停的工作号恢复到前台执行

9、将后台暂停的工作号恢复到后台执行

10、监控系统资源: vmstat

11、查看硬件信息 dmesg

12、查看系统与内核相关信息 uname

13、查看当前系统位数 file

文章来源：IT可乐，欢迎关注公众号 IT可乐

原文链接：<https://www.cnblogs.com/yocean/tag/Linux> 系列教程/

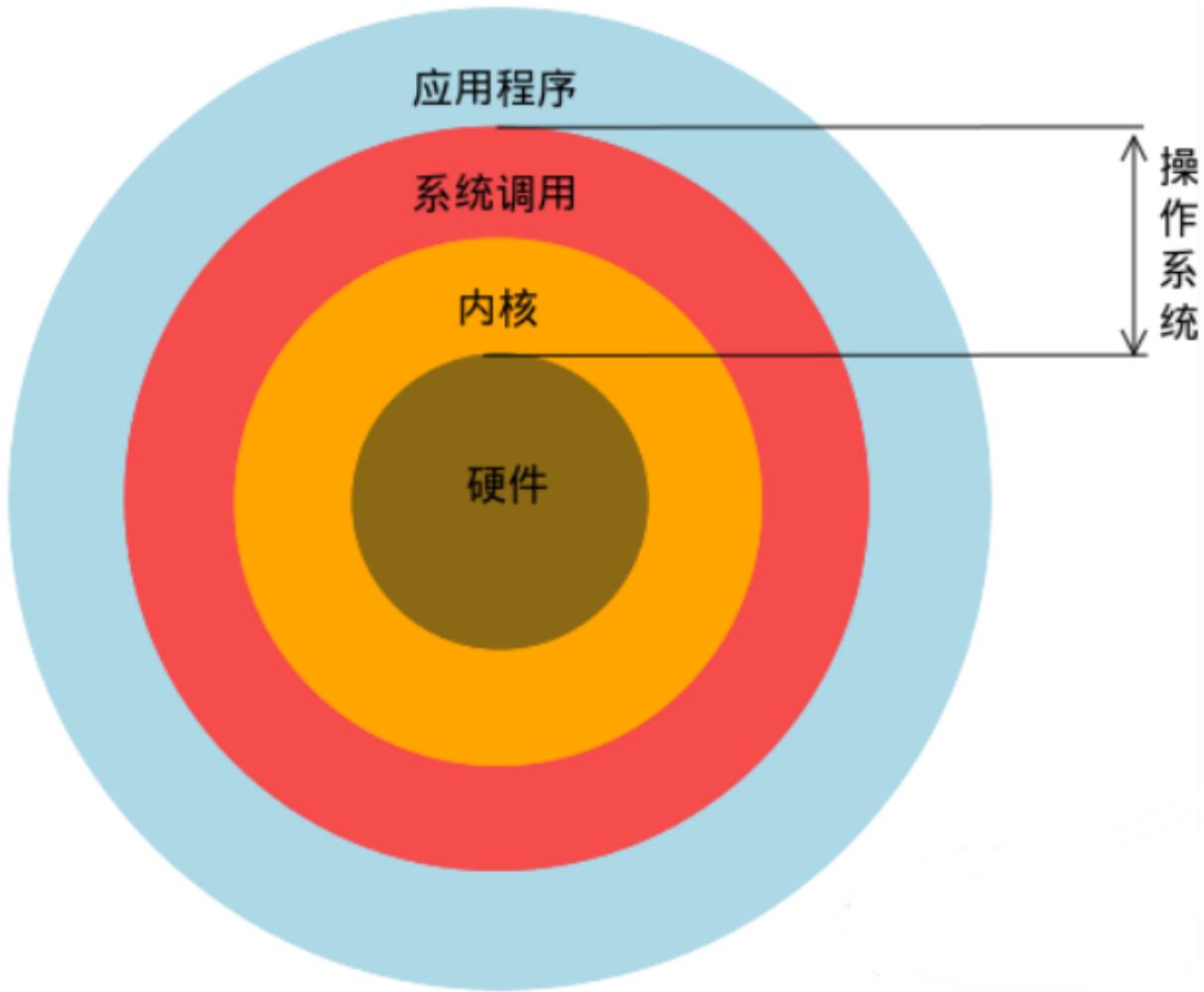
本系列教程将完整的讲解整个 **Linux** 相关的知识，这是楼主学完兄弟连的 Linux 教程之后重新对 Linux 知识体系的整理。个人感觉兄弟连的 Linux 教程可以很好的入门，从最基础的知识开始，对于一个完全不懂 Linux 系统的人，相信在看完整个系列教程之后，都能对 Linux 有一个完完全全的了解。那么废话不多说，本篇博客作为整个教程的第一讲，我们就先来对 Linux 有个简单的整体介绍。

Linux 系统简介

UNIX

可能大家首先看到 Unix 会有点奇怪，我们要讲的不是 Linux 吗？怎么蹦出个 Unix，虽然它和 Linux 长得有点像。因为它们确实有很深的渊源！

Unix 是在 1969 年美国贝尔实验室的 肯.汤普森开发出来的一款操作系统，什么是操作系统？大家正在玩的 Windows 和 Max OS 就是两个操作系统。操作系统是用户和计算机的接口，同时也是计算机硬件和应用程序的接口，也就是说我们和计算机打交道以及计算机底层硬件和应用程序打交道都是通过操作系统。如下所示：



而我们所说的 Unix 也是一个操作系统，其源代码大部分都是用 C 语言写的。它是一个强大的多用户、多任务操作系统，而且支持多种处理器架构。在1984年， Unix 用户协会颁发了使用标准。后来 IEEE 为此制定了 POSIX 标准（即IEEE1003标准）国际标准名称为 ISO/IEC9945，它通过一组最小的功能定义了在 UNIX 操作系统和应用程序之间兼容的语言接口。这个标准很重要，后面很多系统的开发都是遵循这个标准来的。

虽然 Unix 系统这么好用，但是很不幸，它是一个对源代码实行知识产权保护的传统商业软件，也就是说 Unix 系统源代码不开源，而且 Unix 系统也是一个收费软件。这也直接导致了 Linux 系统的诞生。

Linux

UNIX 最初免费发布，在1990年， UNIX 在服务器市场尤其是大学校园成为主流操作系统，许多校园都有 UNIX 主机，当然还包括一些研究它的计算机系的学生。这些学生都渴望能在自己的电脑上运行 UNIX 。不幸的是，从那时候开始， UNIX 开始变得商业化，它的价格也变得非常昂贵。而唯一低廉的选择就是 MINIX ，这是一个功能有限的类似 UNIX 的操作系统，作者 Andrew Tanenbaum 开发它的目的是用于教学。

1991 年 10 月， Linus Torvalds (Linux 之父) 在赫尔辛基大学接触 UNIX ,他希望能在自己的电脑上运行一个类似的操作系统。可是 UNIX 的商业版本非常昂贵，于是他从 MINIX 开始入手，而 Linus Torvalds 对 Minix 不是很满意，于是决定自己编写软件。他以学生时代熟悉的 Unix 作为原型，在一台 Intel 386 PC 上开始了他的工作。他的进展很快，受工作成绩的鼓舞，他将这项成果通过互连网与其他同学共享，主要用于学术领域。他第一次发行的版本很快吸引了一些黑客。尽管最初的 Linux 并没有多少用处，但由于一些黑客的加入使它很快就具有了许多吸引人的特性，甚至一些对操作系统开发不感兴趣

趣的人也开始关注它。每当出现新问题时，有人会立刻找到解决办法并加入其中，很快的，Linux 成为了一个操作系统。值得注意的是 Linux 并没有包括 Unix 源码，它是按照公开的 POSIX 标准重新编写的。Linux 大量使用了由麻省剑桥免费软件基金的 GNU 软件，同时 Linux 自身也是用它们构造而成。



UNIX 和 Linux 的区别

Linux 和 UNIX 的最大的区别是，前者是开放源代码的自由软件，而后者是对源代码实行知识产权保护的传统商业软件。这应该是他们最大的不同，这种不同体现在用户对前者有很高的自主权，而对后者却只能去被动的适应。这种不同还表现在前者的开发是处在一个完全开放的环境之中，而后的开发完全是在一个黑箱之中，只有相关的开发人员才能够接触的产品的原型。具体区别如下：

- UNIX 系统大多是与硬件配套的，而 Linux 则可运行在多种硬件平台上。
- UNIX 是商业软件，收费，而 Linux 是自由软件，免费、公开源代码的。
- Linux 商业化的有 RedHat Linux、SuSe Linux、slackware Linux、国内的红旗等，还有 Turbo Linux。
- Unix 主要有 Sun 的 Solaris、IBM 的 AIX、HP 的 HP-UX，以及 x86 平台的 SCO Unix/Unixware。

下面我们讲解一下 Linux 的搭建过程

在 VMware 上安装 CentOS

注意这里我们选择安装的 Linux 系统是其一种发行版本 CentOS，这里给大家普及一个概念，Linux有非常多的发行版本，从性质上划分，大体分为由商业公司维护的商业版本与由开源社区维护的免费发行版本。商业版本以Redhat为代表，开源社区版本则以debian为代表。下面是三个典型的发行版：

- CentOS是从redhat源代码编译重新发布版。CentOS去除很多与服务器功能无关的应用，系统简单但非常稳定，命令行操作可以方便管理系统和应用，并且有帮助文档和社区的支持。一般新手入门比较好。
- Ubuntu有亮丽的用户界面，完善的包管理系统，强大的软件源支持，丰富的技术社区，并且Ubuntu对计算机硬件的支持好于CentOS和Debian，兼容性强，Ubuntu应用非常多。但是我们需要注意的是图形界面占用的内存非常大。
- Debian也非常适合做服务器操作系统，与Ubuntu比较，它没有太多的花哨，稳定压倒一切，对于服务器系统来说是一条不变的真理，Debian这个linux系统，底层非常稳定，内核和内存的占用都非常小。

我们虽然选择学习的是 CentOS，但是大家也不要在意，基本上学会一个系统的命令后，其余的系统都大同小异。我们是将 CentOS 安装在虚拟 PC，这里模拟虚拟 PC 的软件选择 VMware，所以这里安装的步骤分为两步：

第一步：安装虚拟软件 VMware，百度云盘下载链接：https://pan.baidu.com/s/1l3Eqk_qWbDyfTMd-ZNQI-g 提取码: cmii

第二步：安装虚拟机 CentOS 6.8百度云盘下载地址：<https://pan.baidu.com/s/1MXTWPhpF7b2AzxqGVniy9A> 提取码: xftk

安装 VMware

VMware 是一个虚拟 PC 的软件，可以在现有的操作系统上虚拟出一个新的硬件环境，相当于模拟出一台新的 PC，我们可以在上面构造出一个或多个别的系统，以此来实现一台机器上真正同时运行多个独立的操作系统。VMware 下载链接上面给出了，安装过程都是默认下一步，下一步即可。这里就不给出安装的图示了，安装完成后，双击打开如下：



在 VMware 上安装 CentOS

第 1 步：打开 VMware，点击创建新的虚拟机



欢迎使用新建虚拟机向导

您希望使用什么类型的配置？

典型(推荐)(T)

通过几个简单的步骤创建 Workstation
12.0 虚拟机。

自定义(高级)(C)

创建带有 SCSI 控制器类型、虚拟磁盘类
型以及与旧版 VMware 产品兼容性等高
级选项的虚拟机。

帮助

< 上一步(B)

下一步(N) >

取消

第 2 步：选择典型，点击下一步。出现如下界面，然后选择第三个选项：稍后安装操作系统，点击下一
步

新建虚拟机向导



安装客户机操作系统

虚拟机如同物理机，需要操作系统。您将如何安装客户机操作系统？

安装来源：

安装程序光盘(D):

无可用驱动器

安装程序光盘映像文件(iso)(M):

D:\虚拟机\CentOS-6.8-x86_64-bin-DVD1to2\CentOS-6

[浏览\(R\)...](#)

稍后安装操作系统(S)。

创建的虚拟机将包含一个空白硬盘。

[帮助](#)

[< 上一步\(B\)](#)

[下一步\(N\) >](#)

[取消](#)

第3步：客户机安装操作系统选择Linux,版本根据自己下载的Linux镜像文件来选择，这里我们选择CentOS 64位。然后点击下一步

新建虚拟机向导



选择客户机操作系统

此虚拟机中将安装哪种操作系统?

客户机操作系统

- Microsoft Windows(W)
- Linux(L)
- Novell NetWare(E)
- Solaris(S)
- VMware ESX(X)
- 其他(O)

版本(V)

CentOS 64 位

帮助

< 上一步(B)

下一步(N) >

取消

第 4 步：给虚拟机命名，以及选择虚拟机安装的位置，最好是非中文不含空格的地址。然后点击下一步

新建虚拟机向导

×

命名虚拟机

您要为此虚拟机使用什么名称?

虚拟机名称(V):

Node2

位置(L):

F:\VMSystem\Node2

浏览(R)...

在“编辑”>“首选项”中可更改默认位置。

< 上一步(B)

下一步(N) >

取消

第 5 步：默认，磁盘大小 20 GB 足够。然后点击下一步

新建虚拟机向导

X

指定磁盘容量

磁盘大小为多少？

虚拟机的硬盘作为一个或多个文件存储在主机的物理磁盘中。这些文件最初很小，随着您向虚拟机中添加应用程序、文件和数据而逐渐变大。

最大磁盘大小(GB)(S):

20.0



针对 CentOS 64 位 的建议大小: 20 GB

将虚拟磁盘存储为单个文件(O)

将虚拟磁盘拆分成多个文件(M)

拆分磁盘后，可以更轻松地在计算机之间移动虚拟机，但可能会降低大容量磁盘的性能。

帮助

< 上一步(B)

下一步(N) >

取消

第 6 步：点击完成按钮

新建虚拟机向导

×

已准备好创建虚拟机

单击“完成”创建虚拟机。然后可以安装 CentOS 64 位。

将使用下列设置创建虚拟机：

名称: Node2
位置: F:\VMSystem\Node2
版本: Workstation 12.0
操作系统: CentOS 64 位

硬盘: 20 GB, 拆分
内存: 1024 MB
网络适配器: NAT
其他设备: CD/DVD, USB 控制器, 打印机, 声卡

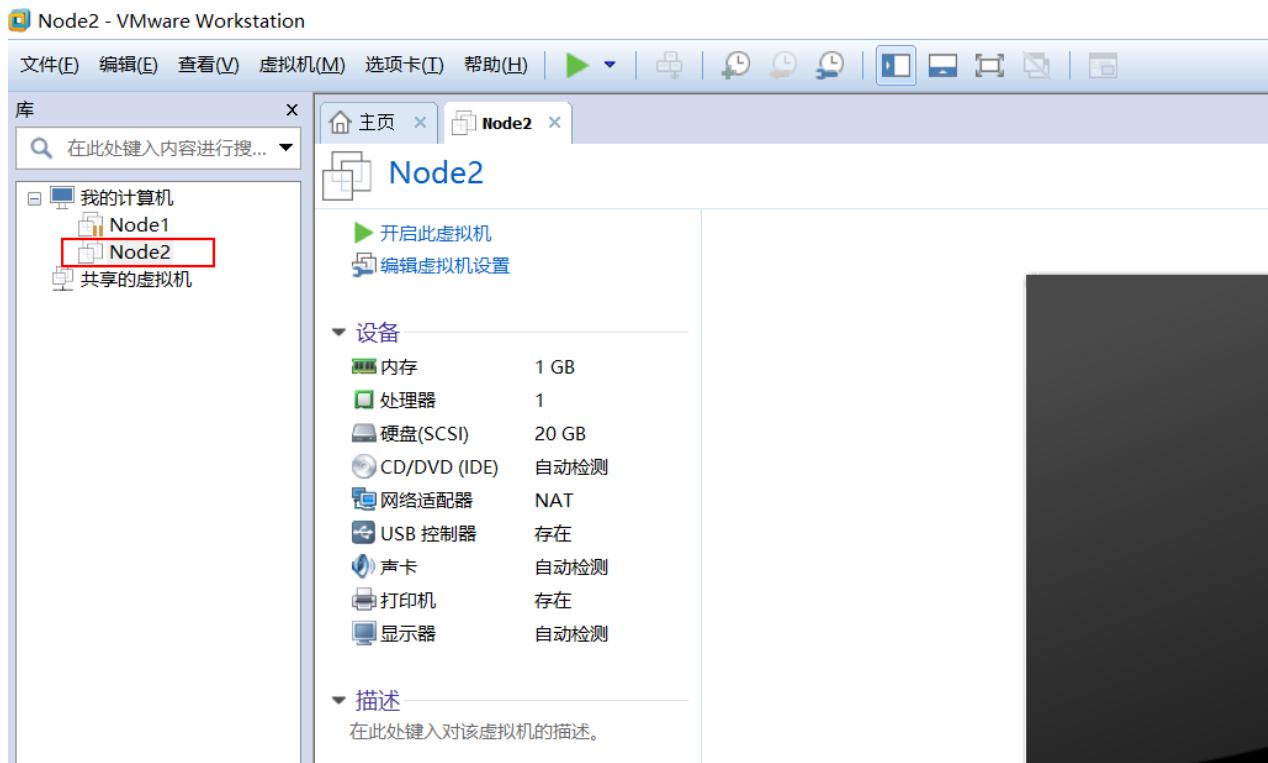
自定义硬件(C)...

< 上一步(B)

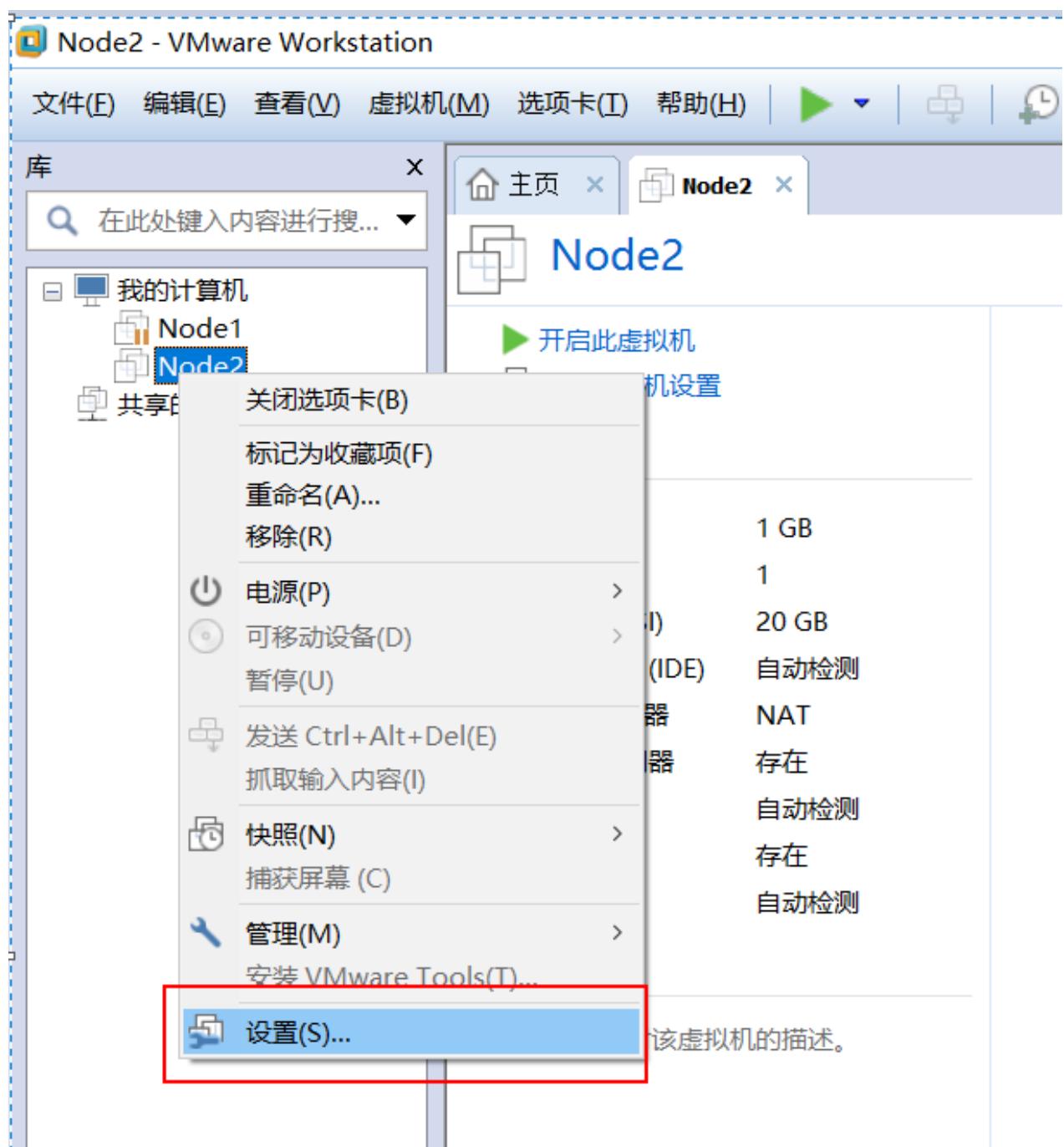
完成

取消

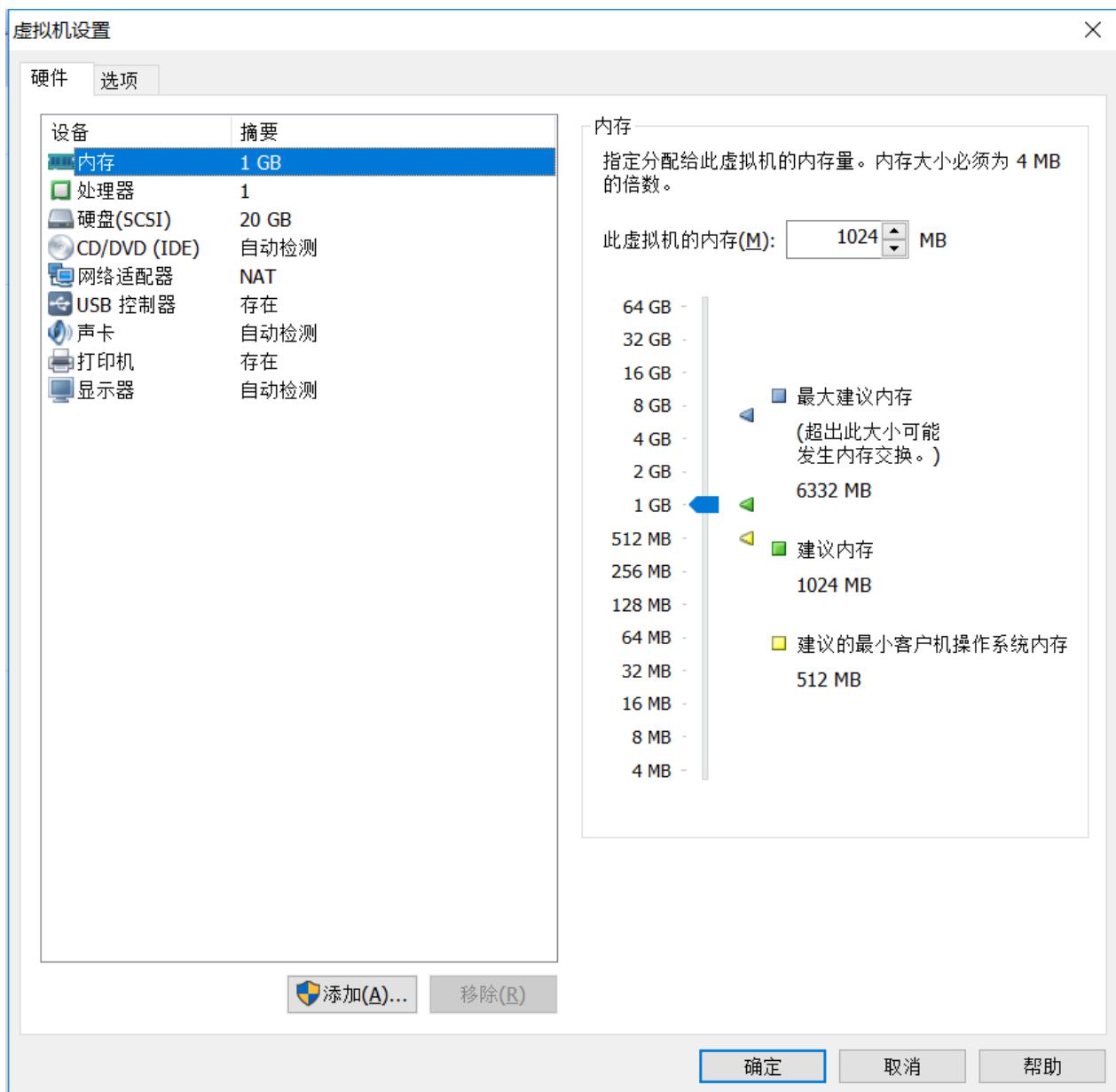
点击完成之后，我们会在 VMware 主界面发现我们刚刚创建的 虚拟机 Node2：



第 7 步：右键虚拟机名称 Node2，选择设置



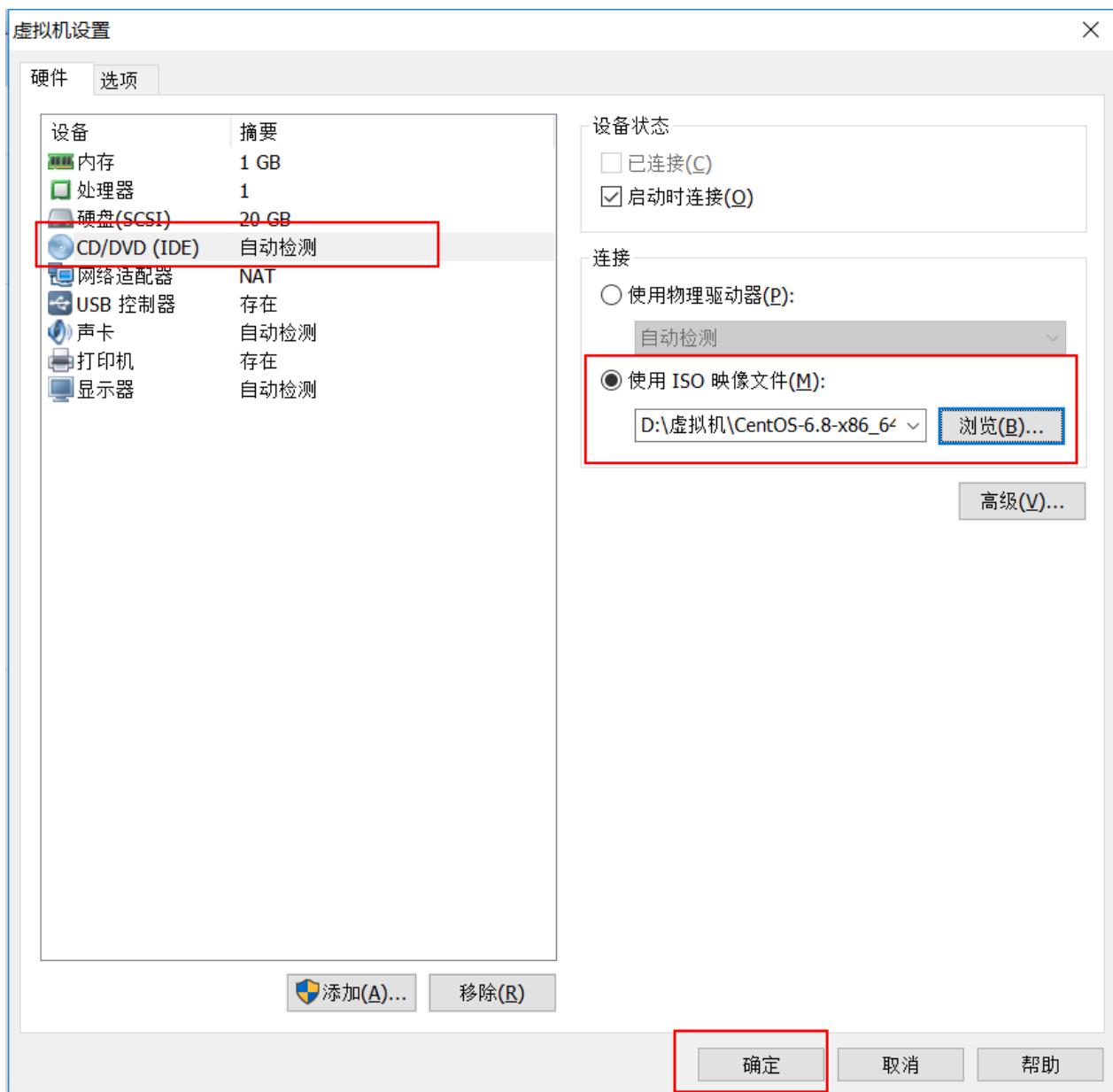
弹出如下界面：



第 8 步：进行虚拟机的各项硬件配置

首先调整内存大小，一般来讲，内存最低不能小于 628 m，最大不能超过 真实物理机内存的一半。我们可以默认选择 1024M 内存大小就够了

其次我们可以对处理器进行调整，对硬盘大小也可以进行调整，但是这里我们都选择默认就行了，直接跳到 CD/DVD(IDE) 选项，选择我们下载的 CentOS 镜像文件，然后点击确定



注意：我们这里下载的 CentOS 6.8 有两个ISO 文件，我们选择第一个，第一个包含 CentOS 系统的主体信息，而第二个 ISO 文件只是一些不常用的软件信息



东版

那么经过上面的步骤，我们对于虚拟机的配置已经设置完成，接下来就进行虚拟机的安装。

第 9 步：到 VMware 主界面，选择 我们安装的 Node2 虚拟机，点击 开启此虚拟机

Node2

▶ 开启此虚拟机

编辑虚拟机设置

设备

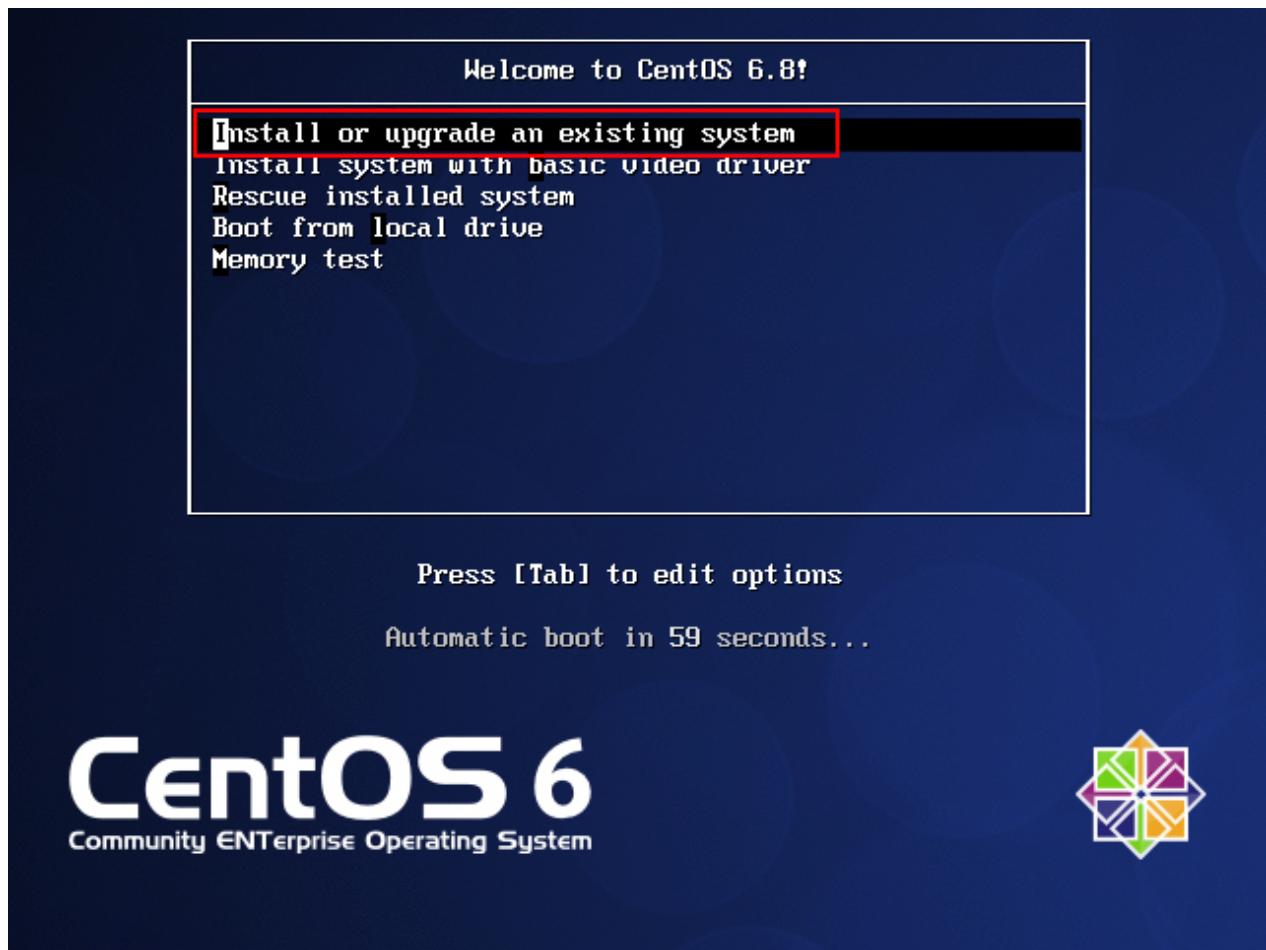
内存	1 GB
处理器	1
硬盘(SCSI)	20 GB
CD/DVD (IDE)	正在使用文件 D:...
网络适配器	NAT
USB 控制器	存在
声卡	自动检测
打印机	存在
显示器	自动检测

描述

在此处键入对该虚拟机的描述。

第 10 步：进入欢迎界面后，选择 第一个选项：Install or upgrade an existing system，然后 Enter。

注意：我们鼠标进入安装界面后，需要按 Ctrl+Alt 才能使得鼠标恢复到自己的主系统。



其余几个选项意思是：

- ◆ “Install or upgrade an existing system”：安装或升级现有系统
- ◆ “Install system with basic video driver”：安装过程采用基本的显卡驱动
- ◆ “Rescue installed system”：进入系统修复模式
- ◆ “Boot from local drive”：退出安装从硬盘启动
- ◆ “Memory test”：存储介质检测

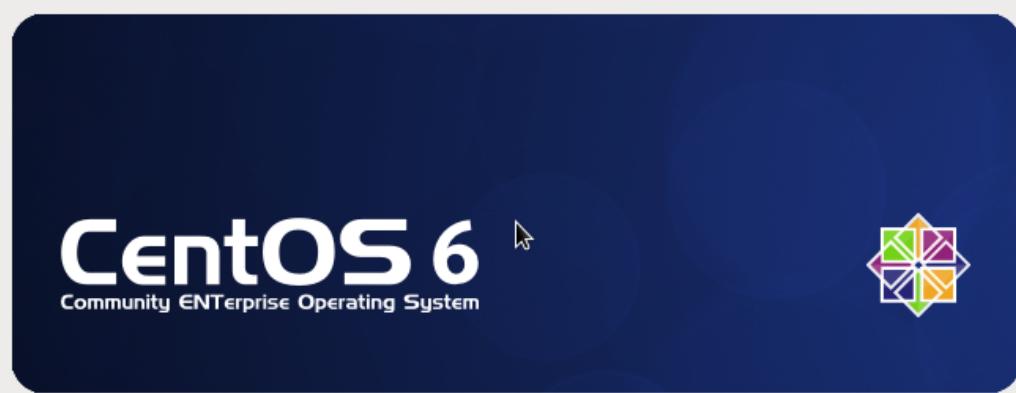
第 11 步：询问我们是否需要检查光盘，我们按右箭头，选择 skip 选项（即不用检测），Enter

Welcome to CentOS for x86_64



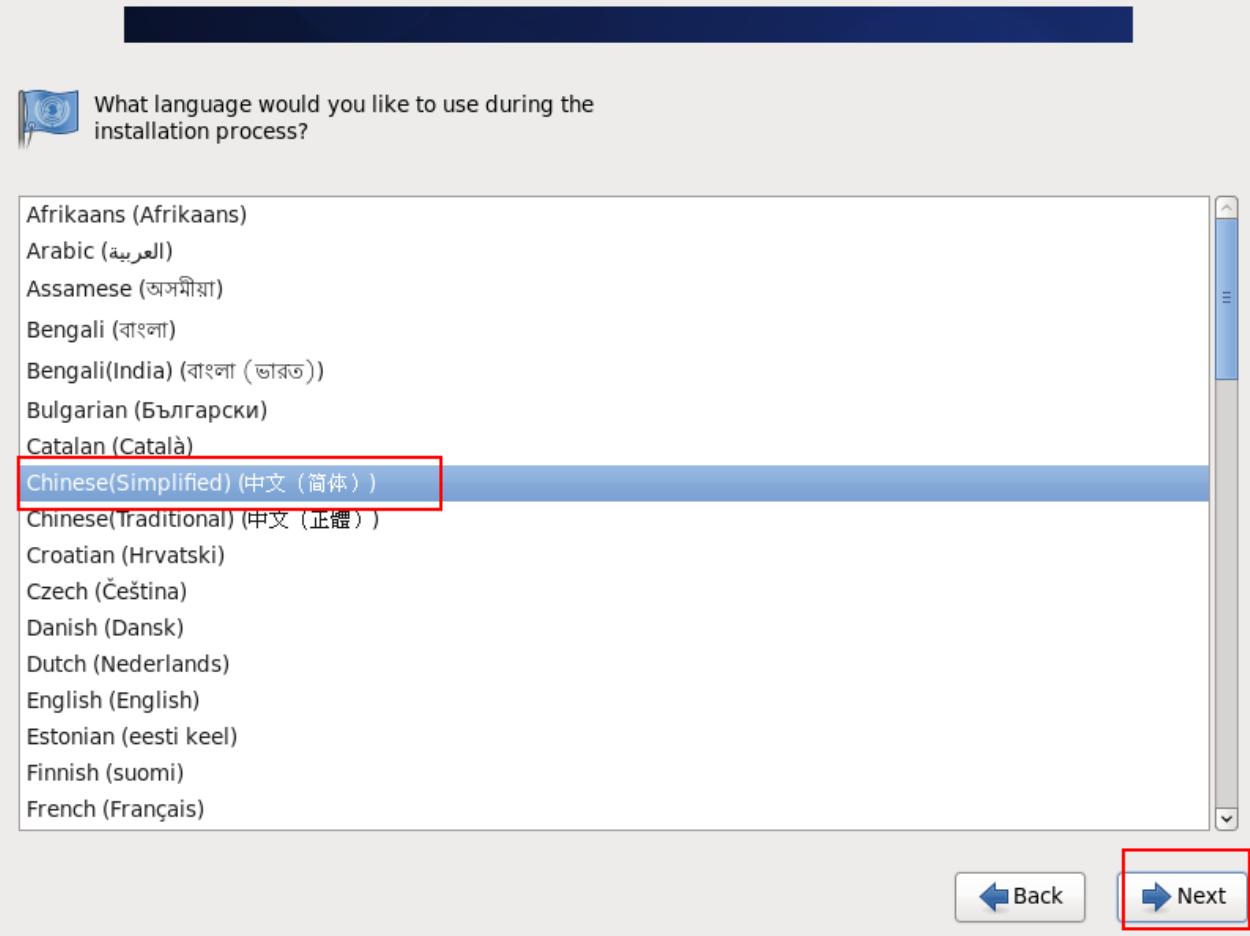
<Tab>/<Alt-Tab> between elements | <Space> selects | <F12> next screen

第 12 步：点击 next



Back Next

第 13 步：选择 中文简体，点击 Next

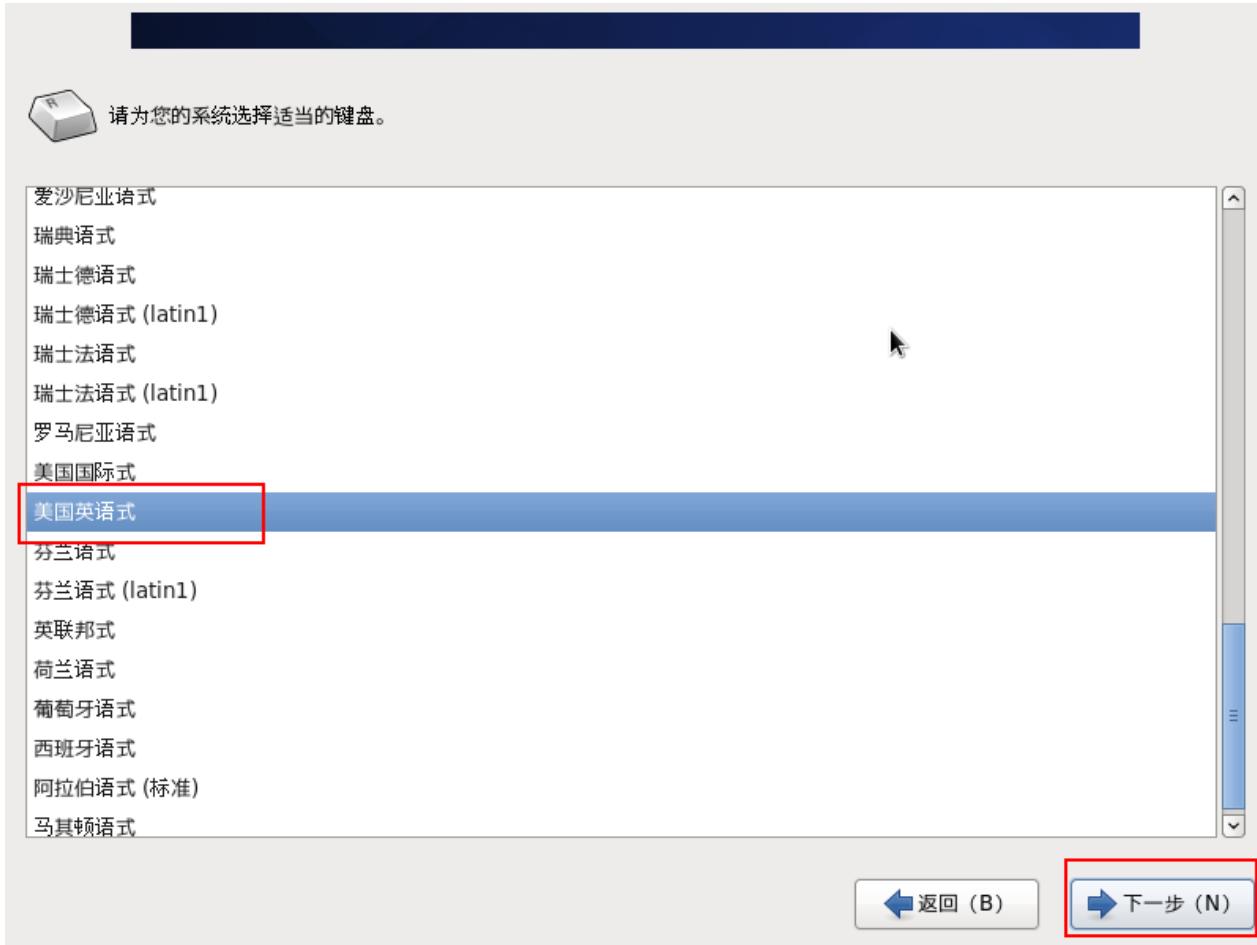


What language would you like to use during the installation process?

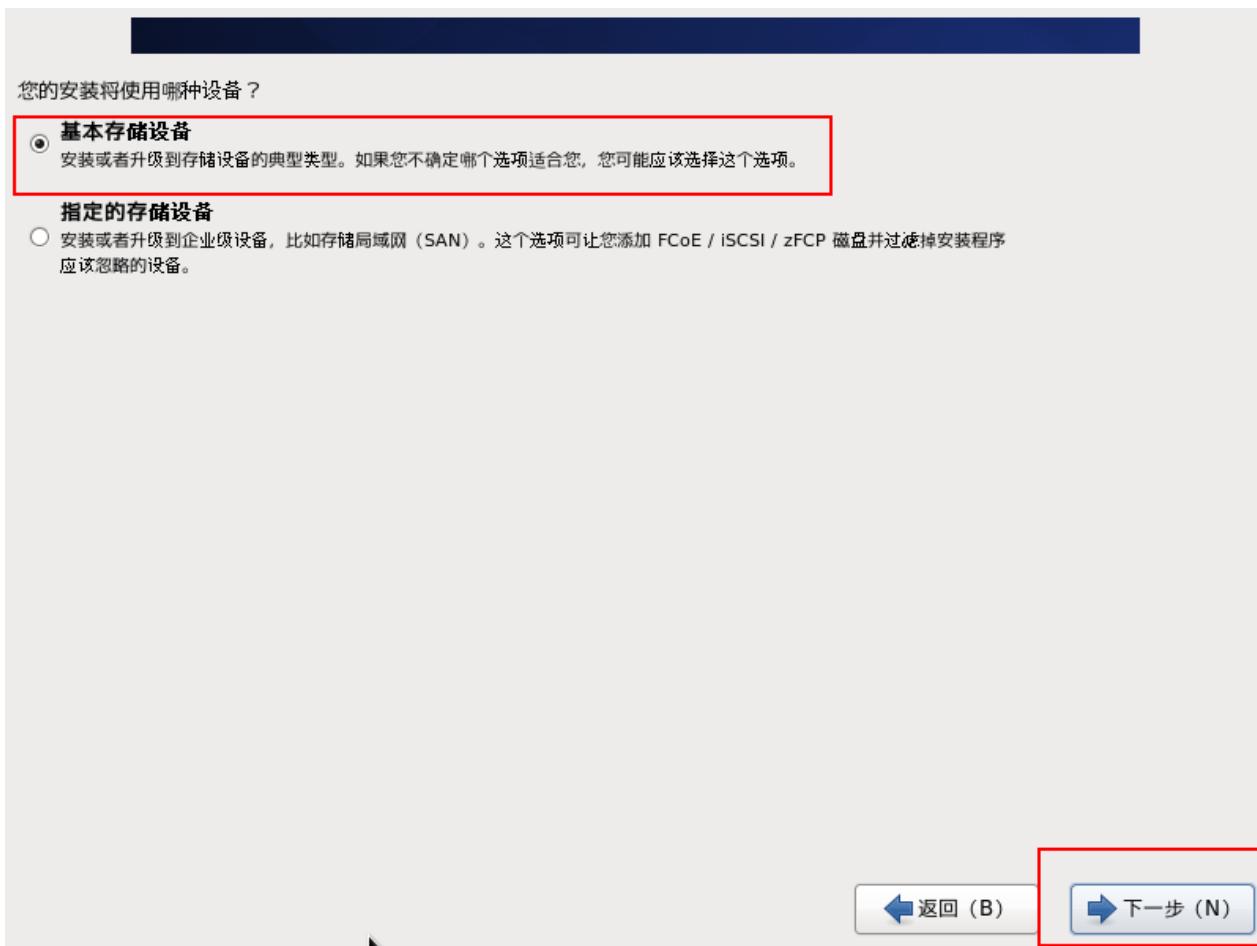
- Afrikaans (Afrikaans)
- Arabic (العربية)
- Assamese (অসমীয়া)
- Bengali (বাংলা)
- Bengali(India) (বাংলা (ভারত))
- Bulgarian (Български)
- Catalan (Català)
- Chinese(Simplified) (中文 (简体))** (highlighted with a red box)
- Chinese(Traditional) (中文 (正體))
- Croatian (Hrvatski)
- Czech (Čeština)
- Danish (Dansk)
- Dutch (Nederlands)
- English (English)
- Estonian (eesti keel)
- Finnish (suomi)
- French (Français)

Back Next

第 14 步：键盘选择 美国英语式，然后点击下一步



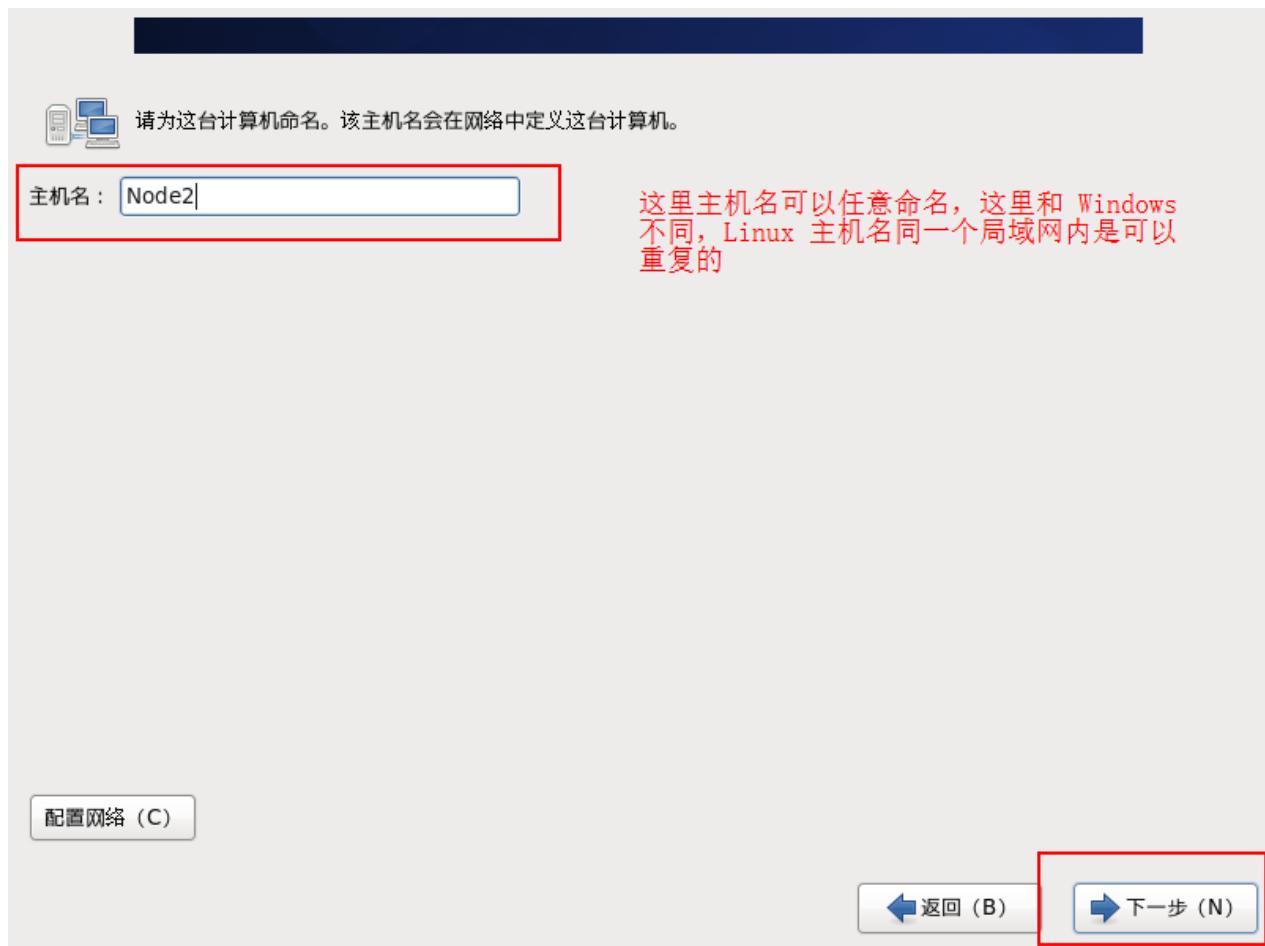
第 15 步：选择 基本存储设备，点击下一步



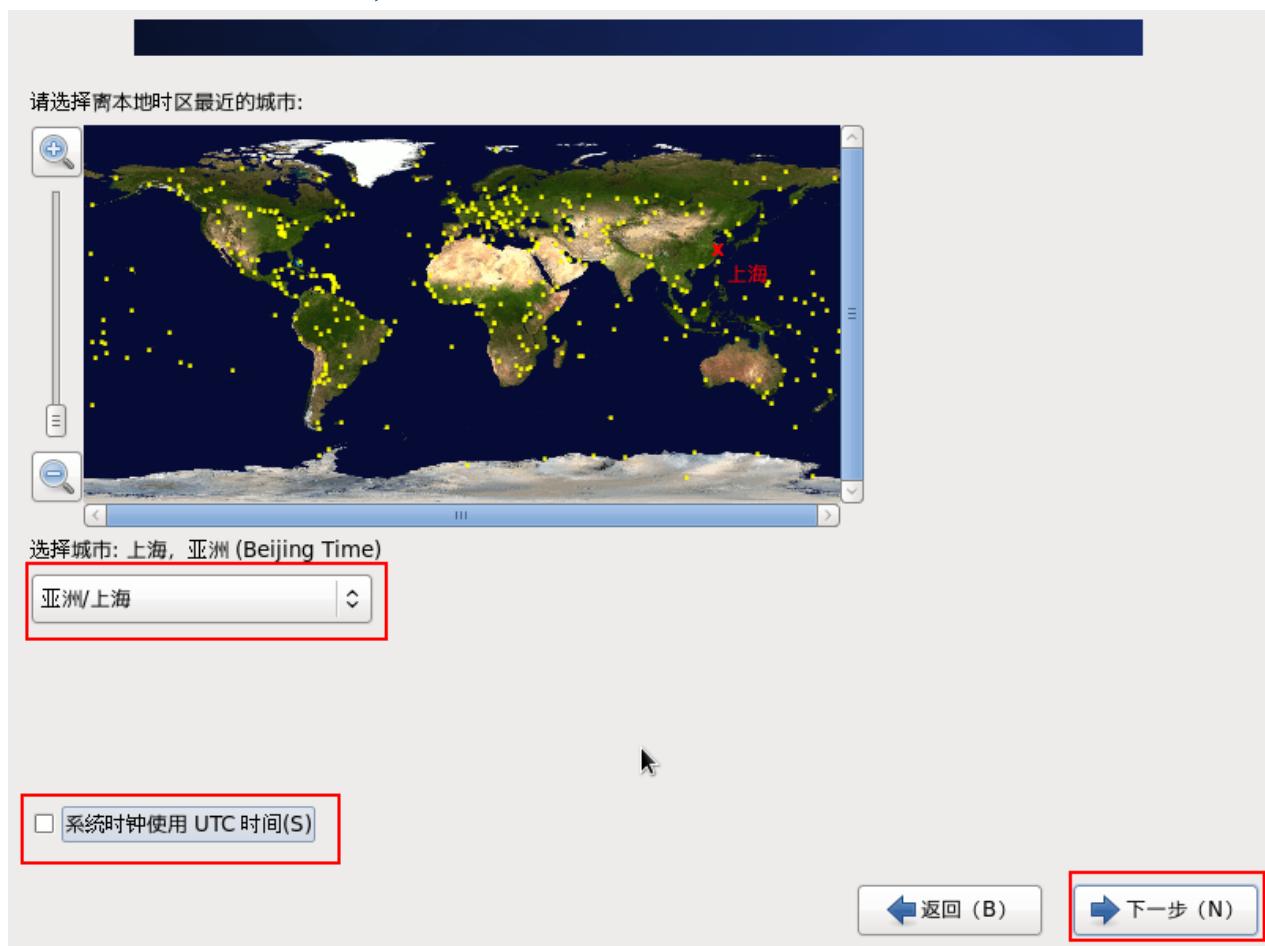
第 16 步：选择第一个，是 忽略所有数据



第 17 步：给主机命名

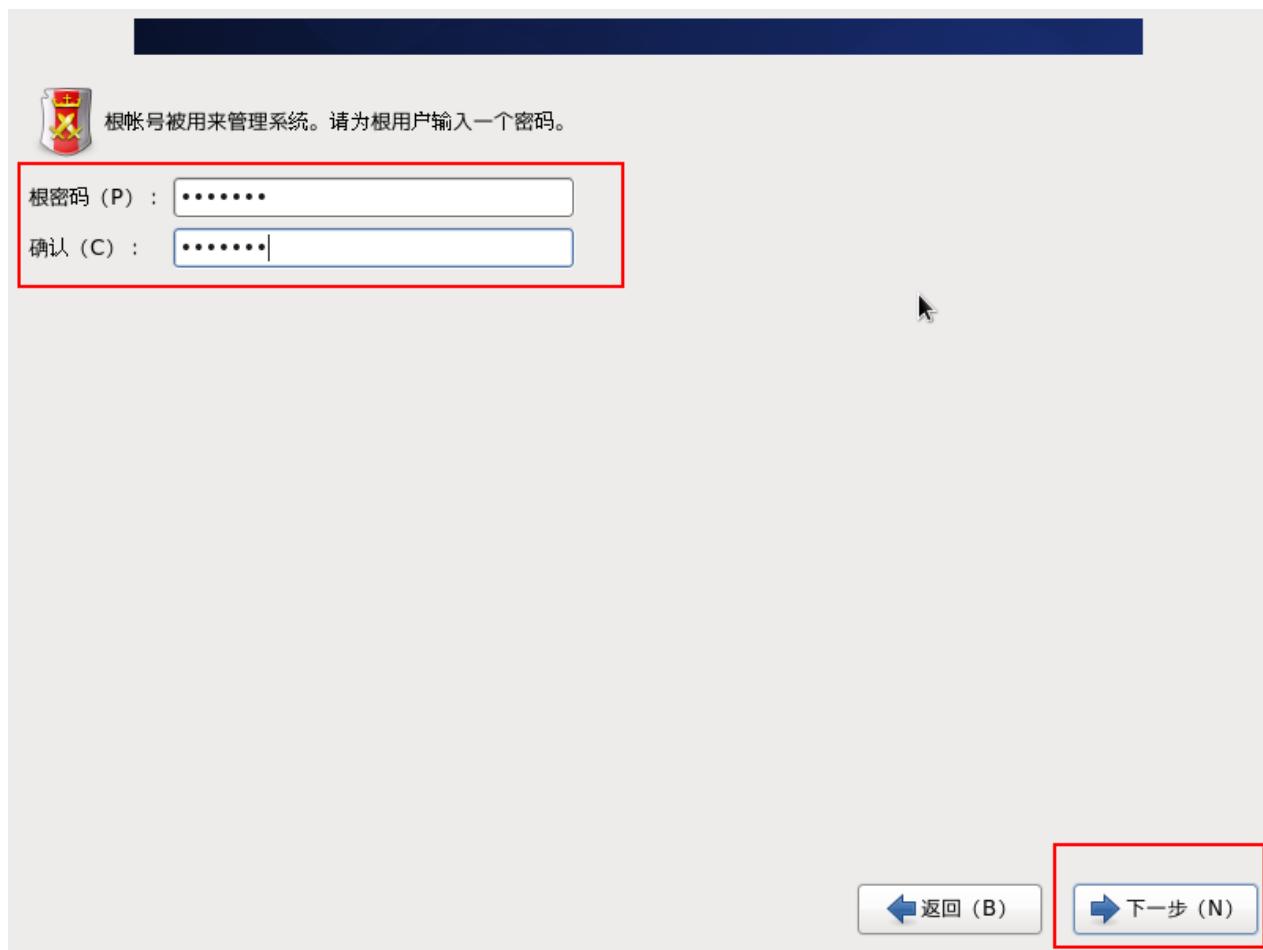


第 18 步：选择时区亚洲/伤害，去掉勾选使用UTC时间。然后点击 下一步 **



第 19 步：给根用户设置密码，然后点击 下一步。

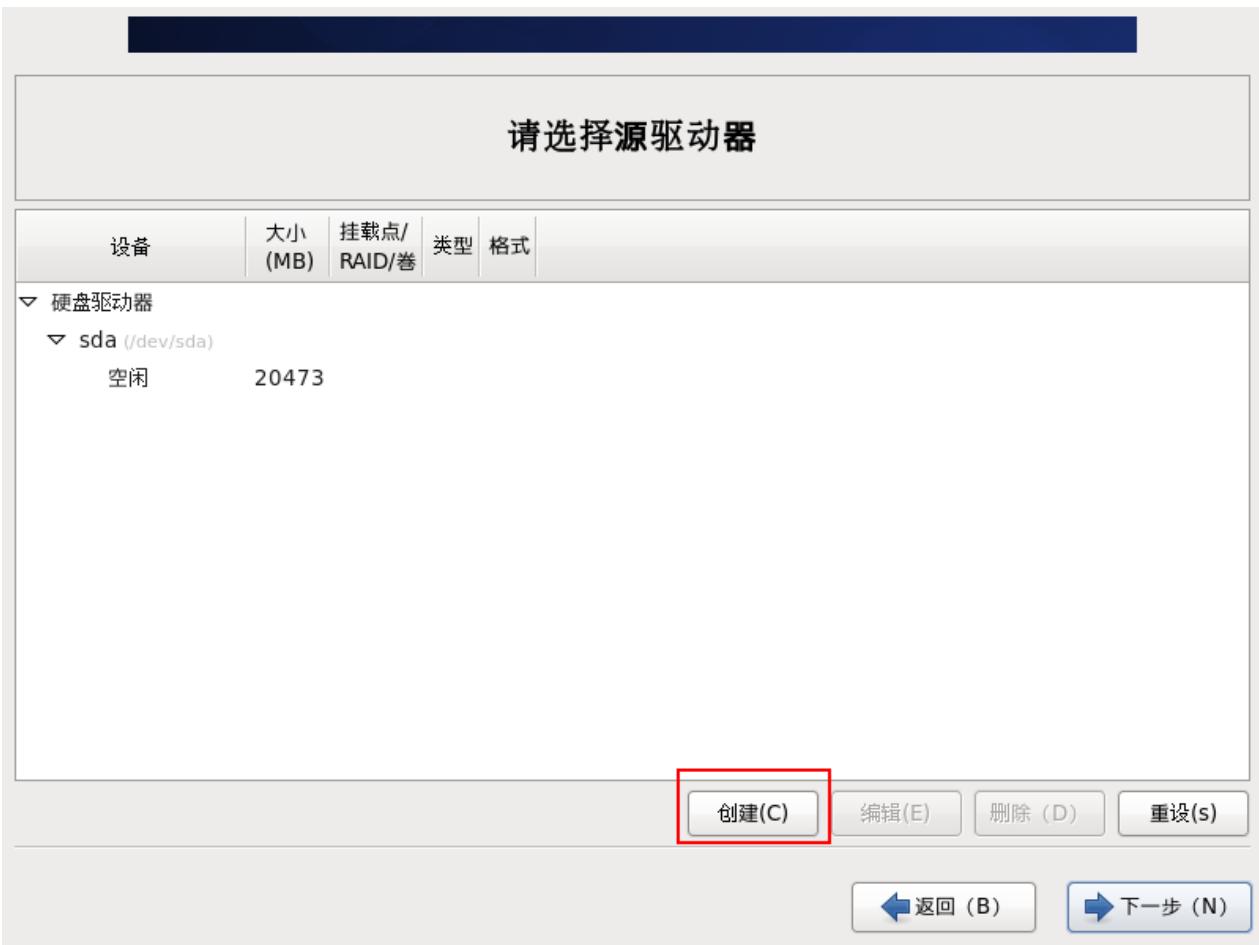
注意：如果密码设置的过于简单，系统会弹出您的密码不够安全，但是你可以选择无论如何都使用，然后继续



第 20 步：选择进行哪种类型的安装，我们选择最后一个 创建自定义布局



第 21 步：给硬盘分区，如下界面，我们点击 创建



为了便于后面的操作，我们这里选择手动分区，顺便给大家普及一下Linux分区的知识：

Linux 系统分区：

必须分区：

- 根分区 /
- 交换分区 swap (可以理解为虚拟内存，当内存不够时，可以临时使用 swap 分区，内存的两倍，不超过 2GB)

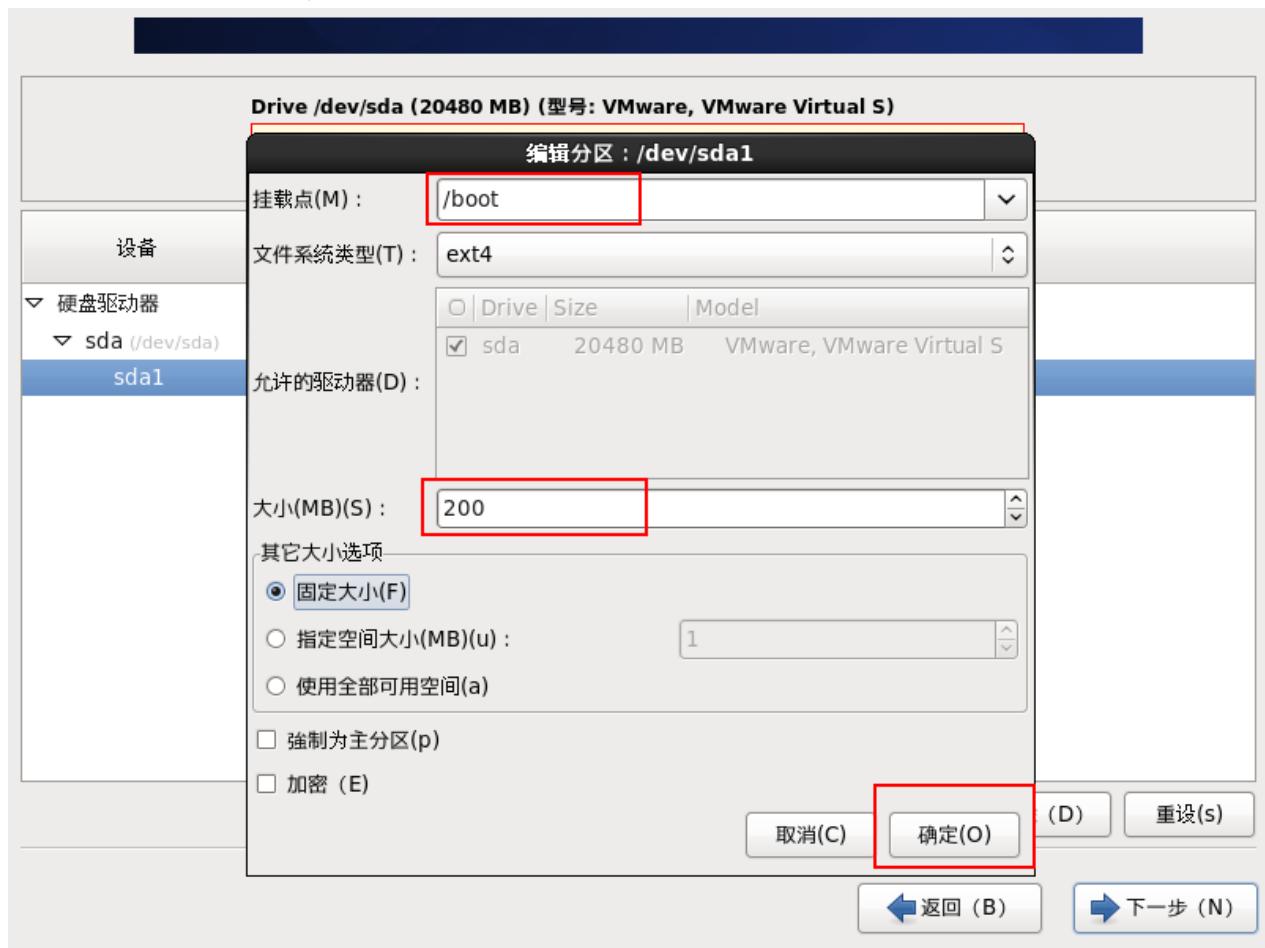
推荐分区：

- 启动分区 boot (保存系统启动时的数据，一般不用太大，200 M足够，防止根分区写满文件之后，系统起不来)
- home 分区，保存用户的信息

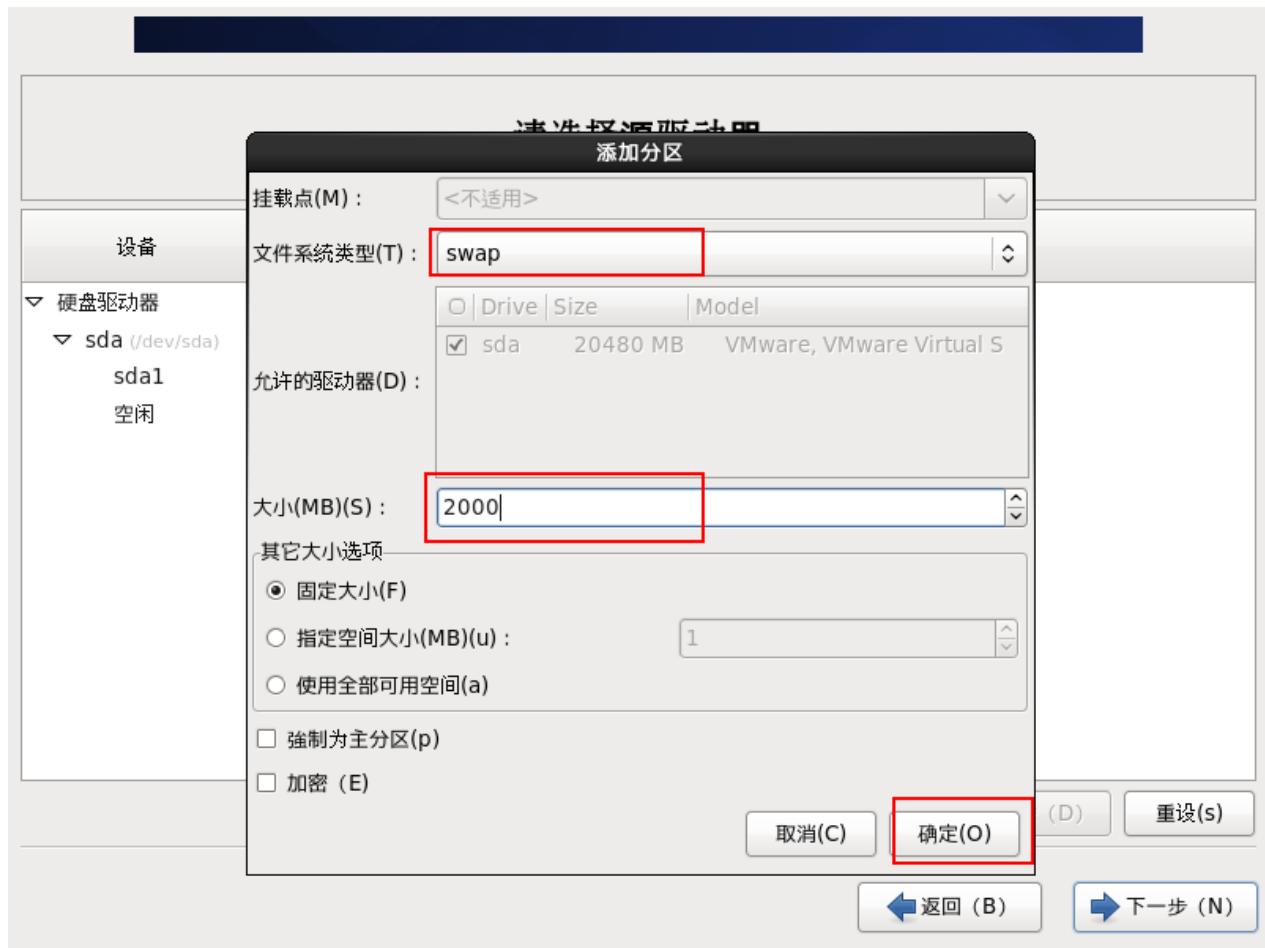
我们在上一步之后选择标准分区，点击创建



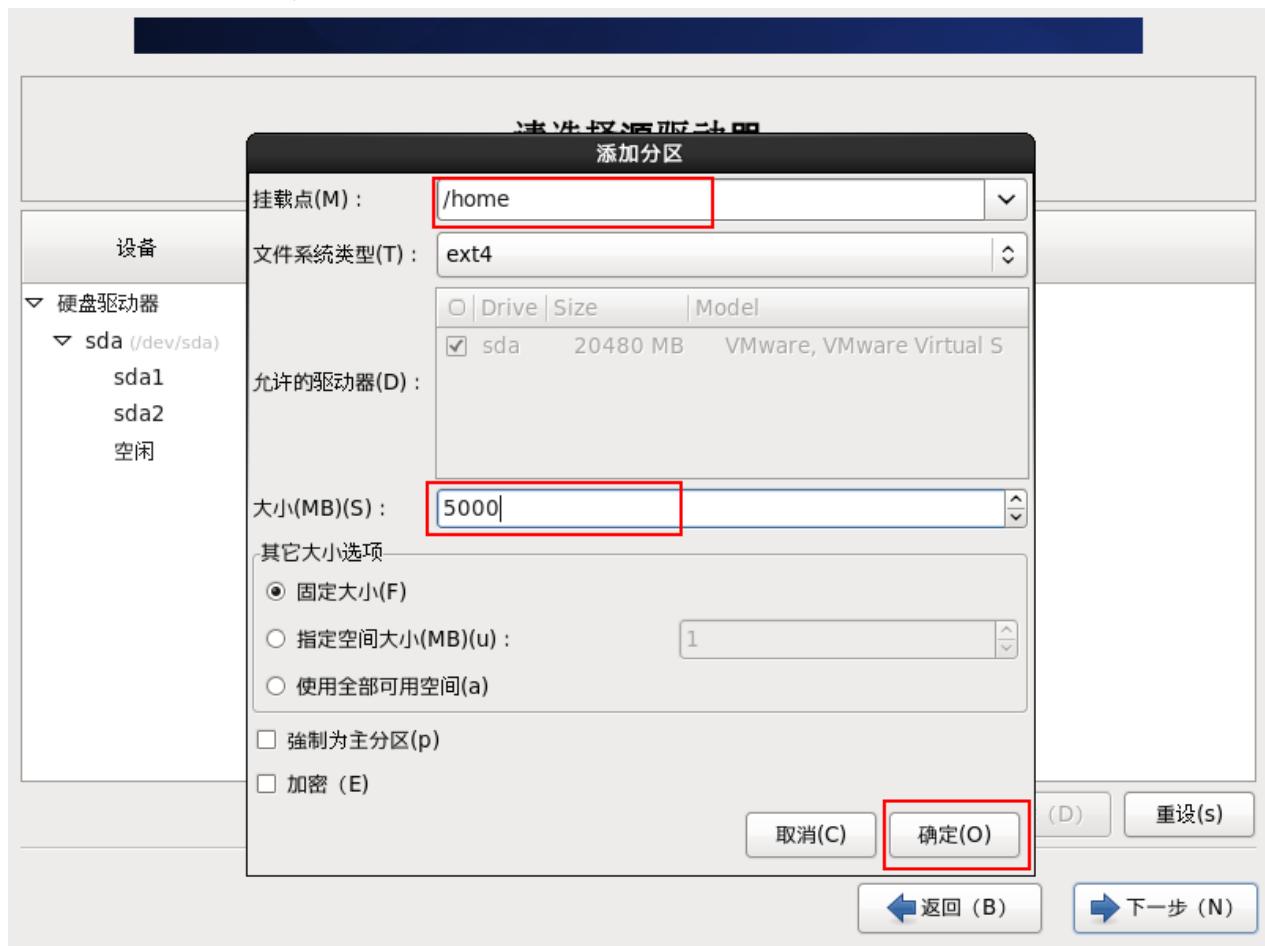
第一步：创建 boot 分区，大小为 200 m



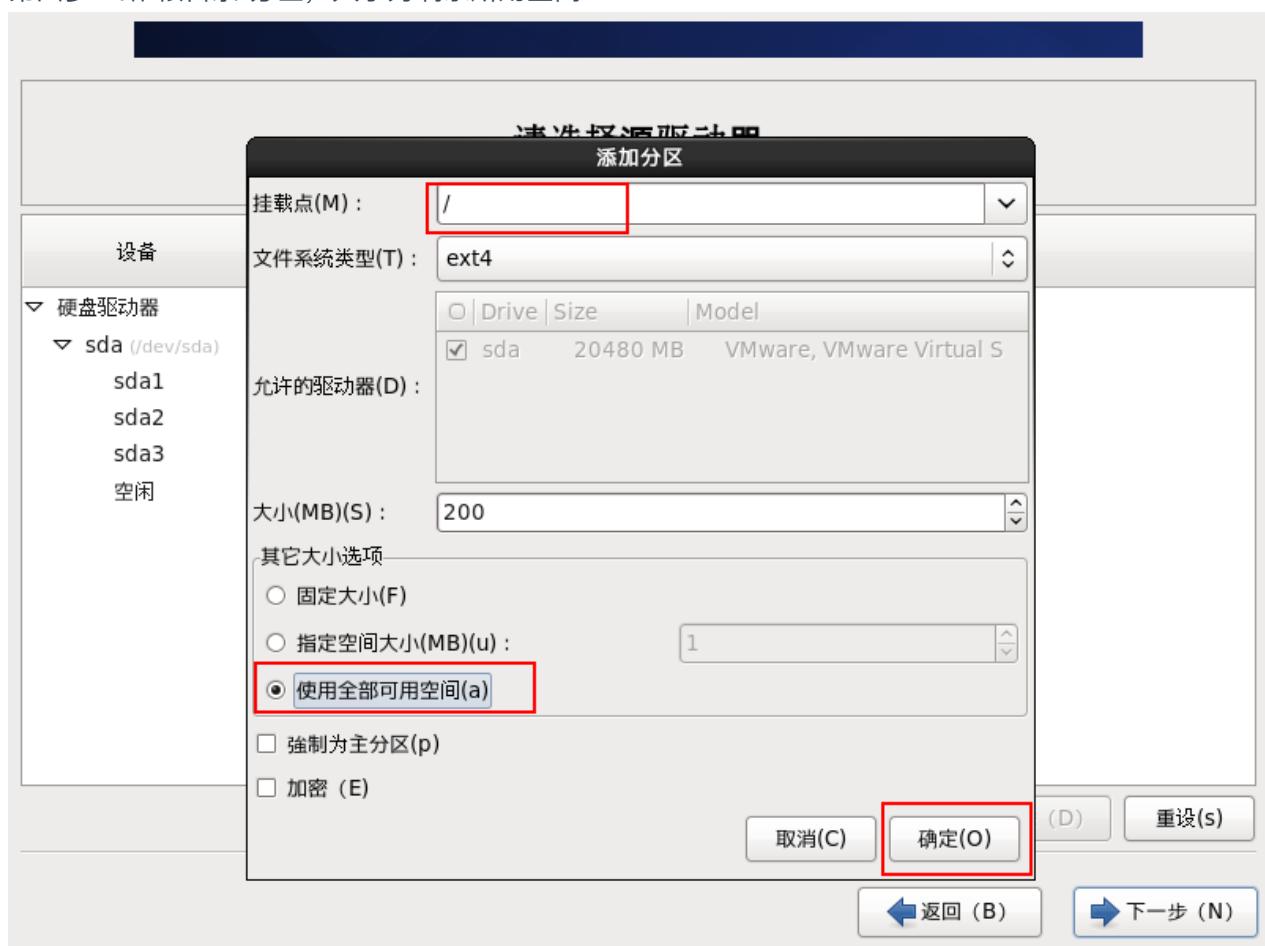
第二步：给 swap 分区，大小为 2000 M



第三步：给 home 分区，大小为 5000 M



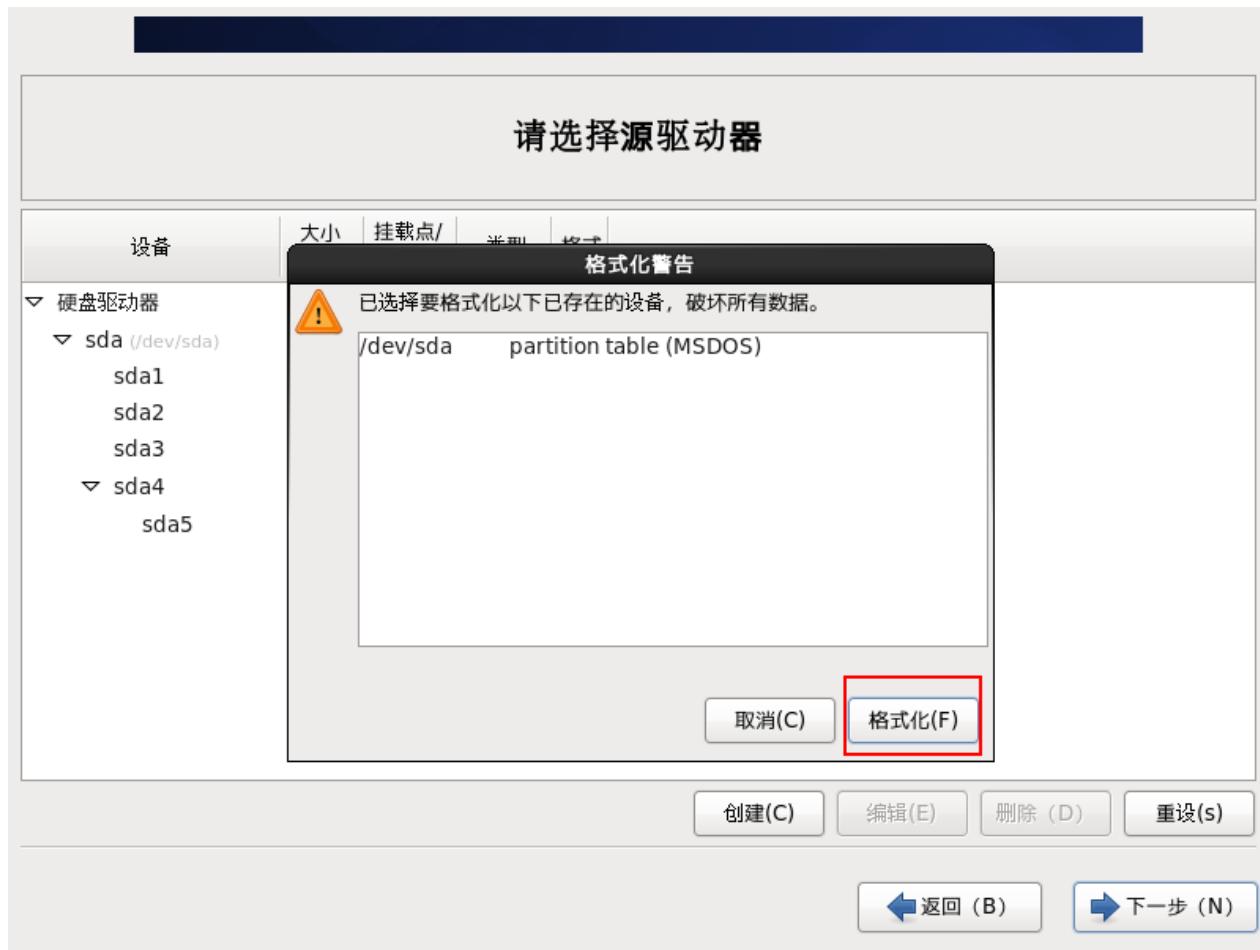
第四步：给 根目录 分区，大小为剩余所用空间



那么我们分区完成，点击 下一步：



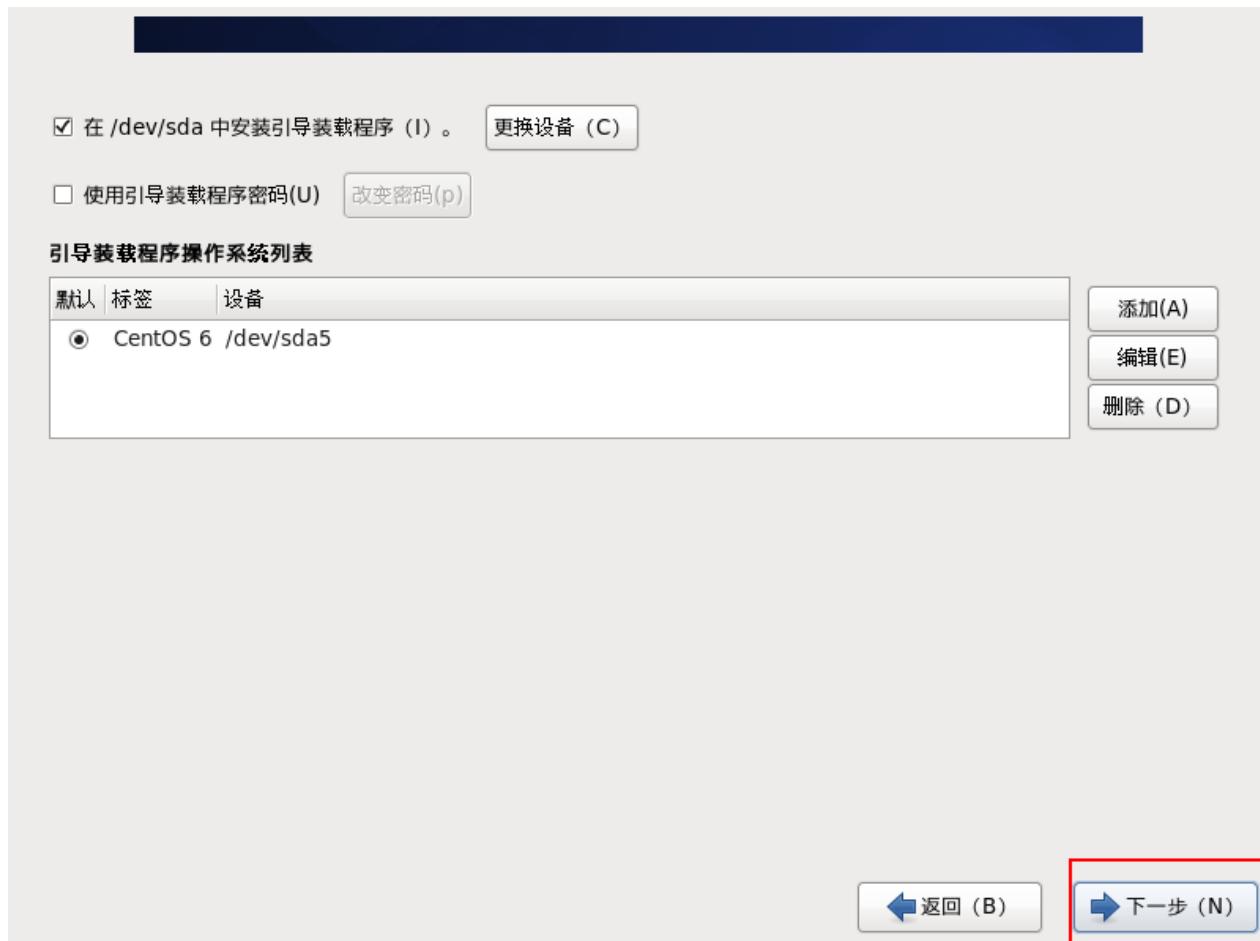
第 22 步：格式化硬盘，选择格式化 **



第 23 步：选择将修改写入磁盘，点击下一步 **

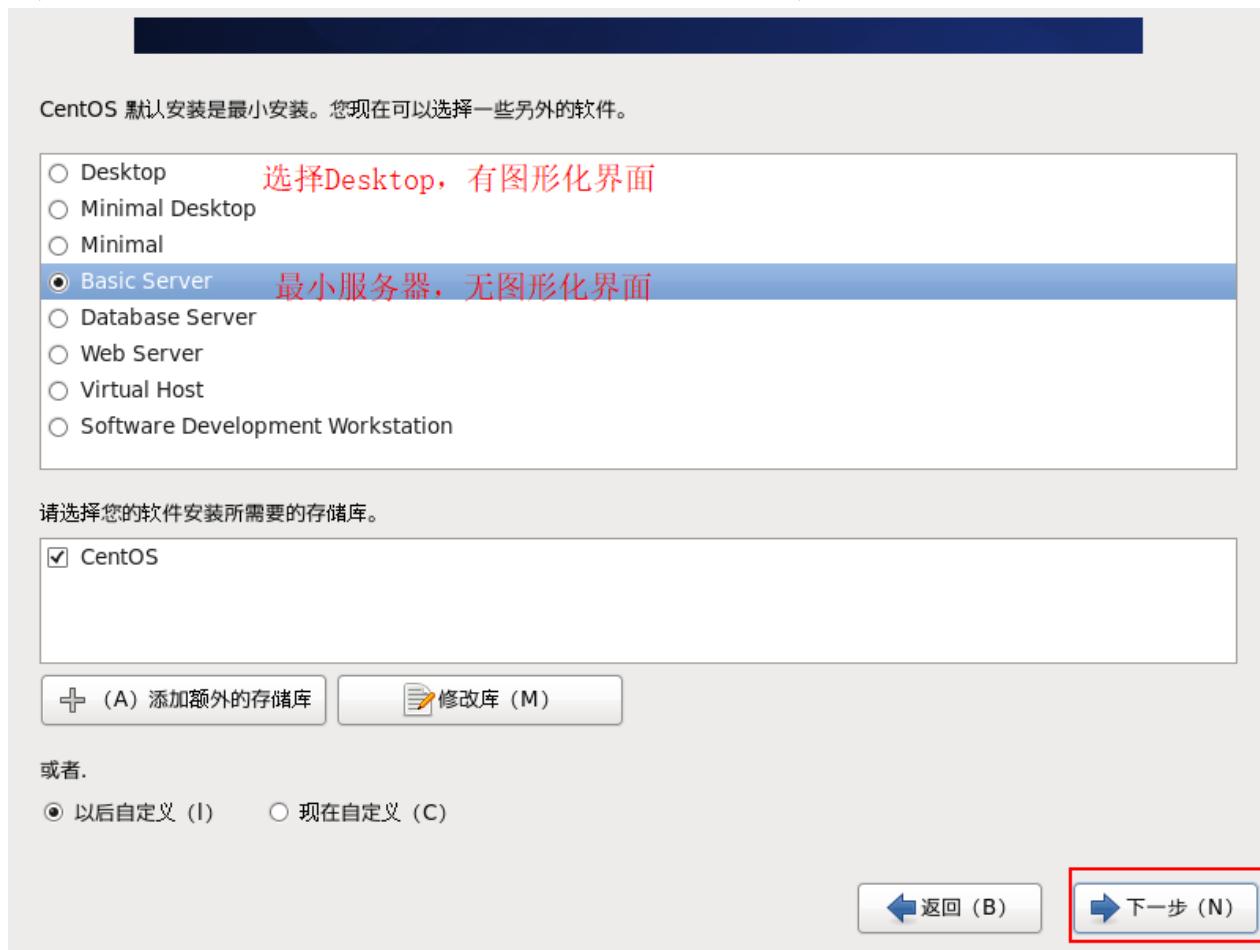


第 24步：默认，点击下一步



第 25 步：选择安装类型

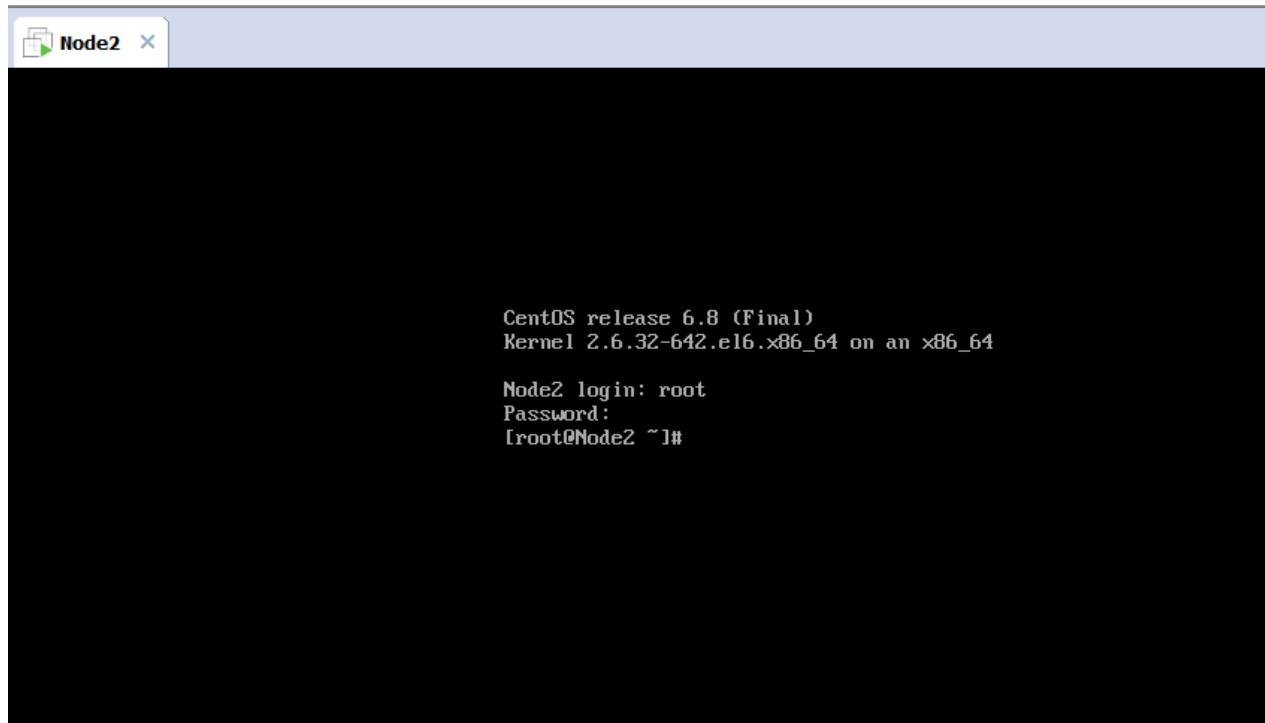
初学者如果想要图形化界面可以选择 前面两个，但是基本上后面的操作建议都用命令行的形式来学习更好，这里我们建议选择 Basic Server（纯字符界面）点击下一步，然后等待安装完成



第 26 步：安装完成后，我们选择重新引导即可，输入用户名密码登录我们所安装的 Linux 系统



第 27 步：输入用户名、密码登录 Linux 系统



上面我们讲了 Linux 系统的详细安装教程，大家跟着教程一步一步的操作，应该能完美的完成安装。那么这篇博客跟大家聊聊如何来学习 Linux。

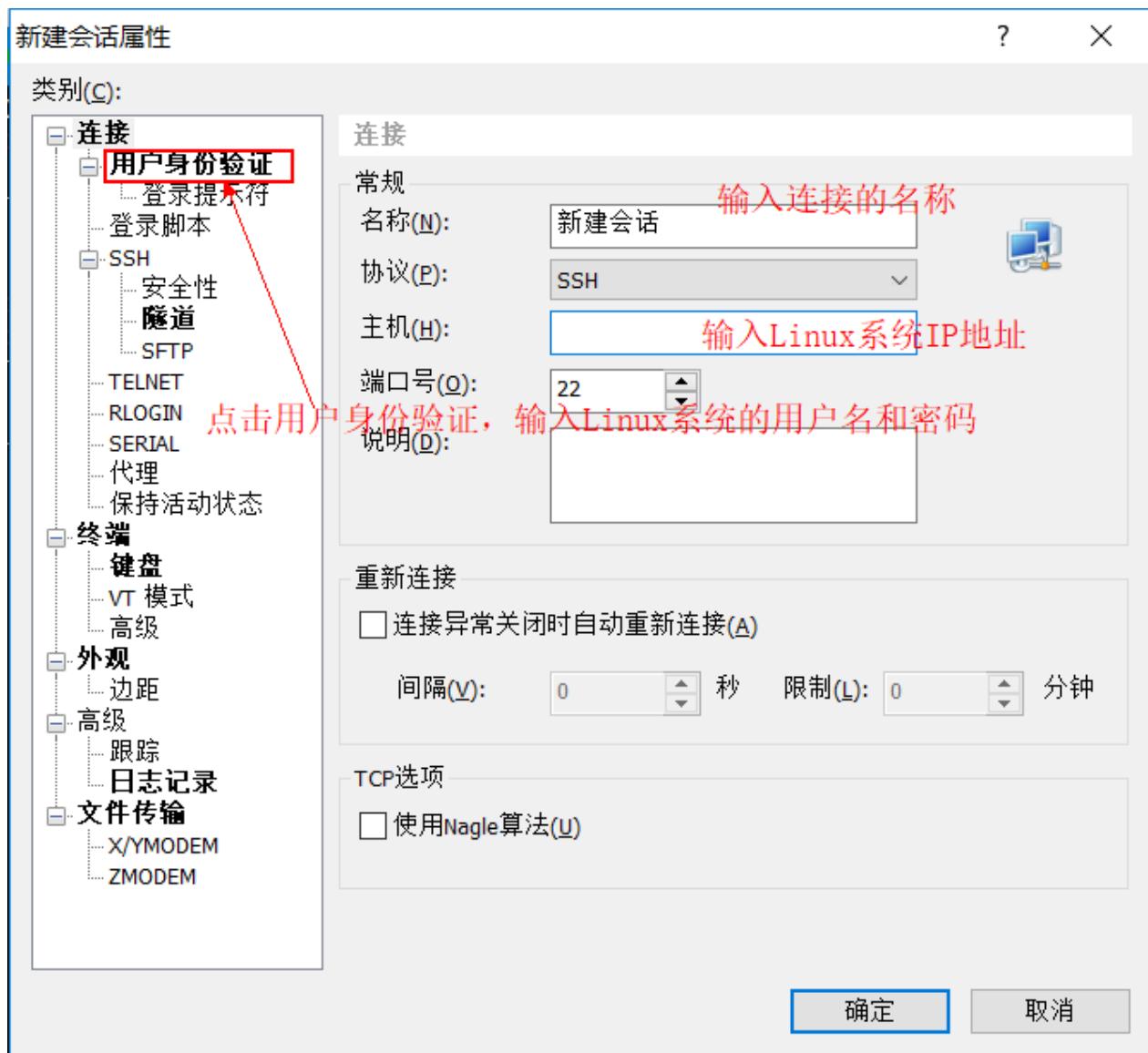
工欲善其事必先利其器

第一个问题：通过前面在虚拟软件中安装的Linux系统，我们发现在Linux虚拟机中操作命令发现特别繁琐，在虚拟机系统和本机系统之间的切换比较麻烦。那么这该怎么解决呢？

这里给大家推荐一个软件：**Xshell 5**

百度网盘下载链接：<http://pan.baidu.com/s/1jlj9s3O> 密码：bgbm

软件的安装步骤很简单，跟着提示不断点击下一步下一步就能完成安装。安装完成之后，打开Xshell5，点击 打开——新建：



配置完成之后，我们就能通过Xshell5来连接Linux系统上，通过这个工具来控制Linux系统是很方便的。而且还能同时控制多个Linux机器。如下：

node0 - root@node0:~ - Xshell 5 (Free for Home/School)

文件(E) 编辑(E) 查看(V) 工具(I) 选项卡(B) 窗口(W) 帮助(H)

ssh://root:*****@192.168.146.250:22

要添加当前会话，点击左侧的箭头按钮。

1 node0 * 2 node0 *

Xshell 5 (Build 0964)
Copyright (c) 2002-2016 NetSarang Computer, Inc. All rights reserved.
Type `help' to learn how to use Xshell prompt.
[c:\~]\$

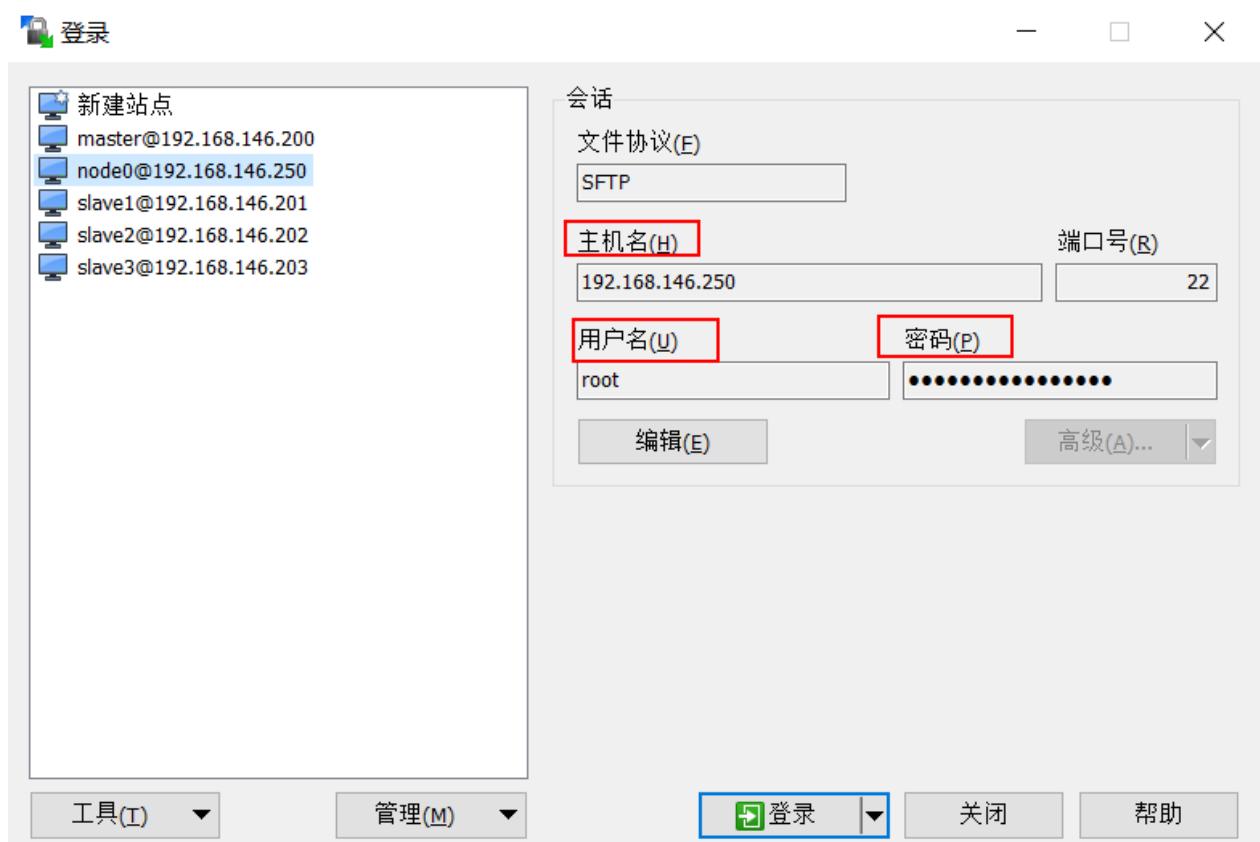
Connecting to 192.168.146.250:22...
Connection established.
To escape to local shell, press 'Ctrl+Alt+]'.
Last login: Tue Oct 17 23:22:18 2017 from 192.168.146.1
[root@node0 ~]#

第二个问题：如果我们想向Linux机器上传文件，我们该怎么办呢？

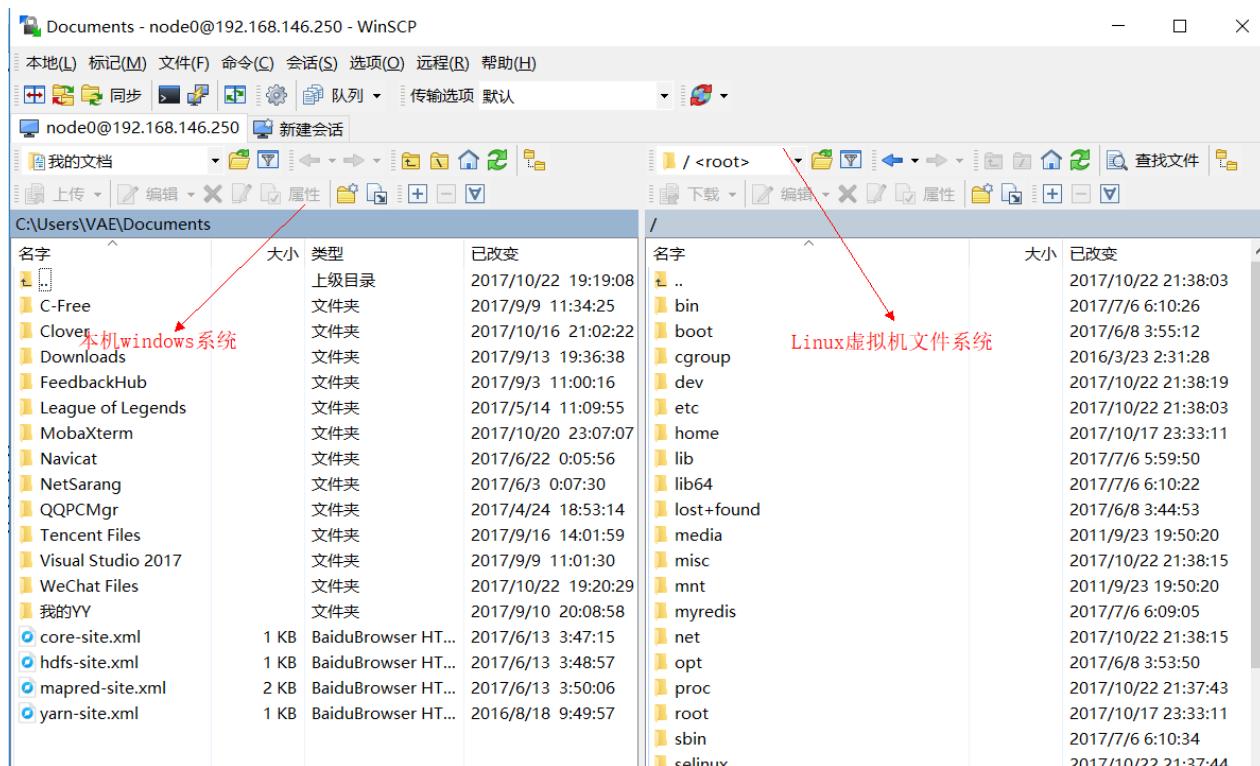
还是给大家推荐一个工具：WinSCP

百度网盘下载链接：<http://pan.baidu.com/s/1geUMqDP> 密码：jrru

同理该软件安装步骤也很简单，安装完成之后，我们打开该软件，进行如下的配置：



点击登录之后，出现如下界面，左边是 windows 系统，右边是 Linux 系统，可以直接进行粘贴复制操作。



英文提示的困惑

这里有两个Linux最常见的报错提示：

Command not found;

No Such file or directory;

这两个英文提示，第一个是找不到命令（我们敲入的命令有误或者该命令还没安装）；第二个是找不到我们输入的文件或者目录。这只是我们在后面操作Linux系统时会遇到的两个很常见的问题报错提示，其实大家在后面遇到英文提示的信息，不要刻意的去忽视，这是用来帮助我们解决问题的很好的提示。而英文大家遇到不会了就去谷歌翻译，百度翻译查查，总归就是那些单词。我们多看几篇，以后在遇到类似的问题就会得心应手了。

忘掉windows的思维方式

相信大家玩操作系统都是从 Windows 开始的，windows的提出理念是 让中年家庭妇女都可以熟练的运用电脑，所以我们在操作 windows 系统时，几乎不需要太多的知识储备，就能比较熟练的操作 windows 系统。但是对于Linux 系统就不行了，windows 特色就是图形化界面设计的非常友好，Linux 系统虽然也有图形化界面，但是Linux 从诞生到现在，一般都是为服务器做共享的，而服务器都是给专业人士来维护的，一般都是用非图形化界面的命令行方式来操作的。所以操作 Linux 系统和 windows 系统的区别比较大，比如 windows 系统我们一般会把系统安装在 C 盘，windows 系统有盘符的说法，而 Linux 没有。

Linux 的基本思想有两点

- 一切都是文件；
- 每个软件都有确定的用途。其中第一条详细来讲就是系统中的所有都归结为一个文件，包括命令、硬件和软件设备、操作系统、进程等等对于操作系统内核而言，都被视为拥有各自特性或类型的文件。所以后面我们讲的Linux 命令介绍，都会给出命令所在的文件目录。但是不管怎么说，大家刚上手由于先入为主的原因，可能对Linux 操作感觉不自在，但是当你习惯之后，你会爱上 Linux 的命令行的。

Linux 常用命令之文件和目录处理命令

Linux 命令的普遍语法格式

命令格式：命令 【-选项】 【参数】 例子： ls -la /etc 说明： ①、个别命令使用不遵循此格式，【】这种符号表示可以省略 ②、当有多个选项时，可以写在一起
 ③、简化选项(一般用一个-)与完整选项 (一般用两个--) 比如 ls -a 等于 ls --all

```
[root@Node3 ~]# ls -a
. anaconda-ks.cfg .bash_logout .bashrc install.log .tcshrc
.. .bash_history .bash_profile .cshrc install.log.syslog
[root@Node3 ~]# ls --all
. anaconda-ks.cfg .bash_logout .bashrc install.log .tcshrc
.. .bash_history .bash_profile .cshrc install.log.syslog
[root@Node3 ~]#
```

上面的便是Linux系统中的一般命令格式，基本上所有命令都是遵循这种语法格式（个别命令除外）。

目录处理命令

一、显示目录文件命令：ls

- ①、命令名称：ls
- ②、英文原意：list
- ③、命令所在路径：/bin/ls
- ④、执行权限：所有用户
- ⑤、功能描述：显示目录文件
- ⑥、语法： ls 选项 【-ald】 【文件或目录】

- a 显示所有文件，包括隐藏文件
- l 详细信息显示
- d 仅显示目录名，而不显示目录下的内容列表
- h 人性化显示 (hommization)
- i 查看任意一个文件的i节点 (类似于身份证唯一信息)
- t 用文件和目录的更改时间排序；可以用第一个显示的文件判断最近修改的文件

注意：. 开头的文件除非是目录，否则就是隐藏文件

```
[root@Node3 ~]# ls -a
. anaconda-ks.cfg .bash_logout .bashrc install.log .tcshrc
.. .bash_history .bash_profile .cshrc install.log.syslog
[root@Node3 ~]# ls -l
total 44
-rw-----. 1 root root 1289 Mar 11 08:05 anaconda-ks.cfg
-rw-r--r--. 1 root root 28250 Mar 11 08:05 install.log
-rw-r--r--. 1 root root 7572 Mar 11 08:03 install.log.syslog
[root@Node3 ~]# ls -d
.
[root@Node3 ~]# ls -dl
dr-xr-x---. 2 root root 4096 Mar 11 08:09 .
[root@Node3 ~]# ls -t
anaconda-ks.cfg install.log install.log.syslog
[root@Node3 ~]# ls -lh
total 44K
-rw-----. 1 root root 1.3K Mar 11 08:05 anaconda-ks.cfg
-rw-r--r--. 1 root root 28K Mar 11 08:05 install.log
-rw-r--r--. 1 root root 7.4K Mar 11 08:03 install.log.syslog
[root@Node3 ~]#
```

上面我们分别列出了ls的各种命令组合显示效果。我们以 **-rw-----. 1 root root 1.3k Mar 11 08:05 anaconda-ks.cfg** 为例解析每个字段：

- ①、-rw----- 第一位表示文件类型，- 表示是二进制文件，d 表示目录，l 表示软连接文件。后面的每三个为一组：

rw- --- ---

u g o

u (user) 所有者 g (group) 所属组 o (other) 其他人

r 读 w写 x执行 -无权限

②、1 引用计数，表示文件被引用过多少次

③、root 这第一个root表示所有者，一般创建一个文件，所有者默认是创建者。

④、root 这第二个root表示所属组。

⑤、1.3K 表示文件字节大小，不带单位表示字节

⑥、ar 11 08:05 表示文件的最后修改时间。注意：Linux没有明确的创建时间，只有最后一次访问时间、文件的状态修改时间、文件的数据修改时间

⑦、anaconda-ks.cfg 表示文件名

代表字符	权限	对文件的含义	对目录的含义
r	读权限	可以查看文件内容	可以列出目录中的内容
w	写权限	可以修改文件内容	可以在目录中创建、删除文件
x	执行权限	可以执行文件	可以进入目录

二、创建目录命令：mkdir

①、命令名称：mkdir

②、英文原意：make directories

③、命令所在路径：/bin/mkdir

④、执行权限：所有用户

⑤、功能描述：创建新目录

⑥、语法： mkdir 【-p】 【目录名】

-p 递归创建

例子：创建单个目录：mkdir /tmp/vae.txt

创建多个目录：mkdir /tmp/a.txt /tmp/b.txt

注意：1、创建的目录已经存在，那么 Linux 会提示我们 Linux 无法创建它。

2、不带任何参数运行 mkdir 命令会在当前目录下创建目录。

3、不带上-p,如果新建的文件上级目录不存在则不会执行成功这种说法是错误的。加或者不加上 -p 前面的目录没有得都会依次创建。

4、创建目录的首要条件是，在想要创建目录的目标路径下你必须具有访问权限。

```
[root@Node3 ~]# cd /tmp
[root@Node3 tmp]# mkdir /vae/ys
[root@Node3 tmp]# mkdir /vae
mkdir: cannot create directory '/vae': File exists
[root@Node3 tmp]# _
```

三、切换目录命令：cd

①、命令名称：cd

②、英文原意：change directory

③、命令所在路径：**shell** 内置命令

④、执行权限：所有用户

⑤、功能描述：切换目录

⑥、语法：cd 【目录名】

例子：切换到指定目录：cd /tmp/vae

回到上一级目录：cd ..

还是在当前目录：cd .

返回上两级目录：cd ../../

返回进入此目录之前所在的目录：cd -

```
[root@Node3 vae]# cd /tmp
[root@Node3 tmp]# cd -
/vae
[root@Node3 vae]# cd ..
[root@Node3 /]# cd /
[root@Node3 /]# cd .
[root@Node3 /]# _
```

四、**shell**内置命令和外部命令的区别

大家可以看到前面的三个命令，ls 命令和 mkdir 命令都有命令的所在路径，而 cd 命令我们说是 shell 内置命令。这两者便是 Linux 内置命令和外部命令。

内部命令实际上是 shell 程序的一部分，其中包含的是一些比较简单的 linux 系统命令，这些命令由 shell 程序识别并在 shell 程序内部完成运行，通常在 linux 系统加载运行时 shell 就被加载并驻留在系统内存中。内部命令是写在 bash 源码里面的，其执行速度比外部命令快，因为解析内部命令 shell 不需要创建子进程。比如：exit, history, cd, echo 等。

外部命令是 linux 系统中的实用程序部分，因为实用程序的功能通常都比较强大，所以其包含的程序量也会很大，在系统加载时并不随系统一起被加载到内存中，而是在需要时才将其调用内存。通常外部命令的实体并不包含在shell 中，但是其命令执行过程是由 shell 程序控制的。shell 程序管理外部命令执行的路径查找、加载存放，并控制命令的执行。外部命令是在bash之外额外安装的，通常放在/bin, /usr/bin, /sbin, /usr/sbin.....等等。可通过 `echo $PATH` 命令查看外部命令的存储路径，比如：ls、vi等。

用type命令可以分辨内部命令与外部命令：

```
[root@node3 tmp]# type cd  
cd is a shell builtin  
[root@node3 tmp]# type mkdir  
mkdir is hashed (/bin/mkdir)  
[root@node3 tmp]#
```

内部命令和外部命令最大的区别之处就是性能。内部命令由于构建在shell中而不必创建多余的进程，要比外部命令执行快得多。因此和执行更大的脚本道理一样，执行包含很多外部命令的脚本会损害脚本的性能。

五、显示当前目录命令：pwd

- ①、命令名称：pwd
- ②、英文原意：print working directory
- ③、命令所在路径：/bin/pwd
- ④、执行权限：所有用户
- ⑤、功能描述：显示当前目录
- ⑥、语法：pwd

例子：显示当前目录：pwd

```
[root@node3 /]# cd /tmp  
[root@node3 tmp]# pwd  
/tmp  
[root@node3 tmp]#
```

六、删除空目录命令：rmdir

- ①、命令名称：rmdir
- ②、英文原意：remove empty directories
- ③、命令所在路径：/bin/rmdir
- ④、执行权限：所有用户
- ⑤、功能描述：删除空目录（如果目录下存在文件则不能删除）
- ⑥、语法： rmdir 【空目录名】

例子：删除指定空目录：rmdir /tmp/a

```
[root@node3 tmp]# mkdir a      在tmp目录下创建a目录
[root@node3 tmp]# ll
总用量 4
drwxr-xr-x. 2 root root 4096 10月 26 08:38 a
[root@node3 tmp]# rmdir /tmp/a  删除tmp目录下的a目录
[root@node3 tmp]# ll
总用量 0                         mkdir命令不带-p参数不能创建不存在的
[root@node3 tmp]# mkdir /tmp/a/b 目录
mkdir: 无法创建目录"/tmp/a/b": 没有那个文件或目录
[root@node3 tmp]# mkdir -p /tmp/a/b mkdir命令-p参数递归创建目录
[root@node3 tmp]# rmdir /tmp/a
rmdir: 删 除 "/tmp/a" 失败: 目录非空 rmdir命令不能删除非空目录
[root@node3 tmp]#
```

注意：由于此命令只能删除空目录，在实际操作中应用的不多，后面我们讲解 rm 命令可以替代。

七、复制文件或目录命令：cp

- ①、命令名称：cp
- ②、英文原意：copy
- ③、命令所在路径：/bin/cp
- ④、执行权限：所有用户
- ⑤、功能描述：复制文件或目录
- ⑥、语法： cp [-rp] 【原文件或目录】 【目标目录】
 - r 复制目录
 - p 保留文件属性

例子：比如我们将 root 目录下的install.log 文件复制到 tmp 目录下

```
[root@Node1 ~]# ls
anaconda-ks.cfg  install.log  install.log.syslog
[root@Node1 ~]# cp /root/install.log /tmp
[root@Node1 ~]# ls -lh /tmp
总用量 28K
-rw-r--r--. 1 root root 28K 6月    3 21:14 install.log
-rw-----. 1 root root    0 6月    2 07:13 yum.log
[root@Node1 ~]#
```

可以用于复制后文件改名，比如我们将 root 目录下的 install.log 文件复制到 tmp 的 copyinstall.log 文件

```
[root@Node1 ~]# ls -lh /tmp
总用量 0
-rw-----. 1 root root 0 6月    2 07:13 yum.log
[root@Node1 ~]# cp /root/install.log /tmp/copyinstall.log
[root@Node1 ~]# ls -lh /tmp
总用量 28K
-rw-r--r--. 1 root root 28K 6月    3 21:18 copyinstall.log
-rw-----. 1 root root    0 6月    2 07:13 yum.log
[root@Node1 ~]#
```

还可同时复制多个文件，注意复制目录的时候要加 -r选项。加-p属性之后会将原文件的一些属性比如修改时间等也原封不动的复制过去。如果不加-p属性，那么复制后的文件修改时间为当前系统时间。

八、剪切文件或目录命令：mv

- ①、命令名称：mv
- ②、英文原意：move
- ③、命令所在路径：/bin/mv
- ④、执行权限：所有用户
- ⑤、功能描述：剪切文件、改名
- ⑥、语法： mv 【原文件或目录】 【目标目录】

例子：在 tmp 目录下创建目录 tmp1，然后在 tmp1 目录下创建目录 tmp1_1,然后 命令 mv /tmp/tmp1/tmp1_1 /tmp 将 tmp1目录下的 tmp1_1 剪切到 tmp 目录下。

```
[root@Node1 tmp]# mkdir tmp1
[root@Node1 tmp]# cd tmp1
[root@Node1 tmp1]# mkdir tmp1_1
[root@Node1 tmp1]# cd ..
[root@Node1 tmp]# mv /tmp/tmp1/tmp1_1 /tmp
[root@Node1 tmp]# ls -lh /tmp
总用量 36K
-rw-r--r--. 1 root root 28K 6月    3 21:18 copyinstall.log
drwxr-xr-x. 2 root root 4.0K 6月    3 21:20 tmp1
drwxr-xr-x. 2 root root 4.0K 6月    3 21:20 tmp1_1
-rw-----. 1 root root    0 6月    2 07:13 yum.log
[root@Node1 tmp]#
```

九、删除文件或目录命令：rm

- ①、命令名称：rm
- ②、英文原意：remove
- ③、命令所在路径：/bin/rm
- ④、执行权限：所有用户

⑤、功能描述：剪切文件、改名

⑥、语法： rm -rf 【文件或目录】

-r 删除目录

-f 强制执行

例子：在 tmp 目录下有两个目录 tmp1 和 tmp1_1, rm -r tmp1_1 是删除这个目录，但是需要输入 y 来确认

rm -rf tmp1 是强制删除 tmp1 目录，不需要输入 y 来确认，这种做法很简单，但是容易误删文件，没有反悔的机会

rm -rf tmp1 tmp2 同时删除两个文件

```
[root@Node1 tmp]# ls -lh
总用量 36K
-rw-r--r--. 1 root root 28K 6月      3 21:18 copyinstall.log
drwxr-xr-x. 2 root root 4.0K 6月      3 21:28 tmp1
drwxr-xr-x. 2 root root 4.0K 6月      3 21:25 tmp1_1
-rw-----. 1 root root    0 6月      2 07:13 yum.log
[root@Node1 tmp]# rm tmp1_1
rm: 无法删除 "tmp1_1": 是一个目录
[root@Node1 tmp]# rm -r tmp1_1
rm: 是否删除目录 "tmp1_1"? y
[root@Node1 tmp]# rm -f tmp1
rm: 无法删除 "tmp1": 是一个目录
[root@Node1 tmp]# rm -rf tmp1
[root@Node1 tmp]# ls -lh
总用量 28K
-rw-r--r--. 1 root root 28K 6月      3 21:18 copyinstall.log
-rw-----. 1 root root    0 6月      2 07:13 yum.log
[root@Node1 tmp]#
```

文件处理命令

一、创建空文件命令： touch

①、命令名称： touch

②、英文原意：

③、命令所在路径： /bin/touch

④、执行权限：所有用户

⑤、功能描述：创建空文件

⑥、语法： touch 【文件名】

例子：在 tmp 目录下创建 tmp.log 文件

```
[root@Node1 tmp]# ls -lh
总用量 28K
-rw-r--r--. 1 root root 28K 6月    3 21:18 copyinstall.log
-rw-----. 1 root root    0 6月    2 07:13 yum.log
[root@Node1 tmp]# touch tmp.log
[root@Node1 tmp]# ls -lh
总用量 28K
-rw-r--r--. 1 root root 28K 6月    3 21:18 copyinstall.log
-rw-r--r--. 1 root root    0 6月    3 21:38 tmp.log
-rw-----. 1 root root    0 6月    2 07:13 yum.log
[root@Node1 tmp]#
```

创建文件时，文件名不要有空格，不然就是创建了两个文件

touch program files 这是创建了两个文件， program 和 files

```
[root@Node1 tmp]# touch program files
[root@Node1 tmp]# ls -lh
总用量 28K
-rw-r--r--. 1 root root 28K 6月    3 21:18 copyinstall.log
-rw-r--r--. 1 root root    0 6月    3 21:40 files
-rw-r--r--. 1 root root    0 6月    3 21:40 program
-rw-r--r--. 1 root root    0 6月    3 21:38 tmp.log
-rw-----. 1 root root    0 6月    2 07:13 yum.log
[root@Node1 tmp]#
```

如果我们想创建一个文件名为 program files， 应该用引号括起来(尽量避免文件名包含空格)

touch "program files"

```
[root@Node1 tmp]# touch "program files"
[root@Node1 tmp]# ls -lh
总用量 28K
-rw-r--r--. 1 root root 28K 6月    3 21:18 copyinstall.log
-rw-r--r--. 1 root root    0 6月    3 21:41 program files
-rw-r--r--. 1 root root    0 6月    3 21:38 tmp.log
-rw-----. 1 root root    0 6月    2 07:13 yum.log
[root@Node1 tmp]#
```

二、显示文件内容命令（适合内容较少的文件）： cat

- ①、命令名称： cat
- ②、英文原意：
- ③、命令所在路径： /bin/cat
- ④、执行权限： 所有用户
- ⑤、功能描述： 显示文件内容（只能显示内容较少的文件）

⑥、语法： cat 【文件名】

-n 显示文件行号

例子：通过不加参数-n和加参数-n，显示/etc/issue的内容

```
[root@Node1 tmp]# cat /etc/issue
CentOS release 6.8 (Final)
Kernel \r on an \m

[root@Node1 tmp]# cat -n /etc/issue
 1  CentOS release 6.8 (Final)
 2  Kernel \r on an \m
 3
[root@Node1 tmp]#
```

注意：此命令只能显示文件内容比较少的文件，如果文件内容很多，用cat命令是不合适的，视觉效果是屏幕不断滚动更新。

三、反向显示文件内容命令（适合内容较少的文件）： tac

①、命令名称： tac

②、英文原意：

③、命令所在路径： /bin/tac

④、执行权限： 所有用户

⑤、功能描述： 显示文件内容（只能显示内容较少的文件）

⑥、语法： tac 【文件名】

例子：显示/etc/issue的内容

```
[root@Node1 tmp]# cat /etc/issue
CentOS release 6.8 (Final)
Kernel \r on an \m

[root@Node1 tmp]# tac /etc/issue
Kernel \r on an \m
CentOS release 6.8 (Final)
[root@Node1 tmp]#
```

四、分页显示文件内容命令（不能向前翻页）： more

①、命令名称： more

②、英文原意：

③、命令所在路径： /bin/more

④、执行权限：所有用户

⑤、功能描述：分页显示文件内容

⑥、语法： more 【文件名】

(空格) 或f 翻页 (一页一页的往后显示)

(Enter) 换行 (一行一行的往后显示)

q 或 Q 退出

例子：查看etc目录下的 services 文件信息：more /etc/services

```
tcpmux          1/tcp          # TCP port service multiplexer
tcpmux          1/udp          # TCP port service multiplexer
rje             5/tcp          # Remote Job Entry
rje             5/udp          # Remote Job Entry
echo            7/tcp          users
echo            7/udp          users
discard         9/tcp          sink null
discard         9/udp          sink null
systat          11/tcp         users
systat          11/udp         users
daytime         13/tcp
daytime         13/udp
qotd            17/tcp          quote
qotd            17/udp          quote
msp              18/tcp          # message send protocol
msp              18/udp          # message send protocol
chargen         19/tcp          ttyst source
chargen         19/udp          ttyst source
ftp-data        20/tcp
--More-- (0%)
```

五、分页显示文件内容命令（可以前后翻页）：less

①、命令名称：less

②、英文原意：

③、命令所在路径：/usr/bin/less

④、执行权限：所有用户

⑤、功能描述：分页显示文件内容

⑥、语法： less 【文件名】

(空格) 或f 或PgDn 翻页 (一页一页的往后显示)

PgUp向前翻页

(Enter) 换行或向下的箭头 (一行一行的往后显示)

向上的箭头 (一行一行的往前显示)

q 或 Q 退出

输入/想搜索的字符，然后回车键

例子：查看etc目录下的 services 文件信息：less/etc/services。这个比较简单，我们就不截图演示了，主要是最后提到的搜索功能，我们在进入命令之后，输入 /想搜索的字符，然后回车键，如下：

```
# /etc/services:  
# $Id: services,v 1.48 2009/11/11 14:32:31 ovasik Exp $  
#  
# Network services, Internet style  
# IANA services version: last updated 2009-11-10  
#  
# Note that it is presently the policy of IANA to assign a single well-known  
# port number for both TCP and UDP; hence, most entries here have two entries  
# even if the protocol doesn't support UDP operations.  
# Updated from RFC 1700, ``Assigned Numbers'' (October 1994). Not all ports  
# are included, only the more common ones.  
#  
# The latest IANA port assignments can be gotten from  
# http://www.iana.org/assignments/port-numbers  
# The Well Known Ports are those from 0 through 1023.  
# The Registered Ports are those from 1024 through 49151  
# The Dynamic and/or Private Ports are those from 49152 through 65535  
#  
# Each line describes one service, and is of the form:  
#  
# service-name port/protocol [aliases ...] [# comment]  
  
tcpmux      1/tcp          # TCP port service multiplexer  
tcpmux      1/udp          # TCP port service multiplexer  
/tcp【 我们输入/tcp，然后回车，搜索文件中tcp的字符】
```

回车之后，如下显示：

```
tcpmux          1/tcp          #
tcpmux          1/udp          #
rje            5/tcp          #
rje            5/udp          #
echo           7/tcp          #
echo           7/udp          #
discard         9/tcp          sink null
discard         9/udp          sink null
systat          11/tcp         users
systat          11/udp         users
daytime         13/tcp         #
daytime         13/udp         #
qotd            17/tcp         quote
qotd            17/udp         quote
msp              18/tcp         #
msp              18/udp         #
chargen         19/tcp         ttystt source
chargen         19/udp         ttystt source
ftp-data        20/tcp         #
ftp-data        20/udp         #
# 21 is registered to ftp, but also used by fsp
ftp              21/tcp         fspd
ftp              21/udp         fspd
ssh              22/tcp         #
: [red]
```

六、显示文件内容命令（指定行数）：head

- ①、命令名称：head
- ②、英文原意：
- ③、命令所在路径：/usr/bin/head
- ④、执行权限：所有用户
- ⑤、功能描述：显示文件的前面几行
- ⑥、语法： head 【文件名】

-n 指定显示的行数

不加 -n 默认显示前 20 行数据

例子：显示 etc 目录下的 services 文件前面 20 行 head -n 20 /etc/services

```
[root@Node1 tmp]# head -n 20 /etc/services
# /etc/services:
# $Id: services,v 1.48 2009/11/11 14:32:31 ovasik Exp $
#
# Network services, Internet style
# IANA services version: last updated 2009-11-10
#
# Note that it is presently the policy of IANA to assign a single well-known
# port number for both TCP and UDP; hence, most entries here have two entries
# even if the protocol doesn't support UDP operations.
# Updated from RFC 1700, ``Assigned Numbers'' (October 1994). Not all ports
# are included, only the more common ones.
#
# The latest IANA port assignments can be gotten from
#      http://www.iana.org/assignments/port-numbers
# The Well Known Ports are those from 0 through 1023.
# The Registered Ports are those from 1024 through 49151
# The Dynamic and/or Private Ports are those from 49152 through 65535
#
# Each line describes one service, and is of the form:
#
[root@Node1 tmp]#
```

七、反向文件内容命令（文件即时更新后也能动态显示，多用于日志文件显示）： **tail**

- ①、命令名称： tail
- ②、英文原意：
- ③、命令所在路径： /usr/bin/tail
- ④、执行权限： 所有用户
- ⑤、功能描述： 显示文件的后面几行
- ⑥、语法： tail 【文件名】

-n 指定显示的行数

-f 动态显示文件末尾内容（即文件实时变化，那么显示内容也会随之变化）

例子：显示 etc目录下的 services 文件后面 20 行 tail -n 20 -f /etc/services

```
[root@node3 c]# tail -n 20 -f /etc/services
ap                  47806/udp          # ALC Protocol
bacnet              47808/tcp          # Building Automation and Control Networks
bacnet              47808/udp          # Building Automation and Control Networks
nimcontroller       48000/tcp          # Nimbus Controller
nimcontroller       48000/udp          # Nimbus Controller
nimspooler           48001/tcp          # Nimbus Spooler
nimspooler           48001/udp          # Nimbus Spooler
nimhub               48002/tcp          # Nimbus Hub
nimhub               48002/udp          # Nimbus Hub
nimgtw               48003/tcp          # Nimbus Gateway
nimgtw               48003/udp          # Nimbus Gateway
3gpp-cbsp            48049/tcp          # 3GPP Cell Broadcast Service Protocol
isnetserv            48128/tcp          # Image Systems Network Services
isnetserv            48128/udp          # Image Systems Network Services
blp5                 48129/tcp          # Bloomberg locator
blp5                 48129/udp          # Bloomberg locator
com-bardac-dw        48556/tcp          # com-bardac-dw
com-bardac-dw        48556/udp          # com-bardac-dw
iqobject              48619/tcp          # iqobject
iqobject              48619/udp          # iqobject
```

如果我们想结束加上 -f显示的tail命令，通过 Ctrl+C。

Linux 常用命令之链接命令和权限管理命令

链接命令

一、生成链接文件命令：ln

- ①、命令名称：ln
- ②、英文原意：link
- ③、命令所在路径：/bin/link
- ④、执行权限：所有用户
- ⑤、功能描述：生成链接文件
- ⑥、语法： ln -s 【源文件】 【目标文件】

-s 创建软链接

不加 -s 创建硬链接

例子：一、创建文件 /etc/issue 的软链接 /tmp/issue.soft：ln -s /etc/issue /tmp/issue.soft

二、创建文件 /etc/issue 的硬链接 /tmp/issue.hard：ln /etc/issue /tmp/issue.hard

```
[root@Node1 ~]# ln -s /etc/issue /tmp/issue.soft
[root@Node1 ~]# ln /etc/issue /tmp/issue.hard
[root@Node1 ~]# ls -lh /tmp
总用量 32K
-rw-r--r--. 1 root root 28K 6月    3 21:18 copyinstall.log
-rw-r--r--. 2 root root 47 5月   19 2016 issue.hard
lrwxrwxrwx. 1 root root 10 6月   3 22:22 issue.soft -> /etc/issue
-rw-r--r--. 1 root root 0 6月    3 21:41 program files
-rw-r--r--. 1 root root 0 6月    3 21:38 tmp.log
-rw-----. 1 root root 0 6月    2 07:13 yum.log
[root@Node1 ~]#
```

我们可以看到：

第一：软链接 前面是 l 开头的 (link) ,而硬链接是 - 开头，表示文件

第二：软链接所有者和所属组具有全部操作的权限， rwxrwxrwx;而硬链接不是。也就是软链接的前面都是 lrwxrwxrwx

第三：软链接类似与 windows 的快捷方式，有一个明显的箭头指向，而指向的是源文件

第四：硬链接文件除了文件名与源文件不一样，其余所有信息都是一样的。类似于 cp 复制操作。但是又和复制不一样，硬链接可以同步更新。

第五：通过 ls -i 操作，来查看 文件的 i 节点。发现硬链接和源文件的 i 节点是相同的，而软链接与源文件的 i 节点是不同的

第六：不允许将硬链接指向目录；不允许跨分区创建硬链接

```
[root@Node1 ~]# ls -lh /boot
总用量 31M
-rw-r--r--. 1 root root 106K 5月   11 2016 config-2.6.32-642.el6.x86_64
drwxr-xr-x. 3 root root 1.0K 6月    2 07:19 efi
drwxr-xr-x. 2 root root 1.0K 6月    2 07:22 grub
-rw-----. 1 root root 25M 6月    2 07:22 initramfs-2.6.32-642.el6.x86_64.img
drwx-----. 2 root root 12K 6月    2 07:13 lost+found
-rw-r--r--. 1 root root 211K 5月   11 2016 symvers-2.6.32-642.el6.x86_64.gz
-rw-r--r--. 1 root root 2.5M 5月   11 2016 System.map-2.6.32-642.el6.x86_64
-rwrxr-xr-x. 1 root root 4.1M 5月   11 2016 vmlinuz-2.6.32-642.el6.x86_64
[root@Node1 ~]# ln /boot/efi /tmp/efi.hard
ln: "/boot/efi": 不允许将硬链接指向目录
[root@Node1 ~]# ln /boot/config-2.6.32-642.el6.x86_64 /tmp/config.log
ln: 创建硬链接"/tmp/config.log" => "/boot/config-2.6.32-642.el6.x86_64": 无效的跨设备连接
[root@Node1 ~]#
```

权限管理命令

一、更改文件或目录权限命令： chmod

①、命令名称： chmod

②、英文原意： change the permissions mode of a file

③、命令所在路径： /bin/chmod

④、执行权限：所有用户

⑤、功能描述：改变文件或目录权限

⑥、语法： chmod 【{ugoa}{+-}{rwx}】 【文件或目录】

【mode=421】 【文件或目录】

-R 递归修改

不是每一个Linux用户都有权限更改某个文件或目录权限，能更改文件或目录权限的只有两种用户

①、文件的所有者。我们通过ls命令查看某个文件的详细信息，可以看到该文件的所有者。

②、root用户，这不用多说，root用户是linux系统权限最大的用户。别人不能干的事，root用户都能干。

对于上面的语法 chmod 【{ugoa}{+-}{rwx}】 【文件或目录】，我们要知道ugoa分别是：u:表示所有者，g:表示所属组，o:表示其他人，a:表示所有人。而rwx表示的意思如下：

对于【mode=421】 【文件或目录】，这是我们将权限用数字表示，其中 r 表示4，w表示2，x表示1，分别是2的0次方，1次方，2次方。那么我们可以这样理解：具有 rwx 权限的数字就是 7，具有 rw- 权限的数字是 6，具有 r-- 权限的数字是 4。

范例1：我们赋予 tmp 目录下的 tmp.log 所有者 x 的权限；赋予 所属组 w 权限，其他人 w 权限。

```
chmod u+x /tmp/tmp.log
```

```
chmod g+w,o+w /tmp/tmp.log
```

```
[root@node3 tmp]# touch tmp.log
[root@node3 tmp]# ll
总用量 0
-rw-r--r--. 1 root root 0 10月 26 22:47 tmp.log
[root@node3 tmp]# chmod u+x /tmp/tmp.log
[root@node3 tmp]# ll
总用量 0
-rwxr--r--. 1 root root 0 10月 26 22:47 tmp.log
[root@node3 tmp]# chmod g+w,o+w /tmp/tmp.log
[root@node3 tmp]# ll
总用量 0
-rwxrw-rw-. 1 root root 0 10月 26 22:47 tmp.log
[root@node3 tmp]#
```

将上面例子改为用 数字来操作，也就是说我们要给 tmp.log 赋予的文件权限是 rwxrw-rw-，用数字表示是766。chmod 766 tmp.log

```
[root@node3 tmp]# touch tmp.log
[root@node3 tmp]# ll
总用量 0
-rw-r--r--. 1 root root 0 10月 26 22:50 tmp.log
[root@node3 tmp]# chmod 766 tmp.log
[root@node3 tmp]# ll
总用量 0
-rwxrw-rw-. 1 root root 0 10月 26 22:50 tmp.log
[root@node3 tmp]#
```

我们还可以递归赋予权限，也就是加上 -R 参数给指定目录下的所有文件或目录赋予指定权限。

范例2：给 tmp 目录下所有文件和目录赋予 776 的权限

```
chmod -R 776 /tmp
```

```
[root@node3 /]# chmod -R 776 /tmp
[root@node3 /]# cd /tmp
[root@node3 tmp]# ll
总用量 0
-rwxrwxrw-. 1 root root 0 10月 26 22:50 tmp.log
[root@node3 tmp]#
```

二、改变文件或目录所有者命令：chown

- ①、命令名称：chown
- ②、英文原意：change file ownership
- ③、命令所在路径：/bin/chown
- ④、执行权限：所有用户
- ⑤、功能描述：改变文件或目录的所有者
- ⑥、语法： chmod 【用户名】 【文件或目录】

注意：能更改文件或目录的所有者用户是 root

这里我们通过useradd 【用户名】 命令创建用户，然后通过passwd 【用户名】 输入密码，这两个命令后面会将。我们通过这两个命令创建 vae 用户

```
[root@Node1 tmp]# useradd vae
[root@Node1 tmp]# passwd vae
更改用户 vae 的密码。
新的 密码:
无效的密码: WAY 过短
无效的密码: 过于简单
重新输入新的 密码:
passwd: 所有的身份验证令牌已经成功更新。
[root@Node1 tmp]#
```

然后我们将tmp.log的所有者更改为 vae 用户: chown vae tmp.log

```
[root@node3 tmp]# ll
总用量 0
-rwxrwxrwx-. 1 root root 0 10月 26 22:50 tmp.log 首先tmp.log所有者是
[root@node3 tmp]# chown vae tmp.log 我们通过此命令更改所有者为 vae
[root@node3 tmp]# ll
总用量 0
-rwxrwxrwx-. 1 vae root 0 10月 26 22:50 tmp.log 所有者已经发生改变
[root@node3 tmp]#
```

三、改变文件或目录所属组命令: chgrp

- ①、命令名称: chgrp
- ②、英文原意: change file group ownership
- ③、命令所在路径: /bin/chown
- ④、执行权限: 所有用户
- ⑤、功能描述: 改变文件或目录的所属组
- ⑥、语法: chgrp 【用户组】 【文件或目录】

注意: 能更改文件或目录的所有者用户是 root

四、显示、设置文件的缺省权限命令: umask

- ①、命令名称: umask
- ②、英文原意: the user file-creation mask
- ③、命令所在路径: shell 内置命令
- ④、执行权限: 所有用户
- ⑤、功能描述: 显示、设置文件的缺省权限
- ⑥、语法: umask [-S]

-S 以rwx形式显示新建文件的缺省权限

注意：可能大家不太明白这个命令的意思，我们分别执行umask和 umask -S ,如下：

```
[root@node3 tmp]# umask  
0022  
[root@node3 tmp]# umask -S  
u=rwx,g=rx,o=rx  
[root@node3 tmp]#
```

其中 umask 执行显示结果是 0022,第一个0表示特殊权限，后面我们会单独进行讲解有哪几种特殊权限。022 表示权限的掩码值，我们用 7 7 7 减去 0 2 2得到755（是每一位相减），表示的就是下面通过加上 -S 输出的 rwxr-xr-x，这个值用数字表示就是 755.

这个意思说明创建一个文件的默认权限所有者为 rwx,所属组为 rx,其他人为 rx。也就是说创建一个新文件默认权限为 rwxr-xr-x，我们创建一个文件来验证一下：

```
[root@node3 tmp]# umask -S  
u=rwx,g=rx,o=rx  
[root@node3 tmp]# touch a.txt  
[root@node3 tmp]# ll a.txt  
-rw-r--r--. 1 root root 0 10月 27 21:10 a.txt  
[root@node3 tmp]#
```

我们发现使用 touch 命令创建了一个文件 a.txt，然后发现权限并不是 rwxr-xr-x，而是 rw-r--r--。对比发现少了三个 x，也就是少了可执行权限。这是为什么呢？

这是因为在 Linux 系统中，所有新创建的文件都是没有可执行权限的。这是出于 Linux 系统的一种自我保护，因为类似的病毒木马程序都是具有可执行权限的。所以在 Linux 系统中，新创建的文件是没有可执行权限的。

那么我们如何设置默认权限呢？比如我们想将新创建的文件权限设置为 rwxr-xr--，也就是7 54。我们用 777 减去754 得到 023。也就是通过执行 umask 023 来完成默认权限设置。

```
[root@node3 tmp]# umask -S  
u=rwx,g=rx,o=rx  
[root@node3 tmp]# umask 023 设置默认文件的权限为754  
[root@node3 tmp]# umask -S  
u=rwx,g=rx,o=r  
[root@node3 tmp]#
```

Linux 常用命令之文件搜索命令

最强大的搜索命令： find

首先进行一点说明，find 命令是我们在 Linux 系统中用来进行文件搜索用的最多的命令，功能特别强大。但是我们要说的是尽量少用 find 命令去执行搜索任务，就算要搜索我们也应该尽量的缩小范围，也不要在服务器使用高峰期进行文件搜索，因为搜索也是很占系统资源的。这就需要我们在进行 Linux 文件整理的时候，尽量规范化，什么文件放在什么目录下都要有比较好的约定。

find 这个命令如果要完全讲清楚，恐怕得花费很长的时间，而且很多用法我们几乎用不到，所以这里我就不一一介绍此命令的每一种用法了，我会详细介绍几种最常用的用法，大家只需要记住这几种就完全够我们日常使用了。

- ①、命令名称：find
- ②、英文原意：
- ③、命令所在路径：/bin/find
- ④、执行权限：所有用户
- ⑤、功能描述：进行各种花式文件搜索
- ⑥、语法：find 【搜索范围】 【匹配条件】

注意：Linux搜索和windows是有明显区别的，Linux严格区分文件大小写。

一、根据文件或目录名称 搜索

find 【搜索目录】 【-name或者-iname】 【搜索字符】： -name 和 -iname 的区别一个区分大小写，一个不区分大小写

- ①、find /etc -name init (精准搜索，名字必须为 init 才能搜索的到)
- ②、find /etc -iname init (精准搜索，名字必须为 init 或者有字母大写也能搜索的到)
- ③、find /etc -name *init (模糊搜索，以 init 结尾的文件或目录名)
- ④、find /etc -name init??? (模糊搜索，? 表示单个字符，即搜索到 init__)

```
[root@node3 tmp]# find /etc -name init
/etc/sysconfig/init
/etc/init
/etc/kdump-adv-conf/kdump_initscripts/init
[root@node3 tmp]# find /etc -name *init
/etc/sysconfig/init
/etc/rc.d/rc.sysinit
/etc/init
/etc/rc.sysinit
/etc/security/namespaces.init
/etc/kdump-adv-conf/kdump_initscripts/init
/etc/pam.d/run_init
/etc/X11/xinit
[root@node3 tmp]# find /etc -name init???
/etc/inittab
[root@node3 tmp]#
```

二、根据文件大小 搜索

比如：在根目录下查找大于 100M 的文件

```
find / -size +204800
```

这里 +n 表示大于， -n 表示小于， n 表示等于

1 数据块 == 512 字节 ==0.5KB, 也就是1KB等于2数据块

100MB == 102400KB==204800数据块

```
[root@node3 tmp]# find / -size +204800
/sys/devices/pci0000:00/0000:00:0f.0/resource1
/sys/devices/pci0000:00/0000:00:0f.0/resource1_wc
find: “/proc/18079/task/18079/fd/5”: 没有那个文件或目录
find: “/proc/18079/task/18079/fdinfo/5”: 没有那个文件或目录
find: “/proc/18079/fd/5”: 没有那个文件或目录
find: “/proc/18079/fdinfo/5”: 没有那个文件或目录
/usr/java/jdk-7u80-linux-x64.tar.gz
/home/hadoop/softwares/hadoop-2.7.3.tar.gz
/home/hadoop/softwares/eclipse-jee-mars-R-win32-x86_64.zip
[root@node3 tmp]# █
```

三、根据所有者和所属组 搜索

①、在home目录下查询所属组为 root 的文件

```
find /home -group root
```

②、在home目录下查询所有者为 root 的文件

```
find /home -user root
```

```
[root@node3 tmp]# find /home -group root
/home
/home/lost+found
[root@node3 tmp]# find /home -user root
/home
/home/lost+found
[root@node3 tmp]# ll /home
总用量 24
drwx----- 6 hadoop hadoop 4096 7月 19 17:54 hadoop
drwx----- 2 root root 16384 6月 8 03:44 lost+found
drwx----- 3 vae vae 4096 10月 26 22:59 vae
[root@node3 tmp]# █
```

四、根据时间属性 搜索

```
find 【路径】 【选项】 【时间】
```

选项有下面三种： -amin 访问时间

-cmin 文件属性被更改

-mmin 文件内容被修改

时间： +n,-n,n分别表示超过n分钟， n分钟以内和n分钟

范例： 在 /etc 目录下查找5 分钟内被修改过属性的文件和目录

```
find /etc -cmin -5
```

五、根据文件类型或i节点搜索

-type 根据文件类型查找： f表示文件， d表示目录， l表示软链接

范例： 查找 /home 目录下文件类型是目录的 find /home -type d

-inum 根据i节点查找

范例： 查找 /tmp 目录下i节点为400342的文件或目录 find /tmp -inum 400342

```
[root@node3 tmp]# ll -i
总用量 0
400342 -rw-r--r--. 1 root root 0 10月 27 21:10 a.txt
393740 -rwxrwxrwx-. 1 vae root 0 10月 26 22:50 tmp.log
[root@node3 tmp]# find /tmp -inum 400342
/tmp/a.txt
[root@node3 tmp]#
```

六、组合条件搜索

这里有两个参数：

①、-a 表示两个条件同时满足 (and)

②、-o 表示两个条件满足任意一个即可 (or)

范例： 查找/etc目录下大于80MB同时小于100MB的文件

```
find /etc -size +163840 -a -size -204800
```

在文件资料库中查找文件命令： locate

①、命令名称： locate

②、英文原意：

③、命令所在路径： /usr/bin/locate

④、执行权限： 所有用户

⑤、功能描述： 在文件资料库中查找文件

⑥、语法： locate 【文件名】

-i 不区分大小写

注意：这里和 find 命令是有区别的，find 是全盘检索，而 locate 是在文件资料库中进行搜索。所以 locate 命令的执行要比 find 命令执行速度快很多。但是这里有个问题，文件资料库是需要不断更新的。我们新创建的文件如果不更新文件资料库，使用 locate 是查找不到的。

updatedb 手动更新资料库，但是对于 /tmp 目录下的新建文件，是更新不到文件资料库的，因为 /tmp 目录不属于文件资料库的收录范围。

```
[root@Node1 tmp]# locate inittab
/etc/inittab
/usr/share/augeas/lenses/dist/inittab.aug
/usr/share/man/man5/inittab.5.gz
/usr/share/vim/vim74/syntax/inittab.vim
[root@Node1 tmp]# mkdir -p /tmp/locateTest
[root@Node1 tmp]# locate locateTest
[root@Node1 tmp]# updatedb
[root@Node1 tmp]# locate locateTest
[root@Node1 tmp]# mkdir -p /home/locateHomeTest
[root@Node1 tmp]# locate locateHomeTest
[root@Node1 tmp]# updatedb
[root@Node1 tmp]# locate locateHomeTest
/home/locateHomeTest
[root@Node1 tmp]#
```

搜索命令所在的目录及别名信息：which

- ①、命令名称：which
- ②、英文原意：
- ③、命令所在路径：/usr/bin/which
- ④、执行权限：所有用户
- ⑤、功能描述：搜索命令所在的目录及别名信息
- ⑥、语法：which 【命令】

范例：查询 ls 命令所在目录以及别名信息

```
[root@Node1 tmp]# which ls
alias ls='ls --color=auto'
/bin/ls
[root@Node1 tmp]#
```

搜索命令所在的目录及帮助文档路径：whereis

- ①、命令名称：whereis

- ②、英文原意：
- ③、命令所在路径：/usr/bin/whereis
- ④、执行权限：所有用户
- ⑤、功能描述：搜索命令所在的目录及帮助文档路径
- ⑥、语法：whereis 【命令】

范例：查询 ls 命令所在目录以及帮助文档路径

```
[root@Node1 tmp]# whereis ls
ls: /bin/ls /usr/share/man/man1p/ls.1p.gz /usr/share/man/man1/ls.1.gz
[root@Node1 tmp]#
```

在文件中搜寻字符串匹配的行并输出：grep

- ①、命令名称：grep
- ②、英文原意：
- ③、命令所在路径：/bin/grep
- ④、执行权限：所有用户
- ⑤、功能描述：在文件中搜寻字符串匹配的行并输出
- ⑥、语法：grep -iv 【指定字符串】 【文件】

-i 不区分大小写

-v 排除指定字符串

范例：查找 /root/install.log 文件中包含 mysql 字符串的行，并输出

```
grep mysql /root/install.log
```

```
[root@Node1 tmp]# grep mysql /root/install.log
安装 mysql-libs-5.1.73-7.el6.x86_64
[root@Node1 tmp]#
```

Linux 常用命令之帮助和用户管理命令

帮助命令

一、获得命令或配置文件帮助信息：man

- ①、命令名称：man
- ②、英文原意：manual

③、命令所在路径： /usr/bin/man

④、执行权限：所有用户

⑤、功能描述：获得帮助信息

⑥、语法： man 【命令或配置文件】

范例1：查看 ls 命令的帮助信息：man ls

```
LS(1)                               User Commands                               LS(1)

NAME
    ls - list directory contents

SYNOPSIS
    ls [OPTION]... [FILE]...

DESCRIPTION
    List information about the FILEs (the current directory by default). Sort
    entries alphabetically if none of -cftuvSUX nor --sort.

    Mandatory arguments to long options are mandatory for short options too.

    -a, --all
        do not ignore entries starting with .

    -A, --almost-all
        do not list implied . and ..

    --author
        with -l, print the author of each file

    -b, --escape
:
```

范例2：查看配置文件 services 的帮助信息：man services。

注意查看配置文件不需要加上绝对路径，如果是 man /etc/services，那么显示的就是services的文件内容。

NAME

services - Internet network services list

DESCRIPTION

services is a plain ASCII file providing a mapping between human-friendly textual names for internet services, and their underlying assigned port numbers and protocol types. Every networking program should look into this file to get the port number (and protocol) for its service. The C library routines **getservent(3)**, **getservbyname(3)**, **getservbyport(3)**, **setservent(3)**, and **endservent(3)** support querying this file from programs.

Port numbers are assigned by the IANA (Internet Assigned Numbers Authority), and their current policy is to assign both TCP and UDP protocols when assigning a port number. Therefore, most entries will have two entries, even for TCP-only services.

Port numbers below 1024 (so-called "low numbered" ports) can only be bound to by root (see **bind(2)**, **tcp(7)**, and **udp(7)**). This is so clients connecting to low numbered ports can trust that the service running on the port is the standard implementation, and not a rogue service run by a user of the machine. Well-known port numbers specified by the IANA are normally located in this root-only space.

:
■

如果以文件即是命令，又是配置文件，比如man passwd，系统是优先显示命令帮助信息的。如果你想看passwd的配置文件信息，可以 man 5 passwd。因为5表示配置文件信息，1表示命令帮助信息。

二、获得shell内置命令的帮助信息：help

- ①、命令名称： help
- ②、英文原意：
- ③、命令所在路径： shell 内置命令
- ④、执行权限：所有用户
- ⑤、功能描述： 获得shell内置命令帮助信息
- ⑥、语法： help 【shell内置命令】

范例：查看 umask 命令的帮助信息： help umask

我们如何判断一个命令是否是shell内置命令呢？前面我们讲过which命令，是用来搜索命令所在的目录及别名信息，如果使用which找不到该命令的所在路径，那么此命令就是shell内置命令。

```
[root@node3 tmp]# which mkdir
/bin/mkdir
[root@node3 tmp]# which umask
/usr/bin/which: no umask in (/usr/lib64/qt-3.3/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/usr/java/jdk1.7/bin:/home/hadoop/hbase0.99/bin:/home/hadoop2.7/bin:/home/hadoop/modules/hadoop2.7/sbin:/root/bin)
[root@node3 tmp]#
```

下面的命令都是shell内置命令：

NAME

```
bash, :, ., [, alias, bg, bind, break, builtin, caller, cd, command, compgen, complete, compopt, continue, declare, dirs, disown, echo, enable, eval, exec, exit, export, false, fc, fg, getopt, hash, help, history, jobs, kill, let, local, logout, mapfile, popd, printf, pushd, pwd, read, readonly, return, set, shift, shopt, source, suspend, test, times, trap, true, type, typeset, ulimit, umask, unalias, unset, wait - bash built-in commands, see bash(1)
```

三、获得命令的中文帮助信息：--help

这里给大家一个简单的技巧，如果我们想简单的查看命令的帮助信息，而不是上面显示的很多内容，那该怎么办呢？

我们执行此命令：命令 --help

范例：查看 ls 的帮助信息

```
[root@node3 tmp]# ls --help
```

用法：ls [选项]... [文件]...

列出 FILE 的信息(默认为当前目录)。

如果不指定-cftuvSUX 或--sort 选项，则根据字母大小排序。

长选项必须使用的参数对于短选项时也是必需使用的。

-a, --all	不隐藏任何以. 开始的项目
-A, --almost-all	列出除. 及.. 以外的任何项目
--author	与-l 同时使用时列出每个文件的作者
-b, --escape	以八进制溢出序列表示不可打印的字符
--block-size=大小	块以指定大小的字节为单位
-B, --ignore-backups	不列出任何以"~"字符结束的项目
-c	配合-lt: 根据ctime 排序并显示ctime(文件状态最后更改的时间)
	配合-l: 显示ctime 但根据名称排序

其他情况：按ctime 排序

-C	每栏由上至下列出项目
--color[=WHEN]	控制是否使用色彩分辨文件。WHEN 可以是

用户管理命令

一、添加新用户：useradd

①、命令名称：useradd

②、英文原意：

③、命令所在路径：/usr/sbin/useradd

④、执行权限：root

⑤、功能描述：添加新用户

⑥、语法： useradd 【用户名】

范例：添加用户tom:useradd tom

```
[root@node3 tmp]# useradd tom  
[root@node3 tmp]# █
```

二、设置用户密码：passwd

- ①、命令名称：passwd
- ②、英文原意：
- ③、命令所在路径：/usr/bin/passwd
- ④、执行权限：root
- ⑤、功能描述：修改用户的密码
- ⑥、语法：passwd 【用户名】

注意：root用户能修改任何用户的密码。而普通用户只能修改自己的密码，而且密码要符合密码规则，不然修改不了

范例：修改用户tom的密码

```
[root@node3 tmp]# passwd tom  
更改用户 tom 的密码。  
新的 密码：  
无效的密码： WAY 过短  
无效的密码： 过于简单  
重新输入新的 密码：  
passwd: 所有的身份验证令牌已经成功更新。  
[root@node3 tmp]# █
```

三、查看登录用户简单信息：who

- ①、命令名称：who
- ②、英文原意：
- ③、命令所在路径：/usr/bin/who
- ④、执行权限：root
- ⑤、功能描述：查看登录用户简单信息
- ⑥、语法：who

范例：查看当前登录用户的信息

```
[root@node3 tmp]# who  
root pts/0 2017-10-24 08:00 (192.168.146.1)  
[root@node3 tmp]# █
```

四、查看登录用户详细信息：w

- ①、命令名称：w
- ②、英文原意：
- ③、命令所在路径：/usr/bin/w
- ④、执行权限：root
- ⑤、功能描述：查看登录用户详细信息
- ⑥、语法：w

范例：查看当前登录用户的详细信息

```
[root@node3 tmp]# w
13:46:28 up 4 days, 5:46, 1 user, load average: 0.00, 0.01, 0.05
USER      TTY      FROM          LOGIN@    IDLE     JCPU   PCPU WHAT
root      pts/0    192.168.146.1    Tue08    0.00s  1.31s  0.67s w
[root@node3 tmp]#
```

第一排 13:46:28:当前系统的时间 up 4 days: Linux连续运行时间4天 1 user: 表示当前1个用户登录
load average:0.00,0.01,0.05:系统的负载情况，分别表示过去一分钟，五分钟和十五分钟系统的负载情况。

USER: 登录用户名

TTY: 登录终端，pts 表示远程终端，tty 表示本地终端

FROM: 登录的主机 IP，如果没有写表示本机登录

Linux 常用命令之压缩和解压缩命令

压缩解压缩格式 .gz

一、将文件压缩为 .gz 格式,只能压缩文件：gzip

- ①、命令名称：gzip
- ②、英文原意：GNU zip
- ③、命令所在路径：/bin/gzip
- ④、执行权限：所有用户
- ⑤、功能描述：压缩文件，压缩后格式为.gz
- ⑥、语法： gzip 【需要压缩的文件】
- ⑦、压缩后文件格式：.gz

注意：只能压缩文件，不能压缩目录；压缩完后不保留原文件

范例：我们在tmp目录下创建一个文件tmpgz，然后通过gzip tmpgz 压缩该文件，压缩后的文件为tmpgz.gz

```
[root@node3 tmp]# touch tmpgz
[root@node3 tmp]# ll
总用量 0
-rw-r--r--. 1 root root 0 10月 28 20:38 tmpgz
[root@node3 tmp]# gzip tmpgz
[root@node3 tmp]# ll
总用量 4
-rw-r--r--. 1 root root 26 10月 28 20:38 tmpgz.gz
[root@node3 tmp]#
```

二、将 .gz 文件解压：gunzip

- ①、命令名称：gunzip
- ②、英文原意：GNU unzip
- ③、命令所在路径：/bin/gunzip
- ④、执行权限：所有用户
- ⑤、功能描述：将格式为.gz的压缩文件解压
- ⑥、语法：gunzip 【压缩文件名】

注意：解压后不保留原文件

范例：我们将上面压缩后的文件tmpgz.gz解压：gunzip tmpgz.gz

```
[root@node3 tmp]# ll
总用量 4
-rw-r--r--. 1 root root 26 10月 28 20:38 tmpgz.gz
[root@node3 tmp]# gunzip tmpgz.gz
[root@node3 tmp]# ll
总用量 0
-rw-r--r--. 1 root root 0 10月 28 20:38 tmpgz
[root@node3 tmp]#
```

压缩解压缩格式 .tar.gz

一、将文件或目录压缩为 .tar.gz 格式：tar -zcf

- ①、命令名称：tar
- ②、英文原意：
- ③、命令所在路径：/bin/tar
- ④、执行权限：所有用户

⑤、功能描述：将文件压缩为.tar.gz格式

⑥、语法：tar 选项 【-zcf】 【压缩后文件名】 【目录】

-c 打包	-v 显示详细信息	-f 指定文件名
-z 打包同时压缩		

⑦、压缩后文件格式：.tar.gz

范例：在/tmp目录下创建a目录，然后在a目录下创建文件a.txt，通过tar -zcvf a.tar.gz a 命令将a目录压缩为a.tar.gz文件

```
[root@node3 tmp]# ll
总用量 4
drwxr-xr-x. 2 root root 4096 10月 28 21:33 a
-rw-r--r--. 1 root root     0 10月 28 20:38 tmpgz
[root@node3 tmp]# tar -zcvf a.tar.gz a
a/
a/a.txt
[root@node3 tmp]# ll
总用量 8
drwxr-xr-x. 2 root root 4096 10月 28 21:33 a
-rw-r--r--. 1 root root   135 10月 28 21:33 a.tar.gz
-rw-r--r--. 1 root root     0 10月 28 20:38 tmpgz
[root@node3 tmp]# 
```

与前面的gzip命令不同，通过tar压缩后是保留原文件或原目录的。

二、将 .tar.gz 文件解压：tar -zxf

①、命令名称：tar

②、英文原意：

③、命令所在路径：/bin/tar

④、执行权限：所有用户

⑤、功能描述：将格式为.tar.gz的压缩文件解压

⑥、语法：tar 选项 【-zxf】 【.tar.gz的压缩文件名】 【指定解压后的文件存放目录，默认当前目录】

-x 解包	
-v 显示详细信息	
-f 指定解压文件	
-z 解压缩	

范例：将a.tar.gz文件解压：tar -zxf a.tar.gz

```
[root@node3 tmp]# tar -zxf a.tar.gz
[root@node3 tmp]# ll
总用量 12
drwxr-xr-x. 2 root root 4096 10月 28 21:33 a
-rw-r--r--. 1 root root 135 10月 28 21:33 a.tar.gz
-rw-r--r--. 1 root root 0 10月 28 20:38 tmpgz
```

压缩解压缩格式 .zip

一、将文件或目录压缩为 .zip 格式：zip

- ①、命令名称：zip
- ②、英文原意：
- ③、命令所在路径：/usr/bin/zip
- ④、执行权限：所有用户
- ⑤、功能描述：将文件或目录压缩为.zip格式
- ⑥、语法： zip 选项 【-r】 【压缩后文件名】 【文件或目录】

-r 压缩目录

- ⑦、压缩后文件格式：.zip

范例：在/tmp目录下创建a目录，然后执行命令：zip -r a.zip a，将a目录压缩为 a.zip 文件

```
[root@node3 tmp]# mkdir a
[root@node3 tmp]# zip -r a.zip a
adding: a/ (stored 0%)
[root@node3 tmp]# ll
总用量 8
drwxr-xr-x. 2 root root 4096 10月 28 21:53 a
-rw-r--r--. 1 root root 154 10月 28 21:54 a.zip
[root@node3 tmp]#
```

通过zip压缩后是保留原文件或原目录的。

二、将 .zip 文件解压：unzip

- ①、命令名称：unzip
- ②、英文原意：
- ③、命令所在路径：/usr/bin/unzip
- ④、执行权限：所有用户
- ⑤、功能描述：将格式为.zip的压缩文件解压
- ⑥、语法： unzip 【.zip的压缩文件名】

范例：将a.zip文件解压：unzip a.zip

```
[root@node3 tmp]# ll  
总用量 4  
-rw-r--r--. 1 root root 154 10月 28 21:54 a.zip  
[root@node3 tmp]# unzip a.zip  
Archive: a.zip  
  creating: a/  
[root@node3 tmp]# ll  
总用量 8  
drwxr-xr-x. 2 root root 4096 10月 28 21:53 a  
-rw-r--r--. 1 root root 154 10月 28 21:54 a.zip  
[root@node3 tmp]#
```

解压之后也是保留原文件的

压缩解压缩格式 .bz2

一、将文件压缩为 .bz2 格式，只能压缩文件： bzip2

- ①、命令名称： bzip2
- ②、英文原意：
- ③、命令所在路径： /usr/bin/bzip2
- ④、执行权限： 所有用户
- ⑤、功能描述： 将文件压缩为.bz2 格式
- ⑥、语法： bzip2 选项 【-k】 【文件】
 - k 产生压缩文件后保留原文件

- ⑦、压缩后文件格式： .bz2

范例： 在/tmp目录下创建a文件， 然后执行命令： bzip2 -k a， 将a文件压缩为a.bz2文件。

```
[root@node3 tmp]# touch a  
[root@node3 tmp]# ll  
总用量 0  
-rw-r--r--. 1 root root 0 10月 28 22:06 a  
[root@node3 tmp]# bzip2 -k a  
[root@node3 tmp]# ll  
总用量 4  
-rw-r--r--. 1 root root 0 10月 28 22:06 a  
-rw-r--r--. 1 root root 14 10月 28 22:06 a.bz2  
[root@node3 tmp]#
```

二、将 .bz2 文件解压： bunzip2

- ①、命令名称： bunzip2
- ②、英文原意：
- ③、命令所在路径： /usr/bin/bunzip2

- ④、执行权限：所有用户
- ⑤、功能描述：将格式为.bz2的压缩文件解压
- ⑥、语法：bunzip2 选项 【-k】 【压缩文件】

-k 解压缩文件后保留原文件

范例：将a.bz2 文件解压：bunzip2 a.bz2

```
[root@node3 tmp]# ll  
总用量 4  
-rw-r--r--. 1 root root 14 10月 28 22:06 a.bz2  
[root@node3 tmp]# bunzip2 a.bz2  
[root@node3 tmp]# ll  
总用量 0  
-rw-r--r--. 1 root root 0 10月 28 22:06 a  
[root@node3 tmp]#
```

不加参数k，解压之后不留原文件

Linux 常用命令之网络和关机重启命令

网络命令

一、给指定用户发送信息：write

- ①、命令名称：write
- ②、英文原意：
- ③、命令所在路径：/usr/bin/write
- ④、执行权限：所有用户
- ⑤、功能描述：给指定用户发送信息，以Ctrl+D 保存结束
- ⑥、语法： write 【用户名】

范例：给vae用户发送信息：write vae

二、给所有用户发送广播信息：wall

- ①、命令名称：wall
- ②、英文原意：write all
- ③、命令所在路径：/usr/bin/wall
- ④、执行权限：所有用户
- ⑤、功能描述：发送广播信息

⑥、语法： wall 【信息内容】

范例：发送广播信息：wall hello linux!!!

三、测试网络连通性：ping

①、命令名称： ping

②、英文原意：

③、命令所在路径： /bin/ping

④、执行权限：所有用户

⑤、功能描述： 测试网络连通性

⑥、语法： ping 【-c n】 【IP地址】

-c n是指定发送次数，如果不指定次数，那么将不断发送连接信息

范例： 测试与百度的连接： ping www.baidu.com

```
[root@node3 tmp]# ping -c 4 www.baidu.com
PING www.a.shifen.com (61.135.169.125) 56(84) bytes of data.
64 bytes from 61.135.169.125: icmp_seq=1 ttl=128 time=37.2 ms
64 bytes from 61.135.169.125: icmp_seq=2 ttl=128 time=21.3 ms
64 bytes from 61.135.169.125: icmp_seq=3 ttl=128 time=30.2 ms
64 bytes from 61.135.169.125: icmp_seq=4 ttl=128 time=30.8 ms

--- www.a.shifen.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3036ms
rtt min/avg/max/mdev = 21.341/29.924/37.291/5.679 ms
[root@node3 tmp]#
```

四、查看和设置网卡信息：ifconfig

①、命令名称： ifconfig

②、英文原意： interface configure

③、命令所在路径： /sbin/ifconfig

④、执行权限： root

⑤、功能描述： 查看和设置网卡信息

⑥、语法： ifconfig 【网卡名称】 【IP地址】

范例1： 查看本机网卡信息： ifconfig

```
[root@node3 tmp]# ifconfig
eth0      Link encap:Ethernet HWaddr 00:0C:29:E0:DB:80
          inet addr:192.168.146.253 Bcast:192.168.146.255 Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fee0:db80/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
            RX packets:116633 errors:0 dropped:0 overruns:0 frame:0
            TX packets:20228 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:7586764 (7.2 MiB) TX bytes:2051167 (1.9 MiB)

lo       Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
            UP LOOPBACK RUNNING MTU:65536 Metric:1
            RX packets:0 errors:0 dropped:0 overruns:0 frame:0
            TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:0
            RX bytes:0 (0.0 b) TX bytes:0 (0.0 b)

[root@node3 tmp]#
```

范例2：查看eth0 的网卡信息：ifconfig eth0

```
[root@node3 tmp]# ifconfig eth0
eth0      Link encap:Ethernet HWaddr 00:0C:29:E0:DB:80
          inet addr:192.168.146.253 Bcast:192.168.146.255 Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fee0:db80/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
            RX packets:116800 errors:0 dropped:0 overruns:0 frame:0
            TX packets:20312 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:7600954 (7.2 MiB) TX bytes:2061645 (1.9 MiB)

[root@node3 tmp]#
```

五、查看发送电子邮件：mail

- ①、命令名称： mail
- ②、英文原意：
- ③、命令所在路径： /bin/mail
- ④、执行权限：所有用户
- ⑤、功能描述：查看发送电子邮件
- ⑥、语法： mail 【用户名】

范例：给root用户发送邮件： mail root

六、列出所有登录系统的用户信息：last

- ①、命令名称： last
- ②、英文原意：

③、命令所在路径：/usr/bin/last

④、执行权限：所有用户

⑤、功能描述：列出所有登录过系统的用户信息

⑥、语法：last

范例：查看所有登录系统的用户：last

```
[root@node3 tmp]# last
root      pts/1          192.168.146.1   Sun Oct 29 10:20  still logged in
root      pts/0          192.168.146.1   Tue Oct 24 08:00  still logged in
reboot    system boot  2.6.32-642.el6.x  Tue Oct 24 08:00 - 10:33 (5+02:33)
reboot    system boot  2.6.32-642.el6.x  Fri Sep  8 19:32 - 10:33 (50+15:01)
root      pts/0          192.168.146.1   Wed Jul 19 15:35 - down   (04:31)
reboot    system boot  2.6.32-642.el6.x  Wed Jul 19 15:34 - 20:06 (04:31)
root      pts/0          192.168.146.1   Wed Jul 19 15:26 - down   (-6:-48)
reboot    system boot  2.6.32-642.el6.x  Wed Jul 19 15:16 - 08:38 (-6:-37)
hadoop    pts/1          node3           Fri Jun 30 04:10 - 04:10 (00:00)
hadoop    pts/1          node1           Fri Jun 30 04:08 - 04:08 (00:00)
hadoop    pts/1          node0           Fri Jun 30 04:05 - 04:05 (00:00)
root      pts/0          192.168.146.1   Fri Jun 30 03:33 - down   (00:40)
reboot    system boot  2.6.32-642.el6.x  Fri Jun 30 03:32 - 04:13 (00:40)
root      ttym1          Fri Jun 30 03:22 - down   (00:09)
reboot    system boot  2.6.32-642.el6.x  Fri Jun 30 03:18 - 03:32 (00:14)
root      pts/0          192.168.146.1   Fri Jun 30 02:56 - down   (00:03)
reboot    system boot  2.6.32-642.el6.x  Fri Jun 30 02:55 - 03:00 (00:04)
root      pts/0          192.168.146.1   Thu Jun 29 06:07 - down   (20:47)
reboot    system boot  2.6.32-642.el6.x  Thu Jun 29 06:06 - 02:55 (20:49)
root      pts/0          192.168.146.1   Thu Jun 29 06:01 - crash  (13+14:04)
```

七、显示数据包到主机间的路径：traceroute

①、命令名称：traceroute

②、英文原意：

③、命令所在路径：/usr/bin/traceroute

④、执行权限：所有用户

⑤、功能描述：显示数据包到主机间的路径

⑥、语法：traceroute 【IP地址】

八、显示网络相关信息：netstat

①、命令名称：netstat

②、英文原意：

③、命令所在路径：/bin/netstat

④、执行权限：所有用户

⑤、功能描述：显示网络相关信息

⑥、语法：netstat 【选项】

- t TCP协议
- u UDP协议
- l 监听
- r 路由
- n 显示IP地址和端口号

范例1：netstat -tlun 查看本机监听的端口

范例2：netstat -an 查看本机所有的网络连接

范例3：netstat -rn 查看本机路由表

九、配置网络：setup

- ①、命令名称：setup
- ②、英文原意：
- ③、命令所在路径：/usr/bin/setup
- ④、执行权限：root
- ⑤、功能描述：配置网络，比如IP地址，子网掩码等
- ⑥、语法：setup



十、挂载命令：mount

- ①、命令名称：mount
- ②、英文原意：

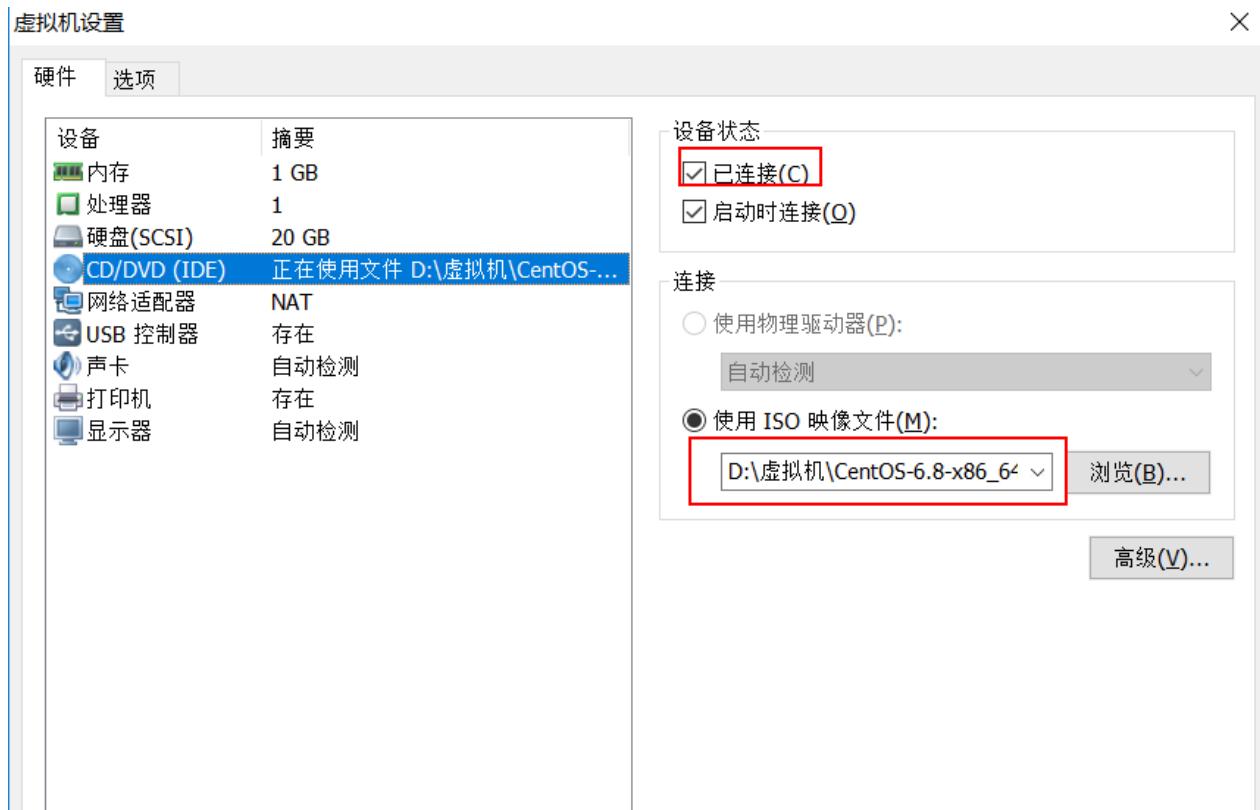
③、命令所在路径：/bin/mount

④、执行权限：所有用户

⑤、功能描述：给光盘、U盘等外界存储设备分配类似于windows系统的盘符，便于访问

⑥、语法：mount 【-t 文件系统】设备文件名 挂载点

范例：在虚拟机光盘里面放入一张CD



将光盘挂载到 /mnt/cdrom 目录下：mount -t iso9660 /dev/sr0 /mnt/cdrom。-t iso9660 可以省略

```
[root@node3 cdrom]# mount -t iso9660 /dev/sr0 /mnt/cdrom
mount: block device /dev/sr0 is write-protected, mounting read-only
```

卸载：umount /dev/sr0

关机重启命令

一、shutdown命令(推荐使用)

①、命令名称：shutdown

②、英文原意：

③、命令所在路径：/sbin/shutdown

④、执行权限：root

⑤、功能描述：进行关机重启操作

⑥、语法：shutdown 【选项】 【时间】

-c 取消前一个关机命令

-h 关机

-r 重启

范例1：马上关机：shutdown -h now

范例2：晚上八点半关机：shutdown -h 20:30

注意：推荐使用该命令进行关机，此命令关机之前会正确的关闭系统的服务。

二、其他关机命令

①、halt

②、poweroff

③、init0

三、其他重启命令

①、reboot

②、init6

四、退出登录命令

①、logout

Linux 软件包管理之RPM命令

Linux 软件包分类

一、源码包

源码包能直接看到源码，安装时需要人为手工设置安装位置，一般是/usr/local/软件名/。源码包的升级版——脚本安装包，人为的改动源码使其有安装界面。

优点：

- ①、开源，如果有足够的能力，可以直接修改源代码。
- ②、安装时可以自由选择所需的功能。
- ③、软件是编译安装，所以更加适合自己的系统，更加稳定也效率更高。
- ④、卸载方便，直接删了你安装软件的那个目录就好了。

缺点：

- ①、安装步骤较多，尤其安装较大的软件集合时，容易出现拼写错误。
- ②、编译时间过长，比后面讲的二进制安装时间长。
- ③、因为是编译安装，安装过程中一旦报错新手很难解决。

二、二进制包

这里的二进制包最要包括centos中的rpm包。 rpm包有默认安装位置。 /etc/ 配置文件安装目录； /usr/bin/ 可执行的命令安装目录； /usr/lib/ 程序所使用的函数库保存位置； /usr/share/doc/ 基本的软件使用手册保存位置； /usr/share/man/ 帮助文件保存位置。

优点：

- ①、包管理系统简单， 只通过几个命令就可以实现包的安装、升级、查询和卸载。
- ②、安装速度比源码包快很多。

缺点： CentOS-6.8-x86_64-bin-DVD1.iso

- ①、已经经过编译了， 看不到源代码。
- ②、功能选择不如源代码包灵活。
- ③、安装过程中有很多依赖的包， 依赖性不好解决。

rpm 包命名规则

我们打开前面安装Linux系统的安装文件 CentOS-6.8-x86_64-bin-DVD1.iso。在package目录下都是 rpm安装文件。

 hunspell-as-1.0.3-3.1.el6.noarch.rpm	87.42 KB
 hunspell-ar-0.20080110-4.1.el6.noarch.rpm	267.14 KB
 hunspell-af-0.20080825-3.1.el6.noarch.rpm	562.69 KB
 hunspell-1.2.8-16.el6.x86_64.rpm	176.52 KB
 httpd-tools-2.2.15-53.el6.centos.x86_64.rpm	78.44 KB
 httpd-manual-2.2.15-53.el6.centos.noarch.rpm	788.86 KB
 httpd-devel-2.2.15-53.el6.centos.x86_64.rpm	155.90 KB
 httpd-2.2.15-53.el6.centos.x86_64.rpm	833.03 KB
 htdig-3.2.0-0.10.b6.el6.x86_64.rpm	839.61 KB
 hsqldb-1.8.0.10-12.el6.noarch.rpm	671.66 KB
 hplip-libs-3.14.6-3.el6.x86_64.rpm	144.23 KB
 hplip-gui-3.14.6-3.el6.x86_64.rpm	1.45 MB
 hplip-common-3.14.6-3.el6.x86_64.rpm	79.30 KB
 hplip-3.14.6-3.el6.x86_64.rpm	6.04 MB
 hpijs-3.14.6-3.el6.x86_64.rpm	5.27 MB
 hmaccalc-0.9.12-2.el6.x86_64.rpm	21.65 KB

我们以 httpd-2.2.15-53.el6.centos.x86_64.rpm 文件为例来看 rpm 包的命名规则。

- ①、httpd:软件包名

- ②、2.2.15：软件版本
- ③、15：软件发布的次数
- ④、el6.centos：适合的Linux平台
- ⑤、x86_64：适合的硬件平台，这里表示64位。
- ⑥、rpm：rpm包的扩展名，注意Linux是没有扩展名的概念，这里是告诉管理员这是一个rpm包文件。

rpm 包安装

rpm -ivh 包全名

选项：

-i (install)	安装
-v (verbose)	显示详细信息
-h (hash)	显示进度
--nodeps	不检测依赖性

范例：以安装 httpd-2.2.15-53.el6.centos.x86_64.rpm 为例：

我们进入到该rpm文件的挂载目录 /mnt/cdrom/Packages，执行命令： rpm -ivh httpd-2.2.15-53.el6.centos.x86_64.rpm

这种安装方法可能需要先安装各种依赖，安装过程特别繁琐，后面讲解yum安装会简单的多。

rpm 包升级

rpm -Uvh 包全名

选项：

-U (upgrade)	升级
---------------------	----

rpm 包卸载

rpm -e 包名

选项：

-e (erase) 卸载

--nodeps 不检查依赖性

查询 rpm 包是否安装

[root@localhost ~]# rpm -q 包名

#查询包是否安装

选项：

-q 查询 (query)

[root@localhost ~]# rpm -qa

#查询所有已经安装的RPM包

选项：

-a 所有 (all)

```
[root@node3 Packages]# rpm -q httpd  
httpd-2.2.15-60.el6.centos.6.x86_64  
[root@node3 Packages]#
```

查询软件包的详细信息

[root@localhost ~]# rpm -qi 包名
选项：

- i 查询软件信息 (information)
- p 查询未安装包信息 (package)

```
[root@node3 Packages]# rpm -qi httpd
Name        : httpd                         Relocations: (not relocatable)
Version     : 2.2.15                         Vendor: CentOS
Release     : 60.el6.centos.6                 Build Date: 2017年10月20日 星期五 00时44分47秒
Install Date: 2017年11月01日 星期三 21时48分43秒   Build Host: c1bl.rdu2.centos.org
Group       : System Environment/Daemons    Source RPM: httpd-2.2.15-60.el6.centos.6.src.rpm
Size        : 3166418                          License: ASL 2.0
Signature   : RSA/SHA1, 2017年10月21日 星期六 01时13分17秒, Key ID 0946fca2c105b9de
Packager    : CentOS BuildSystem <http://bugs.centos.org>
URL         : http://httpd.apache.org/
Summary     : Apache HTTP Server
Description :
The Apache HTTP Server is a powerful, efficient, and extensible
web server.
[root@node3 Packages]#
```

查询软件包的安装位置

[root@localhost ~]# rpm -ql 包名
选项：

- l 列表 (list)
- p 查询未安装包信息 (package)

```
[root@node3 Packages]# rpm -ql httpd
/etc/httpd
/etc/httpd/conf
/etc/httpd/conf.d
/etc/httpd/conf.d/README
/etc/httpd/conf.d/welcome.conf
/etc/httpd/conf/httpd.conf
/etc/httpd/conf/magic
/etc/httpd/logs
/etc/httpd/modules
/etc/httpd/run
/etc/logrotate.d/httpd
/etc/rc.d/init.d/htcacheload
/etc/rc.d/init.d/httpd
/etc/sysconfig/htcacheload
```

查询系统文件属于哪个RPM包

[root@localhost ~]# rpm -qf 系统文件名
选项：

-f 查询系统文件属于哪个软件包（file）

查询软件包的依赖性

[root@localhost ~]# rpm -qR 包名
选项：

-R 查询软件包的依赖性（requires）

-p 查询未安装包信息（package）

Linux 软件包管理之yum在线管理

yum 在线管理

yum (全称为 Yellow dog Updater, Modified) 是一个在 Fedora 和 RedHat 以及 SUSE 中的 Shell 前端软件包管理器。基于 RPM 包管理，能够从指定的服务器自动下载 RPM 包并且安装，可以自动处理依赖性关系，并且一次安装所有依赖的软体包，无须繁琐地一次次下载、安装。yum 提供了查找、安装、删除某一个、一组甚至全部软件包的命令，而且命令简洁而又好记。

从上面的解释我们可以看到 yum 管理是从指定的服务器（网络 yum 源）下载，所以必须要有网络或者自己手动配置一个本地 yum 源（不需要网络，后面会讲解如何手动配置），而且最关键的是 yum 能帮我们解决依赖性关系。

网络 yum 源

在安装好的Linux系统中，进入到 /etc/yum.repos.d/ 目录：

```
[root@node3 mnt]# cd /etc/yum.repos.d/
[root@node3 yum.repos.d]# ll
总用量 24
-rw-r--r--. 1 root root 1991 5月 19 2016 CentOS-Base.repo
-rw-r--r--. 1 root root 647 5月 19 2016 CentOS-Debuginfo.repo
-rw-r--r--. 1 root root 289 5月 19 2016 CentOS-fasttrack.repo
-rw-r--r--. 1 root root 630 5月 19 2016 CentOS-Media.repo
-rw-r--r--. 1 root root 6259 5月 19 2016 CentOS-Vault.repo
[root@node3 yum.repos.d]#
```

一般来讲，以 .repo 结尾的文件都是 yum 源。如果能联网，会使用 CentOS-Base.repo 作为默认的 yum 源，如果不能联网我们使用 CentOS-Media.repo 作为本地光盘 yum 源。

通过 vim 命令打开 CentOS-Base.repo：

```

# CentOS-Base.repo
#
# The mirror system uses the connecting IP address of the client and the
# update status of each mirror to pick mirrors that are updated to and
# geographically close to the client. You should use this for CentOS updates
# unless you are manually picking other mirrors.
#
# If the mirrorlist= does not work for you, as a fall back you can try the
# remarked out baseurl= line instead.
#
#
[base]
name=CentOS-$releasever - Base
mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch&repo=os&infra=$infra
#baseurl=http://mirror.centos.org/centos/$releasever/os/$basearch/
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-6

#released updates
[updates]
name=CentOS-$releasever - Updates
mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch&repo=updates&infra=$infra
#baseurl=http://mirror.centos.org/centos/$releasever/updates/$basearch/
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-6

#additional packages that may be useful
[extras]
name=CentOS-$releasever - Extras
mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch&repo=extras&infra=$infra
#baseurl=http://mirror.centos.org/centos/$releasever/extras/$basearch/
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-6

```

查看红色框住的内容：

- ◆ [base] 容器名称，一定要放在[]中
- ◆ name 容器说明，可以自己随便写
- ◆ mirrorlist 镜像站点，这个可以注释掉
- ◆ baseurl 我们的yum源服务器的地址。默认是CentOS官方的yum源服务器，是可以使用的，如果你觉得慢可以改成你喜欢的yum源地址
- ◆ enabled 此容器是否生效，如果不写或写成enable=1都是生效，写成enable=0就是不生效
- ◆ gpgcheck 如果是1是指RPM的数字证书生效，如果是0则不生效
- ◆ gpgkey 数字证书的公钥文件保存位置。不用修改

mirrorlist和baseurl的地址就是用来下载rpm包的地址，我们使用其中一个就好了。由于默认的地址都是国外的网站，如果嫌网速慢的话，可以更改为国内的yum源地址。

比如阿里的：<http://mirrors.aliyun.com/repo/Centos-7.repo>

光盘 yum 源搭建步骤

一、挂载光盘

```
[root@node3 /]# mount /dev/sr0 /mnt/cdrom/
mount: block device /dev/sr0 is write-protected, mounting read-only
[root@node3 /]#
```

二、让网络 yum 源失效

原理就是让以 .repo 文件都不存在。这里我们将 /etc/yum.repos.d 目录下的 .repo 文件都重命名为.repo.bak

```
[root@node3 /]# cd /etc/yum.repos.d/
[root@node3 yum.repos.d]# ls
CentOS-Base.repo      CentOS-fasttrack.repo  CentOS-Vault.repo
CentOS-Debuginfo.repo  CentOS-Media.repo
[root@node3 yum.repos.d]# mv CentOS-Base.repo  CentOS-Base.repo.bak
[root@node3 yum.repos.d]# mv CentOS-fasttrack.repo CentOS-fasttrack.repo.bak
[root@node3 yum.repos.d]# mv CentOS-Vault.repo CentOS-Vault.repo.bak
[root@node3 yum.repos.d]# mv CentOS-Debuginfo.repo CentOS-Debuginfo.repo.bak
[root@node3 yum.repos.d]# ls
CentOS-Base.repo.bak    CentOS-fasttrack.repo.bak  CentOS-Vault.repo.bak
CentOS-Debuginfo.repo.bak  CentOS-Media.repo  这个是光盘yum源文件，不用重命名
[root@node3 yum.repos.d]#
```

三、修改光盘 yum 源文件

也就是修改上面的 CentOS-Media.repo 文件。

```
[root@node3 yum.repos.d]# vim CentOS-Media.repo

# yum --enablerepo=c6-media [command]
#
# or for ONLY the media repo, do this:
#
# yum --disablerepo=\* --enablerepo=c6-media [command]

[c6-media]
name=CentOS-$releasever - Media
baseurl=file:///mnt/cdrom 地址为自己光盘挂载地址
#           file:///media/cdrom/
#           file:///media/cdrecorder/注释掉两个不存在的地址
gpgcheck=1
enabled=1 将enabled的值改为1，让这个yum源生效
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-6
```

四、输入 yum list 可以查看光盘 yum 源里面的软件包

```

yum-plugin-verify.noarch      1.1.30-37.el6      c6-media
yum-plugin-versionlock.noarch 1.1.30-37.el6      c6-media
yum-presto.noarch            0.6.2-1.el6       c6-media
yum-updateonboot.noarch      1.1.30-37.el6      c6-media
zenity.x86_64                 2.28.0-1.el6       c6-media
zlib.i686                     1.2.3-29.el6      c6-media
zlib-devel.i686               1.2.3-29.el6      c6-media
zlib-devel.x86_64              1.2.3-29.el6      c6-media
zlib-static.x86_64             1.2.3-29.el6      c6-media
zsh.x86_64                     4.3.11-4.el6.centos.2  c6-media
zsh-html.x86_64                4.3.11-4.el6.centos.2  c6-media
[root@node3 yum.repos.d]#

```

光盘yum源的后缀是
c6-media

网络yum源的后缀是
Base

常用的 yum 命令

一、查询所有可用软件包列表: yum list

```
[root@node3 /]# yum list
```

此命令是在配好的yum源服务器上去查询所有可用的软件包

二、查询服务器上和关键字相关的软件包: yum search 关键字

```

[root@node3 /]# yum search httpd
已加载插件: fastestmirror, security
Loading mirror speeds from cached hostfile
 * base: mirrors.btte.net
 * extras: mirrors.btte.net
 * updates: centos.ustc.edu.cn
=====
                                         N/S Matched: httpd =
libmicrohttpd-devel.i686 : Development files for libmicrohttpd
libmicrohttpd-devel.x86_64 : Development files for libmicrohttpd
libmicrohttpd-doc.noarch : Documentation for libmicrohttpd
httpd.x86_64 : Apache HTTP Server
httpd-devel.i686 : Development interfaces for the Apache HTTP server
httpd-devel.x86_64 : Development interfaces for the Apache HTTP server
httpd-manual.noarch : Documentation for the Apache HTTP server
httpd-tools.x86_64 : Tools for use with the Apache HTTP Server
libmicrohttpd.i686 : Lightweight library for embedding a webserver in applications
libmicrohttpd.x86_64 : Lightweight library for embedding a webserver in applications
mod_auth_mellon.x86_64 : A SAML 2.0 authentication module for the Apache Httpd Server
mod_dav_svn.x86_64 : Apache httpd module for Subversion server
mod_dnssd.x86_64 : An Apache HTTPD module which adds Zeroconf support

Name and summary matches only, use "search all" for everything.
[root@node3 /]#

```

三、yum 安装软件包: yum -y install 包名

选项: -y 表示自动回答 yes (如果不加, 每安装一个软件都会让你确认是否安装)

install 表示安装

范例: 这里以安装编译器 gcc 为例 (后面演示安装源码包必须要用到此编译器) : yum -y install gcc

没有报错, 出现如下安装信息则说明安装成功:

```
总计
运行 rpm_check_debug
执行事务测试
事务测试成功
执行事务
正在升级 : libgcc-4.4.7-18.el6.x86_64
正在安装 : ppl-0.10.2-11.el6.x86_64
正在安装 : cloog-ppl-0.15.7-1.2.el6.x86_64
正在安装 : mpfr-2.4.1-6.el6.x86_64
正在安装 : cpp-4.4.7-18.el6.x86_64
正在升级 : libgomp-4.4.7-18.el6.x86_64
正在安装 : gcc-4.4.7-18.el6.x86_64
清理 : libgcc-4.4.7-17.el6.x86_64
清理 : libgomp-4.4.7-17.el6.x86_64
Verifying : cpp-4.4.7-18.el6.x86_64
Verifying : libgomp-4.4.7-18.el6.x86_64
Verifying : mpfr-2.4.1-6.el6.x86_64
Verifying : libgcc-4.4.7-18.el6.x86_64
Verifying : gcc-4.4.7-18.el6.x86_64
Verifying : ppl-0.10.2-11.el6.x86_64
Verifying : cloog-ppl-0.15.7-1.2.el6.x86_64
Verifying : libgcc-4.4.7-17.el6.x86_64
Verifying : libgomp-4.4.7-17.el6.x86_64
```

已安装:

gcc.x86_64 0:4.4.7-18.el6

作为依赖被安装:

cloog-ppl.x86_64 0:0.15.7-1.2.el6

作为依赖被升级:

libgcc.x86_64 0:4.4.7-18.el6

完毕!

[root@node3 ~]#

或者在安装完成之后，输入 rpm -q gcc 弹出如下界面表示安装成功:

```
[root@node3 ~]# rpm -q gcc
gcc-4.4.7-18.el6.x86_64
[root@node3 ~]#
```

四、yum 升级软件包: yum -y update 包名

如果不指定包名，那么将会升级系统中所有的软件包，包括Linux内核。而Linux内核升级之后是需要在本地进行一些配置才能开机，如果是远程连接服务器进行Linux内核升级，那么是不可能启动服务器的。

五、yum 卸载软件包：yum -y remove 包名

卸载和升级也一样，而且由于软件包很多都有依赖性，你卸载A，而B和C都依赖于A，那么B和C都会卸载。假如C和Linux某个系统软件有依赖，而C也卸载掉了，可能造成某个系统功能不能使用。所以我们在卸载软件的时候一定要注意。

yum 软件组管理

```
[root@localhost ~]# yum grouplist  
#列出所有可用的软件组列表
```

```
[root@localhost ~]# yum groupinstall 软件组名  
#安装指定软件组，组名可以由grouplist查询出来
```

```
[root@localhost ~]# yum groupremove 软件组名  
#卸载指定软件组
```

安装某个软件组，会比我们一个一个安装某个软件包要方便的多。

```
[root@node3 ~]# yum grouplist
已加载插件: fastestmirror, security
设置组进程
Loading mirror speeds from cached hostfile
 * base: mirrors.btte.net
 * extras: mirrors.btte.net
 * updates: centos.ustc.edu.cn
base/group_gz
已安装的组:
 MySQL 数据库客户端
 MySQL 数据库服务器
 NFS 文件服务器
 Perl 支持
 万维网服务器
 兼容程序库
 图形管理工具
 基本
 大系统性能
 字体
 安全性工具
 性能工具
 控制台互联网工具
 服务器平台
 桌面平台
 电子邮件服务器
 目录客户端
 硬件监控工具
 科学记数法支持
 继承 UNIX 兼容性
 继承 X Windows 系统的兼容性
 网络文件系统客户端
 联网工具
 调试工具
 输入法
 通用桌面
```

这是在远程的桌面显示的是中文信息，如果直接进入服务器查询，显示的便是英文，我们需要什么直接安装即可。

Linux 用户和用户组管理之相关配置文件

用户信息文件: /etc/passwd

我们通过 vim /etc/passwd 命令，打开 passwd 文件：

```
root:x:0:0:root:/bin/bash
bin:x:1:1:bin:/bin/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
rpc:x:32:32:Rpcbind Daemon:/var/cache/rpcbind:/sbin/nologin
vcsta:x:69:69:virtual console memory owner:/dev:/sbin/nologin
abrt:x:173:173::/etc/abrt:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin
haldaemon:x:68:68:HAL daemon:/:/sbin/nologin
ntp:x:38:38::/etc/ntp:/sbin/nologin
saslauth:x:499:76:Saslauthd user:/var/empty/saslauth:/sbin/nologin
postfix:x:89:89::/var/spool/postfix:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
tcpdump:x:72:72::/sbin/nologin
oprofile:x:16:16:Special user account to be used by OProfile:/home/oprofile:/sbin/nologin
hadoop:x:500:500:/home/hadoop:/bin/bash
mysql:x:27:27:MySQL Server:/var/lib/mysql:/bin/bash
vae:x:501:501:/home/vae:/bin/bash
tom:x:502:502:/home/tom:/bin/bash
apache:x:48:48:Apache:/var/www:/sbin/nologin
~
```

上面的每一行代表一个用户，我们以第一行为例：

```
1 root:x:``0``:``0``:root:/root:/bin/bash
```

上面的root用户通过： 分隔为 7 个字段。

①、第一个字段：root 表示用户名

②、第二个字段：x 表示密码标志，真正的密码是存储在 /etc/shadow 文件中，下面我们会详细讲解。

③、第三个字段：UID,用户ID。这里我们需要说明的是在Linux系统中不一定root用户是超级用户，用户id为0的才是超级用户。

0：表示超级用户，权限最大的用户。

1-499：表示系统用户（伪用户），伪用户是系统用来启动相关服务和命令的，不能用来登录系统，而且不能删除，删除伪用户会造成一些命令不能使用。

500-65535：普通用户。Linux内核2.6以后是可以支持232个用户，基本上是不用担心用户不够的。

④、第四个字段：GID,用户初始组ID。这里需要理解初始组和用户组的概念，初始组就是指用户一登录就立刻拥有这个用户组的相关权限，每个用户的初始组只能有一个，一般就是和这个用户的用户组相同的组名作为这个用户的初始组。附加组值用户可以加入多个其他的用户组，并拥有这些组的权限，附加组可以有多个。

⑤、第五个字段：用户说明

⑥、第六个字段：家目录。

普通用户：/home/用户名/

超级用户：/root/

⑦、第七个字段：登录之后的 shell。shell后面我们会详细介绍，这里简单来说shell就是Linux的命令解释器。

用户密码文件：/etc/shadow

我们通过 vim /etc/shadow 命令，打开 shadow文件：

```
root:$6$bo3LXGTQ8SwsRa6J$.7qTM2GT8EmA8YSkHtIqlVefcUy0Tdv3EBwJLA32U0qL/YHI0e0SQiaCsNq4tSAN2zVbp0bv10FP.sK0euPIg/:17324:0:99999:7:::  
bin:*:15980:0:99999:7:::  
daemon:*:15980:0:99999:7:::  
adm:*:15980:0:99999:7:::  
lp:*:15980:0:99999:7:::  
sync:*:15980:0:99999:7:::  
shutdown:*:15980:0:99999:7:::  
halt:*:15980:0:99999:7:::  
mail:*:15980:0:99999:7:::  
uucp:*:15980:0:99999:7:::  
operator:*:15980:0:99999:7:::  
games:*:15980:0:99999:7:::  
gopher:*:15980:0:99999:7:::  
ftp:*:15980:0:99999:7:::  
nobody:*:15980:0:99999:7:::  
dbus:!!:17324:::::  
rpc:!!:17324:0:99999:7:::  
vcsa:!!:17324:::::  
abrt:!!:17324:::::  
rpcuser:!!:17324:::::  
nfsobody:!!:17324:::::  
haldaemon:!!:17324:::::  
ntp:!!:17324:::::  
saslauth:!!:17324:::::  
postfix:!!:17324:::::  
sshd:!!:17324:::::  
tcpdump:!!:17324:::::  
oprofile:!!:17324:::::  
hadoop:$6$0CD4p/Gw$izUSK4aSkC2afw6WfiI0ocomm5K.FWa4A4y6aHT4dHeZmLkJNnCm7PMkPytdlAnUWd5f30fWcluBgr81J3uyX/:17324:0:99999:7:::  
mysql:!!:17332:::::  
vae:$6$3mItE09u$WnsixmffB20DzNRNCNz5X6y1j7nzA9U0XxhLjxNZFUpdVzB/70AlHa/lLIP3YullN1l/lqXdRRo3iGPk4Wgyc/:17465:0:99999:7:::  
tom:$6$LtQ6PyHE$efBYUpb7x8UK.mFNal0L2CH1s1FI0aqGPu0TpS0tZNf/ZPA8e16L0ruNprECcUxJpiLMrx1a/UhEhylZQpr2t/:17467:0:99999:7:::  
apache:!!:17471:::::
```

同理上面的每一行和 passwd 文件一样，都是表示一个用户。passwd描述的是用户信息，shadow文件描述的是用户密码。我们还是以第一行为例：

```
1 root:$``6``$bo3LXGTQ8SwsRa6J$.7qTM2GT8EmA8YSkHtIqlVefcUy0Tdv3EBwJLA32U0qL/YHI0e0SQ  
iaCsNq4tSAN2zVbp0bv10FP.sK0euPIg/:``17324``:``0``:``99999``:``7``:::
```

上面的root用户通过： 分隔为 9 个字段。

①、第一个字段：root 表示用户名

②、第二个字段：root用户的登录加密密码。

一、加密算法升级为 SHA512 散列加密算法

二、如果密码位是“!”或者“*”代表没有密码，不能登录，基本上伪用户都是没有密码的

③、第三个字段：密码的最后一次修改日期。使用1970年1月1日作为标准时间，每过一天时间戳加

1.

root 用户的修改日期是17324。这里我们通过 date -d "1970-01-01 17324 days"进行换算：

```
[root@node3 /]# date -d "1970-01-01 17324 days"
2017年 06月 07日 星期三 00:00:00 CST
[root@node3 /]#
```

- ④、第四个字段：两次密码的修改间隔时间（和第三字段相比）
- ⑤、第五个字段：密码有效期（和第三字段相比）
- ⑥、第六个字段：密码修改到期前的警告天数（和第五个字段进行比较）
- ⑦、第七个字段：密码过期后的宽限天数（和第五个字段进行比较）
 - 0：代表密码过期后立即失效
 - 1：代表密码永远不会失效
- ⑧、第八个字段：账号失效时间（要用时间戳表示）
- ⑨、第九个字段：保留字段

用户组信息文件：/etc/group

我们通过 vim /etc/group 命令，打开 group 文件：

```
root:x:0:
bin:x:1:bin,daemon
daemon:x:2:bin,daemon
sys:x:3:bin,adm
adm:x:4:adm,daemon
tty:x:5:
disk:x:6:
lp:x:7:daemon
mem:x:8:
kmem:x:9:
wheel:x:10:
mail:x:12:mail,postfix
uucp:x:14:
man:x:15:
games:x:20:
gopher:x:30:
video:x:39:
dip:x:40:
ftp:x:50:
lock:x:54:
audio:x:63:
nobody:x:99:
users:x:100:
dbus:x:81:
rpc:x:32:
utmp:x:22:
```

一般创建一个用户，立即创建一个和用户名相同的用户组，我们还是以第一行 root 为例：

```
1 root:x:``0``:
```

- ①、第一个字段：组名
- ②、第二个字段：组密码标志，密码文件存放在 /etc/gshadow 中
- ③、第三个字段：GID，组id
- ④、第四个字段：组中的附加用户

用户组密码文件：/etc/gshadow

```
root:::  
bin:::bin,daemon  
daemon:::bin,daemon  
sys:::bin,adm  
adm:::adm,daemon  
tty:::  
disk:::  
lp:::daemon  
mem:::  
kmem:::  
wheel:::  
mail:::mail,postfix  
uucp:::  
man:::  
games:::  
gopher:::  
video:::  
dip:::  
ftp:::  
lock:::  
audio:::  
nobody:::  
users:::  
dbus::!::  
rpc::!::  
utmp::!::  
utempter::!::  
floppy::!::
```

这个文件基本上不会用到

- ①、第一个字段：组名
- ②、第二个字段：组密码
- ③、第三个字段：组管理员用户名
- ④、第四个字段：组中的附加用户

用户的家目录

①、普通用户：/home/用户名/，一般创建一个新用户就会自动创建该家目录。所有者和所属组都是此用户，权限是700。

②、超级用户：/root/，所有者和所属组都是root用户，权限是550。注意看上去权限是550，其实权限对于超级用户基本上是没有限制的，所以这里给什么权限都没多大区别。

用户的邮箱

目录是：

```
1 /var/spool/mail/用户名/
```

用户的模板目录

目录是：

```
1 /etc/skel
```

这个目录是每创建一个新用户，就会在其家目录下自动创建 /etc/skel 目录下的所有文件。

范例：我们在 /etc/skel 目录下创建一个 hello 文件，然后创建一个新用户 test，进入到 test 用户的家目录/home/test 就会看到有 hello 文件。

```
[root@node3 ~]# cd /etc/skel/
[root@node3 skel]# ls -a
. .. .bash_logout .bash_profile .bashrc .gnome2
[root@node3 skel]# touch hello 在 /etc/skel 创建hello文件
[root@node3 skel]# ls -a
. .. .bash_logout .bash_profile .bashrc .gnome2 hello
[root@node3 skel]# useradd test 创建新用户 test
[root@node3 skel]# passwd test
更改用户 test 的密码。
新的 密码：
无效的密码: WAY 过短
无效的密码: 过于简单
重新输入新的 密码：
passwd: 所有的身份验证令牌已经成功更新。
[root@node3 skel]# cd /home/test/ 在新创建的test用户家目录中发现存在
[root@node3 test]# ls -a
. .. .bash_logout .bash_profile .bashrc .gnome2 hello
[root@node3 test]# 
```

Linux 用户和用户组管理之用户管理命令

添加用户命令：useradd

①、命令名称：useradd

②、英文原意：

③、命令所在路径： /usr/sbin/useradd

④、执行权限： **root**

⑤、功能描述：添加新用户

⑥、语法： useradd 【选项】 【用户名】

-u UID:手工指定用户的uid

-d 家目录:手工指定用户的家目录

-c 用户说明:手工指定用户说明

-g 组名:手工指定用户的初始组

-G 组名:手工指定用户的附加组

-s shell:手工指定用户的登录shell， 默认是/bin/bash

注意：添加选项的uid，家目录等等前面的配置文件我们都已经讲解了，实际上创建新用户是不用添加任何选项的， 默认就好。

范例：添加用户tom:useradd tom。

```
[root@node3 tmp]# useradd tom  
[root@node3 tmp]#
```

创建一个新用户之后，会在下面文件中自动生成内容：

/etc/passwd：生成用户信息

/etc/shadow：生成密码信息

/etc/group：生成用户组信息

/etc/gshadow：生成用户组密码信息

/home/tom:创建家目录

/var/spool/mail/tom： 创建用户邮箱目录

```
[root@node3 test]# grep "tom" /etc/passwd  
tom:x:502:502::/home/tom:/bin/bash  
[root@node3 test]# grep "tom" /etc/shadow  
tom:$6$LtQ6PyHE$efBYUpb7x8UK.mFNaOL2CH1s1FIQaqGPu0TpS0tZNF/ZPA8e16L0ruNprECcUxJpiLMrx1a/UhEh  
yLZQpr2t/:17467:0:99999:7:::  
[root@node3 test]# grep "tom" /etc/group  
tom:x:502:  
[root@node3 test]# grep "tom" /etc/gshadow  
tom:!::  
[root@node3 test]# ll -d /home/tom  
drwx----- 3 tom tom 4096 10月 29 11:16 /home/tom  
[root@node3 test]# ll /var/spool/mail/tom  
-rw-rw---- 1 tom mail 0 10月 28 13:37 /var/spool/mail/tom  
[root@node3 test]#
```

上面出现了很多默认值， 用户默认值文件如下：

◆ /etc/default/useradd

- GROUP=100 #用户默认组
- HOME=/home #用户家目录
- INACTIVE=-1 #密码过期宽限天数 (7)
- EXPIRE= #密码失效时间 (8)
- SHELL=/bin/bash #默认shell
- SKEL=/etc/skel #模板目录
- CREATE_MAIL_SPOOL=yes #是否建立邮箱

◆ /etc/login.defs

- PASS_MAX_DAYS 99999 #密码有效期 (5)
- PASS_MIN_DAYS 0 #密码修改间隔 (4)
- PASS_MIN_LEN 5 #密码最小5位 (PAM)
- PASS_WARN_AGE 7 #密码到期警告 (6)
- UID_MIN 500 #最小和最大UID范围
- GID_MAX 60000
- ENCRYPT_METHOD SHA512 #加密模式

修改用户密码：passwd

- ①、命令名称：passwd
- ②、英文原意：
- ③、命令所在路径：/usr/bin/passwd
- ④、执行权限：root
- ⑤、功能描述：修改用户的密码
- ⑥、语法： passwd 【选项】 【用户名】

-S 查询用户密码的密码状态，仅root用户可用
-l 暂时锁定用户。仅root用户可用
-u 解锁用户。仅root用户可用

-stdin 可以通过管道符输出的数据作为用户的密码

注意：root用户能修改任何用户的密码，语法为 passwd 用户名。而普通用户只能修改自己的密码，语法为 passwd，后面不能加普通用户名，而且密码要符合密码规则，不然修改不了。

```
[root@node3 test]# passwd -S tom          查看用户密码  
tom PS 2017-10-28 0 99999 7 -1 (密码已设置, 使用 SHA512 加密。)  
[root@node3 test]# passwd -l tom          锁定用户  
锁定用户 tom 的密码。  
passwd: 操作成功  
[root@node3 test]# passwd -u tom          解锁用户  
解锁用户 tom 的密码。  
passwd: 操作成功  
[root@node3 test]# echo "123" | passwd --stdin tom 管道符修改用户密码  
更改用户 tom 的密码。  
passwd: 所有的身份验证令牌已经成功更新。  
[root@node3 test]#
```

修改用户信息：usermod

[root@localhost ~]#usermod [选项] 用户名
选项：

- u UID: 修改用户的UID号
- c 用户说明: 修改用户的说明信息
- G 组名: 修改用户的附加组
- L: 临时锁定用户 (Lock)
- U: 解锁用户锁定 (Unlock)

修改用户密码状态：chage

[root@localhost ~]#chage [选项] 用户名

选项:

- l: 列出用户的详细密码状态
- d 日期: 修改密码最后一次更改日期 (shadow3字段)
- m 天数: 两次密码修改间隔 (4字段)
- M 天数: 密码有效期 (5字段)
- W 天数: 密码过期前警告天数 (6字段)
- I 天数: 密码过后宽限天数 (7字段)
- E 日期: 账号失效时间 (8字段)

删除用户命令: userdel

语法: userdel [-r] 用户名

-r 删除用户的同时删除用户家目录

执行上面的命令, 会自动删除下面的文件:

- ①、删除 /etc/passwd 文件的用户信息
- ②、删除 /etc/shadow 文件的用户密码信息
- ③、删除/etc/group 文件的用户组信息
- ④、删除 /etc/gshadow 文件的用户组密码信息
- ⑤、删除用户的邮件信息 rm -rf /var/spool/mail/用户名
- ⑥、删除用户的家目录 rm -rf /home/用户名

注意: 基本上完整的删除一个用户都是要加上 -r 选项的。如何判断是否完整的删除一个用户, 只需要从新添加该用户一次, 如果报如下错误则没有删除成功:

```
[root@node3 test]# useradd tom  
useradd: 用户“tom”已存在  
[root@node3 test]#
```

查看用户 id

```
[root@node3 test]# id tom  
uid=502(tom) gid=502(tom) 组=502(tom)  
[root@node3 test]#
```

切换用户身份 su

[root@localhost ~]# su [选项] 用户名

选项：

- : 选项只使用“-”代表连带用户的环境变量一起切换

-c 命令：仅执行一次命令，而不切换用户身份

注意：选项 - 千万不能省略，必须要连带用户的环境变量一起切换。从普通用户切换到 root 用户是需要输入密码的，而从root用户切换到普通用户是不需要输入密码的。

```
[root@node3 ~]# su - tom  
[tom@node3 ~]$ su - root -c "useradd tom1"  
密码： 临时使用 root 用户创建用户 tom1  
[tom@node3 ~]$
```

添加用户组：groupadd

语法：groupadd 【选项】组名

-g GID 指定组id

修改用户组：groupmod

语法：groupmod 【选项】组名

-g GID 修改组id

-n 新组名 修改组名

范例：把组名 group1 修改为 group2

groupmod -n group2 group1

删除用户组：groupdel

语法：groupdel 组名

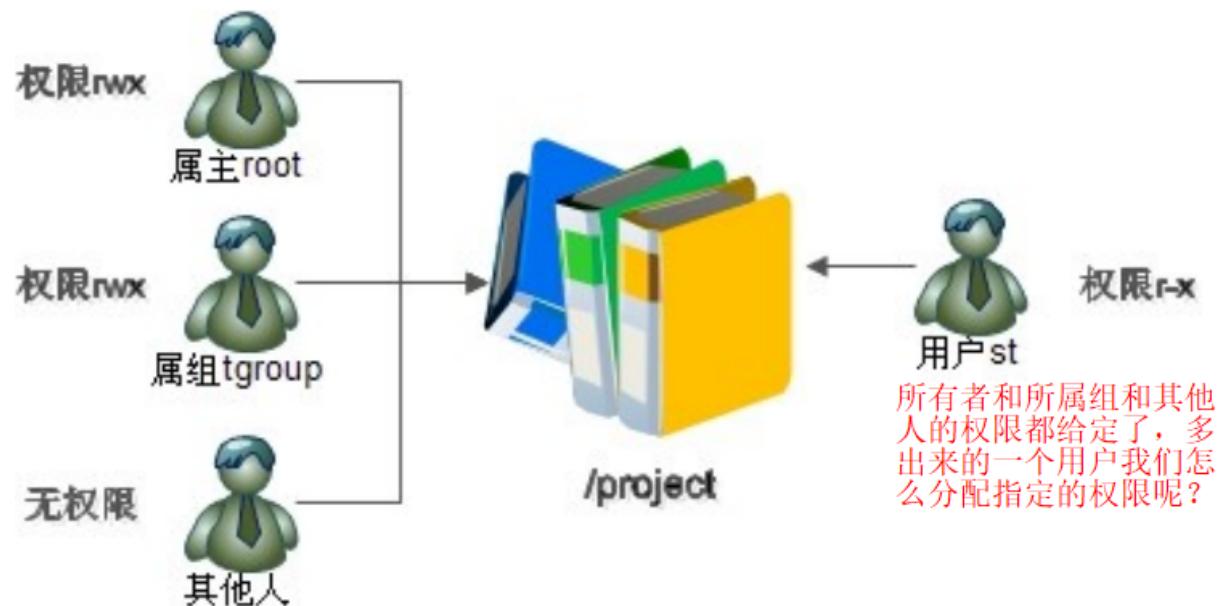
Linux 权限管理之 ACL 权限

什么是 ACL 权限？

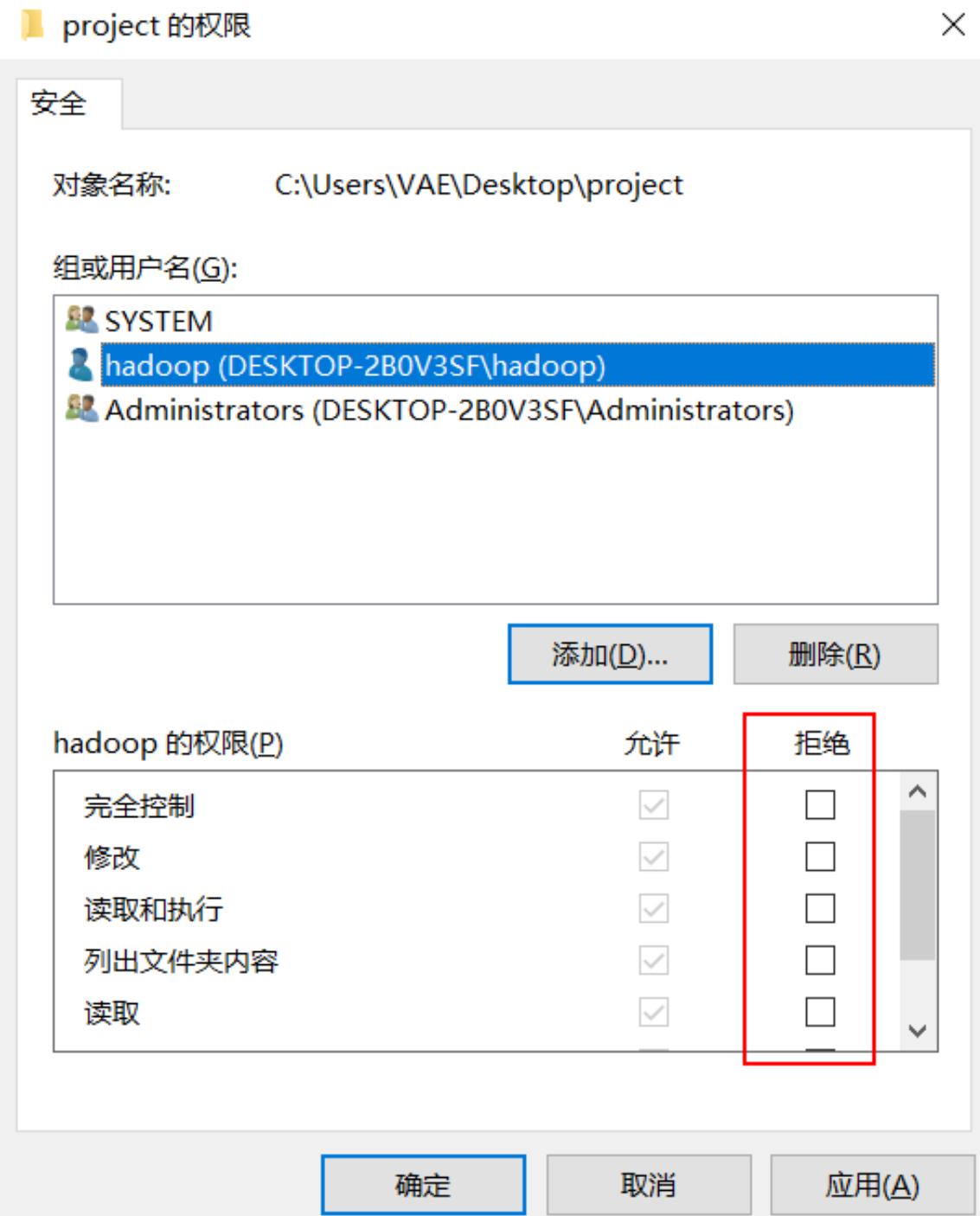
比如有如下场景：

某大牛在 QQ 群内直播讲解 Linux 系统的权限管理，讲解完之后，他在一个公有的 Linux 系统中创建了一个 `/project` 目录，里面存放的是课后参考资料。那么 `/project` 目录对于大牛而言是所有者，拥有读写可执行 (`rwx`) 权限，对于 QQ 群内的所有用户他们都分配的一个所属组里面，也都拥有读写可执行 (`rwx`) 权限，而对于 QQ 群外的其他人，那么我们不给他访问 `/project` 目录的任何权限，那么 `/project` 目录的所有者和所属组权限都是 (`rwx`)，其他人权限无。

问题来了，这时候直播有旁听的人参与（不属于QQ群内），听完之后，我们允许他访问 `/project` 目录查看参考资料，但是不能进行修改，也就是拥有 (`r-x`) 的权限，这时候我们该怎么办呢？我们知道一个文件只能有一个所属组，我们将他分配到 QQ 群所在的所属组内，那么他拥有了写的权限，这是不被允许的；如果将这个旁听的人视为目录`/project` 的其他人，并且将`/project`目录的其他人权限改为 (`r-x`)，那么不是旁听的人也能访问我们 `/project` 目录了，这显然也是不被允许的。怎么解决呢？



我们想想windows系统里面给某个文件分配权限的办法：



如上图，我们想要让某个用户不具备某个权限，直接不给他分配这个目录的相应权限就行了。那么对应到Linux系统也是这样，我们给指定的用户指定目录分配指定的权限，也就是 ACL 权限分配。

查看分区 ACL 权限是否开启：dump2fs

我们看某个文件（Linux系统中目录也是文件，一切皆是文件）是否支持 ACL 权限，首先要看文件所在的分区是否支持 ACL 权限。

一、查看当前系统有哪些分区：df -h

```
[root@node3 tmp]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda5        13G   2.6G  9.5G  22% /
tmpfs           491M     0  491M   0% /dev/shm
/dev/sda1       190M   33M  147M  19% /boot
/dev/sda2        4.7G  760M  3.7G  17% /home
/dev/sr0         3.7G   3.7G     0 100% /mnt/cdrom
[root@node3 tmp]#
```

二、查看指定分区详细文件信息： dumpe2fs -h 分区路径

下面是查看 根分区/ 的详细文件信息

```
[root@node3 tmp]# dumpe2fs -h /dev/sda5
dumpe2fs 1.41.12 (17-May-2010)
Filesystem volume name:  <none>
Last mounted on:        /
Filesystem UUID:        490ed737-f8cf-46a6-ac4b-b7735b79fc63
Filesystem magic number: 0xEF53
Filesystem revision #:   1 (dynamic)
Filesystem features:    has_journal ext_attr resize_inode dir_index filetype needs_recovery extent flex_bg sparse_super large_file huge_file uninit_bg dir_nlink extra_isize
Filesystem flags:        signed_directory_hash
Default mount options:  user_xattr acl
Filesystem state:        clean
Errors behavior:        Continue
Filesystem OS type:      Linux
Inode count:            850304
Block count:             3399168
Reserved block count:   169958
Free blocks:            2680980
```

这里有 acl，一般分区都是默认支持 ACL 权限

开启分区 ACL 权限

一、临时开启分区 ACL 权限

```
1  mount -o remount,acl /
```

重新挂载根分区，并挂载加入 acl 权限。注意这种命令开启方式，如果系统重启了，那么根分区权限会恢复到初始状态。

二、永久开启分区 ACL 权限

修改配置文件 /etc/fstab

```
# /etc/fstab
# Created by anaconda on Thu Jun  8 03:45:10 2017
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
UUID=490ed737-f8cf-46a6-ac4b-b7735b79fc63 /          ext4    defaults,acl  1 1
UUID=b9a4dcfe-bb46-483d-aebd-796828b6b00d /boot       ext4    defaults        1 2
UUID=5a7b789e-c6f3-4bcf-b868-c03811cdb818 /home       ext4    defaults        1 2
UUID=f871cc80-26a3-43ce-9859-60939d17bd44 swap        swap    defaults        0 0
tmpfs          /dev/shm      tmpfs   defaults        0 0
devpts         /dev/pts      devpts  gid=5,mode=620  0 0  想让某个分区拥有acl权限，只需在
sysfs          /sys          sysfs   defaults        0 0  defaults后面加上 ,acl
proc           /proc         proc    defaults        0 0  其实一般default权限是默认开启acl
~
```

上面是修改根分区拥有 acl 权限

```
1   UUID=490ed737-f8cf-46a6-ac4b-b7735b79fc63 /           ext4 defaults,acl     ``1``  
`1
```

重新挂载文件系统或重启系统，使得修改生效

```
1   mount -o remount /
```

设定 ACL 权限：**setfacl** 选项 文件名

选项：

- m 设定ACL权限
- x 删除指定的ACL权限
- b 删除所有的ACL权限
- d 设定默认ACL权限。
- k 删除默认ACL权限
- R 递归设定ACL权限。

一、给用户设定 ACL 权限：**setfacl -m u:用户名:权限 指定文件名**

二、给用户组设定 ACL 权限:**setfacl -m g:组名:权限 指定文件名**

我们给用户或用户组设定 ACL 权限其实并不是真正我们设定的权限，是与 mask 的权限“相与”之后的权限才是用户的真正权限，一般默认mask权限都是rwx，与我们所设定的权限相与就是我们设定的权限。mask 权限下面我们会详细讲解

范例：所有者root用户在根目录下创建一个文件目录/project，然后创建一个QQ群所属组，所属组里面创建两个用户zhangsan和lisi。所有者和所属组权限和其他人权限是770。

然后创建一个旁听用户 pt，给他设定/project目录的 ACL 为 r-x。

```
[root@node3 /]# mkdir /project
[root@node3 /]# useradd zhangsan
[root@node3 /]# useradd lisi
[root@node3 /]# groupadd QQgroup
[root@node3 /]# gpasswd -a zhangsan QQgroup
正在将用户“zhangsan”加入到“QQgroup”组中
[root@node3 /]# gpasswd -a lisi QQgroup
正在将用户“lisi”加入到“QQgroup”组中
[root@node3 /]# chown root:QQgroup /project
[root@node3 /]# chmod 770 /project
[root@node3 /]# ll -d /project
drwxrwx---. 2 root QQgroup 4096 11月  9 08:21 /project
[root@node3 /]#
```

1、在根目录下创建目录 /project
2、创建用户 zhangsan
3、创建用户 lisi
4、创建用户组 QQgroup
5、将用户 zhangsan 加入到 QQgroup 用户组
6、将用户 lisi 加入到 QQgroup 用户组
7、将目录 /project 的所属组改为 QQgroup
8、将目录 /project 的权限改为 770

目录 /project 的所有者和所属组其他权限设定为 770。接下来我们创建旁听用户 pt，并赋予 acl 权限 rx

```
[root@node3 /]# useradd pt
[root@node3 /]# passwd pt    创建旁听用户 pt，并设定密码
更改用户 pt 的密码。
新的 密码：
重新输入新的 密码：
抱歉，密码不匹配。
新的 密码：
无效的密码： 过短
无效的密码： 过于简单
重新输入新的 密码：
passwd: 所有的身份验证令牌已经成功更新。
[root@node3 /]# setfacl -m u:pt:rx /project  设定 pt 用户 /project
[root@node3 /]# ll -d /project
drwxrwx---+ 2 root QQgroup 4096 11月  9 08:21 /project          的 acl 权限为 rx
[root@node3 /]# getfacl /project  查看 /project 目录的 acl 权限
getfacl: Removing leading '/' from absolute path names
# file: project
# owner: root
# group: QQgroup
user::rwx
user:pt:r-x
group::rwx
mask::rwx
other::---
[root@node3 /]#
```

为了验证 pt 用户对于 /project 目录没有写权限，我们用 su 命令切换到 pt 用户，然后进入 /project 目录，在此目录下创建文件，看是否能成功：

```
[root@node3 /]# su - pt
[pt@node3 ~]$ cd /pro
proc/  project/
[pt@node3 ~]$ cd /project/
[pt@node3 project]$ mkdir a
mkdir: 无法创建目录 "a": 权限不够
[pt@node3 project]$
```

上面提示权限不够，说明 acl 权限赋予成功，注意如下所示，如果某个目录或文件下有 + 标志，说明其具有 acl 权限。

```
[pt@node3 project]$ ll -d /project/
drwxrwx---+ 2 root QQgroup 4096 11月  9 08:21 /project/
[pt@node3 project]$ 一个文件或目录有后面的 + , 说明有acl
```

查看 ACL 权限: getfacl 文件名

```
[pt@node3 project]$ getfacl /project
getfacl: Removing leading '/' from absolute path names
# file: project
# owner: root
# group: QQgroup
user::rwx
user:pt:r-x
group::rwx
mask::rwx
other::---

[pt@node3 project]$
```

最大有效权限 mask

前面第 4 点我们讲过，我们给用户或用户组设定 ACL 权限其实并不是真正我们设定的权限，是与 mask 的权限 相与 之后的权限才是用户的真正权限，一般默认mask权限都是 rwx，与我们所设定的权限相与就是我们设定的权限。

我们通过 getfacl 文件名 也能查看 mask 的权限，那么我们怎么设置呢？

```
1  setfacl -m m:权限 文件名
```

```
[root@node3 ~]# setfacl -m m:rx /project    设定 mask 权限为 rx
[root@node3 ~]# getfacl /project
getfacl: Removing leading '/' from absolute path names
# file: project
# owner: root
# group: QQgroup
user::rwx
user:pt:r-x
group::rwx
mask::r-x          #effective:r-x
other::---
[root@node3 ~]#
```

删除 ACL 权限

①、删除指定用户的 ACL 权限

```
1  setfacl -x u:用户名 文件名
```

②、删除指定用户组的 ACL 权限

```
1  setfacl -x g:组名 文件名
```

③、删除文件的所有 ACL 权限

```
1  setfacl -b 文件名
```

递归 ACL 权限

通过加上选项 -R 递归设定文件的 ACL 权限，所有的子目录和子文件也会拥有相同的 ACL 权限。

```
1  setfacl -m u:用户名:权限 -R 文件名
```

默认 ACL 权限

如果给父目录设定了默认的 ACL 权限，那么父目录中所有新建的子文件会继承父目录的 ACL 权限。

```
1  setfacl -m d:u:用户名:权限 文件名
```

Linux 权限管理之文件系统系统属性 chattr 权限和 sudo 命令

设定文件系统属性：chattr

```
1  chattr [+-=][选项] 文件或目录名
```

+：增加权限

-: 删除权限

=: 等于某权限

选项:

- i: 如果对文件设置i属性，那么不允许对文件进行删除、改名，也不能添加和修改数据；如果对目录设置i属性，那么只能修改目录下文件的数据，但不允许建立和删除文件。
- a: 如果对文件设置a属性，那么只能在文件中增加数据，但是不能删除也不能修改数据；如果对目录设置a属性，那么只允许在目录中建立和修改文件，但是不允许删除

```
[root@node3 tmp]# touch test.txt    1、创建文件 test.txt
[root@node3 tmp]# lsattr test.txt    2、lsattr 查看文件系统属性
-----e- test.txt
[root@node3 tmp]# chattr +i test.txt 3、给文件 test.txt 赋予 i 属性
[root@node3 tmp]# lsattr test.txt
----i-----e- test.txt
[root@node3 tmp]# rm -rf test.txt    4、i 属性的文件不允许删除，root用户
rm: 无法删除"test.txt": 不允许的操作
[root@node3 tmp]# echo '添加数据' >> test.txt
-bash: test.txt: 权限不够
[root@node3 tmp]#                                              5、i 属性的文件不允许修改，root 用户
                                                               也不可以
```

chattr 限制权限之后，root 用户也不能例外。这个命令可以防止文件进行误操作。

查看文件的系统属性：lsattr

1 lsattr 选项 文件名

选项: ①、-a 显示所有文件和目录

②、-d 若是目录，仅列出本身的属性，而不是子文件的

```
[root@node3 tmp]# mkdir test
[root@node3 tmp]# lsattr -a test
-----e- test/..
-----e- test/.
[root@node3 tmp]# chattr -ia test
[root@node3 tmp]# lsattr -a test
-----e- test/..
-----e- test/.
[root@node3 tmp]# 
```

sudo 权限

- ①、sudo 的操作对象只能是系统命令。
- ②、把本来由超级用户执行的命令赋予给普通用户执行。

简单来讲就是比如很多只能由超级用户来执行的命令，比如重启，关机等等，有时候不能使用超级用户，那我们该怎么办呢？

第一步：那就进行适当的配置，让超级用户赋予普通用户也能执行这些命令的权限

第二步：加上 sudo 去执行这些命令。

一、超级用户赋予普通用户执行命令权限，配置 /etc/sudoers 文件

我们可以使用 vim /etc/sudoers 命令，或者 visudo 命令

```
##      user      MACHINE=COMMANDS
##
## The COMMANDS section may have other options added to it.
##
## Allow root to run any commands anywhere
root    ALL=(ALL)      ALL
#
## Allows members of the 'sys' group to run networking, software,
## service management apps and more.
# %sys  ALL = NETWORKING, SOFTWARE, SERVICES, STORAGE, DELEGATING, PROCESSES

## Allows people in group wheel to run all commands
# %wheel      ALL=(ALL)      ALL

## Same thing without a password
# %wheel      ALL=(ALL)      NOPASSWD: ALL
```

```
[root@localhost ~]# visudo  
#实际修改的是/etc/sudoers文件
```

```
root    ALL=(ALL)      ALL  
#用户名 被管理主机的地址= (可使用的身份)      授权命令 (绝对路径)  
# %wheel      ALL=(ALL)      ALL  
#%组名 被管理主机的地址= (可使用的身份)      授权命令 (绝对路径)
```

二、授权用户可以重启服务器

```
1    用户名  ALL=/sbin/shutdown -h now
```

```
##  
## Allow root to run any commands anywhere  
root    ALL=(ALL)      ALL  
tom    ALL=/sbin/shutdown -h now  
  
## Allows members of the 'sys' group to run networking, software,  
## service management apps and more.  
# %sys  ALL = NETWORKING, SOFTWARE, SERVICES, STORAGE, DELEGATING, PROC  
  
## Allows people in group wheel to run all commands  
# %wheel      ALL=(ALL)      ALL  
  
## Same thing without a password  
# %wheel      ALL=(ALL)      NOPASSWD: ALL
```

三、查看可用的sudo 命令

```
1    sudo -l
```

```
[tom@node3 ~]$ sudo -l
[sudo] password for tom:
匹配此主机上 tom 的默认条目:
!visiblepw, always_set_home, env_reset, env_keep="COLORS DISPLAY INPUTRC KDEDIR LS_COLORS", env_keep+="MAIL PS1 PS2 QTDIR USERNAME LC_CTYPE", env_keep+="LC_COLLATE LC_IDENTIFICATION LC_MEASUREMENT env_keep+="LC_MONETARY LC_NAME LC_NUMERIC LC_PAPER LC_TELEPHONE" LC_ALL LANGUAGE LINGUAS _XKB_CHARSET XAUTHORITY",
secure_path=/sbin\:/bin\:/usr/sbin\:/usr/bin

用户 tom 可以在该主机上运行以下命令:
[root] /sbin/shutdown -h now
[tom@node3 ~]$
```

四、普通用户执行 sudo 赋予的命令

```
[tom@node3 ~]$ sudo /sbin/shutdown -h now
```

Linux文件系统管理之文件系统常用命令

1、为什么要给硬盘分区?

前面我们讲解 Linux 系统的安装时，我们手动给硬盘划分了4个分区，分为了根目录/，/home 分区,/boot分区，还有/swap交换分区，那么为什么要给硬盘分区呢？

①、易于管理和使用

一个没有分区的硬盘就像一个大柜子，如果我们在柜子里放些衣物和化妆品就会显得很杂乱没有规则不容易管理和拿取，这时如果我们找来木匠把柜子分割成不同的空间，用来分别储存衣物和化妆品就会让我们很容易管理和拿取衣服和化妆品。同样的一个硬盘如果不分割空间而直接储存各种文件会让我们难以管理和使用。

②、有利于数据安全

如果漏雨，一个没有分割的大柜子里面的东西肯定全部无法避免被雨水侵蚀的命运，而把柜子分割开来则会因每个空间相对独立，先侵蚀的只会是正好漏雨的空间如果及时采取措施那其它空间里的东西将得以保存。如果中病毒，一个没有分区的硬盘，里面保存的数据肯定全部都会被感染或者损坏，而如果把硬盘分区，然后把文件分开存放，在中毒后我们有充分的时间来采取措施防止病毒和清除病毒，即使需要重做系统也只会丢失系统所在的数据而其它数据将得以保存。

③、节约寻找文件的时间

在没有分区的硬盘里面找文件就想在一个大柜子里面找衣服一样，总会翻动很多东西才能找到自己想要的。如果我们把硬盘分区，在需要某个文件时可以直到特定的分区去寻找，这样避免了我们翻找过多的文件。

2、Linux系统分区类型

①、主分区：总共最多只能分 4 个。

②、扩展分区：只能有一个，也算作主分区的一种，也就是说主分区加上扩展分区最多有 4 个。但是扩展分区不能存储数据和格式化，必须在划分为逻辑分区才能使用。

③、逻辑分区：逻辑分区是扩展分区中划分的，如果是 IDE 硬盘，Linux 最多支持59个逻辑分区，如果是 SCSI 硬盘Linux最多支持 11 个逻辑分区。



分区的设备文件名	
主分区1	/dev/sda1
主分区2	/dev/sda2
主分区3	/dev/sda3
扩展分区	/dev/sda4
逻辑分区1	/dev/sda5
逻辑分区2	/dev/sda6
逻辑分区3	/dev/sda7

注意：如果只有一个主分区，一个扩展分区，扩展分区下有三个逻辑分区。那么主分区的设备文件名为/dev/sda1,扩展分区的设备文件名为 /dev/sda2。而逻辑分区直接是 /dev/sda5，也就是说系统默认的设备文件名从/dev/sda1—/dev/sda4是给主分区和扩展分区命名的，而逻辑分区的设备文件名是从/dev/sda5开始的。

3、Linux 文件系统的格式

ext2：是ext文件系统的升级版本，Red Hat Linux7.2版本以前的系统默认都是ext2文件系统。1993年发布，最大支持16TB的分区和最大2TB的文件（
1TB=1024GB=1024*1024KB）

ext3: ext3文件系统是ext2文件系统的升级版本，最大的区别就是带日志功能，以在系统突然停止时提高文件系统的可靠性。支持最大16TB的分区和最大2TB的文件

ext4: 它是ext3文件系统的升级版。ext4 在性能、伸缩性和可靠性方面进行了大量改进。EXT4 的变化可以说是翻天覆地的，比如向下兼容 EXT3、最大1EB文件系统和16TB文件、无限数量子目录、Extents连续数据块概念、多块分配、延迟分配、持久预分配、快速FSCK、日志校验、无日志模式、在线碎片整理、inode增强、默认启用barrier等。是CentOS 6.3的默认文件系统

(1EB=1024PB=1024*1024TB)

4、文件系统的常用命令

①、文件系统查看命令：df

```
1 df 【选项】 【挂载点】
```

选项：

- a 显示所有的文件系统信息，包括特殊文件系统，如 /proc、/sysfs
- h 使用习惯单位显示容量，如KB，MB或GB等
- T 显示文件系统类型
- m 以MB为单位显示容量
- k 以KB为单位显示容量。默认就是以KB为单位

使用的比较多的就是 df -h

```
[root@node3 /]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda5        13G  2.6G  9.5G  22% /
tmpfs           491M    0  491M   0% /dev/shm
/dev/sda1       190M   33M  147M  19% /boot
/dev/sda2        4.7G  760M  3.7G  17% /home
[root@node3 /]# df -h /
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda5        13G  2.6G  9.5G  22% /
[root@node3 /]#
```

②、统计目录或文件大小: du

```
1 du 【选项】 【目录或文件名】
```

选项:

- a 显示每个子文件的磁盘占用量。默认只统计子目录的磁盘占用量
- h 使用习惯单位显示磁盘占用量，如KB， MB或GB等
- s 统计总占用量，而不列出子目录和子文件的占用量

我们说通过 ls 命令也能统计文件大小，但是 ls 命令只是文件的一级目录，而 du 能显示目录以及所有子目录和文件的大小。

```
[root@node3 home]# du -sh /home
751M  /home
[root@node3 home]#
```

df 命令能统计文件或目录大小，而前面讲的df命令也能查看分区大小，这里我们以 /home 分区为例：

```
[root@node3 home]# df -h /home
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda2        4.7G  760M  3.7G  17% /home
[root@node3 home]# du -sh /home
751M  /home
```

df 统计的/home目录使用大小
du 统计的/home目录使用大小
为什么df统计的比du大？

df 统计的/home分区使用大小为 760M，而du 命令统计的/home分区使用大小为 751M，这是为什么呢？

①、df 命令是从文件系统考虑的，不管要考虑文件占用的空间，还要统计被命令或程序占用的空间（最常见的就是文件已经删除，但是程序并没有释放空间）。

②、du 命令是面向文件的，只会计算文件或目录占用的空间。

也就是说，实际系统的剩余空间大小是以 df 命令统计为准的。这也告诉我们 Linux 虽然系统很稳定，但是对于经常高负载的服务器，还是应该定期重启，维护服务器的高效运转。

③、文件系统修复命令：fsck

```
1 fsck 【选项】分区设备文件名
```

选项：

-a：不用显示用户提示，自动修复文件系统

-y：自动修复。和-a作用一致，不过有些文件系统只支持-y

系统在启动时会自动进行文件系统修复，这里我们最好不要手动执行文件系统修复命令，很容易造成意外的错误。

④、显示磁盘状态命令：dumpe2fs

```
1 dumpe2fs 分区设备文件名
```

```
[root@node3 home]# dumpe2fs /dev/sda1
dumpe2fs 1.41.12 (17-May-2010)
Filesystem volume name: <none>
Last mounted on: /boot 挂载点
Filesystem UUID: b9a4dcfe-b646-483d-aebd-796828b6b00d
Filesystem magic number: 0xEF53
Filesystem revision #: 1 (dynamic)
Filesystem features: has_journal ext_attr resize_inode dir_index filetype needed_extents flex_bg sparse_super huge_file uninit_bg dir_nlink extra_isize
Filesystem flags: signed_directory_hash
Default mount options: user_xattr acl 默认挂载选项
Filesystem state: clean
Errors behavior: Continue
Filesystem OS type: Linux
Inode count: 51200
Block count: 204800
Reserved block count: 10240
Free blocks: 160470
Free inodes: 51162
First block: 1
Block size: 1024
Fragment size: 1024
Reserved GDT blocks: 256
Blocks per group: 8192
Fragments per group: 8192
Inodes per group: 2048
Inode blocks per group: 256
Flex block group size: 16
Filesystem created: Thu Jun  8 03:44:49 2017
```

5、挂载命令

①、查询系统中已经挂载的设备：mount

```
1 mount [-l]
```

选项：-l 会显示卷标名称，也就是设备文件名的别名

```
[root@node3 home]# mount -l
/dev/sda5 on / type ext4 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
tmpfs on /dev/shm type tmpfs (rw,rootcontext="system_u:object_r:tmpfs_t:s0")
/dev/sda1 on /boot type ext4 (rw)
/dev/sda2 on /home type ext4 (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
[root@node3 home]#
```

②、依据配置文件 /etc/fstab 的内容自动挂载：mount -a

下面是 /etc/fstab 的文件内容：

```
# /dev/sda1 1
UUID=b9a4dcfe-b646-483d-aebd-796828b6b00d /boot 2
UUID=5a7b789e-c6f3-4bcf-b868-c03811cdb818 /home 2
UUID=f871cc80-26a3-43ce-9859-60939d17bd44 swap 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
```

注意：我们最好不要将光盘或者U盘的挂载配置写在 /etc/fstab 文件中，因为系统启动的时候会自动挂载这个文件中配置好的内容，你不可能光盘或者U盘一直都和系统连着，如果没有，则挂载出错，系统有可能奔溃。

③、挂载命令格式

[root@localhost ~]# mount [-t 文件系统] [-L 卷标名] \
[-o 特殊选项] 设备文件名 挂载点

选项：

-t 文件系统：加入文件系统类型来指定挂载的类型，可以ext3、ext4、iso9660等文件系统

-L 卷标名：挂载指定卷标的分区，而不是安装设备文件名挂载

-o 特殊选项：可以指定挂载的额外选项

参数	说明
atime/noatime	更新访问时间/不更新访问时间。访问分区文件时，是否更新文件的访问时间，默认为更新
async/sync	异步/同步，默认为异步
auto/noauto	自动/手动，mount -a命令执行时，是否会自动安装/etc/fstab文件内容挂载，默认为自动
defaults	定义默认值，相当于rw,suid,dev,exec,auto,nouser,async这七个选项
exec/noexec	执行/不执行，设定是否允许在文件系统中执行可执行文件，默认是exec允许
remount	重新挂载已经挂载的文件系统，一般用于指定修改特殊权限
rw/ro	读写/只读，文件系统挂载时，是否具有读写权限，默认是rw
suid/nosuid	具有/不具有SUID权限，设定文件系统是否具有SUID和SGID的权限，默认是具有
user/nouser	允许/不允许普通用户挂载，设定文件系统是否允许普通用户挂载，默认是不允许，只有root可以挂载分区
usrquota	写入代表文件系统支持用户磁盘配额，默认不支持
grpquota	写入代表文件系统支持组磁盘配额，默认不支持

范例：我们重新挂载 /home 分区，加上-o noexec，也就是说使得 /home 分区的可执行文件不能执行。然后创建一个脚本（后面会讲解，这里只需要知道是一个可执行文件），然后看其是否能执行。

第一步：在/home目录下创建脚本hello.sh，简单的输出 hello world

```
#!/bin/bash
echo "hello world"
~
```

我们给其赋予可执行权限，然后执行此脚本：

```
[root@node3 home]# vim hello.sh
[root@node3 home]# chmod 755 hello.sh
[root@node3 home]# ./hello.sh
hello world
[root@node3 home]#
```

第二步：重新挂载 /home分区，加上 -o noexec，在执行此脚本发现权限不够了，注意我们还是使用的超级用户root

```
[root@node3 home]# vim hello.sh
[root@node3 home]# chmod 755 hello.sh
[root@node3 home]# ./hello.sh
hello world
[root@node3 home]# mount -o remount,noexec /home
[root@node3 home]# ./hello.sh
-bash: ./hello.sh: 权限不够
[root@node3 home]#
```

第三步：将/home分区还原，然后在执行此脚本，发现又可以了

```
[root@node3 home]# mount -o remount,exec /home
[root@node3 home]# ./hello.sh
hello world
[root@node3 home]#
```

6、挂载光盘与U盘

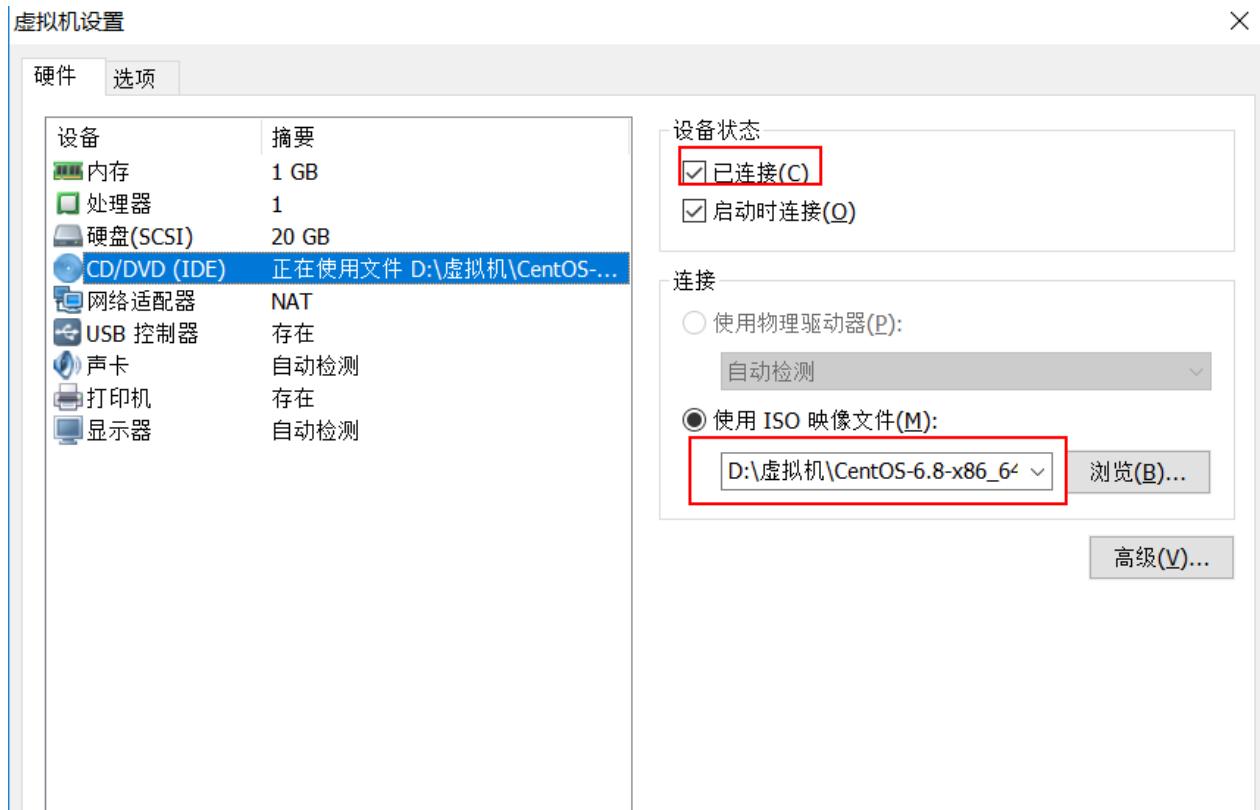
①、挂载光盘

一、建立挂载点

```
1 mkdir /mnt/cdrom
```

二、将光盘放入光驱

对于虚拟机我们执行以下操作即可：



三、挂载光盘

```
1 mount -t iso9660 /dev/cdrom /mnt/cdrom
```

或者执行下面命令

```
1 mount /dev/sr0 /mnt/cdrom
```

为什么有两个设备文件名/dev/cdrom 和/dev/sr0 呢？

```
[root@node3 home]# ll /dev/cdrom
lrwxrwxrwx. 1 root root 3 11月  5 19:09 /dev/cdrom -> sr0
[root@node3 home]#
```

我们可以看到这是一个软链接，光盘的设备文件名是系统自动检测的，以及固定了，我们只需要记住就好了。

②、挂载 U 盘

第一步：让虚拟机识别到 U 盘

注意我们是在真实物理机上安装的虚拟Linux系统，为了让虚拟机能检测到U盘，我们要鼠标点进到虚拟机中，不能用远程连接工具。

第二步：执行 fdisk -l 查看U 盘的设备文件名

第三步：挂载 U 盘

```
1 mount -t vfat /dev/sdb1 /mnt/usb
```

③、卸载命令

```
1 umount 设备文件名或者挂载点
```

```
[root@node3 home]# umount /mnt/cdrom
```

7、支持 NTFS 文件系统

我们知道 Linux 默认是不支持 NTFS 文件系统的，所以早期的苹果笔记本如果插上移动硬盘，是不能对硬盘的内容进行修改，只能读取的。

那么如何解决 Linux 系统不支持 NTFS 文件系统呢？

第一种方法是重新编译内核，这种方法要求较高，这里我们就不做演示了。

第二种方法是安装 NTFS-3G 插件，如下：

```
[root@localhost ~]# tar -zxvf ntfs-3g_ntfsprogs-2013.1.13.tgz  
#解压  
[root@localhost ~]# cd ntfs-3g_ntfsprogs-2013.1.13  
#进入解压目录  
[root@localhost ntfs-3g_ntfsprogs-2013.1.13]# ./configure  
#编译器准备。没有指定安装目录，安装到默认位置中  
[root@localhost ntfs-3g_ntfsprogs-2013.1.13]# make  
#编译  
[root@localhost ntfs-3g_ntfsprogs-2013.1.13]# make install  
#编译安装
```

安装完成之后，如下进行使用：

```
[root@localhost ~]# mount -t ntfs-3g 分区设备文件名 挂载点
```

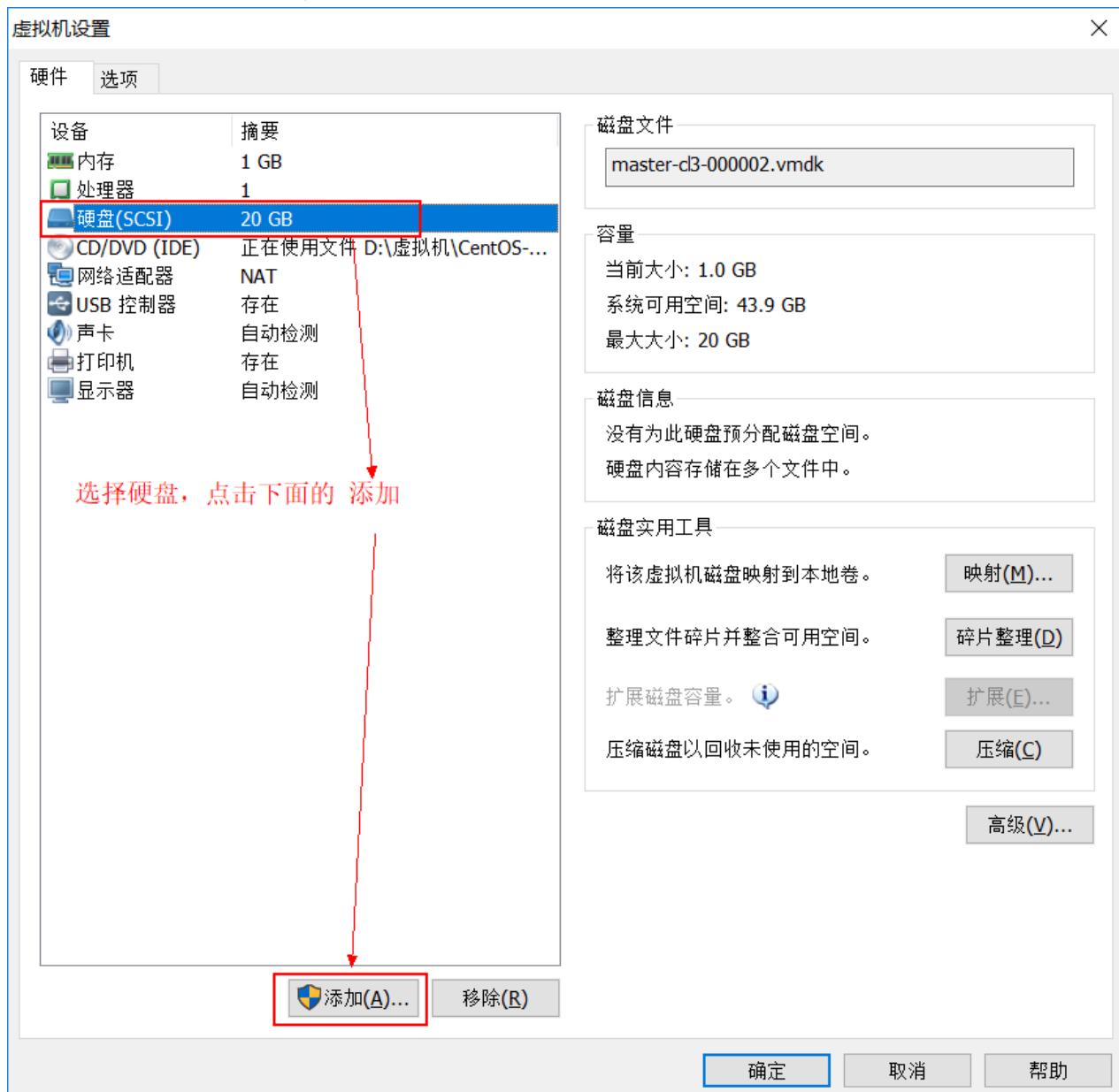
Linux文件系统管理之手工分区

1、添加新硬盘

我们在虚拟机上进行添加，注意要先关闭虚拟机。在进行虚拟机安装的时候，我们给 Linux 系统分配了一块20GB的硬盘，现在添加一块 10GB的。



关闭虚拟机后我们选择硬盘，点击下面的 添加 按钮。



然后依次下一步，下一步，硬盘大小输入 10 GB即可，最后点击完成。



2、查看新硬盘：fdisk -l

```
[root@node3 ~]# fdisk -l

Disk /dev/sda: 21.5 GB, 21474836480 bytes
255 heads, 63 sectors/track, 2610 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00008425a

      Device Boot      Start        End    Blocks   Id  System
/dev/sda1  *           1         26     204800   83  Linux
Partition 1 does not end on cylinder boundary.
/dev/sda2            26        664    5120000   83  Linux
Partition 2 does not end on cylinder boundary.
/dev/sda3            664        918    2048000   82  Linux swap / Solaris
Partition 3 does not end on cylinder boundary.
/dev/sda4            918       2611   13597696    5  Extended
/dev/sda5            919       2611   13596672   83  Linux

Disk /dev/sdb: 10.7 GB, 10737418240 bytes
255 heads, 63 sectors/track, 1305 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

[root@node3 ~]#
```

这就是我们刚刚添加的硬盘，设备文件名已经识别了

3、使用 fdisk 命令分区

我们已经添加了硬盘，并且硬盘已经被系统识别了，现在就对其进行分区。执行下面命令

```
1  fdish /dev/sdb
```

```
[root@node3 ~]# fdisk /dev/sdb
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF disklabel
Building a new DOS disklabel with disk identifier 0x805f9c0d.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)

WARNING: DOS-compatible mode is deprecated. It's strongly recommended to
         switch off the mode (command 'c') and change display units to
         sectors (command 'u').

Command (m for help): m
```

上面，我们输入 m 可以查看帮助，帮助如下：

fdisk交互指令说明	
命令	说明
a	设置可引导标记
b	编辑bsd磁盘标签
c	设置DOS操作系统兼容标记
d	删除一个分区
l	显示已知的文件系统类型。82为Linux swap分区，83为Linux分区
m	显示帮助菜单
n	新建分区
o	建立空白DOS分区表
p	显示分区列表
q	不保存退出
s	新建空白SUN磁盘标签
t	改变一个分区的系统ID
u	改变显示记录单位
v	验证分区表
w	保存退出
x	附加功能（仅专家）

第一步：按 n 新建分区

```
Command (m for help): n
Command action
  e   extended 按 e 表示创建扩展分区，而
  p   primary partition (1-4) 按p表示创建
                                主分区，最多只能有4个主
```

第二步：按 p 创建主分区

```
Command (m for help): n
Command action
  e  extended
  p  primary partition (1-4)
p                                         创建一个分区，大小为2G
Partition number (1-4): 1
First cylinder (1-1305, default 1): 1
Last cylinder, +cylinders or +size{K,M,G} (1-1305, default 1305): +2G

Command (m for help): █
```

第三步：主分区创建完成，这里我们在演示创建扩展分区，按e创建扩展分区

```
Command (m for help): n
Command action
  e  extended
  p  primary partition (1-4)
e
Partition number (1-4): 2
First cylinder (263-1305, default 263):
Using default value 263
Last cylinder, +cylinders or +size{K,M,G} (263-1305, default 1305):
Using default value 1305

Command (m for help): p

Disk /dev/sdb: 10.7 GB, 10737418240 bytes
255 heads, 63 sectors/track, 1305 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x805f9c0d

   Device Boot      Start        End    Blocks   Id  System
/dev/sdb1            1       262    2104483+  83  Linux
/dev/sdb2          263      1305    8377897+   5  Extended
Command (m for help): █
```

这时候，如果在按n，则会出现创建逻辑分区的选项：

```
Command (m for help): n
Command action
  l  logical (5 or over)      1 表示逻辑分区
  p  primary partition (1-4)
█
```

```

Command (m for help): n
Command action
  l  logical (5 or over)
  p  primary partition (1-4)
l
First cylinder (263-1305, default 263):
Using default value 263
Last cylinder, +cylinders or +size{K,M,G} (263-1305, default 1305): +2G

Command (m for help): p

Disk /dev/sdb: 10.7 GB, 10737418240 bytes
255 heads, 63 sectors/track, 1305 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x805f9c0d

      Device Boot      Start        End      Blocks   Id  System
/dev/sdb1            1       262     2104483+   83  Linux
/dev/sdb2          263      1305     8377897+   5  Extended
/dev/sdb5          263       524     2104483+   83  Linux

Command (m for help): 

```

分区完成之后，我们要输入 w，保存退出

```

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
[root@node3 ~]# 

```

第四步：要想让分区有效，可以重启，或者执行下面命令

```
1 partprobe
```

这是重新读取分区信息。下面的警告信息不管用。

```

[root@node3 ~]# partprobe
Warning: WARNING: the kernel failed to re-read the partition table on /dev/sda (设备或资源忙).
As a result, it may not reflect all of your changes until after reboot.
Warning: 无法以读写方式打开 /dev/sr0 (只读文件系统)。/dev/sr0 已按照只读方式打开。
Warning: 无法以读写方式打开 /dev/sr0 (只读文件系统)。/dev/sr0 已按照只读方式打开。
Error: 无效的分区表 - /dev/sr0 出现递归分区。
[root@node3 ~]# 

```

第五步：查看分区信息 fdisk -l

```
[root@node3 ~]# fdisk -l

Disk /dev/sda: 21.5 GB, 21474836480 bytes
255 heads, 63 sectors/track, 2610 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x0008425a

      Device Boot      Start        End    Blocks   Id  System
/dev/sda1  *           1         26     204800   83  Linux
Partition 1 does not end on cylinder boundary.
/dev/sda2            26        664    5120000   83  Linux
Partition 2 does not end on cylinder boundary.
/dev/sda3            664       918    2048000   82  Linux swap / Solaris
Partition 3 does not end on cylinder boundary.
/dev/sda4            918      2611   13597696    5  Extended
/dev/sda5            919      2611   13596672   83  Linux

Disk /dev/sdb: 10.7 GB, 10737418240 bytes
255 heads, 63 sectors/track, 1305 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x805f9c0d

      Device Boot      Start        End    Blocks   Id  System
/dev/sdb1            1         262    2104483+   83  Linux
/dev/sdb2            263       1305    8377897+    5  Extended
/dev/sdb5            263        524    2104483+   83  Linux
[root@node3 ~]#
```

第六步：格式化分区（注意不能格式化扩展分区）

```
1  mkfs -t ext4 /dev/sdb1
```

```
[root@node3 ~]# mkfs -t ext4 /dev/sdb1
mke2fs 1.41.12 (17-May-2010)
文件系统标签=
操作系统:Linux
块大小=4096 (log=2)
分块大小=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
131648 inodes, 526120 blocks
26306 blocks (5.00%) reserved for the super user
第一个数据块=0
Maximum filesystem blocks=541065216
17 block groups
32768 blocks per group, 32768 fragments per group
7744 inodes per group
Superblock backups stored on blocks:
      32768, 98304, 163840, 229376, 294912

正在写入inode表: 完成
Creating journal (16384 blocks): 完成
Writing superblocks and filesystem accounting information: 完成

This filesystem will be automatically checked every 22 mounts or
180 days, whichever comes first. Use tune2fs -c or -i to override.
[root@node3 ~]#
```

第七步：建立挂载点，并挂载分区

```
1 mkdir /disk1``mount /dev/sdb1 /disk1
```

```
[root@node3 ~]# mkdir /disk1
[root@node3 ~]# mount /dev/sdb1 /disk1
[root@node3 ~]# cd /disk1/
[root@node3 disk1]# ll
总用量 16
drwx----- 2 root root 16384 11月 11 14:42 lost+found
[root@node3 disk1]#
```

```
[root@node3 disk1]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda5        13G  2.6G  9.5G  22% /
tmpfs           491M    0  491M   0% /dev/shm
/dev/sda1       190M   33M  147M  19% /boot
/dev/sda2        4.7G  760M  3.7G  17% /home
/dev/sdb1        2.0G  3.1M  1.9G   1% /disk1
/dev/sdb5        2.0G  3.1M  1.9G   1% /disk5
[root@node3 disk1]#
```

4、分区自动挂载

上面我们将硬盘分区，并进行了挂载，但是我们发现如果系统重启，那么分区又不见了，又需要我们重新挂载，那么怎么能实现系统重启自动挂载呢？

①、/etc/fstab 文件

- ◆ 第一字段：分区设备文件名或UUID（硬盘通用唯一识别码）
- ◆ 第二字段：挂载点
- ◆ 第三字段：文件系统名称
- ◆ 第四字段：挂载参数
- ◆ 第五字段：指定分区是否被dump备份，0代表不备份，1代表每天备份，2代表不定期备份
- ◆ 第六字段：指定分区是否被fsck检测，0代表不检测，其他数字代表检测的优先级，那么当然1的优先级比2高

添加如下信息：

UUID=490ed737-f8cf-46a6-ac4b-b7735b79fc63	/	ext4	defaults,acl	.
UUID=b9a4dcfe-b646-483d-aebd-796828b6b00d	/boot	ext4	defaults	1 2
UUID=5a7b789e-c6f3-4bcf-b868-c03811cdb818	/home	ext4	defaults	1 2
UUID=f871cc80-26a3-43ce-9859-60939d17bd44	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
/dev/sdb1	/disk1	ext4	defaults	1 2

Linux的shell概述以及如何执行脚本

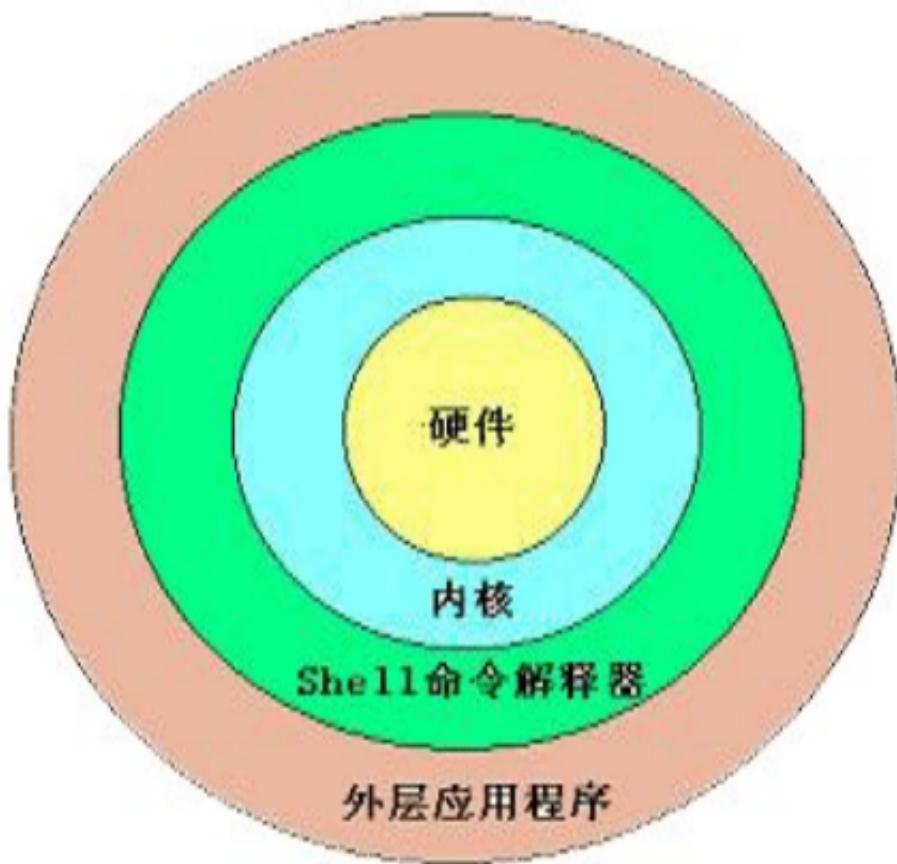
1、Shell 是什么？

和大家通常说的 shell 是一门编程语言的说法其实是不完全正确的。

①、shell 是一个命令行解释器，它为用户提供了一个向 Linux 内核发送请求以便运行程序的界面系统级程序，用户可以用 shell 来启动、挂起、停止甚至是编写一些程序。

比如，我们现在操作的界面就是 shell，我们输入一些命令，通过 shell 去翻译为机器语言，然后由硬件去执行这些命令。

②、shell 是一个功能很强大的编程语言，它易编写、易调试，而且灵活性强。shell 是解释执行的脚本语言，在 shell 中可以直接调用 Linux 系统命令。



其实 windows 类似于 shell 的就是它的图形交互界面，我们在 windows 系统中打开文件，删除文件等操作，就是向 windows 系统发送命令。

2、Shell 的分类

- ◆ Bourne Shell：从1979起Unix就开始使用 Bourne Shell， Bourne Shell的主文件名为 sh。
- ◆ C Shell： C Shell主要在BSD版的Unix系统中使用， 其语法和C语言相类似而得名
- ◆ Shell的两种主要语法类型有Bourne和C，这两种语法彼此不兼容。 Bourne家族主要包括sh、ksh、Bash、psh、zsh； C家族主要包括：csh、tcsh

◆ Bash: Bash与sh兼容，现在使用的Linux就是使用Bash作为用户的基本Shell。

3、查看Linux系统支持的 shell: /etc/shells

```
[root@node3 disk1]# more /etc/shells
/bin/sh
/bin/bash
/sbin/nologin
/bin/dash
/bin/tcsh
/bin/csh
[root@node3 disk1]#
```

我们可以在shell解释器中直接输入shell名称进行切换

```
[root@node3 disk1]# sh
sh-4.1# exit
exit
[root@node3 disk1]# csh
[root@node3 /disk1]# exit
exit
[root@node3 disk1]#
```

4、echo 输出命令

这个命令如果学过Java的类似于 System.out.println(), 如果学过C语言的类似于 printf(), 在shell 当中语法形式如下:

```
1 echo 【选项】 【输出内容】
```

-e 表示支持反斜杠控制的字符转换，也就是转义字符。

控制字符	作用
\\\	输出\本身
\a	输出警告音
\b	退格键，也就是向左删除键
\c	取消输出行末的换行符。和“-n”选项一致
\e	ESCAPE键
\f	换页符
\n	换行符
\r	回车键
\t	制表符，也就是Tab键
\v	垂直制表符
\0nnn	按照八进制ASCII码表输出字符。其中0为数字零，nnn是三位八进制数
\xhh	按照十六进制ASCII码表输出字符。其中hh是两位十六进制数

范例：

```
[root@node3 disk1]# echo "hello world" 原样打印
hello world
[root@node3 disk1]# echo -e "ab\bc" 删除左侧字符
ac
[root@node3 disk1]# echo -e "a\tb\tc\nd\tc\tf"
a      b      c          制表符与换行符
d      e      f
[root@node3 disk1]#
```

5、脚本执行方式

首先我们编写一个shell 脚本。通过 vi hello.sh, 打开 hello.sh 文件，然后在文件中添加如下内容：

```
#!/bin/bash
echo "hello world"
~
```

①、我们说Linux系统是不区分文件后缀名的，这里我们学习的是 bash,所以创建文件 hello.sh，后缀名最好加上.sh（虽然不加也没问题），便于我们识别。

②、脚本的第一行 #!,这是一个约定的标记，它告诉系统这个脚本需要用什么解释器去执行，即使用哪一种 shell，所以学习 bash，第一行固定都是 #!/bin/bash。这是不能省略的。

创建完毕之后，接下来执行该脚本，有如下两种方式：

①、作为可执行程序

```
1 cd /tmp          #进入到脚本所在的目录``chmod +x ./hello.sh #使得脚本具有可执行权限
``./hello.sh      #执行脚本
```

注意，一定要写成 `./hello.sh`，而不是 `hello.sh`，运行其它二进制的程序也一样，直接写 `hello.sh`，linux 系统会去 PATH 里寻找有没有叫 `hello.sh` 的（这是后面会讲的环境变量的配置），而只有 `/bin`, `/sbin`, `/usr/bin`, `/usr/sbin` 等在 PATH 里，你的当前目录通常不在 PATH 里，所以写成 `hello.sh` 是会找不到命令的，要用 `./hello.sh` 告诉系统说，就在当前目录找。

```
[root@node3 tmp]# cd /tmp
[root@node3 tmp]# chmod +x ./hello.sh
[root@node3 tmp]# ./hello.sh
hello world
[root@node3 tmp]#
```

②、作为解释器参数

也就是直接运行解释器，其参数就是 shell 脚本的文件名，如下：

```
1 /bin/sh hello.sh``或者``bash hello.sh
```

注意：这种方式执行脚本，并不需要第一行写上 `#!/bin/bash`。

Linux的bash基本功能

1、历史命令

```
[root@localhost ~]# history [选项] [历史命令保存文件]
选项:
```

- c: 清空历史命令
- w: 把缓存中的历史命令写入历史命令保存文件
`~/.bash_history`

◆ 历史命令默认会保存1000条，可以在环境变量配置文件`/etc/profile`中进行修改

历史命令的调用

- ◆ 使用上、下箭头调用以前的历史命令
- ◆ 使用“!n”重复执行第n条历史命令
- ◆ 使用“!!”重复执行上一条命令
- ◆ 使用“!字符串”重复执行最后一条以该字符串开头的命令

```
[root@node3 tmp]# history
1 ifconfig
2 vi /etc/sysconfig/network-scripts/ifcfg-eth0
3 ifconfig
4 vi /etc/resolv.conf
5 ifconfig
6 service network restart
7 vi /etc/sysconfig/network-scripts/ifcfg-eth0
8 service network restart
9 ifconfig
10 ping 192.168.56.1
11 ping 192.168.146.200
12 ping www.baidu.com
13 vi /etc/resolv.conf
14 ping www.baidu.com
15 service network restart
16 ping www.baidu.com
17 vi /etc/resolv.conf
18 ifconfig
```

2、命令与文件补全：Tab

- ◆ 在Bash中，命令与文件补全是非常方便与常用的功能，我们只要在输入命令或文件时，按“Tab”键就会自动进行补全

在输入一些命令的时候，最好用Tab键去补全，防止命令输入错误。

3、命令的别名：alias

```
[root@localhost ~]# alias 别名='原命令'  
#设定命令别名
```

```
[root@localhost ~]# alias  
#查询命令别名
```

```
[root@node3 tmp]# alias          直接敲 alias 能查看当前系统中定义的别名  
alias cp='cp -i'  
alias l.='ls -d .* --color=auto'  
alias ll='ls -l --color=auto'  
alias ls='ls --color=auto'  
alias mv='mv -i'  
alias rm='rm -i'  
alias which='alias | /usr/bin/which --tty-only --read-alias --show-dot --show-tilde'  
[root@node3 tmp]# alias vi='vim'  alias vi='vim' 将vi命令等价于vim命令  
[root@node3 tmp]#
```

让别名永远生效: vim /root/.bashrc

```
# .bashrc  
  
# User specific aliases and functions  
  
alias rm='rm -i'  
alias cp='cp -i'  
alias mv='mv -i'  
alias vi='vim'  
# Source global definitions  
if [ -f /etc/bashrc ]; then  
    . /etc/bashrc  
fi
```

删除别名

```
1 unalias 别名
```

4、命令的执行顺序

- 1 第一顺位执行用绝对路径或相对路径执行的命令。
- 2 第二顺位执行别名。
- 3 第三顺位执行Bash的内部命令。
- 4 第四顺位执行按照\$PATH环境变量定义的目录查找顺序找到的第一个命令。

5、bash 常用快捷键

快 捷 键	作 用
ctrl+A	把光标移动到命令行开头。如果我们输入的命令过长，想要把光标移动到命令行开头时使用。
ctrl+E	把光标移动到命令行结尾。
ctrl+C	强制终止当前的命令。
ctrl+L	清屏，相当于clear命令。
ctrl+U	删除或剪切光标之前的命令。我输入了一行很长的命令，不用使用退格键一个一个字符的删除，使用这个快捷键会更加方便
ctrl+K	删除或剪切光标之后的内容。
ctrl+Y	粘贴ctrl+U或ctrl+K剪切的内容。
ctrl+R	在历史命令中搜索，按下ctrl+R之后，就会出现搜索界面，只要输入搜索内容，就会从历史命令中搜索。
ctrl+D	退出当前终端。
ctrl+Z	暂停，并放入后台。这个快捷键牵扯工作管理的内容，我们在系统管理章节详细介绍。
ctrl+S	暂停屏幕输出。
ctrl+Q	恢复屏幕输出。

6、输入输出重定向

①、标准输入输出

设备	设备文件名	文件描述符	类型
键盘	/dev/stdin	0	标准输入
显示器	/dev/stdout	1	标准输出
显示器	/dev/stderr	2	标准错误输出

②、输出重定向：将命令执行结果本该显示在屏幕上的存储到别的地方

类型	符号	作用
标准输出重定向	命令 > 文件	以覆盖的方式，把命令的正确输出输出到指定的文件或设备当中。
	命令 >> 文件	以追加的方式，把命令的正确输出输出到指定的文件或设备当中。
标准错误输出重定向	错误命令 2>文件	以覆盖的方式，把命令的错误输出输出到指定的文件或设备当中。
	错误命令 2>>文件	以追加的方式，把命令的错误输出输出到指定的文件或设备当中。

正确输出和错误输出同时保存	命令 > 文件 2>&1	以覆盖的方式，把正确输出和错误输出都保存到同一个文件当中。
	命令 >> 文件 2>&1	以追加的方式，把正确输出和错误输出都保存到同一个文件当中。
	命令 &>文件	以覆盖的方式，把正确输出和错误输出都保存到同一个文件当中。
	命令 &>>文件	以追加的方式，把正确输出和错误输出都保存到同一个文件当中。
	命令>>文件1 2>>文件2	把正确的输出追加到文件1中，把错误的输出追加到文件2中。

③、输入重定向：本该由键盘输入的信息改为由文件进行输入

输入重定向用的很少。在讲解输入重定向之前，我们先介绍一个命令：

[root@localhost ~]# wc [选项] [文件名]

选项：

- c 统计字节数
- w 统计单词数
- l 统计行数

```
[root@node3 tmp]# wc
abc
abc
abc
3      3     12
[root@node3 tmp]#
```

- ◆ 命令<文件 把文件作为命令的输入
- ◆ 命令<< 标识符
- ...
- 标识符 把标识符之间内容作为命令的输入

范例：统计 file 文件

```
[root@node3 tmp]# wc < file
 1 5 30
[root@node3 tmp]# wc << hello
> afdb
> afdsf
> afds
> hello
 3 3 16
[root@node3 tmp]#
```

7、多命令顺序执行

多命令执行符	格式	作用
:	命令1；命令2	多个命令顺序执行，命令之间没有任何逻辑联系
&&	命令1 && 命令2	逻辑与 当命令1正确执行，则命令2才会执行 当命令1执行不正确，则命令2不会执行
	命令1 命令2	逻辑或 当命令1执行不正确，则命令2才会执行 当命令1正确执行，则命令2不会执行

第一个；，两个命令是没有任何逻辑关系的，即使第一个命令出错了，第二个命令还是会执行。

```
[root@node3 tmp]# lsa;date
-bash: lsa: command not found
2017年 11月 11日 星期六 22:47:15 CST
[root@node3 tmp]#
```

8、管道符 命令1 | 命令2

命令格式：

```
[root@localhost ~]# 命令1 | 命令2  
#命令1的正确输出作为命令2的操作对象
```

```
[root@node3 tmp]# ll -a /etc | more
```

9、通配符

通配符	作用
?	匹配一个任意字符
*	匹配0个或任意多个任意字符，也就是可以匹配任何内容
[]	匹配中括号中任意一个字符。例如：[abc]代表一定匹配一个字符，或者是a，或者是b，或者是c。
[-]	匹配中括号中任意一个字符，-代表一个范围。例如：[a-z]代表匹配一个小写字母。
[^]	逻辑非，表示匹配不是中括号内的一个字符。例如：[^0-9]代表匹配一个不是数字的字符。

范例：

```
[root@localhost ~]# cd /tmp/  
[root@localhost tmp]# rm -rf *  
[root@localhost tmp]# touch abc  
[root@localhost tmp]# touch abcd  
[root@localhost tmp]# touch 012  
[root@localhost tmp]# touch 0abc  
[root@localhost tmp]# ls ?abc  
[root@localhost tmp]# ls [0-9]*  
[root@localhost tmp]# ls [^0-9]*
```

10、bash 中的其他特殊符号

符 号	作 用
,	单引号。在单引号中所有的特殊符号，如“\$”和“`”(反引号)都没有特殊含义。
””	双引号。在双引号中特殊符号都没有特殊含义，但是“\$”、“`”和“\”是例外，拥有“调用变量的值”、“引用命令”和“转义符”的特殊含义。
..	反引号。反引号括起来的内容是系统命令，在Bash中会先执行它。和\$()作用一样，不过推荐使用\$(), 因为反引号非常容易看错。
\$()	和反引号作用一样，用来引用系统命令。
#	在Shell脚本中，#开头的行代表注释。
\$	用于调用变量的值，如需要调用变量name的值时，需要用\$name的方式得到变量的值。
\	转义符，跟在\之后的特殊符号将失去特殊含义，变为普通字符。如\\$将输出“\$”符号，而不当做是变量引用。

```
[root@node3 tmp]# name=vae
[root@node3 tmp]# echo '$name'    单引号里面的内容都原样输出
$name
[root@node3 tmp]# echo "$name"    双引号里面的特殊
vae                         字符会进行转义输出
[root@node3 tmp]# echo '$(date)'  $(date)
$(date)
[root@node3 tmp]# echo "$(date)"  2017年 11月 11日 星期六 23:03:30 CST
2017年 11月 11日 星期六 23:03:30 CST
[root@node3 tmp]#
```

Linux 的 bash 变量

1、什么是变量

变量是计算机内存的单元，其中存放的值可以改变。

当 shell 脚本需要保存一些信息时，如一个文件名或者一个数字，就把它放在一个变量里。每个变量都有一个名字，可以根据名字来引用变量。

使用变量可以保存有用信息，使系统获知用户的相关设置，变量也可以保存暂时信息。

2、变量的声明规则

- ◆ 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。如果变量名是“2name”则是错误的。
- ◆ 在Bash中，变量的默认类型都是字符串型，如果要进行数值运算，则必须指定变量类型为数值型。
- ◆ 变量用等号连接值，等号左右两侧不能有空格。
- ◆ 变量的值如果有空格，需要使用单引号或双引号包括。
- ◆ 在变量的值中，可以使用“\”转义符。
- ◆ 如果需要增加变量的值，那么可以进行变量值的叠加。不过变量需要用双引号包含“\$变量名”或用\${变量名}包含。
- ◆ 如果是把命令的结果作为变量值赋予变量，则需要使用反引号或\$()包含命令。
- ◆ 环境变量名建议大写，便于区分。

```
[root@node3 tmp]# name=tom          1、正确声明变量
[root@node3 tmp]# 2name=tom          2、变量不能以数字开头
-bash: 2name=tom: command not found
[root@node3 tmp]# name = tom         3、变量声明等号两边不能有
-bash: name: command not found    空格
[root@node3 tmp]# name=tom bob      4、变量的值如果有空格，必
-bash: bob: command not found    须用单引号或双引号括起来
[root@node3 tmp]#
```

3、变量的分类

- ◆ 用户自定义变量
- ◆ 环境变量：这种变量中主要保存的是和系统操作环境相关的数据。
- ◆ 位置参数变量：这种变量主要是用来向脚本当中传递参数或数据的，变量名不能自定义，变量作用是固定的。
- ◆ 预定义变量：是Bash中已经定义好的变量，变量名不能自定义，变量作用也是固定的。

4、用户自定义变量的用法

用户自定义变量也就是本地变量。只在当前 shell 中生效。

①、变量定义

```
1 变量名=变量值
```

```
[root@node3 tmp]# name=tom  
[root@node3 tmp]# name="tom bob"  
[root@node3 tmp]# █
```

②、变量调用

```
1 $变量名
```

```
[root@node3 tmp]# name="tom bob"  
[root@node3 tmp]# echo $name  
tom bob  
[root@node3 tmp]# █
```

③、变量查看

```
1 set
```

set 是查看当前系统中定义的所有变量。

```
SELINUX_ROLE_REQUESTED=
SELINUX_USE_CURRENT_RANGE=
SHELL=/bin/bash
SHLOPTS=braceexpand:emacs :hashall:histexpand:history:
SHLVL=1
SSH_CLIENT='192.168.146.1 50718 22'
SSH_CONNECTION='192.168.146.1 50718 192.168.146.253 22'
SSH_TTY=/dev/pts/1
TERM=xterm
UID=0
USER=root
_=name
colors=/etc/DIR_COLORS
name='tom bob'
```

④、变量删除

```
1 unset 变量名
```

5、环境变量的用法

环境变量主要保存的是和系统操作环境相关的变量。

前面讲的用户自定义变量（本地变量）只在当前shell 中生效，而环境变量会在当前shell和这个shell的所有子shell当中生效，如果把环境变量写入相应的配置文件中，那么这个环境变量就会在所有的shell中生效。

①、通过 pstree 命令区分当前shell 的级别是父还是子

```
[root@node3 tmp]# csh      通过 csh 切换到csh
[root@node3 /tmp]# pstree
init--abrt
|-acpid
|-atd
|-auditd--{auditd}
|-automount--4*[{automount}]
|-certmonger
|-crond
|-cupsd
|-dbus-daemon--{dbus -daemon}
|-halld--halld-runner--halld-addon-acpi
|   |-halld-addon-inpu
|   |-halld-addon-rfki
|-{halld}
|-master--pickup
|   |-qmgr
|-mcelog
|-6*[mingetty]
|-rpc.statd
|-rpcbind
|-rsyslogd--3*[{rsyslogd}]
|-sshd--sshd--sftp-server
|   |-sshd--bash  bash是父, csh是子
|   |-sshd--bash--csh--pstree
|-udevd--2*[udevd]
[root@node3 /tmp]#
```

②、声明环境变量

```
1  export 变量名=变量值
```

③、查询所有环境变量

```
1  env
```

前面讲的 set 命令是查看所有变量，而 env 是查看环境变量。

④、查看、删除指定环境变量

```
1  $变量名      #查看环境变量``unset 变量名  #删除环境变量
```

这两个和本地变量一样。

```
[root@node3 tmp]# export name=vae  
[root@node3 tmp]# echo $name  
vae  
[root@node3 tmp]# bash  
[root@node3 tmp]# echo $name  
vae  
[root@node3 tmp]#
```

1、通过export声明环境变量
2、调用环境变量
3、切换到子shell，也是bash
4、在子shell中也能调用父shell的环境变量

⑤、系统查找命令的路径环境变量 \$PATH

我们知道调用命令必须要是在当前目录，或者是用绝对路径进行。但是实际上我们调用某个命令直接使用命令名就可以了，比如cd,ls等等这些常用的，这是为什么呢？

原因就是在 *PATH*里面我们已经定义好了，我们执行某个名称的命令，系统会首先去 *PATH*里面我们已经定义好了，我们执行某个名称的命令，系统会首先去 *PATH*里面查找，如果找不到才会报找不到命令错误。

首先看一下\$PATH:

```
[root@node3 tmp]# echo $PATH  
/usr/lib64/qt-3.3/bin:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/usr/java/jdk1.7/bin  
:/home/hadoop/hbase0.99/bin:/home/hadoop/modules/hadoop2.7/bin:/home/hadoop/modules/hadoop2.7/sbin:/root/bin  
[root@node3 tmp]#
```

可以看到，\$PATH 里面都是保存的一些路径，然后通过:分隔。

前面我们写过一个 hello.sh 的脚本，我们知道要想执行它，必须首先赋予可执行权限，然后要么在当前目录通过./hello.sh来执行，要么通过绝对路径/tmp/hello.sh来执行。如果我们想直接通过 hello.sh 来执行呢？

解决办法就将hello.sh的所在路径添加到 \$PATH 变量中，如下：

```
[root@node3 /]# hello.sh  
bash: hello.sh: command not found  
[root@node3 /]# PATH="$PATH":/tmp  
[root@node3 /]# hello.sh  
hello world  
[root@node3 /]#
```

⑥、定义系统提示符的变量 \$PS1

\d: 显示日期，格式为“星期月日”
\h: 显示简写主机名。如默认主机名“localhost”
\t: 显示24小时制时间，格式为“HH:MM:SS”
\T: 显示12小时制时间，格式为“HH:MM:SS”
\A: 显示24小时制时间，格式为“HH:MM”
\u: 显示当前用户名
\w: 显示当前所在目录的完整名称
\W: 显示当前所在目录的最后一个目录
\#: 执行的第几个命令
\\$: 提示符。如果是root用户会显示提示符为“#”，如果是普通用户会显示提示符为“\$”

```
[root@node3 /]# echo $PS1
[\u@\h \W]\$
[root@node3 /]# PS1='[当前时间\u@\t \w]\$'
[当前时间root@13:51:09 /]#
```

6、位置参数变量的用法

位置参数变量	作用
\$n	n为数字，\$0代表命令本身，\$1-\$9代表第一到第九个参数，十以上的参数需要用大括号包含，如\${10}.
\$*	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
\$@	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
\$#	这个变量代表命令行中所有参数的个数

```

[root@node3 tmp]#vim test.sh          编写 test.sh脚本，并用cat命令
[root@node3 tmp]#cat test.sh          查看
#!/bin/bash
echo $0
echo $1
echo $2
[root@node3 tmp]#chmod +x test.sh    赋予 test.sh 脚本执行权限
[root@node3 tmp]#./test.sh 1 2        执行test.sh，并后面带两个参数
./test.sh
1
2
[root@node3 tmp]#                         $0 表示命令本身
                                         $1 表示第一个参数
                                         $2 表示第二个参数

```

7、预定义变量的用法

其实预定义变量也是位置参数变量的一种，有如下几种用法：

预定义变量	作用
\$?	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。
\$\$	当前进程的进程号（PID）
\$!	后台运行的最后一个进程的进程号（PID）

8、声明变量类型 declare

[root@localhost ~]# declare [+/-][选项] 变量名

选项：

- : 给变量设定类型属性
- +: 取消变量的类型属性
- i: 将变量声明为整数型 (integer)
- x: 将变量声明为环境变量
- p: 显示指定变量的被声明的类型

9、数值运算的三种方法

①、declare -i

```
[root@node3 tmp]#a=1  
[root@node3 tmp]#b=2  
[root@node3 tmp]#c=$a+$b  
[root@node3 tmp]#echo $c  
1+2  
[root@node3 tmp]#declare -i c=$a+$b  
[root@node3 tmp]#echo $c  
3  
[root@node3 tmp]#
```

1、声明变量 a b
2、声明变量 c=\$a+\$b
3、声明变量 c 为整型

②、expr 或 let 数值运算工具

```
[root@node3 tmp]#d=$(expr $a+$b)  
[root@node3 tmp]#echo $d  
1+2  
[root@node3 tmp]#d=$(expr $a + $b)  
[root@node3 tmp]#echo d  
d  
[root@node3 tmp]#echo $d  
3  
[root@node3 tmp]#
```

使用 expr 求和的时候，
+ 两侧必须要有空格

③、((运算式))或[[运算式]]或[运算式]

```
[root@node3 tmp]#echo $($a+$b)  
3  
[root@node3 tmp]#echo ${a+$b}  
3  
[root@node3 tmp]#
```

10、运算符及其优先级顺序

优先级	运算符	说明
13	-, +	单目负、单目正
12	!, ~	逻辑非、按位取反或补码
11	* , / , %	乘、除、取模
10	+,-	加、减
9	<< , >>	按位左移、按位右移
8	<=, >=, <, >	小于或等于、大于或等于、小于、大于
7	==, !=	等于、不等于
6	&	按位与
5	^	按位异或
4		按位或
3	&&	逻辑与
2		逻辑或
1	=, +=, -= =, *=, /=, %=, &=, ^= =, <<=, >>=	赋值、运算且赋值

```
[root@node3 tmp]#echo $((1+2/2))
2
[root@node3 tmp]#echo $((1&&0))
0
[root@node3 tmp]#
```

11、变量测试与替换

变量置换方式	变量y没有设置	变量y为空值	变量y设置值
x=\${y-新值}	x=新值	x为空	x=\$y
x=\${y:-新值}	x=新值	x=新值	x=\$y
x=\${y:+新值}	x为空	x=新值	x=新值
x=\${y:+新值}	x为空	x为空	x=新值
x=\${y:=新值}	x=新值 y=新值	x为空 y值不变	x=\$y y值不变
x=\${y:=新值}	x=新值 y=新值	x=新值 y=新值	x=\$y y值不变
x=\${y?新值}	新值输出到标准 错误输出（就是 屏幕）	x为空	x=\$y
x=\${y:?新值}	新值输出到标准 错误输出	新值输出到标准 错误输出	x=\$y

12、环境变量配置文件

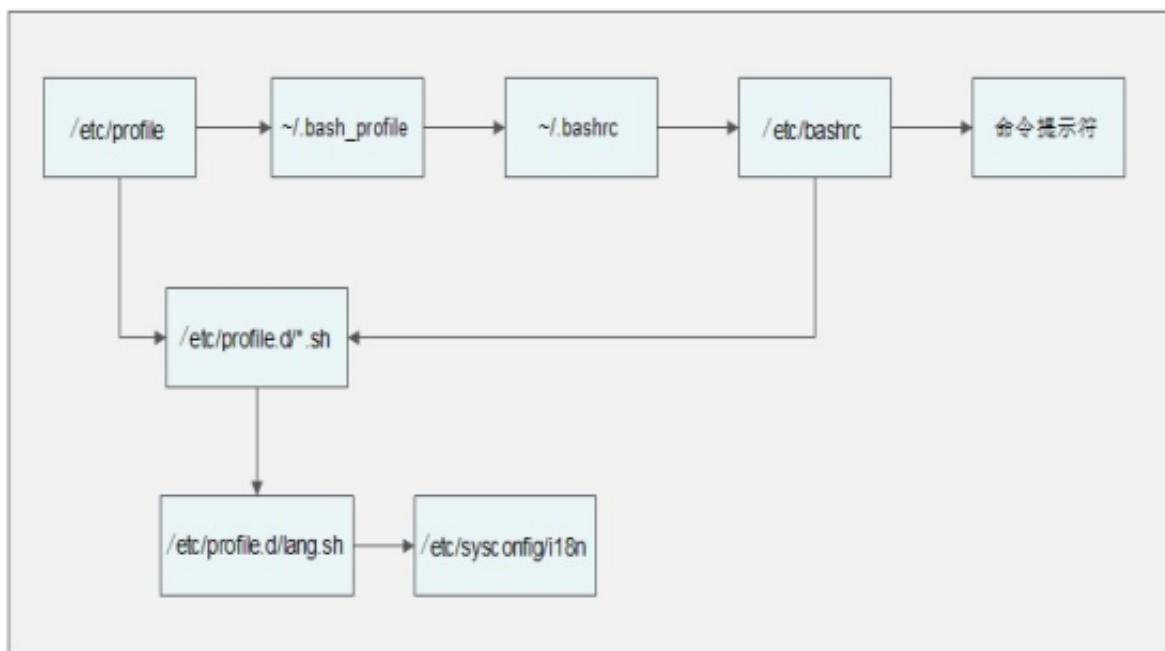
前面我们在将环境变量的时候，我们将脚本的路径加入到 *PATH* 路径中，然后就可以直接通过脚本的名称来执行脚本。但是如果系统重启之后，发现又必须要重新加入到 *PATH* 路径中，然后就可以直接通过脚本的名称来执行脚本。但是如果系统重启之后，发现又必须要重新加入到 *PATH* 变量中才可以。那么有没有让它一直生效的方法呢？

解决办法就是在环境变量配置文件中加入脚本的路径。

环境变量配置文件中主要定义对系统的操作环境生效的系统默认环境变量，比如 *PATH*, *HISTSIZE*, *PS1*, *HOSTNAME* 等默认环境变量。分别有以下配置文件：

- ◆ /etc/profile
- ◆ /etc/profile.d/*.sh
- ◆ ~/.bash_profile
- ◆ ~/.bashrc
- ◆ /etc/bashrc

下图是环境变量文件的读取顺序：（用户每次重新登录的时候就会重新读取下面的配置文件）



我们知道 *PATH* 变量，用户每次登录，就会从上面的文件读取顺序读取所有配置文件，最后得到 *PATH* 变量，用户每次登录，就会从上面的文件读取顺序读取所有配置文件，最后得到 *PATH* 变量的值。

注意：越往后面的配置文件，里面配置的变量内容优先级越高。

/etc/profile的作用：

- ◆ USER变量：
- ◆ LOGNAME变量：
- ◆ MAIL变量：
- ◆ PATH变量：
- ◆ HOSTNAME变量：
- ◆ HISTSIZE变量：
- ◆ umask：
- ◆ 调用/etc/profile.d/*.sh文件

~/.bash_profile的作用

- ◆ 调用了~/.bashrc文件。
- ◆ 在PATH变量后面加入了“:\$HOME/bin”这个目录

~/.bashrc的作用

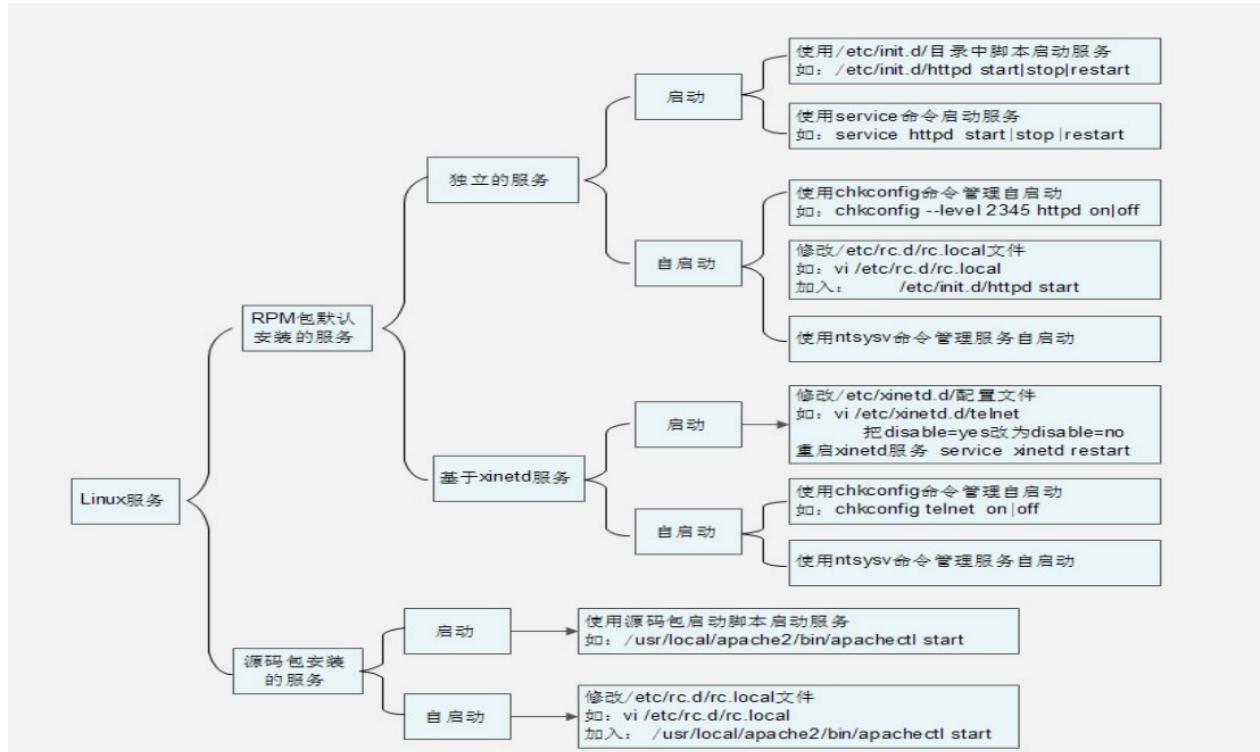
- ◆ 定义默认别名
- ◆ 调用/etc/bashrc

/etc/bashrc的作用

- ◆ PS1变量
- ◆ umask
- ◆ PATH变量
- ◆ 调用/etc/profile.d/*.sh文件

Linux的服务管理

Linux服务管理总览



我们可以看到Linux的服务分为 RPM包安装的服务以及源码包安装的服务，而RPM包服务又分为独立服务和基于 xinetd 服务。本篇博客就分别介绍着三种服务的启动和如何自启动。

2、启动和自启动

- ①、启动：启动某个服务就是在当前系统让服务运行，并提供功能。
- ②、自启动：自启动某个服务，就是在系统开机或重启动之后，随着系统启动而自动启动的服务。

3、RPM包和源码包服务启动差别根本原因

其实不管是RPM包，还是源码包，只不过是初始安装方式不一样而已，如果已经在Linux系统上安装完成之后，那么启动服务都是可以通过如下方式启动：

```
1 /绝对路径/启动脚本名 start
```

之所以RPM包还有诸如 service 服务名 start 等启动方式是由于其安装位置差异造成的。

- ①、源码包：

安装路径一般是 /usr/local。查看源码包的启动方式可以进入到源码包的安装目录，查看安装说明文档 INSTALL

```

[root@node3 local]# cd /usr/local/src/
[root@node3 src]# ls
httpd-2.2.34.tar.gz
[root@node3 src]# tar -zxf httpd-2.2.34.tar.gz
[root@node3 src]# ls
httpd-2.2.34 httpd-2.2.34.tar.gz 安装说明          使用说明
[root@node3 src]# cd httpd-2.2.34      安装说明
[root@node3 httpd-2.2.34]# ls
ABOUT_APACHE   CHANGES      httpd.dsp      libhttpd.dep  NOTICE
acinclude.m4    config.layout httpd.mak      libhttpd.dsp  NWGNUMakefile
Apache.dsw     configure    httpd.spec    libhttpd.mak  os
build        configure.in  include      LICENSE      README
BuildAll.dsp   docs        INSTALL     Makefile.in   README.platforms
BuildBin.dsp   emacs-style   InstallBin.dsp Makefile.win  README-win32.txt
buildconf    httpd.dep    LAYOUT       modules      ROADMAP
[root@node3 httpd-2.2.34]# 

```

这里我们可以查看INSTALL 文件，查看安装说明：

```

For complete installation documentation, see [ht]docs/manual/install.html or
http://httpd.apache.org/docs/2.2/install.html

$ ./configure --prefix=PREFIX
$ make
$ make install
$ PREFIX/bin/apachectl start

NOTES: * Replace PREFIX with the filesystem path under which
       Apache should be installed. A typical installation
       might use "/usr/local/apache2" for PREFIX (without the
       quotes).

* If you are a developer who will be linking your code with
       Apache or using a debugger to step through server code,
       ./configure's --with-included-apr option may be advantageous,
       as it removes the possibility of version or compile-option

```

也就是源码包 httpd 的启动方式是：/usr/local/src/bin/apachectl start

②、RPM包

默认安装位置如下：

- ◆ /etc/init.d/: 启动脚本位置
- ◆ /etc/sysconfig/: 初始化环境配置文件位置
- ◆ /etc/: 配置文件位置
- ◆ /etc/xinetd.conf: xinetd配置文件
- ◆ /etc/xinetd.d/: 基于xinetd服务的启动脚本
- ◆ /var/lib/: 服务产生的数据放在这里
- ◆ /var/log/: 日志

比如 RPM 包安装的httpd服务启动方式有两种：

```
1 ①、/etc/rc.d/init.d/httpd start` `②、service httpd start
```

第二种命令是第一种命令的简化形式。这是红帽专有命令。service 会去 rpm 包默认的安装位置去找可执行文件，所以service类似的启动服务管理命令只能管理rpm安装包。如果更改了rpm包的默认安装位置，可能造成通过service启动服务失败。

下面我们分别对启动和自启动RPM包以及源码包服务进行讲解。

4、独立服务

独立服务属于RPM包服务，启动方法如下：

◆ /etc/init.d/独立服务名 start|stop|status|restart|

◆ service 独立服务名 start|stop|restart||status

第一种方法是启动服务的万能方法，而通过service启动服务是红帽专有命令。start 表示启动服务，stop 表示停止服务，status表示查看指定服务是否启动，restart 是重启指定服务。

独立服务自启动：

◆ chkconfig [--level 运行级别] [独立服务名] [on|off]

◆ 修改/etc/rc.d/rc.local文件

◆ 使用ntsysv命令管理自启动

```
[root@node3 /]# chkconfig --list | grep httpd
httpd      0:关闭  1:关闭  2:启用  3:启用  4:启用  5:启用  6:关闭
[root@node3 /]# chkconfig --level 2345 httpd on
[root@node3 /]# chkconfig --level 2345 httpd off
[root@node3 /]# chkconfig --list | grep httpd
httpd      0:关闭  1:关闭  2:关闭  3:关闭  4:关闭  5:关闭  6:关闭
[root@node3 /]# chkconfig --level 2345 httpd on
[root@node3 /]# chkconfig --list | grep httpd
httpd      0:关闭  1:关闭  2:启用  3:启用  4:启用  5:启用  6:关闭
[root@node3 /]# chkconfig --level 2345 httpd off
[root@node3 /]# chkconfig --list | grep httpd
httpd      0:关闭  1:关闭  2:关闭  3:关闭  4:关闭  5:关闭  6:关闭
[root@node3 /]#
```

1、查看 httpd 服务是否自启动，一般2345启动就说明是自启动服务
2、chkconfig --level 2345 http on/off 分别开启/关闭该服务是否自启动

第二种方法：

```
[root@node3 /]# ll /etc/rc.local  
lrwxrwxrwx. 1 root root 13 6月 8 03:49 /etc/rc.local -> rc.d/rc.local
```

修改/etc/rc.local或者/etc/rc.d/rc.local 都可以

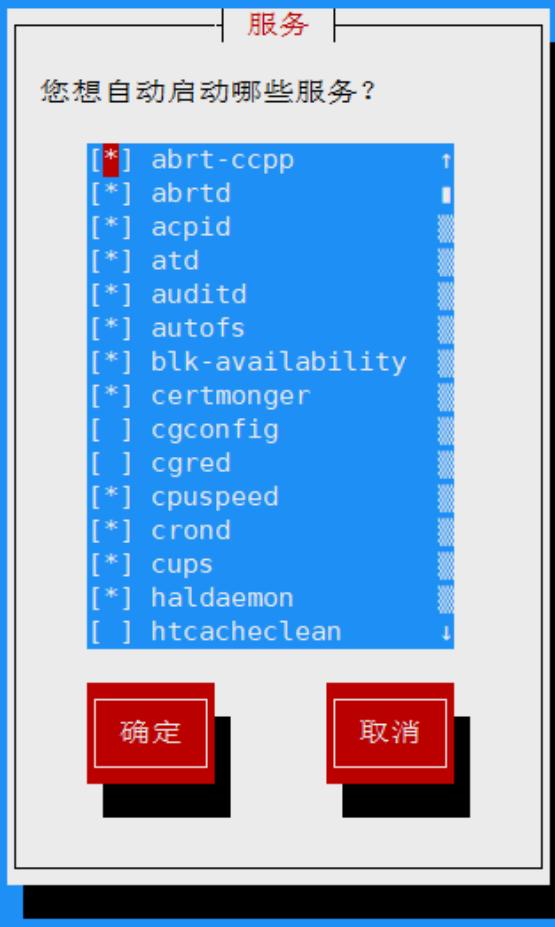
```
#!/bin/sh  
#  
# This script will be executed *after* all the other init scripts.  
# You can put your own initialization stuff in here if you don't  
# want to do the full Sys V style init stuff.  
  
touch /var/lock/subsys/local  
/etc/rc.d/init.d/httpd start  
~  
~  
~  
~
```

在 /etc/rc.d/rc.local 文件中添加上面的代码，系统每次重启时就会读取这个配置文件，然后启动配置好的服务

第三种方法是图形配置界面：

输入 ntsysv 命令，弹出如下界面进行配置就好了。

ntsysv 1.3.49.5 - (C) 2000-2001 Red Hat, Inc.



5、xinetd 服务

这个服务也是属于 RPM包，现在Linux系统中基于 xinetd 服务越来越少了，启动服务我们只需要修改 /etc/xinetd.d/服务名 这个文件的 disable = no 即可

```
[root@localhost ~]# vi /etc/xinetd.d/telnet
service telnet           ←服务的名称为telnet
{
    flags      = REUSE   ←标志为REUSE，设定TCP/IP socket可重用
    socket_type = stream ←使用TCP协议数据包
    wait       = no      ←允许多个连接同时连接
    user       = root    ←启动服务的用户为root
    server     = /usr/sbin/in.telnetd          ←服务的启动程序
    log_on_failure += USERID                  ←登陆失败后，记录用户的ID
    disable    = no          ←服务不启动
}
```

自启动也可以通过 chkconfig 服务名 on 来设置。还可以通过 ntsysv 配置。

6、源码包服务

启动：

- ◆ 使用绝对路径，调用启动脚本来启动。不同的源码包的启动脚本不同。可以查看源码包的安装说明，查看启动脚本的方法。

```
/usr/local/apache2/bin/apachectl start|stop
```

自启动：

```
[root@localhost ~]# vi /etc/rc.d/rc.local
```

加入

```
/usr/local/apache2/bin/apachectl start
```

让源码包服务能被服务管理命令识别：即能通过 service 来启动

- ◆ 让源码包的apache服务能被service命令管理启动

```
ln -s /usr/local/apache2/bin/apachectl /etc/init.d/apache
```

- ◆ 让源码包的apache服务能被chkconfig与ntsysv命令管理自启动

```
vi /etc/init.d/apache
```

```
# chkconfig: 35 86 76
```

#指定httpd脚本可以被chkconfig命令管理。格式是：

 chkconfig: 运行级别 启动顺序 关闭顺序

```
# description: source package apache
```

#说明，内容随意

```
[root@localhost ~]# chkconfig --add apache  
#把源码包apache加入chkconfig命令
```

Linux 的系统管理

1、进程管理

进程简单来说就是系统中正在执行的一个程序或命令，每个进程都是一个运行的实体，都有自己的地址空间，并占用一定的系统资源。

通过管理进程，我们做的主要工作是：

- ①、判断服务器健康状态。
- ②、查看系统中的所有进程。
- ③、杀死进程。

2、查看系统中的所有进程:ps aux

```
[root@localhost ~]# ps aux  
#查看系统中所有进程，使用BSD操作系统格式  
[root@localhost ~]# ps -le  
#查看系统中所有进程，使用Linux标准命令格式。
```

[root@node3 /]# ps aux										
USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.0	0.1	19364	1544	?	Ss	Nov13	0:01	/sbin/init
root	2	0.0	0.0	0	0	?	S	Nov13	0:00	[kthreadd]
root	3	0.0	0.0	0	0	?	S	Nov13	0:00	[migration/0]
root	4	0.0	0.0	0	0	?	S	Nov13	0:00	[ksoftirqd/0]
root	5	0.0	0.0	0	0	?	S	Nov13	0:00	[stopper/0]
root	6	0.0	0.0	0	0	?	S	Nov13	0:00	[watchdog/0]
root	7	0.0	0.0	0	0	?	S	Nov13	1:59	[events/0]
root	8	0.0	0.0	0	0	?	S	Nov13	0:00	[events/0]
root	9	0.0	0.0	0	0	?	S	Nov13	0:00	[events_long/0]
root	10	0.0	0.0	0	0	?	S	Nov13	0:00	[events_power_ef]
root	11	0.0	0.0	0	0	?	S	Nov13	0:00	[cgroup]
root	12	0.0	0.0	0	0	?	S	Nov13	0:00	[khelper]
root	13	0.0	0.0	0	0	?	S	Nov13	0:00	[netns]
root	14	0.0	0.0	0	0	?	S	Nov13	0:00	[async/mgr]
root	15	0.0	0.0	0	0	?	S	Nov13	0:00	[pm]
root	16	0.0	0.0	0	0	?	S	Nov13	0:00	[sync_supers]
root	17	0.0	0.0	0	0	?	S	Nov13	0:00	[bdi-default]
root	18	0.0	0.0	0	0	?	S	Nov13	0:00	[kintegrityd/0]

每一列解释如下：

- ◆ USER：该进程是由哪个用户产生的；
- ◆ PID：进程的ID号；
- ◆ %CPU：该进程占用CPU资源的百分比，占用越高，进程越耗费资源；
- ◆ %MEM：该进程占用物理内存的百分比，占用越高，进程越耗费资源；
- ◆ VSZ：该进程占用虚拟内存的大小，单位KB；
- ◆ RSS：该进程占用实际物理内存的大小，单位KB；
- ◆ TTY：该进程是在哪个终端中运行的。其中tty1-tty7代表本地控制台终端，tty1-tty6是本地的字符界面终端，tty7是图形终端。pts/0-255代表虚拟终端。

- ◆ STAT：进程状态。常见的状态有：R：运行、S：睡眠、T：停止状态、s：包含子进程、+：位于后台
- ◆ START：该进程的启动时间
- ◆ TIME：该进程占用CPU的运算时间，注意不是系统时间
- ◆ COMMAND：产生此进程的命令名

3、查看系统健康状态：top

```
[root@localhost ~]# top [选项]
```

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：

? 或h: 显示交互模式的帮助

P: 以CPU使用率排序， 默认就是此项

M: 以内存的使用率排序

N: 以PID排序

q: 退出top

```
[root@node3 /]# top
top - 08:19:27 up 1 day, 10:31, 2 users, load average: 0.00, 0.01, 0.05
Tasks: 97 total, 1 running, 96 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.0%us, 0.0%sy, 0.0%ni, 100.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 1004136k total, 334488k used, 669648k free, 57376k buffers
Swap: 2047996k total, 0k used, 2047996k free, 111092k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1	root	20	0	19364	1544	1228	S	0.0	0.2	0:01.89	init
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
3	root	RT	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
4	root	20	0	0	0	0	S	0.0	0.0	0:00.08	ksoftirqd/0
5	root	RT	0	0	0	0	S	0.0	0.0	0:00.00	stopper/0
6	root	RT	0	0	0	0	S	0.0	0.0	0:00.29	watchdog/0
7	root	20	0	0	0	0	S	0.0	0.0	1:59.66	events/0
8	root	20	0	0	0	0	S	0.0	0.0	0:00.00	events/0
9	root	20	0	0	0	0	S	0.0	0.0	0:00.00	events_long/0
10	root	20	0	0	0	0	S	0.0	0.0	0:00.00	events_power_ef
11	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cgroup
12	root	20	0	0	0	0	S	0.0	0.0	0:00.00	khelper
13	root	20	0	0	0	0	S	0.0	0.0	0:00.00	netns
14	root	20	0	0	0	0	S	0.0	0.0	0:00.00	async/mgr
15	root	20	0	0	0	0	S	0.0	0.0	0:00.00	pm

第一行信息为任务队列信息

内容	说明
12:26:46	系统当前时间
up 1 day, 13:32	系统的运行时间，本机已经运行1天13小时32分钟
2 users	当前登录了两个用户

第二行为进程信息

内容	说明
Tasks: 95 total	系统中的进程总数
1 running	正在运行的进程数
94 sleeping	睡眠的进程
0 stopped	正在停止的进程
0 zombie	僵尸进程。如果不是0，需要手工检查僵尸进程

第三行为CPU信息

内容	说明
Cpu(s): 0.1%us	用户模式占用的CPU百分比
0.1%sy	系统模式占用的CPU百分比
0.0%ni	改变过优先级的用户进程占用的CPU百分比
99.7%id	空闲CPU的CPU百分比
0.1%wa	等待输入/输出的进程的占用CPU百分比
0.0%hi	硬中断请求服务占用的CPU百分比
0.1%si	软中断请求服务占用的CPU百分比
0.0%st	st (Steal time) 虚拟时间百分比。就是当有虚拟机时，虚拟CPU等待实际CPU的时间百分比。

第四行为物理内存信息

内容	说明
Mem: 625344k total	物理内存的总量，单位KB
571504k used	已经使用的物理内存数量
53840k free	空闲的物理内存数量，我们使用的是虚拟机，总共只分配了628MB内存，所以只有53MB的空闲内存了
65800k buffers	作为缓冲的内存数量

第五行为交换分区（swap）信息

内容	说明
Swap: 524280k total	交换分区（虚拟内存）的总大小
0k used	已经使用的交互分区的大小
524280k free	空闲交换分区的大小
409280k cached	作为缓存的交互分区的大小

4、查看进程数：pstree

[root@localhost ~]# pstree [选项]

选项：

-p: 显示进程的PID

-u: 显示进程的所属用户

```
[root@node3 /]# pstree
init─ abrtd
      └─ acpid
        └─ atd
          └─ auditd─ {auditd}
            └─ automount─ 4*[{automount}]
              └─ certmonger
                └─ crond
                  └─ cupsd
                    └─ dbus-daemon─ {dbus-daemon}
                      └─ hald─ hald-runner─ hald-addon-acpi
                        └─ hald-addon-inpu
                          └─ hald-addon-rfki
                            └─ {hald}
              └─ master─ pickup
                └─ qmgr
              └─ mcelog
              └─ 6*[mingetty]
              └─ rpc.statd
              └─ rpcbind
              └─ rsyslogd─ 3*[{rsyslogd}]
              └─ sshd─ sshd─ bash─ pstree
                └─ sshd─ bash
              └─ udevd─ 2*[udevd]
```

5、终止进程:kill

①、查看可用的进程信号： kill -l

```
[root@node3 /]# kill -l
 1) SIGHUP      2) SIGINT      3) SIGQUIT      4) SIGILL      5) SIGTRAP
 6) SIGABRT     7) SIGBUS      8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
[root@node3 /]#
```

信号代号	信号名称	说明
1	SIGHUP	该信号让进程立即关闭，然后重新读取配置文件之后重启。
2	SIGINT	程序终止信号，用于终止前台进程。相当于输出 <code>ctrl+c</code> 快捷键。
8	SIGFPE	在发生致命的算术运算错误时发出。不仅包括浮点运算错误，还包括溢出及除数为0等其它所有的算术的错误。
9	SIGKILL	用来立即结束程序的运行。本信号不能被阻塞、处理和忽略。一般用于强制终止进程。
14	SIGALRM	时钟定时信号，计算的是实际的时间或时钟时间。 <code>alarm</code> 函数使用该信号。
15	SIGTERM	正常结束进程的信号， <code>kill</code> 命令的默认信号。有时如果进程已经发生问题，这个信号是无法正常终止进程的，我们才会尝试 <code>SIGKILL</code> 信号，也就是信号9。
18	SIGCONT	该信号可以让暂停的进程恢复执行，本信号不能被阻断。
19	SIGSTOP	该信号可以暂停前台进程，相当于输入 <code>ctrl+z</code> 快捷键。本信号不能被阻断。

②、根据进程 pid 杀死进程

`kill -15` 进程号 默认正常结束进程，其中选项 `-15` 可以省略。

```
[root@localhost ~]# kill -1 22354
#重启进程
```

```
[root@localhost ~]# kill -9 22368
#强制杀死进程
```

③、根据进程名 杀死进程

```
[root@localhost ~]# killall [选项][信号] 进程名
#按照进程名杀死进程
```

选项：

- i: 交互式，询问是否要杀死某个进程
- I: 忽略进程名的大小写

```
[root@localhost ~]# pkill [选项] [信号] 进程名  
#按照进程名终止进程
```

选项：

-t 终端号： 按照终端号踢出用户

按照终端号踢出用户

```
[root@localhost ~]# w  
#使用w命令查询本机已经登录的用户
```

```
[root@localhost ~]# pkill -t -9 pts/1  
#强制杀死从pts/1虚拟终端登录的进程
```

6、将进程放入后台运行

①、在命令后加 &，使得命令在后台运行

◆ tar -zcf etc.tar.gz /etc &

②、快捷键 Ctrl+z，使得命令在后台暂停

◆ [root@localhost ~]# top
#在top命令执行的过程中，按下ctrl+z快捷键

7、查看后台的工作

```
[root@localhost ~]# jobs [-1]
```

选项：

-1： 显示工作的PID

注：“+”号代表最近一个放入后台的工作，也是工作恢复时，默认恢复的工作。“-”号代表倒数第二个放入后台的工作

8、将后台暂停的工作号恢复到前台执行

```
[root@localhost ~]# fg %工作号
```

参数：

%工作号： %号可以省略，但是注意工作号和PID的区别

9、将后台暂停的工作号恢复到后台执行

```
[root@localhost ~]# bg %工作号
```

注：后台恢复执行的命令，是不能和前台有交互的，否则不能恢复到后台执行

10、监控系统资源：vmstat

```
[root@localhost ~]# vmstat [刷新延时 刷新次数]
```

例如：

```
[root@localhost proc]# vmstat 1 3
```

```
[root@node3 /]# vmstat 1 3 1秒钟刷新一次，一共刷新三次
procs -----memory-----swap-- io-----system--cpu-----
r b swpd free buff cache si so bi bo in cs us sy id wa st
1 0 0 658828 66656 111484 0 0 1 1 9 9 0 0 100 0 0
0 0 0 658812 66656 111484 0 0 0 0 14 14 0 0 100 0 0
0 0 0 658812 66656 111484 0 0 0 0 14 16 0 0 100 0 0
[root@node3 /]#
```

11、查看硬件信息 dmesg

```
[root@localhost ~]# dmesg
```

```
[root@localhost ~]# dmesg | grep CPU
```

```
[root@node3 /]# dmesg | grep CPU
SMP: Allowing 128 CPUs, 127 hotplug CPUs
NR_CPUS:4096 nr_cpumask_bits:128 nr_cpu_ids:128 nr_node_ids:1
PERCPU: Embedded 31 pages/cpu @fffff880003400000 s96600 r8192 d22184 u131072
CPU: Physical Processor ID: 0
mce: CPU supports 0 MCE banks
CPU0: Intel(R) Core(TM) i5-6300HQ CPU @ 2.30GHz stepping 03
CPUID marked event: 'cpu cycles' unavailable
CPUID marked event: 'instructions' unavailable
CPUID marked event: 'bus cycles' unavailable
CPUID marked event: 'cache references' unavailable
CPUID marked event: 'cache misses' unavailable
CPUID marked event: 'branch instructions' unavailable
CPUID marked event: 'branch misses' unavailable
Brought up 1 CPUs
microcode: CPU0 sig=0x506e3, pf=0x1, revision=0x23
[root@node3 /]#
```

还有个命令也可以查看 CPU 信息：

```
1 cat /proc/cpuinfo
```

12、查看系统与内核相关信息 uname

```
[root@localhost ~]# uname [选项]
```

选项：

-a: 查看系统所有相关信息；

-r: 查看内核版本；

-s: 查看内核名称。

13、查看当前系统位数 file

通过 file 命令查看任意一个外部命令的信息，比如 file /bin/ls

```
[root@node3 /]# file /bin/ls
/bin/ls: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked
r GNU/Linux 2.6.18, stripped
[root@node3 /]# file /bin/ln
/bin/ln: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked
r GNU/Linux 2.6.18, stripped
[root@node3 /]#
```

欢迎关注公众号



程序员 cxuan



Java 建设者