**Linux**

## 一、基础知识

### 1 Linux 简介

Linux内核最初只是由芬兰人李纳斯·托瓦兹（Linus Torvalds）在赫尔辛基大学上学时出于个人爱好而编写的。Linux是一套免费使用和自由传播的类Unix操作系统，**是一个基于POSIX和UNIX的多用户、多任务、支持多线程和多CPU的操作系统。**Linux能运行主要的UNIX工具软件、应用程序和网络协议。它支持32位和64位硬件。Linux继承了Unix以网络为核心的设计思想，是一个性能稳定的多用户网络操作系统。

Linux以它的高效性和灵活性著称。它能够在 PC计算机上实现全部的 Unix特性， 具有多任务、多用户的能力。 Linux是在 GNU公共许可权限下免费获得的，是一 个符合 POSIX标准的操作系统。 Linux操作系统软件包不仅包括完整的 Linux操 作系统，而且还包括了文本编辑器、高级语言编译器等应用软件。它还包括带有 多个窗口管理器的 X-Windows图形用户界面，如同我们使用 Windows NT一样，允 许我们使用窗口、图标和菜单对系统进行操作。 用C语言，汇编语言开发，支持shell脚本。

### 2 Linux的发行版

Linux的发行版说简单点就是将Linux内核与应用软件做一个打包。目前市面上较知名的发行版有：**Ubuntu、RedHat、CentOS**、Debian、Fedora、SuSE、OpenSUSE、Arch Linux、SolusOS 等。

### 3 Linux应用领域

今天各种场合都有使用各种Linux发行版，从嵌入式设备到超级计算机，并且在服务器领域确定了地位，通常服务器使用LAMP（Linux + Apache + MySQL + PHP）或LNMP（Linux + Nginx+ MySQL + PHP）组合。目前Linux不仅在家庭与企业中使用，并且在政府中也很受欢迎。

（1）巴西联邦政府由于支持Linux而世界闻名。

（2）有新闻报道俄罗斯军队自己制造的Linux发布版的，做为G.H.ost项目已经取得成果.

（3）印度的Kerala联邦计划在向全联邦的高中推广使用Linux。

（4）中华人民共和国为取得技术独立，在龙芯过程中排他性地使用Linux。

（5）在西班牙的一些地区开发了自己的Linux发布版，并且在政府与教育领域广泛使用，如Extremadura地区的gnuLinEx和Andalusia地区的Guadalinex。

（6）葡萄牙同样使用自己的Linux发布版Caixa Mágica，用于Magalh?es笔记本电脑和e-escola政府软件。

（7）法国和德国同样开始逐步采用Linux。

### 4 Linux vs Windows

|  |  |  |
| --- | --- | --- |
| **比较** | **Windows** | **Linux** |
| 界面 | 界面统一，外壳程序固定所有Windows程序菜单几乎一致，快捷键也几乎相同 | 图形界面风格依发布版不同而不同，可能互不兼容。GNU/Linux的终端机是从UNIX传承下来，基本命令和操作方法也几乎一致。 |
| 驱动程序 | 驱动程序丰富，版本更新频繁。默认安装程序里面一般包含有该版本发布时流行的硬件驱动程序，之后所出的新硬件驱动依赖于硬件厂商提供。对于一些老硬件，如果没有了原配的驱动有时很难支持。另外，有时硬件厂商未提供所需版本的Windows下的驱动，也会比较头痛。 | 由志愿者开发，由Linux核心开发小组发布，很多硬件厂商基于版权考虑并未提供驱动程序，尽管多数无需手动安装，但是涉及安装则相对复杂，使得新用户面对驱动程序问题（是否存在和安装方法）会一筹莫展。但是在开源开发模式下，许多老硬件尽管在Windows下很难支持的也容易找到驱动。HP、Intel、AMD等硬件厂商逐步不同程度支持开源驱动，问题正在得到缓解。 |
| 使用 | 使用比较简单，容易入门。图形化界面对没有计算机背景知识的用户使用十分有利。 | 图形界面使用简单，容易入门。文字界面，需要学习才能掌握。 |
| 学习 | 系统构造复杂、变化频繁，且知识、技能淘汰快，深入学习困难。 | 系统构造简单、稳定，且知识、技能传承性好，深入学习相对容易。 |
| 软件 | 每一种特定功能可能都需要商业软件的支持，需要购买相应的授权。 | 大部分软件都可以自由获取，同样功能的软件选择较少。 |

### 5 linux安装见



### 6 linux特点

**1）. 技术成熟，可靠性高**

经过30来年开放式道路的发展，UNIX的一些基本技术已变得十分成熟，有的已成为各类操作系统的常用技术。实践表明，UNIX是能达到大型主机（mainframe）可靠性要求的少数操作系统之一。目前许多UNIX大型主机和服务器在国外的大型企业中每天24小时，每年365天不间断地运行。例如，不少大企业或政府部门，即所谓肩负关键使命的场合/部门将其整个企业/部门信息系统建立并运行在以UNIX为主服务器的Client/Server结构上。但到目前为止，世界上还没有一家大型企业将其重要的信息系统完全建立在NT上。

**2）. 极强的可伸缩性**

UNIX系统是世界上唯一能在笔记本电脑、PC、工作站，直至巨型机上运行的操作系统，而且能在所有主要CPU芯片搭建的体系结构上运行（包括Intel/AMD及HP-PA、MIPS、PowerPC、UltraSPARC、ALPHA等RISC芯片）。至今为止，世界上没有第二个操作系统能达到这一点。此外，由于UNIX系统能很好地支持SMP、MPP和Cluster等技术，使其可伸缩性又有了很大的增强。目前，商品化UNIX系统能支持的SMP，CPU数已达到几百甚至更多个，MPP系统中的节点甚至已超过1024个UNIX支持的异种平台Cluster技术也已投入使用。UNIX的伸缩性远远超过了NT操作系统目前所能达到的水平

**3）. 网络功能强**

网络功能是UNIX系统的一又一重要特色，作为Internet网技术和异种机连接重要手段的TCP/IP协议就是在UNIX上开发和发展起来的。TCP/IP是所有UNIX系统不可分割的组成部分。因此，UNIX服务器在Internet服务器中占80％以上，占绝对优势。此外，UNIX还支持所有常用的网络通信协议，包括NFS、DCE、IPX/SPX、SLIP、PPP等，使得UNIX系统能方便地与已有的主机系统，以及各种广域网和局域网相连接，这也是UNIX具有出色的互操作性（Interoperability）的根本原因。

**4）. 强大的数据库支持能力**

由于UNIX具有强大的支持数据库的能力和良好的开发环境，因此多年来，所有主要数据库厂商，包括Oracle、Informix、Sybase、Progress等，都把UNIX作为主要的数据库开发和运行平台，并创造出一个又一个性价比的新记录。UNIX服务器正在成为大型企业数据中心替代大型主机的主要平台。

**5）. 开发功能强**

UNIX系统从一开始就为软件开发人员提供了丰富的开发工具。成为工程工作站的首选和主要的操作系统和开发环境。可以说，工程工作站的出现和成长与UNIX是分不开的。至今为止，UNIX工作站仍是软件开发厂商和工程研究设计部门的主要工作平台。有重大意义的软件新技术的出现几乎都在UNIX上，如TCP/IP、WWW、OODBMS等。

**6）. 开放性好**

开放性是UNIX最重要的本质特性。开放系统概念的形成与UNIX是密不可分的。UNIX是开放系统的先驱和代表。由于开放系统深入人心，几乎所厂商都宣称自己的产品是开放系统，确实每一种系统都能满足某种开放的特性，如可移植性、可兼容性、可伸缩性、互操作性等。但所有这些系统与开放系统的本质特征—不受某些厂商的垄断和控制相去甚远，只有UNIX完全符合这一条件。

**7）. 标准化**

过去，Unix界被分析家和用户批判，因为没有为所有Unix操作系统提供统一的标准。其实，到目前为止，国际标准化组织（ISO）、工业团体恰恰是以UNIX基础制订了一系列标准化，如ISO/IEC的POSIX标准、IEEE POSIX标准、X/Open组织的XPG3/4工业标准以及后来的Spec 1170(因为它包含了1170个应用编程接口，后来改名为UNIX’95)标准。不少人对标准及标准化组织的作用及职权产生了误解。事实上，当标准化组织企图驾驭互相竞争的力量，和企图为用户规定他们的要求时是注定要失败的。比方说，标准只能用于给出道路的规则，而不应用于制造汽车。如果厂家被强迫完全遵从单一的标准，而不允许他们产品有特色，则用户将受害，Unix将变成象任何单一厂家的产品一样，没有任何特色。 Unix标准组织的真实目标是为用户和厂家定义一种规定Unix形态的基础。

### 7 著名厂商

**(1)红帽系列**

**Red Hat linux:** 大名鼎鼎的红帽，现在已经完结。以此为基础有以下分支:

**Red Hat Enterprise:** (www.redhat.com) 这个是企业级的linux，主要面向服务器。作为商业版，有比较好的配套软件和技术支持。RH的教材也堪称经典。

**Fedora:** (http://fedoraproject.org) 由社区维护，去除了一些商业软件。红帽实际上赞助了这个项目，以便以此作为技术测试平台。

**CentOS:** (www.centos.org) 这个版本不来自红帽的公司，是收集红帽公开的源码组成的免费版本，由社区维护，和红帽完全兼容。版本号升级较慢，但有持续的技术支持，所以适合于不愿意频繁升级的情形，鸟哥中就推荐以CentOS来建站。

**(2)SUSE系列**

**SUSE Linux Enterprise: (www.suse.com)** 和红帽商业版类似，是德国公司制作的，据说在欧洲比较流行，但我欧洲的朋友表示没怎么听说过。SUSE系列比较有特色的是YAST2，就是用来设置Linux的界面，对初级管理员比较方便。

**openSUSE:** (www.opensuse.org) SUSE的免费版本。以前SUSE不是很推这个免费版本，支持不好。现在似乎态度大大转变。就我个人的使用体验来说，还是觉得社区支持不足。

**(3)Debian系列**

**Debian:** (www.debian.org) 完全免费，社区维护的Linux版本，有很大的用户群，所以遇到问题，基本都可以找到社区用户的支持。

**Ubuntu:** (www.ubuntu.com) 由一个基金提供支持的免费Linux版本。 继承自Debian。界面很友好。发邮件过去，还会送你免费安装CD (不知道现在还有没有)。现在的版本加载的东西有些多，速度有些慢。写《大教堂和市集》的Eric Raymond说它是最好的Linux。

**Mint (http://www.linuxmint.com),** 基于ubuntu。它提供了更加丰富的预装应用，以减少用户搜索并安装应用的麻烦。其使用的应用版本比较新，可能不是很稳定。

### 8 常用工具

**（1）Xmanager**

Xmanager是一款小巧、便捷的浏览远端X窗口系统的工具。在工作中经常使用Xmanager来登录远端的Solaris系统，在X窗口系统上作图形化的操作。但是，Xmanager默认并不提供对于中文的支持，为了能够浏览远端的中文Solaris系统，就必须对Xmanager作一些定制化操作，提供中文的浏览能力。

@ Xmanager PCX Server是一个运行于MS Windows平台上的高性能的X window服务器。可以在本地PC上同时运行Unix/Linux和Windows图形应用程序。

@ Xshell是一个用于MS Windows平台的强大的SSH，TELNET，和RLOGIN终端仿真软件。它使得用户能轻松和安全地从Windows PC上访问Unix/Linux主机。

@ Xftp是一个用于MS Windows平台的强大的FTP和SFTP文件传输程序。Xftp能安全地在Unix/Linux和Windows PC之间传输文件。

@ Xlpd是一个用于MS Windows平台的LPD(行式打印机虚拟后台程序)应用程序。安装了Xlpd后，带有打印机的本地PC就成为了一个打印服务器，来自不同远程系统的打印任务都能在网络环境中得到请求和处理。(共享软件)

@ Xstart是一个窗口化的登陆界面，要求填入session，host，protocol，user name， password。可进行远程登陆。

**（3）Xshell**

Xshell是一个强大的安全终端模拟软件，它支持SSH1, SSH2, 以及Microsoft Windows 平台的TELNET 协议。Xshell 通过互联网到远程主机的安全连接以及它创新性的设计和特色帮助用户在复杂的网络环境中享受他们的工作。Xshell可以在Windows界面下用来访问远端不同系统下的服务器，从而比较好的达到远程控制终端的目的.

TELNET协议：未加密的Internet远程登陆服务的标准协议和主要方式。ftp、pop和telnet在本质上都是不安全的，因为它们在网络上用明文传送口令和数据，别有用心的人非常容易就可以截获这些口令和数据。

SSH1协议：SSH命令是加密的并以几种方式进行保密。SSH1又分为1.3和1.5两个版本。SSH1采用DES、3DES、Blowfish和RC4等对称加密算法保护数据安全传输，而对称加密算法的密钥是通过非对称加密算法（RSA）来完成交换的。SSH1使用循环冗余校验码（CRC）来保证数据的完整性，但是后来发现这种方法有缺陷。

SSH2协议：SSH2避免了RSA的专利问题，并修补了CRC的缺陷。SSH2用数字签名算法（DSA）和Diffie-Hellman（DH）算法代替RSA来完成对称密钥的交换，用消息证实代码（HMAC）来代替CRC。同时SSH2增加了AES和Twofish等对称加密算法。

**（4）secureCRT**

Xshell 和SecureCRT的比较：

Screen不会闪屏，并且可以回滚，这个功能比较好用。Script的执行顺序可以调整，SecureCRT的这个功能相较而言比较死板，不过可以同时发送指令到多个session，这个功能较好。键盘映射的兼容性比较好，不需要自己去改映射，可以展现tunnel等。支持布局切换，像gnome-terminal。缺憾是对Unicode制表符支持不够好，内置的sftp不怎么符合用户的操作习惯（该公司有另外的xftp）。许可相对便宜，对个人、教育用户是免费的。

名称：192.168.0.241\_uoss

协议：SSH

主机：192.168.0.241

端口号：22

方法：Password

用户名：hiya

密码：235425

### 9 vmware

VMware（中文名威睿”，纽约证券交易所“代码：VMW） 虚拟机软件，是全球桌面到数据中心虚拟化解决方案的领导厂商。全球不同规模的客户依靠VMware来降低成本和运营费用、确保业务持续性、加强安全性并走向绿色。2008年，VMware年收入达到19亿美元，拥有逾150,000的用户和接近22,000多家合作伙伴，是增长最快的上市软件公司之一。VMware总部设在加利福尼亚州的帕罗奥多市（Palo Alto）。

VMware（纽约证交所代码：VMW）在虚拟化和云计算基础架构领域处于全球领先地位，所提供的经客户验证的解决方案可通过降低复杂性以及更灵活、敏捷地交付服务来提高IT效率。VMware使企业可以采用能够解决其独有业务难题的云计算模式。VMware提供的方法可在保留现有投资并提高安全性和控制力的同时，加快向云计算的过度。 VMware拥有 400,000多家客户和55,000多家合作伙伴，它的解决方案可帮助各种规模的组织降低成本、提高业务灵活性并确保选择自由。

### 10 linux优点

1）、稳定的系统：打个比方，安装Linux的主机连续运行一年以上不曾宕机、不必关机是很平常的事，我的windows系统今早打开时是黑屏，强行关机之后才恢复，原因不明

2）、免费或少许费用：这个没必要强调了，只要你在中国，使用免费，需要技术支持费用。并且开源。

3）、安全性、漏洞的修补：Linux有众多电脑高手在使用，所以维护者众多，更新维护很快，而windows则是所有人都会用，且不开源

4）、多任务、多用户：你可以在一个Linux主机上规划出不同等级的用户，而且每个用户登录系统时工作环境可以不同，此外你还可以允许不同用户在同一时间登陆主机以使用主机的资源

5）、用户与用户组的规划：在linux的机器上，文件可分为可读可写可执行三类（windows上也可以设置这些属性，在命令行里，会在以后的windows博客里写出来）此外，这些属性还可以分为三类：分别是文件拥有者、文件所属用户组、其他非拥有着与用户组者，这对于项目或者其他项目开发者具有良好的保密性

6）、不耗资源：这是最吸引眼球的地方，目前市面上任何一款个人计算机都可以达到使用Linux搭建一个服务上百人以上的主机

7）、小，这个也是很棒的，你有见过你家的家电是windows系统的吗？现在win8所占内存甚大，而Linux只要几百kb的代码就可以驱动整个计算机的硬件

### 11 linux缺点

**1） 没有特定的支援厂商**

因为在Linux上面的所有套件几乎都是自由软体，而每个自由软体的开发者可能并不是公司团体，而是非营利性质的团体。如此一来，在你Linux主机上面的软体若发生问题，该如何是好？好在由于目前Linux商业界的整合还不错，目前在台湾比较具名的Red Hat与SuSE均有设立了服务点。你可以经由这个服务点来直接向他们购买/咨询相关的软硬体问题呢！不过，如果你并非选择有专门商业公司的Linux distributions时？怎么办？没有专人到府服务呢～这点倒是还不需要太担心，因为拜网路风行之赐，你要问的问题几乎在网路上都可以找到答案喔！

**2） 游戏的支援度不足**

在现代这个时候，敢说你们家的桌上型电脑里面完全没有游戏的小朋友应该不多了！游戏软体也是个应用程式，所以它与作业系统的关系就相当密切了。可惜的是目前很多游戏开发商并没有在Linux平台上面开发大型游戏，这间接导致Linux无法进入一般家庭说。

**3） 专业软体的支援度不足**

目前很多专业绘图软体公司所推出的专业软体并不支援Linux作业系统，这让同学很难在不同的平台上面操作相同的软体！唉！很伤脑筋～

**4） 教育训练作的还不够好**

目前政策方面还是相当的摇摆不定，希望未来能够给自由软体一些机会。老实说，这些缺点绝大部分都不是Linux本身的问题，倒是一些政策面与商业方面的考量，才是最大的困扰.不过，Linux与其他的作业系统一样，就是一个工具而已！希望大家能够在快乐中学习到Linux的精髓。

### 12 windows和linux比较

**1）. 可完全免费得到**

Linux操作系统可以从互联网上免费下载使用，只要您有快速的网络连接就行；而且，Linux上跑的绝大多数应用程序也是免费可得的。用了Linux就再也不用背”使用盗版软件”的黑锅了。

**2）. 可以运行在386以上及各种RISC体系结构机器上**

Linux最早诞生于微机环境，一系列版本都充分利用了X86CPU的任务切换能力，使X86CPU的效能发挥得淋淋尽致，而这一点连Windows都没有做到。此外，它可以很好地运行在由各种主流RISC芯片

（ALPHA、MIPS、PowerPC、UltraSPARC、HP-PA等）搭建的机器上。

**3）. Linux是UNIX的完整实现**

从发展的背景看，Linux与其他操作系统的区别是，Linux是从一个比较成熟的操作系统发展而来的，而其他操作系统，如WindowsNT等，都是自成体系，无对应的相依托的操作系统。这一区别使得Linux的用户能大大地从Unix团体贡献中获利。无论是Unix的作者还是Unix的用户，都认为只有Unix才是一个真正的操作系统，许多计算机系统（从个人计算机到超级计算机）都存在Unix版本，Unix的用户可以从很多方面得到支持和帮助。因此，Linux作为Unix的一个克隆，同样会得到相应的支持和帮助，直接拥有Unix在用户中建立的牢固的地位。 UNIX上的绝大多数命令都可以在Linux里找到并有所加强。UNIX的可靠性、稳定性以及强大的网络功能也在Linux身上一一体现。

**4）. 真正的多任务多用户**

只有很少的操作系统能提供真正的多任务能力，尽管许多操作系统声明支持多任务，但并不完全准确，如Windows。而Linux则充分利用了X86CPU的任务切换机制，实现了真正多

任务、多用户环境，允许多个用户同时执行不同的程序，并且可以给紧急任务以较高的优先级。

**5）. 完全符合POSIX标准**

POSIX是基于UNIX的第一个操作系统簇国际标准，Linux遵循这一标准这使UNIX下许多应用程序可以很容易地移植到Linux下，相反也是这样。

**6）. 具有图形用户界面**

Linux的图形用户界面是Xwindow系统。Xwindow可以做MSWindows下的所有事情，而且更有趣、更丰富，用户甚至可以在几种不同风格的窗口之间来回切换。

**7）. 具有强大的网络功能**

实际上，Linux就是依靠互联网才迅速发展了起来，Linux具有强大的网络功能也是自然而然的事情。它可以轻松地与TCP/IP、LANManager、Windows for Workgroups、Novell Netware或Windows NT网络集成在一起，还可以通过以太网或调制解调器连接到Internet上。 Linux不仅能够作为网络工作站使用，更可以胜任各类服务器，如X应用服务器、文件服务器、打印服务器、邮件服务器、新闻服务器等等。

**8）. 是完整的UNIX开发平台**

Linux支持一系列的UNIX开发工上，几乎所有的主流程序设计语言都已移植到Linux上并可免费得到.

## 二、Linux 系统启动过程

### 1 Linux系统的启动过程并不是大家想象中的那么复杂，其过程可以分为5个阶段：

@内核的引导。

@运行 init。

@系统初始化。

@建立终端 。

@用户登录系统。

### 2 init程序的类型：

@SysV: init, CentOS 5之前, 配置文件： /etc/inittab。

@Upstart: init,CentOS 6, 配置文件： /etc/inittab, /etc/init/\*.conf。

@Systemd：systemd, CentOS 7,配置文件： /usr/lib/systemd/system、 /etc/systemd/system。

### 3 内核引导

当计算机打开电源后，首先是BIOS开机自检，按照BIOS中设置的启动设备（通常是硬盘）来启动。操作系统接管硬件以后，首先读入 /boot 目录下的内核文件。



### 4 运行init

init 进程是系统所有进程的起点，你可以把它比拟成系统所有进程的老祖宗，没有这个进程，系统中任何进程都不会启动。init 程序首先是需要读取配置文件 /etc/inittab。



### 5 运行级别

**（1）概述**

许多程序需要开机启动。它们在Windows叫做"服务"（service），在Linux就叫做"守护进程"（daemon）。init进程的一大任务，就是去运行这些开机启动的程序。但是，不同的场合需要启动不同的程序，比如用作服务器时，需要启动Apache，用作桌面就不需要。Linux允许为不同的场合，分配不同的开机启动程序，这就叫做"运行级别"（runlevel）。也就是说，启动时根据"运行级别"，确定要运行哪些程序。



**（2）Linux系统有7个运行级别(runlevel)**

运行级别0：系统停机状态，系统默认运行级别不能设为0，否则不能正常启动

运行级别1：单用户工作状态，root权限，用于系统维护，禁止远程登陆

运行级别2：多用户状态(没有NFS)

运行级别3：完全的多用户状态(有NFS)，登陆后进入控制台命令行模式

运行级别4：系统未使用，保留

运行级别5：X11控制台，登陆后进入图形GUI模式

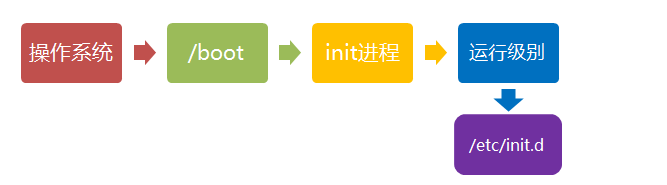
运行级别6：系统正常关闭并重启，默认运行级别不能设为6，否则不能正常启动

### 6 系统初始化

在init的配置文件中有这么一行： si::sysinit:/etc/rc.d/rc.sysinit　它调用执行了/etc/rc.d/rc.sysinit，而rc.sysinit是一个bash shell的脚本，它主要是完成一些系统初始化的工作，rc.sysinit是每一个运行级别都要首先运行的重要脚本。它主要完成的工作有：激活交换分区，检查磁盘，加载硬件模块以及其它一些需要优先执行任务。

**l5:5:wait:/etc/rc.d/rc 5**

这一行表示以5为参数运行/etc/rc.d/rc，/etc/rc.d/rc是一个Shell脚本，它接受5作为参数，去执行/etc/rc.d/rc5.d/目录下的所有的rc启动脚本，/etc/rc.d/rc5.d/目录中的这些启动脚本实际上都是一些连接文件，而不是真正的rc启动脚本，真正的rc启动脚本实际上都是放在/etc/rc.d/init.d/目录下。这些rc启动脚本有着类似的用法，它们一般能接受start、stop、restart、status等参数。



/etc/rc.d/rc5.d/中的rc启动脚本通常是K或S开头的连接文件，对于以 S 开头的启动脚本，将以start参数来运行。而如果发现存在相应的脚本也存在K打头的连接，而且已经处于运行态了(以/var/lock/subsys/下的文件作为标志)，则将首先以stop为参数停止这些已经启动了的守护进程，然后再重新运行。这样做是为了保证是当init改变运行级别时，所有相关的守护进程都将重启。至于在每个运行级中将运行哪些守护进程，用户可以通过chkconfig或setup中的"System Services"来自行设定。

### 7 建立终端

rc执行完毕后，返回init。这时基本系统环境已经设置好了，各种守护进程也已经启动了。

init接下来会打开6个终端，以便用户登录系统。在inittab中的以下6行就是定义了6个终端：

1:2345:respawn:/sbin/mingetty tty1

2:2345:respawn:/sbin/mingetty tty2

3:2345:respawn:/sbin/mingetty tty3

4:2345:respawn:/sbin/mingetty tty4

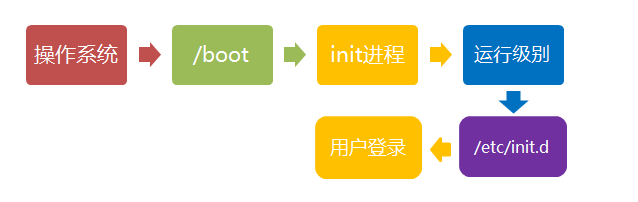
5:2345:respawn:/sbin/mingetty tty5

6:2345:respawn:/sbin/mingetty tty6

从上面可以看出在2、3、4、5的运行级别中都将以respawn方式运行mingetty程序，mingetty程序能打开终端、设置模式。

同时它会显示一个文本登录界面，这个界面就是我们经常看到的登录界面，在这个登录界面中会提示用户输入用户名，而用户输入的用户将作为参数传给login程序来验证用户的身份。

### 8 用户登录系统



一般来说，用户的登录方式有三种：

**（1）命令行登录**

**（2）ssh登录**

**（3）图形界面登录**

对于运行级别为5的图形方式用户来说，他们的登录是通过一个图形化的登录界面。登录成功后可以直接进入 KDE、Gnome 等窗口管理器。

而本文主要讲的还是文本方式登录的情况：当我们看到mingetty的登录界面时，我们就可以输入用户名和密码来登录系统了。

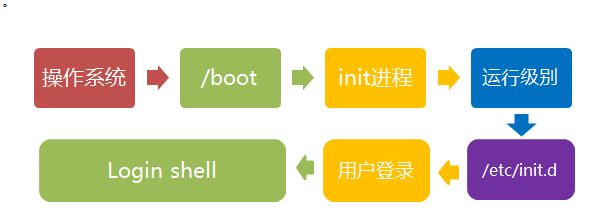
Linux 的账号验证程序是 login，login 会接收 mingetty 传来的用户名作为用户名参数。

然后 login 会对用户名进行分析：如果用户名不是 root，且存在 /etc/nologin 文件，login 将输出 nologin 文件的内容，然后退出。

这通常用来系统维护时防止非root用户登录。只有/etc/securetty中登记了的终端才允许 root 用户登录，如果不存在这个文件，则 root 用户可以在任何终端上登录。

/etc/usertty文件用于对用户作出附加访问限制，如果不存在这个文件，则没有其他限制。

### 9 图形模式与文字模式的切换方式



Linux预设提供了六个命令窗口终端机让我们来登录。

默认我们登录的就是第一个窗口，也就是tty1，这个六个窗口分别为tty1,tty2 … tty6，你可以按下Ctrl + Alt + F1 ~ F6 来切换它们。

如果你安装了图形界面，默认情况下是进入图形界面的，此时你就可以按Ctrl + Alt + F1 ~ F6来进入其中一个命令窗口界面。

当你进入命令窗口界面后再返回图形界面只要按下Ctrl + Alt + F7 就回来了。

如果你用的vmware 虚拟机，命令窗口切换的快捷键为 Alt + Space + F1~F6. 如果你在图形界面下请按Alt + Shift + Ctrl + F1~F6 切换至命令窗口。

### 10 Linux 关机

在linux领域内大多用在服务器上，很少遇到关机的操作。毕竟服务器上跑一个服务是永无止境的，除非特殊情况下，不得已才会关机。

正确的关机流程为：sync > shutdown > reboot > halt

关机指令为：shutdown ，你可以man shutdown 来看一下帮助文档。

例如你可以运行如下命令关机：

sync 将数据由内存同步到硬盘中。

shutdown 关机指令，你可以man shutdown 来看一下帮助文档。例如你可以运行如下命令关机：

shutdown –h 10 ‘This server will shutdown after 10 mins’ 这个命令告诉大家，计算机将在10分钟后关机，并且会显示在登陆用户的当前屏幕中。

Shutdown –h now 立马关机

Shutdown –h 20:25 系统会在今天20:25关机

Shutdown –h +10 十分钟后关机

Shutdown –r now 系统立马重启

Shutdown –r +10 系统十分钟后重启

reboot 就是重启，等同于 shutdown –r now

halt 关闭系统，等同于shutdown –h now 和 poweroff

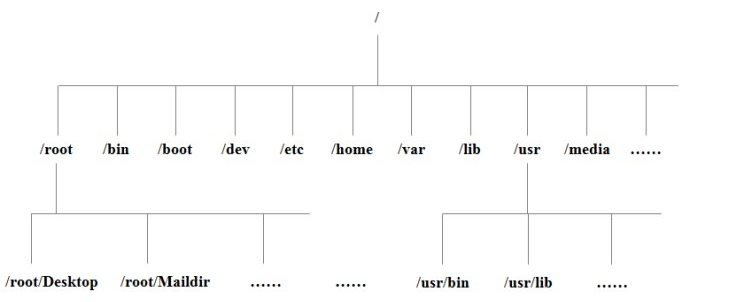
最后总结一下，不管是重启系统还是关闭系统，首先要运行 sync 命令，把内存中的数据写到磁盘中。关机的命令有 shutdown –h now halt poweroff 和 init 0 , 重启系统的命令有 shutdown –r now reboot init 6。

## Linux 系统目录结构

### 1 ls目录清单



### 2 目录树形结构图



**/bin：**

bin是Binary的缩写, 这个目录存放着最经常使用的命令。

**/boot：**

这里存放的是启动Linux时使用的一些核心文件，包括一些连接文件以及镜像文件。

**/dev ：**

dev是Device(设备)的缩写, 该目录下存放的是Linux的外部设备，在Linux中访问设备的方式和访问文件的方式是相同的。

**/etc：**

这个目录用来存放所有的系统管理所需要的配置文件和子目录。

**/home：**

用户的主目录，在Linux中，每个用户都有一个自己的目录，一般该目录名是以用户的账号命名的。

**/lib：**

这个目录里存放着系统最基本的动态连接共享库，其作用类似于Windows里的DLL文件。几乎所有的应用程序都需要用到这些共享库。

**/lost+found：**

这个目录一般情况下是空的，当系统非法关机后，这里就存放了一些文件。

**/media：**

linux系统会自动识别一些设备，例如U盘、光驱等等，当识别后，linux会把识别的设备挂载到这个目录下。

**/mnt：**

系统提供该目录是为了让用户临时挂载别的文件系统的，我们可以将光驱挂载在/mnt/上，然后进入该目录就可以查看光驱里的内容了。

**/opt：**

这是给主机额外安装软件所摆放的目录。比如你安装一个ORACLE数据库则就可以放到这个目录下。默认是空的。

**/proc：**

这个目录是一个虚拟的目录，它是系统内存的映射，我们可以通过直接访问这个目录来获取系统信息。这个目录的内容不在硬盘上而是在内存里，我们也可以直接修改里面的某些文件，比如可以通过下面的命令来屏蔽主机的ping命令，使别人无法ping你的机器：

**echo 1 > /proc/sys/net/ipv4/icmp\_echo\_ignore\_all**

**/root：**

该目录为系统管理员，也称作超级权限者的用户主目录。

**/sbin：**

s就是Super User的意思，这里存放的是系统管理员使用的系统管理程序。

**/selinux：**

这个目录是Redhat/CentOS所特有的目录，Selinux是一个安全机制，类似于windows的防火墙，但是这套机制比较复杂，这个目录就是存放selinux相关的文件的。

**/srv：**

该目录存放一些服务启动之后需要提取的数据。

**/sys：**

这是linux2.6内核的一个很大的变化。该目录下安装了2.6内核中新出现的一个文件系统 sysfs 。sysfs文件系统集成了下面3种文件系统的信息：针对进程信息的proc文件系统、针对设备的devfs文件系统以及针对伪终端的devpts文件系统。该文件系统是内核设备树的一个直观反映。当一个内核对象被创建的时候，对应的文件和目录也在内核对象子系统中被创建。

**/tmp：**

这个目录是用来存放一些临时文件的。

**/usr：**

这是一个非常重要的目录，用户的很多应用程序和文件都放在这个目录下，类似于windows下的program files目录。

**/usr/bin：**

系统用户使用的应用程序。

**/usr/sbin：**

超级用户使用的比较高级的管理程序和系统守护程序。

**/usr/src：**内核源代码默认的放置目录。

**/var：**

这个目录中存放着在不断扩充着的东西，我们习惯将那些经常被修改的目录放在这个目录下。包括各种日志文件。在linux系统中，有几个目录是比较重要的，平时需要注意不要误删除或者随意更改内部文件。

**/etc：** 上边也提到了，这个是系统中的配置文件，如果你更改了该目录下的某个文件可能会导致系统不能启动。

**/bin, /sbin, /usr/bin, /usr/sbin:** 这是系统预设的执行文件的放置目录，比如 ls 就是在/bin/ls 目录下的。值得提出的是，/bin, /usr/bin 是给系统用户使用的指令（除root外的通用户），而/sbin, /usr/sbin 则是给root使用的指令。

**/var：** 这是一个非常重要的目录，系统上跑了很多程序，那么每个程序都会有相应的日志产生，而这些日志就被记录到这个目录下，具体在/var/log 目录下，另外mail的预设放置也是在这里。

### 3 注意事项

**（1）在linux系统中，有几个目录是比较重要的，平时需要注意不要误删除或者随意更改内部文件。**

/etc： 上边也提到了，这个是系统中的配置文件，如果你更改了该目录下的某个文件可能会导致系统不能启动。

/bin, /sbin, /usr/bin, /usr/sbin: 这是系统预设的执行文件的放置目录，比如 ls 就是在/bin/ls 目录下的。

值得提出的是，/bin, /usr/bin 是给系统用户使用的指令（除root外的通用户），而/sbin, /usr/sbin 则是给root使用的指令。

/var： 这是一个非常重要的目录，系统上跑了很多程序，那么每个程序都会有相应的日志产生，而这些日志就被记录到这个目录下，具体在/var/log 目录下，另外mail的预设放置也是在这里。

**（2）在 Linux 或 Unix 操作系统中，所有的文件和目录都被组织成以一个根节点开始的倒置的树状结构。**

文件系统的最顶层是由根目录开始的，系统使用 / 来表示根目录。在根目录之下的既可以是目录，也可以是文件，而每一个目录中又可以包含子目录文件。如此反复就可以构成一个庞大的文件系统。在Linux文件系统中有两个特殊的目录，一个用户所在的工作目录，也叫当前目录，可以使用一个点 . 来表示；另一个是当前目录的上一级目录，也叫父目录，可以使用两个点 .. 来表示。

. ：代表当前的目录，也可以使用 ./ 来表示；

.. ：代表上一层目录，也可以 ../ 来代表。

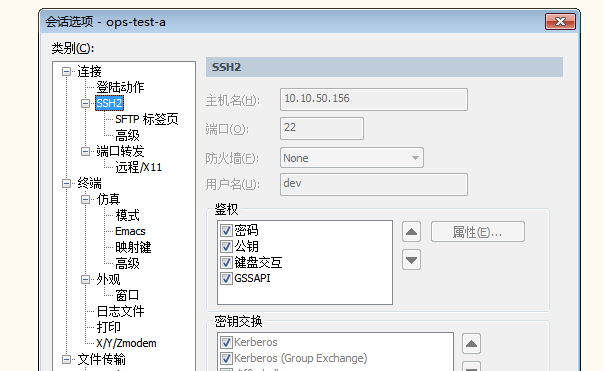
如果一个目录或文件名以一个点 . 开始，表示这个目录或文件是一个隐藏目录或文件(如：.bashrc)。即以默认方式查找时，不显示该目录或文件。

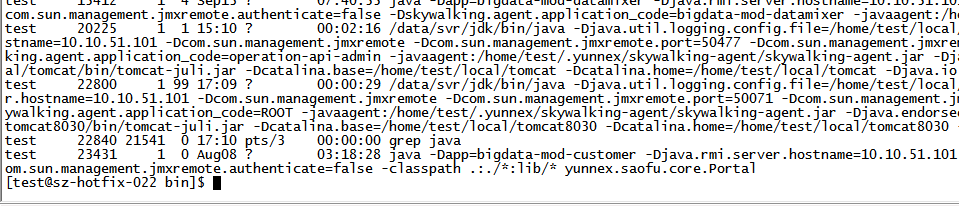
## Linux 远程登录

### 1 概述

Linux一般作为服务器使用，而服务器一般放在机房，你不可能在机房操作你的Linux服务器。这时我们就需要远程登录到Linux服务器来管理维护系统。Linux系统中是通过ssh服务实现的远程登录功能，默认ssh服务端口号为 22。Window系统上 Linux 远程登录客户端有SecureCRT, Putty, SSH Secure Shell等，本文以Putty为例来登录远程服务器







### 2 安装ssh：

yum install ssh

### 3 启动ssh：

service sshd start

### 4 登录远程服务器：

ssh -p 50022 my@127.0.0.1

输入密码：

my@127.0.0.1:

-p 后面是端口

my 是服务器用户名

127.0.0.1 是服务器 ip

回车输入密码即可登录

## 五、Linux 文件基本属性

### 1 文件属性

Linux系统是一种典型的多用户系统，不同的用户处于不同的地位，拥有不同的权限。为了保护系统的安全性，Linux系统对不同的用户访问同一文件（包括目录文件）的权限做了不同的规定。在Linux中我们可以使用ll或者ls –l命令来显示一个文件的属性以及文件所属的用户和组。

[root@www /]# ls -l

total 64

dr-xr-xr-x 2 root root 4096 Dec 14 2012 bin

dr-xr-xr-x 4 root root 4096 Apr 19 2012 boot

……

实例中，bin文件的第一个属性用"d"表示。"d"在Linux中代表该文件是一个目录文件。

在Linux中第一个字符代表这个文件是目录、文件或链接文件等等。

当为[ d ]则是目录

当为[ - ]则是文件；

若是[ l ]则表示为链接文档(link file)；

若是[ b ]则表示为装置文件里面的可供储存的接口设备(可随机存取装置)；

若是[ c ]则表示为装置文件里面的串行端口设备，例如键盘、鼠标(一次性读取装置)。

接下来的字符中，以三个为一组，且均为『rwx』 的三个参数的组合。其中，[ r ]代表可读(read)、[ w ]代表可写(write)、[ x ]代表可执行(execute)。 要注意的是，这三个权限的位置不会改变，如果没有权限，就会出现减号[ - ]而已。

每个文件的属性由左边第一部分的10个字符来确定（如下图）。

363003\_1227493859FdXT

从左至右用0-9这些数字来表示。第0位确定文件类型，第1-3位确定属主（该文件的所有者）拥有该文件的权限。第4-6位确定属组（所有者的同组用户）拥有该文件的权限，第7-9位确定其他用户拥有该文件的权限。其中，第1、4、7位表示读权限，如果用"r"字符表示，则有读权限，如果用"-"字符表示，则没有读权限；第2、5、8位表示写权限，如果用"w"字符表示，则有写权限，如果用"-"字符表示没有写权限；第3、6、9位表示可执行权限，如果用"x"字符表示，则有执行权限，如果用"-"字符表示，则没有执行权限。

### 2 Linux文件属主和属组

[root@www /]# ls -l

total 64

drwxr-xr-x 2 root root 4096 Feb 15 14:46 cron

drwxr-xr-x 3 mysql mysql 4096 Apr 21 2014 mysql

……

对于文件来说，它都有一个特定的所有者，也就是对该文件具有所有权的用户。同时，在Linux系统中，用户是按组分类的，一个用户属于一个或多个组。文件所有者以外的用户又可以分为文件所有者的同组用户和其他用户。因此，Linux系统按文件所有者、文件所有者同组用户和其他用户来规定了不同的文件访问权限。在以上实例中，mysql 文件是一个目录文件，属主和属组都为 mysql，属主有可读、可写、可执行的权限；与属主同组的其他用户有可读和可执行的权限；其他用户也有可读和可执行的权限。对于 root 用户来说，一般情况下，文件的权限对其不起作用。

### 3 更改文件属性

**（1）、chgrp：更改文件属组**

语法：

chgrp [-R] 属组名 文件名

参数选项

-R：递归更改文件属组，就是在更改某个目录文件的属组时，如果加上-R的参数，那么该目录下的所有文件的属组都会更改。

**（2）、chown：更改文件属主，也可以同时更改文件属组**

语法：

chown [–R] 属主名 文件名

chown [-R] 属主名：属组名 文件名

进入 /root 目录（~）将install.log的拥有者改为bin这个账号：

[root@www ~] cd ~

[root@www ~]# chown bin install.log

[root@www ~]# ls -l

-rw-r--r-- 1 bin users 68495 Jun 25 08:53 install.log

将install.log的拥有者与群组改回为root：

[root@www ~]# chown root:root install.log

[root@www ~]# ls -l

-rw-r--r-- 1 root root 68495 Jun 25 08:53 install.log

**（3）、chmod：更改文件9个属性**

Linux文件属性有两种设置方法，一种是数字，一种是符号。

Linux文件的基本权限就有九个，分别是owner/group/others三种身份各有自己的read/write/execute权限。

先复习一下刚刚上面提到的数据：文件的权限字符为：『-rwxrwxrwx』， 这九个权限是三个三个一组的！其中，我们可以使用数字来代表各个权限，各权限的分数对照表如下：

r:4

w:2

x:1

每种身份(owner/group/others)各自的三个权限(r/w/x)分数是需要累加的，例如当权限为： [-rwxrwx---] 分数则是：

owner = rwx = 4+2+1 = 7

group = rwx = 4+2+1 = 7

others= --- = 0+0+0 = 0

所以等一下我们设定权限的变更时，该文件的权限数字就是770啦！变更权限的指令chmod的语法是这样的：

chmod [-R] xyz 文件或目录

选项与参数：

xyz : 就是刚刚提到的数字类型的权限属性，为 rwx 属性数值的相加。

-R : 进行递归(recursive)的持续变更，亦即连同次目录下的所有文件都会变更

举例来说，如果要将.bashrc这个文件所有的权限都设定启用，那么命令如下：

[root@www ~]# ls -al .bashrc

-rw-r--r-- 1 root root 395 Jul 4 11:45 .bashrc

[root@www ~]# chmod 777 .bashrc

[root@www ~]# ls -al .bashrc

-rwxrwxrwx 1 root root 395 Jul 4 11:45 .bashrc

那如果要将权限变成 -rwxr-xr-- 呢？那么权限的分数就成为 [4+2+1][4+0+1][4+0+0]=754。

**（4）、符号类型改变文件权限**

还有一个改变权限的方法呦！从之前的介绍中我们可以发现，基本上就九个权限分别是(1)user (2)group (3)others三种身份啦！ 那么我们就可以藉由u, g, o来代表三种身份的权限！此外， a 则代表 all 亦即全部的身份！那么读写的权限就可以写成r, w, x！也就是可以使用底下的方式来看：

chmod u

g

o

a +(加入)

-(除去)

=(设定) r

w

x 文件或目录

如果我们需要将文件权限设置为 -rwxr-xr-- ，可以使用 chmod u=rwx,g=rx,o=r 文件名 来设定:

# touch test1 // 创建 test1 文件

# ls -al test1 // 查看 test1 默认权限

-rw-r--r-- 1 root root 0 Nov 15 10:32 test1

# chmod u=rwx,g=rx,o=r test1 // 修改 test1 权限

# ls -al test1

-rwxr-xr-- 1 root root 0 Nov 15 10:32 test1

而如果是要将权限去掉而不改变其他已存在的权限呢？例如要拿掉全部人的可执行权限，则：

# chmod a-x test1

# ls -al test1

-rw-r--r-- 1 root root 0 Nov 15 10:32 test1

## 六、Linux 文件与目录管理

### 1 路径

我们知道Linux的目录结构为树状结构，最顶级的目录为根目录 /。其他目录通过挂载可以将它们添加到树中，通过解除挂载可以移除它们。在开始本教程前我们需要先知道什么是绝对路径与相对路径。

@绝对路径：路径的写法，由根目录 / 写起，例如： /usr/share/doc 这个目录。

@相对路径：路径的写法，不是由 / 写起，例如由 /usr/share/doc 要到 /usr/share/man 底下时，可以写成： cd ../man 这就是相对路径的写法啦！

### 2 处理目录的常用命令

接下来我们就来看几个常见的处理目录的命令吧：

ls: 列出目录

cd：切换目录

pwd：显示目前的目录

mkdir：创建一个新的目录

rmdir：删除一个空的目录

cp: 复制文件或目录

rm: 移除文件或目录

你可以使用 man [命令] 来查看各个命令的使用文档，如 ：man cp。

**（1）ls (列出目录)**

在Linux系统当中， ls 命令可能是最常被运行的。

语法：

[root@www ~]# ls [-aAdfFhilnrRSt] 目录名称

[root@www ~]# ls [--color={never,auto,always}] 目录名称

[root@www ~]# ls [--full-time] 目录名称

选项与参数：

-a ：全部的文件，连同隐藏档( 开头为 . 的文件) 一起列出来(常用)

-d ：仅列出目录本身，而不是列出目录内的文件数据(常用)

-l ：长数据串列出，包含文件的属性与权限等等数据；(常用)

将家目录下的所有文件列出来(含属性与隐藏档)

[root@www ~]# ls -al ~

**（2）cd (切换目录)**

cd是Change Directory的缩写，这是用来变换工作目录的命令。

语法：cd [相对路径或绝对路径]

#使用 mkdir 命令创建 runoob 目录

[root@www ~]# mkdir runoob

#使用绝对路径切换到 runoob 目录

[root@www ~]# cd /root/runoob/

#使用相对路径切换到 runoob 目录

[root@www ~]# cd ./runoob/

# 表示回到自己的家目录，亦即是 /root 这个目录

[root@www runoob]# cd ~

# 表示去到目前的上一级目录，亦即是 /root 的上一级目录的意思；

[root@www ~]# cd ..

接下来大家多操作几次应该就可以很好的理解 cd 命令的。

pwd (显示目前所在的目录)

**（3）pwd** 是 Print Working Directory 的缩写，也就是显示目前所在目录的命令。

[root@www ~]# pwd [-P]

选项与参数：-P ：显示出确实的路径，而非使用连结 (link) 路径。

实例：单纯显示出目前的工作目录：

[root@www ~]# pwd

/root <== 显示出目录啦～

实例显示出实际的工作目录，而非连结档本身的目录名而已。

[root@www ~]# cd /var/mail <==注意，/var/mail是一个连结档

[root@www mail]# pwd

/var/mail <==列出目前的工作目录

[root@www mail]# pwd -P

/var/spool/mail <==怎么回事？有没有加 -P 差很多～

[root@www mail]# ls -ld /var/mail

lrwxrwxrwx 1 root root 10 Sep 4 17:54 /var/mail -> spool/mail

# 看到这里应该知道为啥了吧？因为 /var/mail 是连结档，连结到 /var/spool/mail

# 所以，加上 pwd -P 的选项后，会不以连结档的数据显示，而是显示正确的完整路径啊！

**（4）mkdir (创建新目录)**

如果想要创建新的目录的话，那么就使用mkdir (make directory)吧。

语法：mkdir [-mp] 目录名称

选项与参数：

-m ：配置文件的权限喔！直接配置，不需要看默认权限 (umask) 的脸色～

-p ：帮助你直接将所需要的目录(包含上一级目录)递归创建起来！

实例：请到/tmp底下尝试创建数个新目录看看：

[root@www ~]# cd /tmp

[root@www tmp]# mkdir test <==创建一名为 test 的新目录

[root@www tmp]# mkdir test1/test2/test3/test4

mkdir: cannot create directory `test1/test2/test3/test4':

No such file or directory <== 没办法直接创建此目录啊！

[root@www tmp]# mkdir -p test1/test2/test3/test4

加了这个 -p 的选项，可以自行帮你创建多层目录！

实例：创建权限为 rwx--x--x 的目录。

[root@www tmp]# mkdir -m 711 test2

[root@www tmp]# ls -l

drwxr-xr-x 3 root root 4096 Jul 18 12:50 test

drwxr-xr-x 3 root root 4096 Jul 18 12:53 test1

drwx--x--x 2 root root 4096 Jul 18 12:54 test2

上面的权限部分，如果没有加上 -m 来强制配置属性，系统会使用默认属性。

如果我们使用 -m ，如上例我们给予 -m 711 来给予新的目录 drwx--x--x 的权限。

**（5）rmdir (删除空的目录)**

语法：rmdir [-p] 目录名称

选项与参数： -p ：连同上一级『空的』目录也一起删除

删除 runoob 目录

[root@www tmp]# rmdir runoob/

将 mkdir 实例中创建的目录(/tmp 底下)删除掉！

[root@www tmp]# ls -l <==看看有多少目录存在？

drwxr-xr-x 3 root root 4096 Jul 18 12:50 test

drwxr-xr-x 3 root root 4096 Jul 18 12:53 test1

drwx--x--x 2 root root 4096 Jul 18 12:54 test2

[root@www tmp]# rmdir test <==可直接删除掉，没问题

[root@www tmp]# rmdir test1 <==因为尚有内容，所以无法删除！

rmdir: `test1': Directory not empty

[root@www tmp]# rmdir -p test1/test2/test3/test4

[root@www tmp]# ls -l <==您看看，底下的输出中test与test1不见了！

drwx--x--x 2 root root 4096 Jul 18 12:54 test2

利用 -p 这个选项，立刻就可以将 test1/test2/test3/test4 一次删除。

不过要注意的是，这个 rmdir 仅能删除空的目录，你可以使用 rm 命令来删除非空目录。

**（6）cp (复制文件或目录)**

cp 即拷贝文件和目录。

语法:

[root@www ~]# cp [-adfilprsu] 来源档(source) 目标档(destination)

[root@www ~]# cp [options] source1 source2 source3 .... directory

选项与参数：

-a：相当於 -pdr 的意思，至於 pdr 请参考下列说明；(常用)

-d：若来源档为连结档的属性(link file)，则复制连结档属性而非文件本身；

-f：为强制(force)的意思，若目标文件已经存在且无法开启，则移除后再尝试一次；

-i：若目标档(destination)已经存在时，在覆盖时会先询问动作的进行(常用)

-l：进行硬式连结(hard link)的连结档创建，而非复制文件本身；

-p：连同文件的属性一起复制过去，而非使用默认属性(备份常用)；

-r：递归持续复制，用於目录的复制行为；(常用)

-s：复制成为符号连结档 (symbolic link)，亦即『捷径』文件；

-u：若 destination 比 source 旧才升级 destination ！

用 root 身份，将 root 目录下的 .bashrc 复制到 /tmp 下，并命名为 bashrc

[root@www ~]# cp ~/.bashrc /tmp/bashrc

[root@www ~]# cp -i ~/.bashrc /tmp/bashrc

cp: overwrite `/tmp/bashrc'? n <==n不覆盖，y为覆盖

**（7）rm (移除文件或目录)**

语法：rm [-fir] 文件或目录

选项与参数：

-f ：就是 force 的意思，忽略不存在的文件，不会出现警告信息；

-i ：互动模式，在删除前会询问使用者是否动作

-r ：递归删除啊！最常用在目录的删除了！这是非常危险的选项！！！

将刚刚在 cp 的实例中创建的 bashrc 删除掉！

[root@www tmp]# rm -i bashrc

rm: remove regular file `bashrc'? y

如果加上 -i 的选项就会主动询问喔，避免你删除到错误的档名！

**（8）mv (移动文件与目录，或修改名称)**

语法：

[root@www ~]# mv [-fiu] source destination

[root@www ~]# mv [options] source1 source2 source3 .... directory

选项与参数：

-f ：force 强制的意思，如果目标文件已经存在，不会询问而直接覆盖；

-i ：若目标文件 (destination) 已经存在时，就会询问是否覆盖！

-u ：若目标文件已经存在，且 source 比较新，才会升级 (update)

复制一文件，创建一目录，将文件移动到目录中

[root@www ~]# cd /tmp

[root@www tmp]# cp ~/.bashrc bashrc

[root@www tmp]# mkdir mvtest

[root@www tmp]# mv bashrc mvtest

将某个文件移动到某个目录去，就是这样做！

将刚刚的目录名称更名为 mvtest2

[root@www tmp]# mv mvtest mvtest2

### 3 Linux 文件内容查看

Linux系统中使用以下命令来查看文件的内容：

cat 由第一行开始显示文件内容

tac 从最后一行开始显示，可以看出 tac 是 cat 的倒著写！

nl 显示的时候，顺道输出行号！

more 一页一页的显示文件内容

less 与 more 类似，但是比 more 更好的是，他可以往前翻页！

head 只看头几行

tail 只看尾巴几行

**（1）cat**

由第一行开始显示文件内容

语法：cat [-AbEnTv]

选项与参数：

-A ：相当於 -vET 的整合选项，可列出一些特殊字符而不是空白而已；

-b ：列出行号，仅针对非空白行做行号显示，空白行不标行号！

-E ：将结尾的断行字节 $ 显示出来；

-n ：列印出行号，连同空白行也会有行号，与 -b 的选项不同；

-T ：将 [tab] 按键以 ^I 显示出来；

-v ：列出一些看不出来的特殊字符

检看 /etc/issue 这个文件的内容：

[root@www ~]# cat /etc/issue

CentOS release 6.4 (Final)

Kernel \r on an \m

**（2）tac**

tac与cat命令刚好相反，文件内容从最后一行开始显示，可以看出 tac 是 cat 的倒着写！如：

[root@www ~]# tac /etc/issue

Kernel \r on an \m

CentOS release 6.4 (Final)

**（3）nl显示行号**

语法：nl [-bnw] 文件

选项与参数：

-b ：指定行号指定的方式，主要有两种：

-b a ：表示不论是否为空行，也同样列出行号(类似 cat -n)；

-b t ：如果有空行，空的那一行不要列出行号(默认值)；

-n ：列出行号表示的方法，主要有三种：

-n ln ：行号在荧幕的最左方显示；

-n rn ：行号在自己栏位的最右方显示，且不加 0 ；

-n rz ：行号在自己栏位的最右方显示，且加 0 ；

-w ：行号栏位的占用的位数。

实例一：用 nl 列出 /etc/issue 的内容

[root@www ~]# nl /etc/issue

**1 CentOS release 6.4 (Final)**

**2 Kernel \r on an \m**

**（4）more**

一页一页翻动

[root@www ~]# more /etc/man.config

#

# Generated automatically from man.conf.in by the

# configure script.

#

# man.conf from man-1.6d

....(中间省略)....

--More--(28%) <== 重点在这一行喔！你的光标也会在这里等待你的命令

在 more 这个程序的运行过程中，你有几个按键可以按的：

空白键 (space)：代表向下翻一页；

Enter ：代表向下翻『一行』；

/字串 ：代表在这个显示的内容当中，向下搜寻『字串』这个关键字；

:f ：立刻显示出档名以及目前显示的行数；

q ：代表立刻离开 more ，不再显示该文件内容。

b 或 [ctrl]-b ：代表往回翻页，不过这动作只对文件有用，对管线无用。

**（5）less**

一页一页翻动，以下实例输出/etc/man.config文件的内容：

[root@www ~]# less /etc/man.config

#

# Generated automatically from man.conf.in by the

# configure script.

#

# man.conf from man-1.6d

....(中间省略)....

: <== 这里可以等待你输入命令！

less运行时可以输入的命令有：

空白键 ：向下翻动一页；

[pagedown]：向下翻动一页；

[pageup] ：向上翻动一页；

/字串 ：向下搜寻『字串』的功能；

?字串 ：向上搜寻『字串』的功能；

n ：重复前一个搜寻 (与 / 或 ? 有关！)

N ：反向的重复前一个搜寻 (与 / 或 ? 有关！)

q ：离开 less 这个程序；

**（6）head**

取出文件前面几行

语法：head [-n number] 文件

选项与参数：

-n ：后面接数字，代表显示几行的意思

[root@www ~]# head /etc/man.config

默认的情况中，显示前面 10 行！若要显示前 20 行，就得要这样：

[root@www ~]# head -n 20 /etc/man.config

**（7）tail**

取出文件后面几行

语法：tail [-n number] 文件

选项与参数：

-n ：后面接数字，代表显示几行的意思

-f ：表示持续侦测后面所接的档名，要等到按下[ctrl]-c才会结束tail的侦测

[root@www ~]# tail /etc/man.config

# 默认的情况中，显示最后的十行！若要显示最后的 20 行，就得要这样：

[root@www ~]# tail -n 20 /etc/man.config

## 七、Linux 用户和用户组管理

### 1 Linux 用户系统

Linux系统是一个多用户多任务的分时操作系统，任何一个要使用系统资源的用户，都必须首先向系统管理员申请一个账号，然后以这个账号的身份进入系统。用户的账号一方面可以帮助系统管理员对使用系统的用户进行跟踪，并控制他们对系统资源的访问；另一方面也可以帮助用户组织文件，并为用户提供安全性保护。每个用户账号都拥有一个惟一的用户名和各自的口令。用户在登录时键入正确的用户名和口令后，就能够进入系统和自己的主目录。

实现用户账号的管理，要完成的工作主要有如下几个方面：

**用户账号的添加、删除与修改。**

**用户口令的管理。**

**用户组的管理。**

### 2 Linux系统用户账号的管理

用户账号的管理工作主要涉及到用户账号的添加、修改和删除。

添加用户账号就是在系统中创建一个新账号，然后为新账号分配用户号、用户组、主目录和登录Shell等资源。刚添加的账号是被锁定的，无法使用。

**（1）添加新的用户账号使用useradd命令，其语法如下：**

useradd 选项 用户名

参数说明：

-c comment 指定一段注释性描述。

-d 目录 指定用户主目录，如果此目录不存在，则同时使用-m选项，可以创建主目录。

-g 用户组 指定用户所属的用户组。

-G 用户组，用户组 指定用户所属的附加组。

-s Shell文件 指定用户的登录Shell。

-u 用户号 指定用户的用户号，如果同时有-o选项，则可以重复使用其他用户的标识号。

实例1

# useradd –d /usr/sam -m sam

此命令创建了一个用户sam，其中-d和-m选项用来为登录名sam产生一个主目录/usr/sam（/usr为默认的用户主目录所在的父目录）。

实例2

# useradd -s /bin/sh -g group –G adm,root gem

此命令新建了一个用户gem，该用户的登录Shell是 /bin/sh，它属于group用户组，同时又属于adm和root用户组，其中group用户组是其主组。

这里可能新建组：#groupadd group及groupadd adm

增加用户账号就在/etc/passwd文件中为新用户增加一条记录，同时更新其他系统文件如/etc/shadow, /etc/group等。

Linux提供了集成的系统管理工具userconf，它可以用来对用户账号进行统一管理。

**（2）删除帐号**

如果一个用户的账号不再使用，可以从系统中删除。删除用户账号就是要将/etc/passwd等系统文件中的该用户记录删除，必要时还删除用户的主目录。

删除一个已有的用户账号使用userdel命令，其格式如下：

userdel 选项 用户名

常用的选项是 -r，它的作用是把用户的主目录一起删除。

例如：

# userdel -r sam

此命令删除用户sam在系统文件中（主要是/etc/passwd, /etc/shadow, /etc/group等）的记录，同时删除用户的主目录。

**（3）修改帐号**

修改用户账号就是根据实际情况更改用户的有关属性，如用户号、主目录、用户组、登录Shell等。

修改已有用户的信息使用usermod命令，其格式如下：

usermod 选项 用户名

常用的选项包括-c, -d, -m, -g, -G, -s, -u以及-o等，这些选项的意义与useradd命令中的选项一样，可以为用户指定新的资源值。

另外，有些系统可以使用选项：-l 新用户名

这个选项指定一个新的账号，即将原来的用户名改为新的用户名。

例如：

# usermod -s /bin/ksh -d /home/z –g developer sam

此命令将用户sam的登录Shell修改为ksh，主目录改为/home/z，用户组改为developer。

**（4）用户口令的管理**

用户管理的一项重要内容是用户口令的管理。用户账号刚创建时没有口令，但是被系统锁定，无法使用，必须为其指定口令后才可以使用，即使是指定空口令。

指定和修改用户口令的Shell命令是passwd。超级用户可以为自己和其他用户指定口令，普通用户只能用它修改自己的口令。命令的格式为：

passwd 选项 用户名

可使用的选项：

-l 锁定口令，即禁用账号。

-u 口令解锁。

-d 使账号无口令。

-f 强迫用户下次登录时修改口令。

如果默认用户名，则修改当前用户的口令。

例如，假设当前用户是sam，则下面的命令修改该用户自己的口令：

$ passwd

Old password:\*\*\*\*\*\*

New password:\*\*\*\*\*\*\*

Re-enter new password:\*\*\*\*\*\*\*

如果是超级用户，可以用下列形式指定任何用户的口令：

# passwd sam

New password:\*\*\*\*\*\*\*

Re-enter new password:\*\*\*\*\*\*\*

普通用户修改自己的口令时，passwd命令会先询问原口令，验证后再要求用户输入两遍新口令，如果两次输入的口令一致，则将这个口令指定给用户；而超级用户为用户指定口令时，就不需要知道原口令。

为了系统安全起见，用户应该选择比较复杂的口令，例如最好使用8位长的口令，口令中包含有大写、小写字母和数字，并且应该与姓名、生日等不相同。

为用户指定空口令时，执行下列形式的命令：

# passwd -d sam

此命令将用户sam的口令删除，这样用户sam下一次登录时，系统就不再询问口令。

passwd命令还可以用-l(lock)选项锁定某一用户，使其不能登录，例如：

# passwd -l sam

### 3 Linux系统用户组的管理

每个用户都有一个用户组，系统可以对一个用户组中的所有用户进行集中管理。不同Linux 系统对用户组的规定有所不同，如Linux下的用户属于与它同名的用户组，这个用户组在创建用户时同时创建。

用户组的管理涉及用户组的添加、删除和修改。组的增加、删除和修改实际上就是对/etc/group文件的更新。

**（1）增加一个新的用户组使用groupadd命令。其格式如下：**

groupadd 选项 用户组

可以使用的选项有：

-g GID 指定新用户组的组标识号（GID）。

-o 一般与-g选项同时使用，表示新用户组的GID可以与系统已有用户组的GID相同。

实例1：

# groupadd group1

此命令向系统中增加了一个新组group1，新组的组标识号是在当前已有的最大组标识号的基础上加1。

实例2：

# groupadd -g 101 group2

此命令向系统中增加了一个新组group2，同时指定新组的组标识号是101。

**（2）如果要删除一个已有的用户组，使用groupdel命令，其格式如下：**

groupdel 用户组

例如：

# groupdel group1

此命令从系统中删除组group1。

**（3）修改用户组的属性使用groupmod命令。其语法如下：**

groupmod 选项 用户组

常用的选项有：

-g GID 为用户组指定新的组标识号。

-o 与-g选项同时使用，用户组的新GID可以与系统已有用户组的GID相同。

-n新用户组 将用户组的名字改为新名字

实例1：

# groupmod -g 102 group2

此命令将组group2的组标识号修改为102。

实例2：

# groupmod –g 10000 -n group3 group2

此命令将组group2的标识号改为10000，组名修改为group3。

**（4）如果一个用户同时属于多个用户组，那么用户可以在用户组之间切换，以便具有其他用户组的权限。**

用户可以在登录后，使用命令newgrp切换到其他用户组，这个命令的参数就是目的用户组。例如：

$ newgrp root

这条命令将当前用户切换到root用户组，前提条件是root用户组确实是该用户的主组或附加组。类似于用户账号的管理，用户组的管理也可以通过集成的系统管理工具来完成。

### 4 与用户账号有关的系统文件

完成用户管理的工作有许多种方法，但是每一种方法实际上都是对有关的系统文件进行修改。与用户和用户组相关的信息都存放在一些系统文件中，这些文件包括/etc/passwd, /etc/shadow, /etc/group等。

**（1）、/etc/passwd文件是用户管理工作涉及的最重要的一个文件。**

Linux系统中的每个用户都在/etc/passwd文件中有一个对应的记录行，它记录了这个用户的一些基本属性。

这个文件对所有用户都是可读的。它的内容类似下面的例子：

＃ cat /etc/passwd

root:x:0:0:Superuser:/:

daemon:x:1:1:System daemons:/etc:

bin:x:2:2:Owner of system commands:/bin:

sys:x:3:3:Owner of system files:/usr/sys:

adm:x:4:4:System accounting:/usr/adm:

uucp:x:5:5:UUCP administrator:/usr/lib/uucp:

auth:x:7:21:Authentication administrator:/tcb/files/auth:

cron:x:9:16:Cron daemon:/usr/spool/cron:

listen:x:37:4:Network daemon:/usr/net/nls:

lp:x:71:18:Printer administrator:/usr/spool/lp:

sam:x:200:50:Sam san:/usr/sam:/bin/sh

从上面的例子我们可以看到，/etc/passwd中一行记录对应着一个用户，每行记录又被冒号(:)分隔为7个字段，其格式和具体含义如下：

用户名:口令:用户标识号:组标识号:注释性描述:主目录:登录Shell

**1）"用户名"是代表用户账号的字符串。**

通常长度不超过8个字符，并且由大小写字母和/或数字组成。登录名中不能有冒号(:)，因为冒号在这里是分隔符。

为了兼容起见，登录名中最好不要包含点字符(.)，并且不使用连字符(-)和加号(+)打头。

**2）“口令”一些系统中，存放着加密后的用户口令字。**

虽然这个字段存放的只是用户口令的加密串，不是明文，但是由于/etc/passwd文件对所有用户都可读，所以这仍是一个安全隐患。因此，现在许多Linux 系统（如SVR4）都使用了shadow技术，把真正的加密后的用户口令字存放到/etc/shadow文件中，而在/etc/passwd文件的口令字段中只存放一个特殊的字符，例如“x”或者“\*”。

**3）“用户标识号”是一个整数，系统内部用它来标识用户。**

一般情况下它与用户名是一一对应的。如果几个用户名对应的用户标识号是一样的，系统内部将把它们视为同一个用户，但是它们可以有不同的口令、不同的主目录以及不同的登录Shell等。

通常用户标识号的取值范围是0～65 535。0是超级用户root的标识号，1～99由系统保留，作为管理账号，普通用户的标识号从100开始。在Linux系统中，这个界限是500。

**4）“组标识号”字段记录的是用户所属的用户组。**

它对应着/etc/group文件中的一条记录。

**5)“注释性描述”字段记录着用户的一些个人情况。**

例如用户的真实姓名、电话、地址等，这个字段并没有什么实际的用途。在不同的Linux 系统中，这个字段的格式并没有统一。在许多Linux系统中，这个字段存放的是一段任意的注释性描述文字，用做finger命令的输出。

**6)“主目录”，也就是用户的起始工作目录。**

它是用户在登录到系统之后所处的目录。在大多数系统中，各用户的主目录都被组织在同一个特定的目录下，而用户主目录的名称就是该用户的登录名。各用户对自己的主目录有读、写、执行（搜索）权限，其他用户对此目录的访问权限则根据具体情况设置。

**7)用户登录后，要启动一个进程，负责将用户的操作传给内核，这个进程是用户登录到系统后运行的命令解释器或某个特定的程序，即Shell。**

Shell是用户与Linux系统之间的接口。Linux的Shell有许多种，每种都有不同的特点。常用的有sh(Bourne Shell), csh(C Shell), ksh(Korn Shell), tcsh(TENEX/TOPS-20 type C Shell), bash(Bourne Again Shell)等。系统管理员可以根据系统情况和用户习惯为用户指定某个Shell。如果不指定Shell，那么系统使用sh为默认的登录Shell，即这个字段的值为/bin/sh。用户的登录Shell也可以指定为某个特定的程序（此程序不是一个命令解释器）。

利用这一特点，我们可以限制用户只能运行指定的应用程序，在该应用程序运行结束后，用户就自动退出了系统。有些Linux 系统要求只有那些在系统中登记了的程序才能出现在这个字段中。

**8)系统中有一类用户称为伪用户（pseudo users）。**

这些用户在/etc/passwd文件中也占有一条记录，但是不能登录，因为它们的登录Shell为空。它们的存在主要是方便系统管理，满足相应的系统进程对文件属主的要求。

常见的伪用户如下所示：

**伪 用 户 含 义**

bin 拥有可执行的用户命令文件

sys 拥有系统文件

adm 拥有帐户文件

uucp UUCP使用

lp lp或lpd子系统使用

nobody NFS使用

### 5 拥有帐户文件

**（1）除了上面列出的伪用户外，还有许多标准的伪用户，例如：audit, cron, mail, usenet等，它们也都各自为相关的进程和文件所需要。**

由于/etc/passwd文件是所有用户都可读的，如果用户的密码太简单或规律比较明显的话，一台普通的计算机就能够很容易地将它破解，因此对安全性要求较高的Linux系统都把加密后的口令字分离出来，单独存放在一个文件中，这个文件是/etc/shadow文件。 有超级用户才拥有该文件读权限，这就保证了用户密码的安全性。

**（2）/etc/shadow中的记录行与/etc/passwd中的一一对应，它由pwconv命令根据/etc/passwd中的数据自动产生**

它的文件格式与/etc/passwd类似，由若干个字段组成，字段之间用":"隔开。这些字段是：

登录名:加密口令:最后一次修改时间:最小时间间隔:最大时间间隔:警告时间:不活动时间:失效时间:标志

"登录名"是与/etc/passwd文件中的登录名相一致的用户账号

"口令"字段存放的是加密后的用户口令字，长度为13个字符。如果为空，则对应用户没有口令，登录时不需要口令；如果含有不属于集合 { ./0-9A-Za-z }中的字符，则对应的用户不能登录。

"最后一次修改时间"表示的是从某个时刻起，到用户最后一次修改口令时的天数。时间起点对不同的系统可能不一样。例如在SCO Linux 中，这个时间起点是1970年1月1日。

"最小时间间隔"指的是两次修改口令之间所需的最小天数。

"最大时间间隔"指的是口令保持有效的最大天数。

"警告时间"字段表示的是从系统开始警告用户到用户密码正式失效之间的天数。

"不活动时间"表示的是用户没有登录活动但账号仍能保持有效的最大天数。

"失效时间"字段给出的是一个绝对的天数，如果使用了这个字段，那么就给出相应账号的生存期。期满后，该账号就不再是一个合法的账号，也就不能再用来登录了。

下面是/etc/shadow的一个例子：

＃ cat /etc/shadow

root:Dnafw28zf38w:8764:0:168:7:::

daemon:\*::0:0::::

bin:\*::0:0::::

sys:\*::0:0::::

adm:\*::0:0::::

uucp:\*::0:0::::

nuucp:\*::0:0::::

auth:\*::0:0::::

cron:\*::0:0::::

listen:\*::0:0::::

lp:\*::0:0::::

sam:EkdiSECLWPdSa:9740:0:0::::

**（3）用户组的所有信息都存放在/etc/group文件中。**

将用户分组是Linux 系统中对用户进行管理及控制访问权限的一种手段。

每个用户都属于某个用户组；一个组中可以有多个用户，一个用户也可以属于不同的组。

当一个用户同时是多个组中的成员时，在/etc/passwd文件中记录的是用户所属的主组，也就是登录时所属的默认组，而其他组称为附加组。

用户要访问属于附加组的文件时，必须首先使用newgrp命令使自己成为所要访问的组中的成员。

用户组的所有信息都存放在/etc/group文件中。此文件的格式也类似于/etc/passwd文件，由冒号(:)隔开若干个字段，这些字段有：

组名:口令:组标识号:组内用户列表

"组名"是用户组的名称，由字母或数字构成。与/etc/passwd中的登录名一样，组名不应重复。

"口令"字段存放的是用户组加密后的口令字。一般Linux 系统的用户组都没有口令，即这个字段一般为空，或者是\*。

"组标识号"与用户标识号类似，也是一个整数，被系统内部用来标识组。

"组内用户列表"是属于这个组的所有用户的列表/b]，不同用户之间用逗号(,)分隔。这个用户组可能是用户的主组，也可能是附加组。

/etc/group文件的一个例子如下：

root::0:root

bin::2:root,bin

sys::3:root,uucp

adm::4:root,adm

daemon::5:root,daemon

lp::7:root,lp

users::20:root,sam

### 6 添加批量用户

添加和删除用户对每位Linux系统管理员都是轻而易举的事，比较棘手的是如果要添加几十个、上百个甚至上千个用户时，我们不太可能还使用useradd一个一个地添加，必然要找一种简便的创建大量用户的方法。Linux系统提供了创建大量用户的工具，可以让您立即创建大量用户，方法如下：

**（1）先编辑一个文本用户文件。**

每一列按照/etc/passwd密码文件的格式书写，要注意每个用户的用户名、UID、宿主目录都不可以相同，其中密码栏可以留做空白或输入x号。一个范例文件user.txt内容如下：

user001::600:100:user:/home/user001:/bin/bash

user002::601:100:user:/home/user002:/bin/bash

user003::602:100:user:/home/user003:/bin/bash

user004::603:100:user:/home/user004:/bin/bash

user005::604:100:user:/home/user005:/bin/bash

user006::605:100:user:/home/user006:/bin/bash

**（2）以root身份执行命令 /usr/sbin/newusers，从刚创建的用户文件user.txt中导入数据，创建用户：**

# newusers < user.txt

然后可以执行命令 vipw 或 vi /etc/passwd 检查 /etc/passwd 文件是否已经出现这些用户的数据，并且用户的宿主目录是否已经创建。

**（3）执行命令/usr/sbin/pwunconv。**

将 /etc/shadow 产生的 shadow 密码解码，然后回写到 /etc/passwd 中，并将/etc/shadow的shadow密码栏删掉。这是为了方便下一步的密码转换工作，即先取消 shadow password 功能。

# pwunconv

**（4）编辑每个用户的密码对照文件。**

范例文件 passwd.txt 内容如下：

user001:密码

user002:密码

user003:密码

user004:密码

user005:密码

user006:密码

**（5）以root身份执行命令 /usr/sbin/chpasswd。**

创建用户密码，chpasswd 会将经过 /usr/bin/passwd 命令编码过的密码写入 /etc/passwd 的密码栏。

# chpasswd < passwd.txt

**（6）确定密码经编码写入/etc/passwd的密码栏后。**

执行命令 /usr/sbin/pwconv 将密码编码为 shadow password，并将结果写入 /etc/shadow。

# pwconv

这样就完成了大量用户的创建了，之后您可以到/home下检查这些用户宿主目录的权限设置是否都正确，并登录验证用户密码是否正确。

## 八、Linux 磁盘管理

### 1 Linux磁盘管理常用三个命令

df：列出文件系统的整体磁盘使用量

du：检查磁盘空间使用量

fdisk：用于磁盘分区

**Linux 操作系统的文件数据除了文件实际内容外，通常含有非常多的属性，例如 Linux 操作系统的文件权限（rwx）与文件属性（拥有者、群组、时间参数等）。文件系统通常会将这两部分的数据分别存放在不同的区块，权限与属性存放在 inode 中，至于实际数据则放置到 data block 区块中。另外，还有一个超级区块（superblock）会记录整个文件系统的整体信息，包括 inode 与 block 的总量、使用量、剩余量等。**

inode：记录文件的属性，一个文件占用一个 inode，同时记录此文件的数据所在的 block。

在 Linux 中，可以使用 stat 命令查看某个文件的 inode 信息：

stat /etc/passwd

linux-peanut:~/Desktop # stat /etc/passwd

File: `/etc/passwd'

Size: 2269 Blocks: 8 IO Block: 4096 regular file

Device: 802h/2050d Inode: 1149740 Links: 1

Access: (0644/-rw-r--r--) Uid: ( 0/ root) Gid: ( 0/ root)

Access: 2018-04-09 17:16:42.000000000 +0800

Modify: 2018-04-09 17:16:36.000000000 +0800

Change: 2018-04-09 17:16:36.000000000 +0800

Birth: -

可见，文件的绝大部分信息都存储在 inode 中。

### 2 df

df命令参数功能：检查文件系统的磁盘空间占用情况。可以利用该命令来获取硬盘被占用了多少空间，目前还剩下多少空间等信息。

语法：df [-ahikHTm] [目录或文件名]

选项与参数：

-a ：列出所有的文件系统，包括系统特有的 /proc 等文件系统；

-k ：以 KBytes 的容量显示各文件系统；

-m ：以 MBytes 的容量显示各文件系统；

-h ：以人们较易阅读的 GBytes, MBytes, KBytes 等格式自行显示；

-H ：以 M=1000K 取代 M=1024K 的进位方式；

-T ：显示文件系统类型, 连同该 partition 的 filesystem 名称 (例如 ext3) 也列出；

-i ：不用硬盘容量，而以 inode 的数量来显示

**实例 1**

将系统内所有的文件系统列出来！

[root@www ~]# df

Filesystem 1K-blocks Used Available Use% Mounted on

/dev/hdc2 9920624 3823112 5585444 41% /

/dev/hdc3 4956316 141376 4559108 4% /home

/dev/hdc1 101086 11126 84741 12% /boot

tmpfs 371332 0 371332 0% /dev/shm

在 Linux 底下如果 df 没有加任何选项，那么默认会将系统内所有的 (不含特殊内存内的文件系统与 swap) 都以 1 Kbytes 的容量来列出来！

**实例 2**

将容量结果以易读的容量格式显示出来

[root@www ~]# df -h

Filesystem Size Used Avail Use% Mounted on

/dev/hdc2 9.5G 3.7G 5.4G 41% /

/dev/hdc3 4.8G 139M 4.4G 4% /home

/dev/hdc1 99M 11M 83M 12% /boot

tmpfs 363M 0 363M 0% /dev/shm

### 3 du

Linux du命令也是查看使用空间的，但是与df命令不同的是Linux du命令是对文件和目录磁盘使用的空间的查看，还是和df命令有一些区别的，这里介绍Linux du命令。

语法：du [-ahskm] 文件或目录名称

选项与参数：

-a ：列出所有的文件与目录容量，因为默认仅统计目录底下的文件量而已。

-h ：以人们较易读的容量格式 (G/M) 显示；

-s ：列出总量而已，而不列出每个各别的目录占用容量；

-S ：不包括子目录下的总计，与 -s 有点差别。

-k ：以 KBytes 列出容量显示；

-m ：以 MBytes 列出容量显示；

**实例 1**

列出目前目录下的所有文件容量

[root@www ~]# du

8 ./test4 <==每个目录都会列出来

8 ./test2

....中间省略....

12 ./.gconfd <==包括隐藏文件的目录

220 . <==这个目录(.)所占用的总量

直接输入 du 没有加任何选项时，则 du 会分析当前所在目录的文件与目录所占用的硬盘空间。

**实例 2**

将文件的容量也列出来

[root@www ~]# du -a

12 ./install.log.syslog <==有文件的列表了

8 ./.bash\_logout

8 ./test4

8 ./test2

....中间省略....

12 ./.gconfd

220 .

### 4 fdisk

fdisk 是 Linux 的磁盘分区表操作工具。

语法：fdisk [-l] 装置名称

选项与参数：

-l ：输出后面接的装置所有的分区内容。若仅有 fdisk -l 时， 则系统将会把整个系统内能够搜寻到的装置的分区均列出来。

**实例 1**

列出所有分区信息

[root@AY120919111755c246621 tmp]# fdisk -l

Disk /dev/xvda: 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: 0x00000000

Device Boot Start End Blocks Id System

/dev/xvda1 \* 1 2550 20480000 83 Linux

/dev/xvda2 2550 2611 490496 82 Linux swap / Solaris

Disk /dev/xvdb: 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: 0x56f40944

Device Boot Start End Blocks Id System

/dev/xvdb2 1 2610 20964793+ 83 Linux

### 5 磁盘格式化

磁盘分割完毕后自然就是要进行文件系统的格式化，格式化的命令非常的简单，使用 mkfs（make filesystem） 命令。

语法：mkfs [-t 文件系统格式] 装置文件名

选项与参数：

-t ：可以接文件系统格式，例如 ext3, ext2, vfat 等(系统有支持才会生效)

实例 1

查看 mkfs 支持的文件格式

[root@www ~]# mkfs[tab][tab]

mkfs mkfs.cramfs mkfs.ext2 mkfs.ext3 mkfs.msdos mkfs.vfat

按下两个[tab]，会发现 mkfs 支持的文件格式如上所示。

### 6 磁盘检验

fsck（file system check）用来检查和维护不一致的文件系统。

若系统掉电或磁盘发生问题，可利用fsck命令对文件系统进行检查。

语法：fsck [-t 文件系统] [-ACay] 装置名称

选项与参数：

-t : 给定档案系统的型式，若在 /etc/fstab 中已有定义或 kernel 本身已支援的则不需加上此参数

-s : 依序一个一个地执行 fsck 的指令来检查

-A : 对/etc/fstab 中所有列出来的 分区（partition）做检查

-C : 显示完整的检查进度

-d : 打印出 e2fsck 的 debug 结果

-p : 同时有 -A 条件时，同时有多个 fsck 的检查一起执行

-R : 同时有 -A 条件时，省略 / 不检查

-V : 详细显示模式

-a : 如果检查有错则自动修复

-r : 如果检查有错则由使用者回答是否修复

-y : 选项指定检测每个文件是自动输入yes，在不确定那些是不正常的时候，可以执行 # fsck -y 全部检查修复。

实例 1

查看系统有多少文件系统支持的 fsck 命令：

[root@www ~]# fsck[tab][tab]

fsck fsck.cramfs fsck.ext2 fsck.ext3 fsck.msdos fsck.vfat

### 7 磁盘挂载与卸除

Linux 的磁盘挂载使用 mount 命令，卸载使用 umount 命令。

磁盘挂载语法：mount [-t 文件系统] [-L Label名] [-o 额外选项] [-n] 装置文件名 挂载点

实例 1

用默认的方式，将刚刚创建的 /dev/hdc6 挂载到 /mnt/hdc6 上面！

[root@www ~]# mkdir /mnt/hdc6

[root@www ~]# mount /dev/hdc6 /mnt/hdc6

[root@www ~]# df

Filesystem 1K-blocks Used Available Use% Mounted on

.....中间省略.....

/dev/hdc6 1976312 42072 1833836 3% /mnt/hdc6

磁盘卸载命令 umount 语法：

umount [-fn] 装置文件名或挂载点

选项与参数：

-f ：强制卸除！可用在类似网络文件系统 (NFS) 无法读取到的情况下；

-n ：不升级 /etc/mtab 情况下卸除。

卸载/dev/hdc6

[root@www ~]# umount /dev/hdc6

## 九、Linux vi/vim

### 1 Vim含义

Vim是从 vi 发展出来的一个文本编辑器。代码补完、编译及错误跳转等方便编程的功能特别丰富，在程序员中被广泛使用。简单的来说， vi 是老式的字处理器，不过功能已经很齐全了，但是还是有可以进步的地方。 vim 则可以说是程序开发者的一项很好用的工具。

### 2 vi/vim 的使用

基本上 vi/vim 共分为三种模式，分别是**命令模式（Command mode），输入模式（Insert mode）和底线命令模式（Last line mode）。**

**（1）命令模式：**

用户刚刚启动 vi/vim，便进入了命令模式。此状态下敲击键盘动作会被Vim识别为命令，而非输入字符。比如我们此时按下i，并不会输入一个字符，i被当作了一个命令。以下是常用的几个命令：

i 切换到输入模式，以输入字符。

x 删除当前光标所在处的字符。

: 切换到底线命令模式，以在最底一行输入命令。

若想要编辑文本：启动Vim，进入了命令模式，按下i，切换到输入模式。命令模式只有一些最基本的命令，因此仍要依靠底线命令模式输入更多命令。

**（2）输入模式**

在命令模式下按下i就进入了输入模式。

在输入模式中，可以使用以下按键：

字符按键以及Shift组合，输入字符

ENTER，回车键，换行

BACK SPACE，退格键，删除光标前一个字符

DEL，删除键，删除光标后一个字符

方向键，在文本中移动光标

HOME/END，移动光标到行首/行尾

Page Up/Page Down，上/下翻页

Insert，切换光标为输入/替换模式，光标将变成竖线/下划线

ESC，退出输入模式，切换到命令模式

**（3）底线命令模式**

在命令模式下按下:（英文冒号）就进入了底线命令模式。

底线命令模式可以输入单个或多个字符的命令，可用的命令非常多。

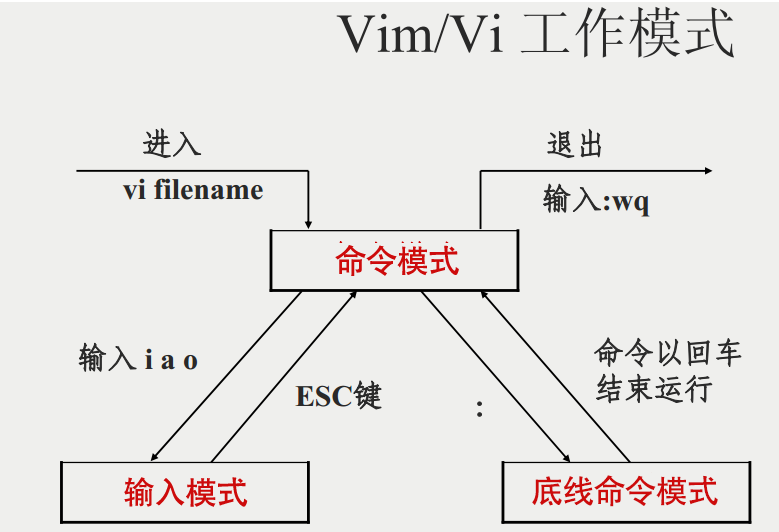
在底线命令模式中，基本的命令有（已经省略了冒号）：

q 退出程序

w 保存文件

按ESC键可随时退出底线命令模式。

简单的说，我们可以将这三个模式想成底下的图标来表示：



### 3 vi/vim 使用实例

**（1）使用 vi/vim 进入一般模式**

如果你想要使用 vi 来建立一个名为 test.txt 的文件时，你可以这样做：

$ vi runoob.txt

**（2）直接输入 vi 文件名** 就能够进入 vi 的一般模式了。请注意，记得 vi 后面一定要加文件名，不管该文件存在与否！



**（3）按下 i** 进入输入模式(也称为编辑模式)，开始编辑文字。在一般模式之中，只要按下 i, o, a 等字符就可以进入输入模式了！在编辑模式当中，你可以发现在左下角状态栏中会出现 –INSERT- 的字样，那就是可以输入任意字符的提示。这个时候，键盘上除了 Esc 这个按键之外，其他的按键都可以视作为一般的输入按钮了，所以你可以进行任何的编辑。

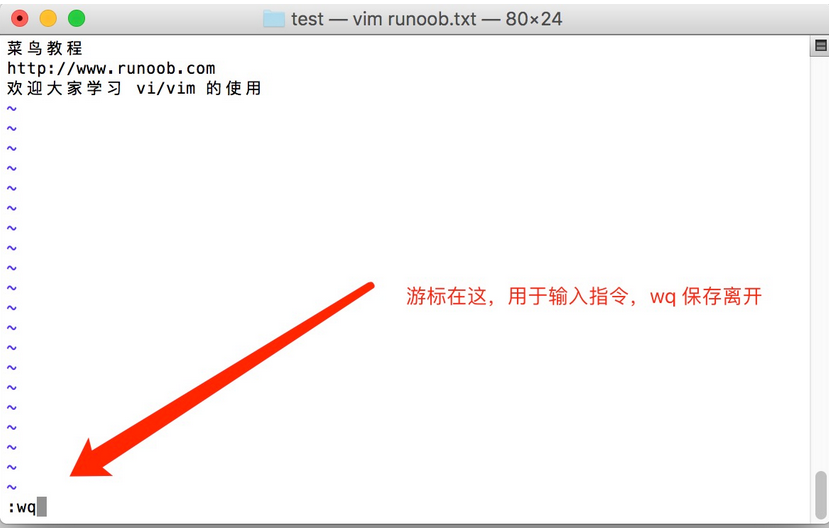


**（4）按下 ESC 按钮回到一般模式**

好了，假设我已经按照上面的样式给他编辑完毕了，那么应该要如何退出呢？是的！没错！就是给他按下 Esc 这个按钮即可！马上你就会发现画面左下角的 – INSERT – 不见了！

**（5）在一般模式中按下 :wq 储存后离开 vi**

OK，我们要存档了，存盘并离开的指令很简单，输入 :wq 即可保存离开！



### 4 vi/vim 按键说明

**第一部份：一般模式可用的光标移动、复制粘贴、搜索替换等**

|  |  |
| --- | --- |
| **移动光标的方法** | |
| h 或 向左箭头键(←) | 光标向左移动一个字符 |
| j 或 向下箭头键(↓) | 光标向下移动一个字符 |
| k 或 向上箭头键(↑) | 光标向上移动一个字符 |
| l 或 向右箭头键(→) | 光标向右移动一个字符 |
| 如果你将右手放在键盘上的话，你会发现 hjkl 是排列在一起的，因此可以使用这四个按钮来移动光标。 如果想要进行多次移动的话，例如向下移动 30 行，可以使用 "30j" 或 "30↓" 的组合按键， 亦即加上想要进行的次数(数字)后，按下动作即可！ | |
| [Ctrl] + [f] | 屏幕『向下』移动一页，相当于 [Page Down]按键 (常用) |
| [Ctrl] + [b] | 屏幕『向上』移动一页，相当于 [Page Up] 按键 (常用) |
| [Ctrl] + [d] | 屏幕『向下』移动半页 |
| [Ctrl] + [u] | 屏幕『向上』移动半页 |
| + | 光标移动到非空格符的下一行 |
| - | 光标移动到非空格符的上一行 |
| n<space> | 那个 n 表示『数字』，例如 20 。按下数字后再按空格键，光标会向右移动这一行的 n 个字符。例如 20<space> 则光标会向后面移动 20 个字符距离。 |
| 0 或功能键[Home] | 这是数字『 0 』：移动到这一行的最前面字符处 (常用) |
| $ 或功能键[End] | 移动到这一行的最后面字符处(常用) |
| H | 光标移动到这个屏幕的最上方那一行的第一个字符 |
| M | 光标移动到这个屏幕的中央那一行的第一个字符 |
| L | 光标移动到这个屏幕的最下方那一行的第一个字符 |
| G | 移动到这个档案的最后一行(常用) |
| nG | n 为数字。移动到这个档案的第 n 行。例如 20G 则会移动到这个档案的第 20 行(可配合 :set nu) |
| gg | 移动到这个档案的第一行，相当于 1G 啊！ (常用) |
| n<Enter> | n 为数字。光标向下移动 n 行(常用) |
| 搜索替换 | |
| /word | 向光标之下寻找一个名称为 word 的字符串。例如要在档案内搜寻 vbird 这个字符串，就输入 /vbird 即可！ (常用) |
| ?word | 向光标之上寻找一个字符串名称为 word 的字符串。 |
| n | 这个 n 是英文按键。代表重复前一个搜寻的动作。举例来说， 如果刚刚我们执行 /vbird 去向下搜寻 vbird 这个字符串，则按下 n 后，会向下继续搜寻下一个名称为 vbird 的字符串。如果是执行 ?vbird 的话，那么按下 n 则会向上继续搜寻名称为 vbird 的字符串！ |
| N | 这个 N 是英文按键。与 n 刚好相反，为『反向』进行前一个搜寻动作。 例如 /vbird 后，按下 N 则表示『向上』搜寻 vbird 。 |
| 使用 /word 配合 n 及 N 是非常有帮助的！可以让你重复的找到一些你搜寻的关键词！ | |
| :n1,n2s/word1/word2/g | n1 与 n2 为数字。在第 n1 与 n2 行之间寻找 word1 这个字符串，并将该字符串取代为 word2 ！举例来说，在 100 到 200 行之间搜寻 vbird 并取代为 VBIRD 则： 『:100,200s/vbird/VBIRD/g』。(常用) |
| :1,$s/word1/word2/g | 从第一行到最后一行寻找 word1 字符串，并将该字符串取代为 word2 ！(常用) |
| :1,$s/word1/word2/gc | 从第一行到最后一行寻找 word1 字符串，并将该字符串取代为 word2 ！且在取代前显示提示字符给用户确认 (confirm) 是否需要取代！(常用) |
| 删除、复制与贴上 | |
| x, X | 在一行字当中，x 为向后删除一个字符 (相当于 [del] 按键)， X 为向前删除一个字符(相当于 [backspace] 亦即是退格键) (常用) |
| nx | n 为数字，连续向后删除 n 个字符。举例来说，我要连续删除 10 个字符， 『10x』。 |
| dd | 删除游标所在的那一整行(常用) |
| ndd | n 为数字。删除光标所在的向下 n 行，例如 20dd 则是删除 20 行 (常用) |
| d1G | 删除光标所在到第一行的所有数据 |
| dG | 删除光标所在到最后一行的所有数据 |
| d$ | 删除游标所在处，到该行的最后一个字符 |
| d0 | 那个是数字的 0 ，删除游标所在处，到该行的最前面一个字符 |
| yy | 复制游标所在的那一行(常用) |
| nyy | n 为数字。复制光标所在的向下 n 行，例如 20yy 则是复制 20 行(常用) |
| y1G | 复制游标所在行到第一行的所有数据 |
| yG | 复制游标所在行到最后一行的所有数据 |
| y0 | 复制光标所在的那个字符到该行行首的所有数据 |
| y$ | 复制光标所在的那个字符到该行行尾的所有数据 |
| p, P | p 为将已复制的数据在光标下一行贴上，P 则为贴在游标上一行！ 举例来说，我目前光标在第 20 行，且已经复制了 10 行数据。则按下 p 后， 那 10 行数据会贴在原本的 20 行之后，亦即由 21 行开始贴。但如果是按下 P 呢？ 那么原本的第 20 行会被推到变成 30 行。 (常用) |
| J | 将光标所在行与下一行的数据结合成同一行 |
| c | 重复删除多个数据，例如向下删除 10 行，[ 10cj ] |
| u | 复原前一个动作。(常用) |
| [Ctrl]+r | 重做上一个动作。(常用) |
| 这个 u 与 [Ctrl]+r 是很常用的指令！一个是复原，另一个则是重做一次～ 利用这两个功能按键，你的编辑，嘿嘿！很快乐的啦！ | |
| . | 不要怀疑！这就是小数点！意思是重复前一个动作的意思。 如果你想要重复删除、重复贴上等等动作，按下小数点『.』就好了！ (常用) |

**第二部份：一般模式切换到编辑模式的可用的按钮说明**

|  |  |
| --- | --- |
| 进入输入或取代的编辑模式 | |
| i, I | 进入输入模式(Insert mode)： i 为『从目前光标所在处输入』， I 为『在目前所在行的第一个非空格符处开始输入』。 (常用) |
| a, A | 进入输入模式(Insert mode)： a 为『从目前光标所在的下一个字符处开始输入』， A 为『从光标所在行的最后一个字符处开始输入』。(常用) |
| o, O | 进入输入模式(Insert mode)： 这是英文字母 o 的大小写。o 为『在目前光标所在的下一行处输入新的一行』； O 为在目前光标所在处的上一行输入新的一行！(常用) |
| r, R | 进入取代模式(Replace mode)： r 只会取代光标所在的那一个字符一次；R会一直取代光标所在的文字，直到按下 ESC 为止；(常用) |
| 上面这些按键中，在 vi 画面的左下角处会出现『--INSERT--』或『--REPLACE--』的字样。 由名称就知道该动作了吧！！特别注意的是，我们上面也提过了，你想要在档案里面输入字符时， 一定要在左下角处看到 INSERT 或 REPLACE 才能输入喔！ | |
| [Esc] | 退出编辑模式，回到一般模式中(常用) |

**第三部份：一般模式切换到指令行模式的可用的按钮说明**

|  |  |
| --- | --- |
| 指令行的储存、离开等指令 | |
| :w | 将编辑的数据写入硬盘档案中(常用) |
| :w! | 若文件属性为『只读』时，强制写入该档案。不过，到底能不能写入， 还是跟你对该档案的档案权限有关啊！ |
| :q | 离开 vi (常用) |
| :q! | 若曾修改过档案，又不想储存，使用 ! 为强制离开不储存档案。 |
| 注意一下啊，那个惊叹号 (!) 在 vi 当中，常常具有『强制』的意思～ | |
| :wq | 储存后离开，若为 :wq! 则为强制储存后离开 (常用) |
| ZZ | 这是大写的 Z 喔！若档案没有更动，则不储存离开，若档案已经被更动过，则储存后离开！ |
| :w [filename] | 将编辑的数据储存成另一个档案（类似另存新档） |
| :r [filename] | 在编辑的数据中，读入另一个档案的数据。亦即将 『filename』 这个档案内容加到游标所在行后面 |
| :n1,n2 w [filename] | 将 n1 到 n2 的内容储存成 filename 这个档案。 |
| :! command | 暂时离开 vi 到指令行模式下执行 command 的显示结果！例如 『:! ls /home』即可在 vi 当中察看 /home 底下以 ls 输出的档案信息！ |
| vim 环境的变更 | |
| :set nu | 显示行号，设定之后，会在每一行的前缀显示该行的行号 |
| :set nonu | 与 set nu 相反，为取消行号！ |

## 十、linux yum 命令

### 1 yum（ Yellow dog Updater, Modified）

一个在Fedora和RedHat以及SUSE中的Shell前端软件包管理器。基於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。

### 2 yum 语法

yum [options] [command] [package ...]

options：可选，选项包括-h（帮助），-y（当安装过程提示选择全部为"yes"），-q（不显示安装的过程）等等。

command：要进行的操作。

package操作的对象。

### 3 yum常用命令

1.列出所有可更新的软件清单命令：yum check-update

2.更新所有软件命令：yum update

3.仅安装指定的软件命令：yum install <package\_name>

4.仅更新指定的软件命令：yum update <package\_name>

5.列出所有可安裝的软件清单命令：yum list

6.删除软件包命令：yum remove <package\_name>

7.查找软件包 命令：yum search <keyword>

8.清除缓存命令:

yum clean packages: 清除缓存目录下的软件包

yum clean headers: 清除缓存目录下的 headers

yum clean oldheaders: 清除缓存目录下旧的 headers

yum clean, yum clean all (= yum clean packages; yum clean oldheaders) :清除缓存目录下的软件包及旧的headers

### 4 实例 1

安装 pam-devel

[root@www ~]# yum install pam-devel

Setting up Install Process

Parsing package install arguments

Resolving Dependencies <==先检查软件的属性相依问题

--> Running transaction check

---> Package pam-devel.i386 0:0.99.6.2-4.el5 set to be updated

--> Processing Dependency: pam = 0.99.6.2-4.el5 for package: pam-devel

--> Running transaction check

---> Package pam.i386 0:0.99.6.2-4.el5 set to be updated

filelists.xml.gz 100% |=========================| 1.6 MB 00:05

filelists.xml.gz 100% |=========================| 138 kB 00:00

-> Finished Dependency Resolution

……(省略)

### 5 实例 2

移除 pam-devel

[root@www ~]# yum remove pam-devel

Setting up Remove Process

Resolving Dependencies <==同样的，先解决属性相依的问题

--> Running transaction check

---> Package pam-devel.i386 0:0.99.6.2-4.el5 set to be erased

--> Finished Dependency Resolution

Dependencies Resolved

=============================================================================

Package Arch Version Repository Size

=============================================================================

Removing:

pam-devel i386 0.99.6.2-4.el5 installed 495 k

Transaction Summary

=============================================================================

Install 0 Package(s)

Update 0 Package(s)

Remove 1 Package(s) <==还好，并没有属性相依的问题，单纯移除一个软件

Is this ok [y/N]: y

Downloading Packages:

Running rpm\_check\_debug

Running Transaction Test

Finished Transaction Test

Transaction Test Succeeded

Running Transaction

Erasing : pam-devel ######################### [1/1]

Removed: pam-devel.i386 0:0.99.6.2-4.el5

Complete!

## 十一、Linux 命令大全

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **1、文件管理** | | | | |
| [cat](http://www.runoob.com/linux/linux-comm-cat.html) | [chattr](http://www.runoob.com/linux/linux-comm-chattr.html) | [chgrp](http://www.runoob.com/linux/linux-comm-chgrp.html) | [chmod](http://www.runoob.com/linux/linux-comm-chmod.html) |
| [chown](http://www.runoob.com/linux/linux-comm-chown.html) | [cksum](http://www.runoob.com/linux/linux-comm-cksum.html) | [cmp](http://www.runoob.com/linux/linux-comm-cmp.html) | [diff](http://www.runoob.com/linux/linux-comm-diff.html) |
| [diffstat](http://www.runoob.com/linux/linux-comm-diffstat.html) | [file](http://www.runoob.com/linux/linux-comm-file.html) | [find](http://www.runoob.com/linux/linux-comm-find.html) | [git](http://www.runoob.com/linux/linux-comm-git.html) |
| [gitview](http://www.runoob.com/linux/linux-comm-gitview.html) | [indent](http://www.runoob.com/linux/linux-comm-indent.html) | [cut](http://www.runoob.com/linux/linux-comm-cut.html) | [ln](http://www.runoob.com/linux/linux-comm-ln.html) |
| [less](http://www.runoob.com/linux/linux-comm-less.html) | [locate](http://www.runoob.com/linux/linux-comm-locate.html) | [lsattr](http://www.runoob.com/linux/linux-comm-lsattr.html) | [mattrib](http://www.runoob.com/linux/linux-comm-mattrib.html) |
| [mc](http://www.runoob.com/linux/linux-comm-mc.html) | [mdel](http://www.runoob.com/linux/linux-comm-mdel.html) | [mdir](http://www.runoob.com/linux/linux-comm-mdir.html) | [mktemp](http://www.runoob.com/linux/linux-comm-mktemp.html) |
| [more](http://www.runoob.com/linux/linux-comm-more.html) | [mmove](http://www.runoob.com/linux/linux-comm-mmove.html) | [mread](http://www.runoob.com/linux/linux-comm-mread.html) | [mren](http://www.runoob.com/linux/linux-comm-mren.html) |
| [mtools](http://www.runoob.com/linux/linux-comm-mtools.html) | [mtoolstest](http://www.runoob.com/linux/linux-comm-mtoolstest.html) | [mv](http://www.runoob.com/linux/linux-comm-mv.html) | [od](http://www.runoob.com/linux/linux-comm-od.html) |
| [paste](http://www.runoob.com/linux/linux-comm-paste.html) | [patch](http://www.runoob.com/linux/linux-comm-patch.html) | [rcp](http://www.runoob.com/linux/linux-comm-rcp.html) | [rm](http://www.runoob.com/linux/linux-comm-rm.html) |
| [slocate](http://www.runoob.com/linux/linux-comm-slocate.html) | [split](http://www.runoob.com/linux/linux-comm-split.html) | [tee](http://www.runoob.com/linux/linux-comm-tee.html) | [tmpwatch](http://www.runoob.com/linux/linux-comm-tmpwatch.html) |
| [touch](http://www.runoob.com/linux/linux-comm-touch.html) | [umask](http://www.runoob.com/linux/linux-comm-umask.html) | [which](http://www.runoob.com/linux/linux-comm-which.html) | [cp](http://www.runoob.com/linux/linux-comm-cp.html) |
| [whereis](http://www.runoob.com/linux/linux-comm-whereis.html) | [mcopy](http://www.runoob.com/linux/linux-comm-mcopy.html) | [mshowfat](http://www.runoob.com/linux/linux-comm-mshowfat.html) | [rhmask](http://www.runoob.com/linux/linux-comm-rhmask.html) |
| [scp](http://www.runoob.com/linux/linux-comm-scp.html) | [awk](http://www.runoob.com/linux/linux-comm-awk.html) | [read](http://www.runoob.com/linux/linux-comm-read.html) |  |
| **2、文档编辑** | | | | |
| [col](http://www.runoob.com/linux/linux-comm-col.html) | [colrm](http://www.runoob.com/linux/linux-comm-colrm.html) | [comm](http://www.runoob.com/linux/linux-comm-comm.html) | [csplit](http://www.runoob.com/linux/linux-comm-csplit.html) |
| [ed](http://www.runoob.com/linux/linux-comm-ed.html) | [egrep](http://www.runoob.com/linux/linux-comm-egrep.html) | [ex](http://www.runoob.com/linux/linux-comm-ex.html) | [fgrep](http://www.runoob.com/linux/linux-comm-fgrep.html) |
| [fmt](http://www.runoob.com/linux/linux-comm-fmt.html) | [fold](http://www.runoob.com/linux/linux-comm-fold.html) | [grep](http://www.runoob.com/linux/linux-comm-grep.html) | [ispell](http://www.runoob.com/linux/linux-comm-ispell.html) |
| [jed](http://www.runoob.com/linux/linux-comm-jed.html) | [joe](http://www.runoob.com/linux/linux-comm-joe.html) | [join](http://www.runoob.com/linux/linux-comm-join.html) | [look](http://www.runoob.com/linux/linux-comm-look.html) |
| [mtype](http://www.runoob.com/linux/linux-comm-mtype.html) | [pico](http://www.runoob.com/linux/linux-comm-pico.html) | [rgrep](http://www.runoob.com/linux/linux-comm-rgrep.html) | [sed](http://www.runoob.com/linux/linux-comm-sed.html) |
| [sort](http://www.runoob.com/linux/linux-comm-sort.html) | [spell](http://www.runoob.com/linux/linux-comm-spell.html) | [tr](http://www.runoob.com/linux/linux-comm-tr.html) | [expr](http://www.runoob.com/linux/linux-comm-expr.html) |
| [uniq](http://www.runoob.com/linux/linux-comm-uniq.html) | [wc](http://www.runoob.com/linux/linux-comm-wc.html) | [let](http://www.runoob.com/linux/linux-comm-let.html) |  |
| **3、文件传输** | | | | |
| [lprm](http://www.runoob.com/linux/linux-comm-lprm.html) | [lpr](http://www.runoob.com/linux/linux-comm-lpr.html) | [lpq](http://www.runoob.com/linux/linux-comm-lpq.html) | [lpd](http://www.runoob.com/linux/linux-comm-lpd.html) |
| [bye](http://www.runoob.com/linux/linux-comm-bye.html) | [ftp](http://www.runoob.com/linux/linux-comm-ftp.html) | [uuto](http://www.runoob.com/linux/linux-comm-uuto.html) | [uupick](http://www.runoob.com/linux/linux-comm-uupick.html) |
| [uucp](http://www.runoob.com/linux/linux-comm-uucp.html) | [uucico](http://www.runoob.com/linux/linux-comm-uucico.html) | [tftp](http://www.runoob.com/linux/linux-comm-tftp.html) | [ncftp](http://www.runoob.com/linux/linux-comm-ncftp.html) |
| [ftpshut](http://www.runoob.com/linux/linux-comm-ftpshut.html) | [ftpwho](http://www.runoob.com/linux/linux-comm-ftpwho.html) | [ftpcount](http://www.runoob.com/linux/linux-comm-ftpcount.html) |  |
| **4、磁盘管理** | | | | |
| [cd](http://www.runoob.com/linux/linux-comm-cd.html) | [df](http://www.runoob.com/linux/linux-comm-df.html) | [dirs](http://www.runoob.com/linux/linux-comm-dirs.html) | [du](http://www.runoob.com/linux/linux-comm-du.html) |
| [edquota](http://www.runoob.com/linux/linux-comm-edquota.html) | [eject](http://www.runoob.com/linux/linux-comm-eject.html) | [mcd](http://www.runoob.com/linux/linux-comm-mcd.html) | [mdeltree](http://www.runoob.com/linux/linux-comm-mdeltree.html) |
| [mdu](http://www.runoob.com/linux/linux-comm-mdu.html) | [mkdir](http://www.runoob.com/linux/linux-comm-mkdir.html) | [mlabel](http://www.runoob.com/linux/linux-comm-mlabel.html) | [mmd](http://www.runoob.com/linux/linux-comm-mmd.html) |
| [mrd](http://www.runoob.com/linux/linux-comm-mrd.html) | [mzip](http://www.runoob.com/linux/linux-comm-mzip.html) | [pwd](http://www.runoob.com/linux/linux-comm-pwd.html) | [quota](http://www.runoob.com/linux/linux-comm-quota.html) |
| [mount](http://www.runoob.com/linux/linux-comm-mount.html) | [mmount](http://www.runoob.com/linux/linux-comm-mmount.html) | [rmdir](http://www.runoob.com/linux/linux-comm-rmdir.html) | [rmt](http://www.runoob.com/linux/linux-comm-rmt.html) |
| [stat](http://www.runoob.com/linux/linux-comm-stat.html) | [tree](http://www.runoob.com/linux/linux-comm-tree.html) | [umount](http://www.runoob.com/linux/linux-comm-umount.html) | [ls](http://www.runoob.com/linux/linux-comm-ls.html) |
| [quotacheck](http://www.runoob.com/linux/linux-comm-quotacheck.html) | [quotaoff](http://www.runoob.com/linux/linux-comm-quotaoff.html) | [lndir](http://www.runoob.com/linux/linux-comm-lndir.html) | [repquota](http://www.runoob.com/linux/linux-comm-repquota.html) |
| [quotaon](http://www.runoob.com/linux/linux-comm-quotaon.html) |  |  |  |
| **5、磁盘维护** | | | | |
| [badblocks](http://www.runoob.com/linux/linux-comm-badblocks.html) | [cfdisk](http://www.runoob.com/linux/linux-comm-cfdisk.html) | [dd](http://www.runoob.com/linux/linux-comm-dd.html) | [e2fsck](http://www.runoob.com/linux/linux-comm-e2fsck.html) |
| [ext2ed](http://www.runoob.com/linux/linux-comm-ext2ed.html) | [fsck](http://www.runoob.com/linux/linux-comm-fsck.html) | [fsck.minix](http://www.runoob.com/linux/linux-comm-fsck-minix.html) | [fsconf](http://www.runoob.com/linux/linux-comm-fsconf.html) |
| [fdformat](http://www.runoob.com/linux/linux-comm-fdformat.html) | [hdparm](http://www.runoob.com/linux/linux-comm-hdparm.html) | [mformat](http://www.runoob.com/linux/linux-comm-mformat.html) | [mkbootdisk](http://www.runoob.com/linux/linux-comm-mkbootdisk.html) |
| [mkdosfs](http://www.runoob.com/linux/linux-comm-mkdosfs.html) | [mke2fs](http://www.runoob.com/linux/linux-comm-mke2fs.html) | [mkfs.ext2](http://www.runoob.com/linux/linux-comm-mkfs-ext2.html) | [mkfs.msdos](http://www.runoob.com/linux/linux-comm-mkfs-msdos.html) |
| [mkinitrd](http://www.runoob.com/linux/linux-comm-mkinitrd.html) | [mkisofs](http://www.runoob.com/linux/linux-comm-mkisofs.html) | [mkswap](http://www.runoob.com/linux/linux-comm-mkswap.html) | [mpartition](http://www.runoob.com/linux/linux-comm-mpartition.html) |
| [swapon](http://www.runoob.com/linux/linux-comm-swapon.html) | [symlinks](http://www.runoob.com/linux/linux-comm-symlinks.html) | [sync](http://www.runoob.com/linux/linux-comm-sync.html) | [mbadblocks](http://www.runoob.com/linux/linux-comm-mbadblocks.html) |
| [mkfs.minix](http://www.runoob.com/linux/linux-comm-mkfs-minix.html) | [fsck.ext2](http://www.runoob.com/linux/linux-comm-fsck-ext2.html) | [fdisk](http://www.runoob.com/linux/linux-comm-fdisk.html) | [losetup](http://www.runoob.com/linux/linux-comm-losetup.html) |
| [mkfs](http://www.runoob.com/linux/linux-comm-mkfs.html) | [sfdisk](http://www.runoob.com/linux/linux-comm-sfdisk.html) | [swapoff](http://www.runoob.com/linux/linux-comm-swapoff.html) |  |
| **6、网络通讯** | | | | |
| [apachectl](http://www.runoob.com/linux/linux-comm-apachectl.html) | [arpwatch](http://www.runoob.com/linux/linux-comm-arpwatch.html) | [dip](http://www.runoob.com/linux/linux-comm-dip.html) | [getty](http://www.runoob.com/linux/linux-comm-getty.html) |
| [mingetty](http://www.runoob.com/linux/linux-comm-mingetty.html) | [uux](http://www.runoob.com/linux/linux-comm-uux.html) | [telnet](http://www.runoob.com/linux/linux-comm-telnet.html) | [uulog](http://www.runoob.com/linux/linux-comm-uulog.html) |
| [uustat](http://www.runoob.com/linux/linux-comm-uustat.html) | [ppp-off](http://www.runoob.com/linux/linux-comm-ppp-off.html) | [netconfig](http://www.runoob.com/linux/linux-comm-netconfig.html) | [nc](http://www.runoob.com/linux/linux-comm-nc.html) |
| [httpd](http://www.runoob.com/linux/linux-comm-httpd.html) | [ifconfig](http://www.runoob.com/linux/linux-comm-ifconfig.html) | [minicom](http://www.runoob.com/linux/linux-comm-minicom.html) | [mesg](http://www.runoob.com/linux/linux-comm-mesg.html) |
| [dnsconf](http://www.runoob.com/linux/linux-comm-dnsconf.html) | [wall](http://www.runoob.com/linux/linux-comm-wall.html) | [netstat](http://www.runoob.com/linux/linux-comm-netstat.html) | [ping](http://www.runoob.com/linux/linux-comm-ping.html) |
| [pppstats](http://www.runoob.com/linux/linux-comm-pppstats.html) | [samba](http://www.runoob.com/linux/linux-comm-samba.html) | [setserial](http://www.runoob.com/linux/linux-comm-setserial.html) | [talk](http://www.runoob.com/linux/linux-comm-talk.html) |
| [traceroute](http://www.runoob.com/linux/linux-comm-traceroute.html) | [tty](http://www.runoob.com/linux/linux-comm-tty.html) | [newaliases](http://www.runoob.com/linux/linux-comm-newaliases.html) | [uuname](http://www.runoob.com/linux/linux-comm-uuname.html) |
| [netconf](http://www.runoob.com/linux/linux-comm-netconf.html) | [write](http://www.runoob.com/linux/linux-comm-write.html) | [statserial](http://www.runoob.com/linux/linux-comm-statserial.html) | [efax](http://www.runoob.com/linux/linux-comm-efax.html) |
| [pppsetup](http://www.runoob.com/linux/linux-comm-pppsetup.html) | [tcpdump](http://www.runoob.com/linux/linux-comm-tcpdump.html) | [ytalk](http://www.runoob.com/linux/linux-comm-ytalk.html) | [cu](http://www.runoob.com/linux/linux-comm-cu.html) |
| [smbd](http://www.runoob.com/linux/linux-comm-smbd.html) | [testparm](http://www.runoob.com/linux/linux-comm-testparm.html) | [smbclient](http://www.runoob.com/linux/linux-comm-smbclient.html) | [shapecfg](http://www.runoob.com/linux/linux-comm-shapecfg.html) |
| **7、系统管理** | | | | |
| [adduser](http://www.runoob.com/linux/linux-comm-adduser.html) | [chfn](http://www.runoob.com/linux/linux-comm-chfn.html) | [useradd](http://www.runoob.com/linux/linux-comm-useradd.html) | [date](http://www.runoob.com/linux/linux-comm-date.html) |
| [exit](http://www.runoob.com/linux/linux-comm-exit.html) | [finger](http://www.runoob.com/linux/linux-comm-finger.html) | [fwhios](http://www.runoob.com/linux/linux-comm-fwhios.html) | [sleep](http://www.runoob.com/linux/linux-comm-sleep.html) |
| [suspend](http://www.runoob.com/linux/linux-comm-suspend.html) | [groupdel](http://www.runoob.com/linux/linux-comm-groupdel.html) | [groupmod](http://www.runoob.com/linux/linux-comm-groupmod.html) | [halt](http://www.runoob.com/linux/linux-comm-halt.html) |
| [kill](http://www.runoob.com/linux/linux-comm-kill.html) | [last](http://www.runoob.com/linux/linux-comm-last.html) | [lastb](http://www.runoob.com/linux/linux-comm-lastb.html) | [login](http://www.runoob.com/linux/linux-comm-login.html) |
| [logname](http://www.runoob.com/linux/linux-comm-logname.html) | [logout](http://www.runoob.com/linux/linux-comm-logout.html) | [ps](http://www.runoob.com/linux/linux-comm-ps.html) | [nice](http://www.runoob.com/linux/linux-comm-nice.html) |
| [procinfo](http://www.runoob.com/linux/linux-comm-procinfo.html) | [top](http://www.runoob.com/linux/linux-comm-top.html) | [pstree](http://www.runoob.com/linux/linux-comm-pstree.html) | [reboot](http://www.runoob.com/linux/linux-comm-reboot.html) |
| [rlogin](http://www.runoob.com/linux/linux-comm-rlogin.html) | [rsh](http://www.runoob.com/linux/linux-comm-rsh.html) | [sliplogin](http://www.runoob.com/linux/linux-comm-sliplogin.html) | [screen](http://www.runoob.com/linux/linux-comm-screen.html) |
| [shutdown](http://www.runoob.com/linux/linux-comm-shutdown.html) | [rwho](http://www.runoob.com/linux/linux-comm-rwho.html) | [sudo](http://www.runoob.com/linux/linux-comm-sudo.html) | [gitps](http://www.runoob.com/linux/linux-comm-gitps.html) |
| [swatch](http://www.runoob.com/linux/linux-comm-swatch.html) | [tload](http://www.runoob.com/linux/linux-comm-tload.html) | [logrotate](http://www.runoob.com/linux/linux-comm-logrotate.html) | [uname](http://www.runoob.com/linux/linux-comm-uname.html) |
| [chsh](http://www.runoob.com/linux/linux-comm-chsh.html) | [userconf](http://www.runoob.com/linux/linux-comm-userconf.html) | [userdel](http://www.runoob.com/linux/linux-comm-userdel.html) | [usermod](http://www.runoob.com/linux/linux-comm-usermod.html) |
| [vlock](http://www.runoob.com/linux/linux-comm-vlock.html) | [who](http://www.runoob.com/linux/linux-comm-who.html) | [whoami](http://www.runoob.com/linux/linux-comm-whoami.html) | [whois](http://www.runoob.com/linux/linux-comm-whois.html) |
| [newgrp](http://www.runoob.com/linux/linux-comm-newgrp.html) | [renice](http://www.runoob.com/linux/linux-comm-renice.html) | [su](http://www.runoob.com/linux/linux-comm-su.html) | [skill](http://www.runoob.com/linux/linux-comm-skill.html) |
| [w](http://www.runoob.com/linux/linux-comm-w.html) | [id](http://www.runoob.com/linux/linux-comm-id.html) | [free](http://www.runoob.com/linux/linux-comm-free.html) |  |
| **8、系统设置** | | | | |
| [reset](http://www.runoob.com/linux/linux-comm-reset.html) | [clear](http://www.runoob.com/linux/linux-comm-clear.html) | [alias](http://www.runoob.com/linux/linux-comm-alias.html) | [dircolors](http://www.runoob.com/linux/linux-comm-dircolors.html) |
| [aumix](http://www.runoob.com/linux/linux-comm-aumix.html) | [bind](http://www.runoob.com/linux/linux-comm-bind.html) | [chroot](http://www.runoob.com/linux/linux-comm-chroot.html) | [clock](http://www.runoob.com/linux/linux-comm-clock.html) |
| [crontab](http://www.runoob.com/linux/linux-comm-crontab.html) | [declare](http://www.runoob.com/linux/linux-comm-declare.html) | [depmod](http://www.runoob.com/linux/linux-comm-depmod.html) | [dmesg](http://www.runoob.com/linux/linux-comm-dmesg.html) |
| [enable](http://www.runoob.com/linux/linux-comm-enable.html) | [eval](http://www.runoob.com/linux/linux-comm-eval.html) | [export](http://www.runoob.com/linux/linux-comm-export.html) | [pwunconv](http://www.runoob.com/linux/linux-comm-pwunconv.html) |
| [grpconv](http://www.runoob.com/linux/linux-comm-grpconv.html) | [rpm](http://www.runoob.com/linux/linux-comm-rpm.html) | [insmod](http://www.runoob.com/linux/linux-comm-insmod.html) | [kbdconfig](http://www.runoob.com/linux/linux-comm-kbdconfig.html) |
| [lilo](http://www.runoob.com/linux/linux-comm-lilo.html) | [liloconfig](http://www.runoob.com/linux/linux-comm-liloconfig.html) | [lsmod](http://www.runoob.com/linux/linux-comm-lsmod.html) | [minfo](http://www.runoob.com/linux/linux-comm-minfo.html) |
| [set](http://www.runoob.com/linux/linux-comm-set.html) | [modprobe](http://www.runoob.com/linux/linux-comm-modprobe.html) | [ntsysv](http://www.runoob.com/linux/linux-comm-ntsysv.html) | [mouseconfig](http://www.runoob.com/linux/linux-comm-mouseconfig.html) |
| [passwd](http://www.runoob.com/linux/linux-comm-passwd.html) | [pwconv](http://www.runoob.com/linux/linux-comm-pwconv.html) | [rdate](http://www.runoob.com/linux/linux-comm-rdate.html) | [resize](http://www.runoob.com/linux/linux-comm-resize.html) |
| [rmmod](http://www.runoob.com/linux/linux-comm-rmmod.html) | [grpunconv](http://www.runoob.com/linux/linux-comm-grpunconv.html) | [modinfo](http://www.runoob.com/linux/linux-comm-modinfo.html) | [time](http://www.runoob.com/linux/linux-comm-time.html) |
| [setup](http://www.runoob.com/linux/linux-comm-setup.html) | [sndconfig](http://www.runoob.com/linux/linux-comm-sndconfig.html) | [setenv](http://www.runoob.com/linux/linux-comm-setenv.html) | [setconsole](http://www.runoob.com/linux/linux-comm-setconsole.html) |
| [timeconfig](http://www.runoob.com/linux/linux-comm-timeconfig.html) | [ulimit](http://www.runoob.com/linux/linux-comm-ulimit.html) | [unset](http://www.runoob.com/linux/linux-comm-unset.html) | [chkconfig](http://www.runoob.com/linux/linux-comm-chkconfig.html) |
| [apmd](http://www.runoob.com/linux/linux-comm-apmd.html) | [hwclock](http://www.runoob.com/linux/linux-comm-hwclock.html) | [mkkickstart](http://www.runoob.com/linux/linux-comm-mkkickstart.html) | [fbset](http://www.runoob.com/linux/linux-comm-fbset.html) |
| [unalias](http://www.runoob.com/linux/linux-comm-unalias.html) | [SVGATextMode](http://www.runoob.com/linux/linux-comm-svgatextmode.html) |  |  |
| 9**、备份压缩** | | | | |
| [ar](http://www.runoob.com/linux/linux-comm-ar.html) | [bunzip2](http://www.runoob.com/linux/linux-comm-bunzip2.html) | [bzip2](http://www.runoob.com/linux/linux-comm-bzip2.html) | [bzip2recover](http://www.runoob.com/linux/linux-comm-bzip2recover.html) |
| [gunzip](http://www.runoob.com/linux/linux-comm-gunzip.html) | [unarj](http://www.runoob.com/linux/linux-comm-unarj.html) | [compress](http://www.runoob.com/linux/linux-comm-compress.html) | [cpio](http://www.runoob.com/linux/linux-comm-cpio.html) |
| [dump](http://www.runoob.com/linux/linux-comm-dump.html) | [uuencode](http://www.runoob.com/linux/linux-comm-uuencode.html) | [gzexe](http://www.runoob.com/linux/linux-comm-gzexe.html) | [gzip](http://www.runoob.com/linux/linux-comm-gzip.html) |
| [lha](http://www.runoob.com/linux/linux-comm-lha.html) | [restore](http://www.runoob.com/linux/linux-comm-restore.html) | [tar](http://www.runoob.com/linux/linux-comm-tar.html) | [uudecode](http://www.runoob.com/linux/linux-comm-uudecode.html) |
| [unzip](http://www.runoob.com/linux/linux-comm-unzip.html) | [zip](http://www.runoob.com/linux/linux-comm-zip.html) | [zipinfo](http://www.runoob.com/linux/linux-comm-zipinfo.html) |  |
| **10、设备管理** | | | | |
| [setleds](http://www.runoob.com/linux/linux-comm-setleds.html) | [loadkeys](http://www.runoob.com/linux/linux-comm-loadkeys.html) | [rdev](http://www.runoob.com/linux/linux-comm-rdev.html) | [dumpkeys](http://www.runoob.com/linux/linux-comm-dumpkeys.html) |
| [MAKEDEV](http://www.runoob.com/linux/linux-comm-makedev.html) |  |  |  |

**11 常用命令简单介绍**

**（1）常用命令**

ls　 显示文件或目录

ls -l 列出文件详细信息l(list)

ls -a 列出当前目录下所有文件及目录，包括隐藏的a(all)

mkdir test1; 简单创建

mkdir -m 777 test2; 授予777权限

mkdir -v test3; 创建并且显示

mkdir -p -v aaa/bbb/ccc; 创建并且显示，父目录自动生成

rm demo.java 直接删除文件

rm -R -v aaa 递归删除，删除完成显示

rmdir -v czj 删除空目录，删除完成显示

rmdir -v -p aaa/bbb/ccc/ 递归删除空目录，注意删除的顺序，由子目录到父目录

cd /opt/huawei/hnas\_uoms/uoms/bin 进入目录

cd .. 上级目录

cd ../.. 上两级目录

touch -m hs\_err\_pid2874.log 只更新修改时间

touch demo.c 创建文件

echo 666 输出666换行

echo -n 666 输出666不换行

cat uomsDeploy.sh 查看整个文件内容

cat -n textfile1 > textfile2 把 textfile1 的档案内容加上行号后输入 textfile2 这个档案里

cp file1 file2 将文件file1复制成文件file2

cp -f file1 file2 强制复制的模式

cp -R file1 file2 将目录dir1复制成目录dir2

mv test.log test1.txt 将文件test.log改名为test1.txt

mv test1.txt test3 将文件test1.txt移到test3

mv -f log3.txt log2.txt 将文件file1改名为file2，即使file2存在，也是直接覆盖掉

mv -i log3.txt log2.txt 将文件file1改名为file2，file2存在，询问

mv dir1 dir2 目录的移动

find ./ -size -10G 文件size小于10G的文件或目录

find ./ -size +10K 文件size大于10M的文件或目录

find ./ -nogroup 查找文件的组ID不存在的文件

find ./ -group wade 查找组名为wade的文件或目录

find ./ -user czj 所有者为czj的文件或目录

find ./ -perm -o=r 其它用户权限有读权限的目录或文件

find ./ -perm -g=w 用户组权限有写权限的目录或文件

find ./ -perm -u=e 用户权限有执行权限的目录或文件

ind ./ -perm 664 权限为644的文件或目录

find ./ -empty -type f -print -delete 查找空文件并删除

find ./ -mmin -2 查找文件更新日时在距现在时刻二分以内的文件

find ./ -name \\*.zip 查找文件名匹配\*.zip的文件

find ./ -name test 查找文件名为test的文件

wc testfile testfile\_1 testfile\_2 #统计三个文件的信息 行数、字数、字符数

grep ‘test’ d\* 显示所有以d开头的文件中包含 test的行

grep ‘[a-z]\{5\}’ aa 显示aa所有包含每个字符串至少有5个连续小写字符的字符串的行

pwd 显示当前目录

clear 清除命令

more uomsDeploy.sh 分页显示uomsDeploy.sh；enter向下一行；Ctrl+f向下滚动一屏；Ctrl+b返回上一屏

less uomsDeploy.sh 分页显示uomsDeploy.sh；enter向下一行；空格向下滚动一屏；g第一行；G最后一行。

head -5 uomsDeploy.sh 显示开始5行

tail -10 /etc/passwd 读取最后10行

tail -f /var/log/messages 读取最后几行，不断读取新的内容，实时监控的作用

ctrl+c 退出

全文行号高亮： grep timerLockMapper -n --color run.log

监控最后行号高亮（只显示）：tail -f -n 1000 run.log |grep timerLockMapper -n --color

监控最后行号高亮（全部显示）：tail -f run.log | perl -pe 's/(request)/\e[1;31m$1\e[0m/g'

**（2）系统管理**

who 显示在线登陆用户

whoami 显示当前操作用户

hostname 显示主机名

uname 显示系统信息

top 动态显示当前耗费资源最多进程信息

ps 显示瞬间进程状态 ps -aux

ping www.baidu.com 测试网络连通

netstat 显示网络状态信息

clear 清屏

kill -9 pid 强制终止,杀死进程，可以先用ps 或 top命令查看进程的id，然后再用kill命令杀死进程。

shutdown -r 关机重启

shutdown -h 关机不重启

shutdown now 立刻关机

halt 关机

reboot 重启

**（3）打包压缩**

tar: 打包压缩

-c 归档文件

-x 压缩文件

-z gzip压缩文件

-j bzip2压缩文件

-v 显示压缩或解压缩过程 v(view)

-f 使用档名

tar -cvf /home/abc.tar /home/abc 只打包，不压缩

tar -zcvf /home/abc.tar.gz /home/abc 打包，并用gzip压缩

tar -jcvf /home/abc.tar.bz2 /home/abc 打包，并用bzip2压缩

tar –cvf mysql-5.0.tar mysql-----把整个目录mysql中文件打包到mysql-5.0.tar

tar – rf mysql-5.0.tar mysql.conf ------将文件mysql.conf增加到包mysql-5.0.tar

tar –uf mysql-5.0.tar mysql.conf --------用文件mysql.conf更新包中的文件mysql.conf

tar –xvf mysql-5.0.tar -------打开包mysql-5.0.tar中的文件到当前目录

**（4）vim三种模式：命令模式(Esc)、插入模式(i)、底行模式 (ctrl+c 或者 :)**

命令模式下：

:q 退出

:q! 强制退出

:wq 保存并退出

**（5）用户及权限**

/etc/passwd 存储用户账号

/etc/group 存储组账号

/etc/shadow 存储用户账号的密码

/etc/gshadow 存储用户组账号的密码

useradd 用户名

userdel 用户名

adduser 用户名

groupadd 组名

groupdel 组名

passwd root 给root设置密码

su root

su - root

/etc/profile 系统环境变量

bash\_profile 用户环境变量

.bashrc 用户环境变量

su user 切换用户，加载配置文件.bashrc

su - user 切换用户，加载配置文件/etc/profile ，加载bash\_profile

chgrp : 改变档案所属群组

chown : 改变档案拥有者; 将install.log的拥有者改为bin这个账号： chown bin install.log

chmod : 改变档案的权限, SUID, SGID, SBIT等等的特性

Linux档案的基本权限就有九个，分别是owner/group/others三种身份各有自己的read/write/execute权限

举例：档案的权限字符为 -rwxrwxrwx 这九个权限是三个三个一组的！其中，我们可以使用数字来代表各个权限，各权限的分数对照表如下：

r:4 　　w:2　　　x:1

每种身份(owner/group/others)各自的三个权限(r/w/x)分数是需要累加的，例如当权限为： [-rwxrwx---] 分数则是：

owner = rwx = 4+2+1 = 7

group = rwx = 4+2+1 = 7

others= --- = 0+0+0 = 0

所以我们设定权限的变更时，该档案的权限数字就是770啦！变更权限的指令chmod的语法是这样的：

u User，即文件或目录的拥有者；

g Group，即文件或目录的所属群组；

o Other，除了文件或目录拥有者或所属群组之外，其他用户皆属于这个范围；

a All，即全部的用户，包含拥有者，所属群组以及其他用户；

r 读取权限，数字代号为“4”;

w 写入权限，数字代号为“2”；

x 执行或切换权限，数字代号为“1”；

- 不具任何权限，数字代号为“0”；

chmod [who] [+ | - | =] [mode] 文件名

chmod 777 filename

chmod -R 777 /home/mypackage

$ chmod u+x file 给file的属主增加执行权限

$ chmod 751 file 给file的属主分配读、写、执行(7)的权限，给file的所在组分配读、执行(5)的权限，给其他用户分配执行(1)的权限

$ chmod u=rwx,g=rx,o=x file 上例的另一种形式

$ chmod =r file 为所有用户分配读权限

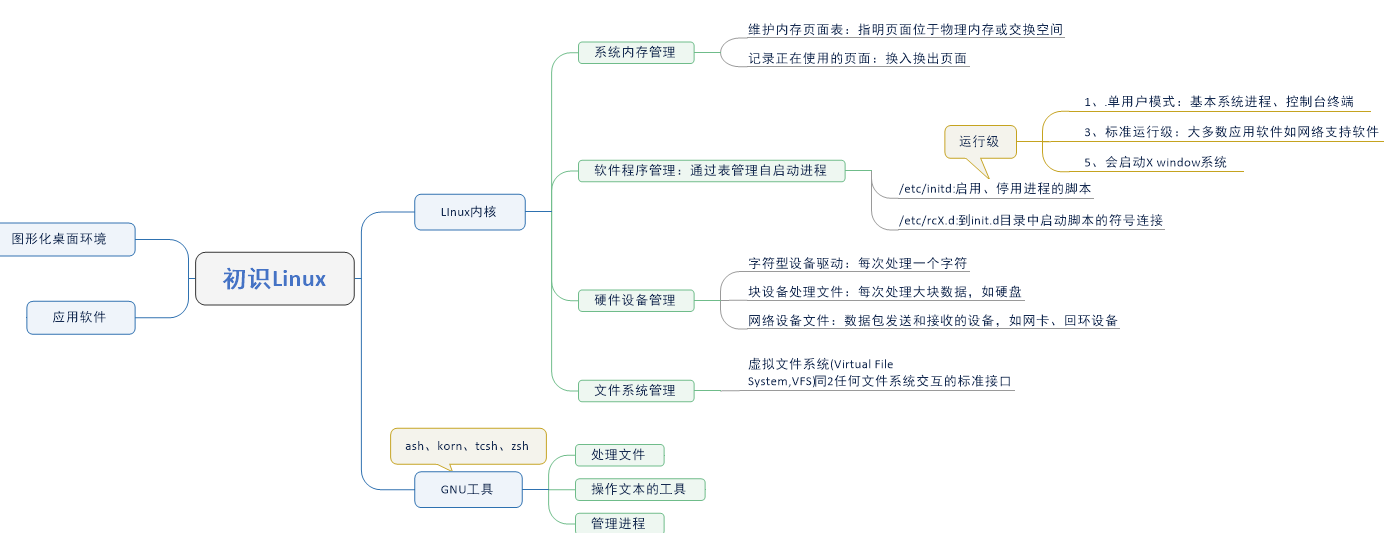
$ chmod 444 file 同上例

$ chmod a-wx,a+r file 同上例

$ chmod -R u+r directory 递归地给directory目录下所有文件和子目录的属主分配读的权限

$ chmod 4755 设置用ID，给属主分配读、写和执行权限，给组和其他用户分配读、执行的权限

## 十二、Shell基础



**shell翻译成壳的意思，它是包裹在**[**linux内核**](https://www.baidu.com/s?wd=linux%E5%86%85%E6%A0%B8&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)**外层的，一个可通过一系列的linux命令对**[**操作系统**](https://www.baidu.com/s?wd=%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)**发出相关指令的人机界面。 shell可以通过其条件语句和循环语句等，把一系列linux命令结合在一起，形成一个相当于**[**面向过程**](https://www.baidu.com/s?wd=%E9%9D%A2%E5%90%91%E8%BF%87%E7%A8%8B&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)**的程序，shell script，来实现一些较为复杂的功能。总括，shell是linux命令集的概称，是属于命令行的人机界面。相当于SQL 的存储过程。**

准确上讲应该说是 shell 脚本文件编程（当然了，在该 shell 脚本文件中也可以包含 linux 系统命令）。shell 脚本文件是一个文本文件，它是将你需要完成的各种复杂功能通过编写 shell 程序来实现。例如：模式匹配（awk） 、字符串替换功能（sed）等。在 awk 和 sed 等工具的使用中肯定需要使用正则表达式的熟练使用，而 linux 命令是指 Linux 系统的基本操作命令，例如：ls 列出文件名命令、cat filename、cd 改变子目录命令等等。这就是它们之间的区别。

### 1 shell是一种弱类型、解释型语言，不需要编译，只需要一个解释器，这里我们用bash。

输入如下命令：

more /etc/passwd

1

可以看到：



**当前用户root登录的时候，默认打开的命令行用户接口就是bash，在这个bash中输入bash又会打开一个子bash接口，可以不断深入的嵌套bash，当然执行退出命令exit时，也会一层一层退出之前打开的bash。**

### 2 Shell含义

由于安全、复杂、繁琐等原因，用户不能直接接触内核（也没有必要），需要另外再开发一个程序，让用户直接使用这个程序；该程序的作用就是接收用户的操作（点击图标、输入命令），并进行简单的处理，然后再传递给内核，这样用户就能间接地使用操作系统内核了。你看，在用户和内核之间增加一层“代理”，既能简化用户的操作，又能保障内核的安全，何乐不为呢？**用户界面和命令行就是这个另外开发的程序，就是这层“代理”。在Linux下，这个命令行程序叫做 Shell。**Shell 是一个应用程序，它连接了用户和 Linux 内核，让用户能够更加高效、安全、低成本地使用 Linux 内核，这就是 Shell 的本质。

Shell 本身并不是内核的一部分，它只是站在内核的基础上编写的一个应用程序，它和 QQ、迅雷、Firefox 等其它软件没有什么区别。然而 Shell 也有着它的特殊性，就是开机立马启动，并呈现在用户面前；用户通过 Shell 来使用 Linux，不启动 Shell 的话，用户就没办法使用 Linux。

### 3 Shell 是如何连接用户和内核的？

Shell 能够接收用户输入的命令，并对命令进行处理，处理完毕后再将结果反馈给用户，比如输出到显示器、写入到文件等，这就是大部分读者对 Shell 的认知。你看，我一直都在使用 Shell，哪有使用内核哦？我也没有看到 Shell 将我和内核连接起来呀？！

其实，Shell 程序本身的功能是很弱的，比如文件操作、输入输出、进程管理等都得依赖内核。我们运行一个命令，大部分情况下 Shell 都会去调用内核暴露出来的接口，这就是在使用内核，只是这个过程被 Shell 隐藏了起来，它自己在背后默默进行，我们看不到而已。

接口其实就是一个一个的函数，使用内核就是调用这些函数。这就是使用内核的全部内容了吗？嗯，是的！除了函数，你没有别的途径使用内核。

比如，我们都知道在 Shell 中输入cat log.txt命令就可以查看 log.txt 文件中的内容，然而，log.txt 放在磁盘的哪个位置？分成了几个数据块？在哪里开始？在哪里终止？如何操作探头读取它？这些底层细节 Shell 统统不知道的，它只能去调用内核提供的 open() 和 read() 函数，告诉内核我要读取 log.txt 文件，请帮助我，然后内核就乖乖地按照 Shell 的吩咐去读取文件了，并将读取到的文件内容交给 Shell，最后再由 Shell 呈现给用户（其实呈现到显示器上还得依赖内核）。整个过程中 Shell 就是一个“中间商”，它在用户和内核之间“倒卖”数据，只是用户不知道罢了。

### 4 Shell 还能连接其它程序

在 Shell 中输入的命令，有一部分是 Shell 本身自带的，这叫做内置命令；有一部分是其它的应用程序（一个程序就是一个命令），这叫做外部命令。

Shell 本身支持的命令并不多，功能也有限，但是 Shell 可以调用其他的程序，每个程序就是一个命令，这使得 Shell 命令的数量可以无限扩展，其结果就是 Shell 的功能非常强大，完全能够胜任 Linux 的日常管理工作，如文本或字符串检索、文件的查找或创建、大规模软件的自动部署、更改系统设置、监控服务器性能、发送报警邮件、抓取网页内容、压缩文件等。更加惊讶的是，Shell 还可以让多个外部程序发生连接，在它们之间很方便地传递数据，也就是把一个程序的输出结果传递给另一个程序作为输入。

大家所说的 Shell 强大，并不是 Shell 本身功能丰富，而是它擅长使用和组织其他的程序。Shell 就是一个领导者，这正是 Shell 的魅力所在。

可以将 Shell 在整个 Linux 系统中的地位描述成下图所示的样子。注意“用户”和“其它应用程序”是通过虚线连接的，因为用户启动 Linux 后直接面对的是 Shell，通过 Shell 才能运行其它的应用程序。

Shell在整个Linux系统中的地位示意图

### 5 Shell 也支持编程

Shell 并不是简单的堆砌命令，我们还可以在 Shell 中编程，这和使用 C++、C#、Java、Python 等常见的编程语言并没有什么两样。

Shell 虽然没有 C++、Java、Python 等强大，但也支持了基本的编程元素，例如：

if...else 选择结构，case...in 开关语句，for、while、until 循环；

变量、数组、字符串、注释、加减乘除、逻辑运算等概念；

函数，包括用户自定义的函数和内置函数（例如 printf、export、eval 等）。

站在这个角度讲，Shell 也是一种编程语言，它的编译器（解释器）是 Shell 这个程序。我们平时所说的 Shell，有时候是指连接用户和内核的这个程序，有时候又是指 Shell 编程。

Shell 主要用来开发一些实用的、自动化的小工具，而不是用来开发具有复杂业务逻辑的中大型软件，例如检测计算机的硬件参数、搭建 Web 运行环境、日志分析等，Shell 都非常合适。使用 Shell 的熟练程度反映了用户对 Linux 的掌握程度，运维工程师、网络管理员、程序员都应该学习 Shell。尤其是 Linux 运维工程师，Shell 更是必不可少的，是必须掌握的技能，它使得我们能够自动化地管理服务器集群，否则你就得一个一个地登录所有的服务器，对每一台服务器都进行相同的设置，而这些服务器可能有成百上千之多，会浪费大量的时间在重复性的工作上。

### 6 Shell 是一种脚本语言

任何代码最终都要被“翻译”成二进制的形式才能在计算机中执行。

有的编程语言，如 C/C++、Pascal、Go语言、汇编等，必须在程序运行之前将所有代码都翻译成二进制形式，也就是生成可执行文件，用户拿到的是最终生成的可执行文件，看不到源码。

这个过程叫做编译（Compile），这样的编程语言叫做编译型语言，完成编译过程的软件叫做编译器（Compiler）。而有的编程语言，如 Shell、JavaScript、Python、PHP等，需要一边执行一边翻译，不会生成任何可执行文件，用户必须拿到源码才能运行程序。程序运行后会即时翻译，翻译完一部分执行一部分，不用等到所有代码都翻译完。

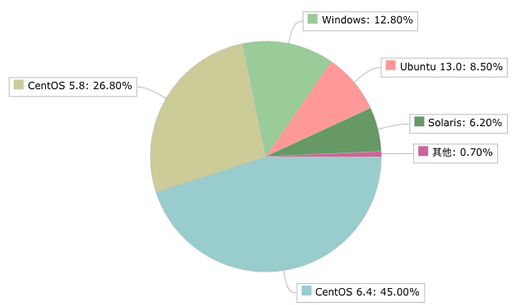
这个过程叫做解释，这样的编程语言叫做解释型语言或者脚本语言（Script），完成解释过程的软件叫做解释器。编译型语言的优点是执行速度快、对硬件要求低、保密性好，适合开发操作系统、大型应用程序、数据库等。脚本语言的优点是使用灵活、部署容易、跨平台性好，非常适合 Web 开发以及小工具的制作。Shell 就是一种脚本语言，我们编写完源码后不用编译，直接运行源码即可。

### 7 Shell是运维人员必须掌握的技能

Linux 运维人员就是负责 Linux 服务器的运行和维护。随着互联网的爆发，Linux 运维在最近几年也迎来了春天，出现了大量的职位需求，催生了一批 Linux 运维培训班。

如今的 IT 服务器领域是 Linux、UNIX、Windows 三分天下，Linux 系统可谓后起之秀，特别是“互联网热”以来，Linux 在服务器端的市场份额不断扩大，每年增长势头迅猛，开始对 Windows 和 UNIX 的地位构成严重威胁。

下图是 2016 年初国内服务器端各个操作系统的市场份额：



可以看出来，Linux 占 80% 左右（包括 CentOS、Ubuntu 等），Windows 占 12.8%，Solaris 占 6.2%。在未来的服务器领域，Linux 是大势所趋。

Linux 在服务器上的应用非常广泛，可以用来搭建 Web 服务器、数据库服务器、负载均衡服务器（CDN）、邮件服务器、DNS 服务器、反向代理服务器、VPN 服务器、路由器等。用 Linux 作为服务器系统不但非常高效和稳定，还不用担心版权问题，不用付费。

正是由于 Linux 服务器的大规模应用，才需要一批专业的人才去管理，这群人就是 Linux 运维工程师（OPS）。

OPS 的主要工作就是搭建起运行环境，让程序员写的代码能够高效、稳定、安全地在服务器上运行，他们属于后勤部门。OPS 的要求并不比程序员低，优秀的 OPS 拥有架设服务器集群的能力，还会编程开发常用的工具。

**（1）OPS 这项工作的细节内容包括：**

安装操作系统，例如 CentOS、Ubuntu 等。

部署代码运行环境，例如网站后台语言采用 PHP，就需要安装 Nginx、Apache、MySQL、PHP 运行时等。

及时修复漏洞，防止服务器被攻击，这包括 Linux 本身漏洞以及各个软件的漏洞。

根据项目需求升级软件，例如 PHP 7.0 在性能方面获得了重大突破，如果现在服务器压力比较大，就可以考虑将旧版的 PHP 5.x 升级到 PHP 7.0。

监控服务器压力，别让服务器宕机。例如淘宝双十一的时候就会瞬间涌入大量用户，导致部分服务器宕机，网页没法访问，甚至连支付宝都不能使用。

分析日志，及时发现代码或者环境的问题，通知相关人员修复。

这些任务只要登录远程服务器，或者去机房连接服务器（下图所示）就能够完成，为什么要用 Shell 编程呢？

图：OPS 在机房中用笔记本连接服务器

因为 OPS 面对的是成千上万台的服务器，不是十台八台，你总不能把同样的工作重复成千上万遍吧，那时估计黄花菜都凉了，市场也成一片红海了。

服务器一旦多了，这些人力工作都需要自动化起来，跑一段代码就能在成千上万台服务器上完成相同的工作，例如服务的监控、代码快速部署、服务启动停止、数据备份、日志分析等。

Shell 脚本很适合处理纯文本类型的数据，而 Linux 中几乎所有的配置文件、日志文件（如 NFS、Rsync、Httpd、Nginx、MySQL 等），以及绝大多数的启动文件都是纯文本类型的文件。

**（2）下面的手链形象地展示了 Shell 在运维工作中的地位：**

运维“手链”的组成：每颗“珍珠”都是一项服务，将珍珠穿起来的“线”就是 Shell。Shell 脚本是实现 Linux 系统自动管理以及自动化运维所必备的工具，Linux 的底层以及基础应用软件的核心大都涉及 Shell 脚本的内容。每一个合格的 Linux 系统管理员或运维工程师，都应该能够熟练的编写 Shell 脚本，只要这样才能提升运维人员的工作效率，减少不必要的重复劳动，为个人的职场发展奠定较好的基础。Shell、Python 和 Perl除了 Shell，能够用于 Linux 运维的脚本语言还有 Python 和 Perl。

**1) Perl 语言**

Perl 比 Shell 强大很多，在 2010 年以前很流行，它的语法灵活、复杂，在实现不同的功能时可以用多种不同的方式，缺点是不易读，团队协作困难。Perl 脚本已经成为历史了，现在的 Linux 运维人员几乎不需要了解 Perl 了，最多可以了解一下 Perl 的安装环境。

**2) Python 语言**

Python 是近几年非常流行的语言，它不但可以用于脚本程序开发，也可以实现 Web 程序开发（知乎、豆瓣、YouTube、Instagram 都是用 Python 开发），甚至还可以实现软件的开发（大名鼎鼎的 OpenStack、SaltStack 都是 Python 语言开发）、游戏开发、大数据开发、移动端开发。现在越来越多的公司要求运维人员会 Python 自动化开发，Python 也成了运维人员必备的技能，每一个运维人员在熟悉了 Shell 之后，都应该再学习 Python 语言。

**3) Shell**

Shell 脚本的优势在于处理偏操作系统底层的业务，例如，Linux 内部的很多应用（有的是应用的一部分）都是使用 Shell 脚本开发的，因为有 1000 多个 Linux 系统命令为它作支撑，特别是 Linux 正则表达式以及三剑客 grep、awk、sed 等命令。

对于一些常见的系统脚本，使用 Shell 开发会更简单、更快速，例如，让软件一键自动化安装、优化，监控报警脚本，软件启动脚本，日志分析脚本等，虽然 Python 也能做到这些，但是考虑到掌握难度、开发效率、开发习惯等因素，它们可能就不如 Shell 脚本流行以及有优势了。对于一些常见的业务应用，使用 Shell 更符合 Linux 运维简单、易用、高效的三大原则。Python 语言的优势在于开发复杂的运维软件、Web 页面的管理工具和 Web 业务的开发（例如 CMDB 自动化运维平台、跳板机、批量管理软件 SaltStack、云计算 OpenStack 软件）等。我们在开发一个应用时，应该根据业务需求，结合不同语言的优势以及自己擅长的语言来选择，扬长避短，从而达到高效开发、易于自己维护的目的。

### 8 Shell类型

Linux 是一个开源的操作系统，由分布在世界各地的多个组织机构或个人共同开发完成，每个组织结构或个人负责一部分功能，最后组合在一起，就构成了今天的 Linux。例如：

Linux 内核最初由芬兰黑客 Linus Torvalds 开发，后来他组建了团队，Linux 内核由这个团队维护。GNU 组织开发了很多核心软件和基础库，例如 GCC 编译器、C语言标准库、文本编辑器 Emacs、进程管理软件、Shell 以及 GNOME 桌面环境等。VIM 编辑器由荷兰人 Bram Moolenaar 开发。

Windows、Mac OS、Android 等操作系统不一样，它们都由一家公司开发，所有的核心软件和基础库都由一家公司做决定，容易形成统一的标准，一般不会开发多款功能类似的软件。

而 Linux 不一样，它是“万国牌”，由多个组织机构开发，不同的组织机构为了发展自己的 Linux 分支可能会开发出功能类似的软件，它们各有优缺点，用户可以自由选择。Shell 就是这样的一款软件，不同的组织机构开发了不同的 Shell，它们各有所长，有的占用资源少，有的支持高级编程功能，有的兼容性好，有的重视用户体验。

Shell 既是一种脚本编程语言，也是一个连接内核和用户的软件。

常见的 Shell 有 **sh、bash、csh、tcsh、ash** 等。

**（1）sh**

sh 的全称是 Bourne shell，由 AT&T 公司的 Steve Bourne开发，为了纪念他，就用他的名字命名了。sh 是 UNIX 上的标准 shell，很多 UNIX 版本都配有 sh。sh 是第一个流行的 Shell。

**（2）csh**

sh 之后另一个广为流传的 shell 是由柏克莱大学的 Bill Joy 设计的，这个 shell 的语法有点类似C语言，所以才得名为 C shell ，简称为 csh。Bill Joy 是一个风云人物，他创立了 BSD 操作系统，开发了 vi 编辑器，还是 Sun 公司的创始人之一。BSD 是 UNIX 的一个重要分支，后人在此基础上发展出了很多现代的操作系统，最著名的有 FreeBSD、OpenBSD 和 NetBSD，就连 Mac OS X 在很大程度上也基于BSD。

**（3）tcsh**

tcsh 是 csh 的增强版，加入了命令补全功能，提供了更加强大的语法支持。

**（4）ash**

一个简单的轻量级的 Shell，占用资源少，适合运行于低内存环境，但是与下面讲到的 bash shell 完全兼容。

**（5）bash**

**bash shell 是 Linux 的默认 shell，本教程也基于 bash 编写**。

bash 由 GNU 组织开发，保持了对 sh shell 的兼容性，是各种 Linux 发行版默认配置的 shell。bash 兼容 sh 意味着，针对 sh 编写的 Shell 代码可以不加修改地在 bash 中运行。尽管如此，bash 和 sh 还是有一些不同之处：一方面，bash 扩展了一些命令和参数；

另一方面，bash 并不完全和 sh 兼容，它们有些行为并不一致，但在大多数企业运维的情况下区别不大，特殊场景可以使用 bash 代替 sh。

**（6）查看 Shell**

Shell 是一个程序，一般都是放在/bin或者/usr/bin目录下，当前 Linux 系统可用的 Shell 都记录在/etc/shells文件中。/etc/shells是一个纯文本文件，你可以在图形界面下打开它，也可以使用 cat 命令查看它。

通过 cat 命令来查看当前 Linux 系统的可用 Shell：

$ cat /etc/shells

/bin/sh

/bin/bash

/sbin/nologin

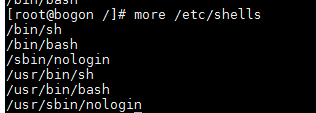
/usr/bin/sh

/usr/bin/bash

/usr/sbin/nologin

/bin/tcsh

/bin/csh



在现代的 Linux 上，sh 已经被 bash 代替，/bin/sh往往是指向/bin/bash的符号链接。

如果你希望查看当前 Linux 的默认 Shell，那么可以输出 SHELL 环境变量：

$ echo $SHELL

/bin/bash



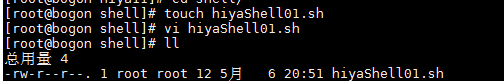
输出结果表明默认的 Shell 是 bash。

echo是一个 Shell 命令，用来输出变量的值，我们将在《Shell echo》一节中详细介绍它的用法。SHELL是 Linux 系统中的环境变量，它指明了当前使用的 Shell 程序的位置，也就是使用的哪个 Shell。

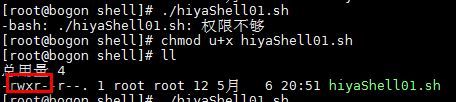
### 9 执行文件

**touch hiyaShell01.sh**

如果fileA存在,使用touch指令可更改这个文件或目录的日期时间,包括存取时间和更改时间; 如果fileA不存在,touch指令会在当前目录下创建这个文件。

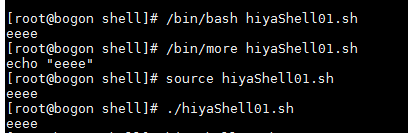


**chmod u+x hiyaShell01.sh** 修改权限可执行



**./hiyaShell01.sh** 执行文件

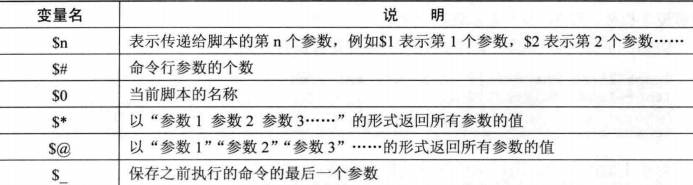


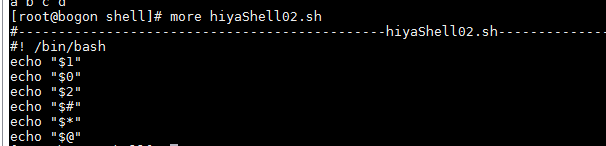


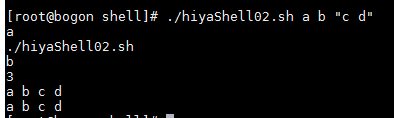
执行文件方法很多种



### 10 传参数

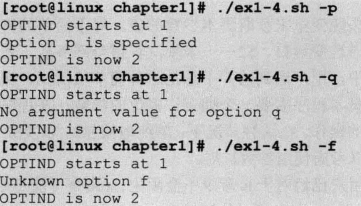






### 11 扩展参数

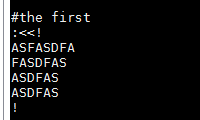
类似 ls -l这种参数

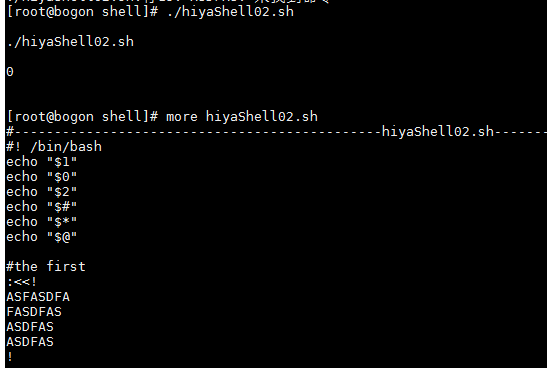


### 12 使用more解释器



### 13 一行注释和多行注释

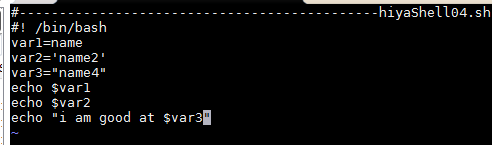


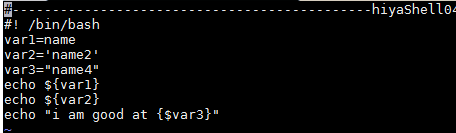


### 14 变量

在 Bash shell 中，每一个变量的值都是字符串，无论你给变量赋值时有没有使用引号，值都会以字符串的形式存储。赋值号=的周围不能有空格。

variable=value  
variable='value'  
variable="value"





单引号和双引号：

以单引号' '包围变量的值时，单引号里面是什么就输出什么，即使内容中有变量和命令（命令需要反引起来）也会把它们原样输出。这种方式比较适合定义显示纯字符串的情况，即不希望解析变量、命令等的场景。  
  
以双引号" "包围变量的值时，输出时会先解析里面的变量和命令，而不是把双引号中的变量名和命令原样输出。这种方式比较适合字符串中附带有变量和命令并且想将其解析后再输出的变量定义。

## 十三、Shell 变量

**1 定义变量时，变量名不加美元符号（$，PHP语言中变量需要），如：**

your\_name="runoob.com"

**2 注意，变量名和等号之间不能有空格，**这可能和你熟悉的所有编程语言都不一样。同时，变量名的命名须遵循如下规则：

@命名只能使用英文字母，数字和下划线，首个字符不能以数字开头。

@中间不能有空格，可以使用下划线（\_）。

@不能使用标点符号。

@不能使用bash里的关键字（可用help命令查看保留关键字）。

**3 有效的 Shell 变量名示例如下：**

RUNOOB

LD\_LIBRARY\_PATH

\_var

var2

**4 无效的变量命名：**

?var=123

user\*name=runoob

**5 除了显式地直接赋值，还可以用语句给变量赋值，如：**

for file in `ls /etc`

或

for file in $(ls /etc)

以上语句将 /etc 下目录的文件名循环出来。

**6 使用变量**

使用一个定义过的变量，只要在变量名前面加美元符号即可，如：

your\_name="qinjx"

echo $your\_name

echo ${your\_name}

变量名外面的花括号是可选的，加不加都行，加花括号是为了帮助解释器识别变量的边界，比如下面这种情况：

for skill in Ada Coffe Action Java; do

echo "I am good at ${skill}Script"

done

如果不给skill变量加花括号，写成echo "I am good at $skillScript"，解释器就会把$skillScript当成一个变量（其值为空），代码执行结果就不是我们期望的样子了。

推荐给所有变量加上花括号，这是个好的编程习惯。

已定义的变量，可以被重新定义，如：

your\_name="tom"

echo $your\_name

your\_name="alibaba"

echo $your\_name

这样写是合法的，但注意，第二次赋值的时候不能写$your\_name="alibaba"，使用变量的时候才加美元符（$）。

**7 只读变量**

使用 readonly 命令可以将变量定义为只读变量，只读变量的值不能被改变。

下面的例子尝试更改只读变量，结果报错：

#!/bin/bash

myUrl="http://www.google.com"

readonly myUrl

myUrl="http://www.runoob.com"

运行脚本，结果如下：

/bin/sh: NAME: This variable is read only.

**8 删除变量**

使用 unset 命令可以删除变量。语法：

unset variable\_name

变量被删除后不能再次使用。unset 命令不能删除只读变量。

实例

#!/bin/sh

myUrl="http://www.runoob.com"

unset myUrl

echo $myUrl

以上实例执行将没有任何输出。

**9 变量类型**

运行shell时，会同时存在三种变量：

1) 局部变量 局部变量在脚本或命令中定义，仅在当前shell实例中有效，其他shell启动的程序不能访问局部变量。

2) 环境变量 所有的程序，包括shell启动的程序，都能访问环境变量，有些程序需要环境变量来保证其正常运行。必要的时候shell脚本也可以定义环境变量。

3) shell变量 shell变量是由shell程序设置的特殊变量。shell变量中有一部分是环境变量，有一部分是局部变量，这些变量保证了shell的正常运行

**10 Shell 字符串**

字符串是shell编程中最常用最有用的数据类型（除了数字和字符串，也没啥其它类型好用了），字符串可以用单引号，也可以用双引号，也可以不用引号。单双引号的区别跟PHP类似。

**（1）单引号**

str='this is a string'

单引号字符串的限制：

单引号里的任何字符都会原样输出，单引号字符串中的变量是无效的；

单引号字串中不能出现单独一个的单引号（对单引号使用转义符后也不行），但可成对出现，作为字符串拼接使用。

**（2）双引号**

your\_name='runoob'

str="Hello, I know you are \"$your\_name\"! \n"

echo $str

Hello, I know you are "runoob"!

输出结果为：

双引号的优点：

双引号里可以有变量

双引号里可以出现转义字符

**（3）拼接字符串**

your\_name="runoob"

# 使用双引号拼接

greeting="hello, "$your\_name" !"

greeting\_1="hello, ${your\_name} !"

echo $greeting $greeting\_1

# 使用单引号拼接

greeting\_2='hello, '$your\_name' !'

greeting\_3='hello, ${your\_name} !'

echo $greeting\_2 $greeting\_3

输出结果为：

hello, runoob ! hello, runoob !

hello, runoob ! hello, ${your\_name} !

**（4）获取字符串长度**

string="abcd"

echo ${#string} #输出 4

**（5）提取子字符串**

以下实例从字符串第 2 个字符开始截取 4 个字符：

string="runoob is a great site"

echo ${string:1:4} # 输出 unoo

**（6）查找子字符串**

查找字符 i 或 o 的位置(哪个字母先出现就计算哪个)：

string="runoob is a great site"

echo `expr index "$string" io` # 输出 4

注意： 以上脚本中 ` 是反引号，而不是单引号 '，不要看错了哦。

**11 Shell 数组**

bash支持一维数组（不支持多维数组），并且没有限定数组的大小。

类似于 C 语言，数组元素的下标由 0 开始编号。获取数组中的元素要利用下标，下标可以是整数或算术表达式，其值应大于或等于 0。

**（1）定义数组**

在 Shell 中，用括号来表示数组，数组元素用"空格"符号分割开。定义数组的一般形式为：

数组名=(值1 值2 ... 值n)

例如：

array\_name=(value0 value1 value2 value3)

或者

array\_name=(

value0

value1

value2

value3

)

还可以单独定义数组的各个分量：

array\_name[0]=value0

array\_name[1]=value1

array\_name[n]=valuen

可以不使用连续的下标，而且下标的范围没有限制。

**（2）读取数组**

读取数组元素值的一般格式是：

${数组名[下标]}

例如：

valuen=${array\_name[n]}

使用 @ 符号可以获取数组中的所有元素，例如：

echo ${array\_name[@]}

**（3）获取数组的长度**

获取数组长度的方法与获取字符串长度的方法相同，例如：

# 取得数组元素的个数

length=${#array\_name[@]}

# 或者

length=${#array\_name[\*]}

# 取得数组单个元素的长度

lengthn=${#array\_name[n]}

**12 Shell 注释**

**（1）以 # 开头的行就是注释，会被解释器忽略。**

通过每一行加一个 # 号设置多行注释，像这样：

#--------------------------------------------

# 这是一个注释

# author：菜鸟教程

# site：www.runoob.com

# slogan：学的不仅是技术，更是梦想！

#--------------------------------------------

##### 用户配置区 开始 #####

#

#

# 这里可以添加脚本描述信息

#

#

##### 用户配置区 结束 #####

如果在开发过程中，遇到大段的代码需要临时注释起来，过一会儿又取消注释，怎么办呢？

每一行加个#符号太费力了，可以把这一段要注释的代码用一对花括号括起来，定义成一个函数，没有地方调用这个函数，这块代码就不会执行，达到了和注释一样的效果。

**（2）多行注释**

多行注释还可以使用以下格式：

:<<EOF

注释内容...

注释内容...

注释内容...

EOF

EOF 也可以使用其他符号:

:<<'

注释内容...

注释内容...

注释内容...

'

:<<!

注释内容...

注释内容...

注释内容...

!

## 十四、Shell 传递参数

### 1 概述

我们可以在执行 Shell 脚本时向脚本传递参数，脚本内获取参数的格式为：$n。n 代表一个数字，1 为执行脚本的第一个参数，2 为执行脚本的第二个参数，以此类推……

### 2 实例

以下实例我们向脚本传递三个参数，并分别输出，其中 $0 为执行的文件名：

#!/bin/bash

# author:菜鸟教程

# url:www.runoob.com

echo "Shell 传递参数实例！";

echo "执行的文件名：$0";

echo "第一个参数为：$1";

echo "第二个参数为：$2";

echo "第三个参数为：$3";

### 3 为脚本设置可执行权限，并执行脚本，输出结果如下所示：

$ chmod +x test.sh

$ ./test.sh 1 2 3

Shell 传递参数实例！

执行的文件名：./test.sh

**第一个参数为：1**

**第二个参数为：2**

**第三个参数为：3**

### 4 另外，还有几个特殊字符用来处理参数：

|  |  |
| --- | --- |
| **参数处理** | **说明** |
| $# | 传递到脚本的参数个数 |
| $\* | 以一个单字符串显示所有向脚本传递的参数。 如"$\*"用「"」括起来的情况、以"$1 $2 … $n"的形式输出所有参数。 |
| $$ | 脚本运行的当前进程ID号 |
| $! | 后台运行的最后一个进程的ID号 |
| $@ | 与$\*相同，但是使用时加引号，并在引号中返回每个参数。 如"$@"用「"」括起来的情况、以"$1" "$2" … "$n" 的形式输出所有参数。 |
| $- | 显示Shell使用的当前选项，与[set命令](http://www.runoob.com/linux/linux-comm-set.html)功能相同。 |
| $? | 显示最后命令的退出状态。0表示没有错误，其他任何值表明有错误。 |

#!/bin/bash

# author:菜鸟教程

# url:www.runoob.com

echo "Shell 传递参数实例！";

echo "第一个参数为：$1";

echo "参数个数为：$#";

echo "传递的参数作为一个字符串显示：$\*";

执行脚本，输出结果如下所示：

$ chmod +x test.sh

$ ./test.sh 1 2 3

Shell 传递参数实例！

第一个参数为：1

参数个数为：3

传递的参数作为一个字符串显示：1 2 3

### 5 $\* 与 $@ 区别：

**相同点：都是引用所有参数。**

**不同点：只有在双引号中体现出来。假设在脚本运行时写了三个参数 1、2、3，，则 " \* " 等价于 "1 2 3"（传递了一个参数），而 "@" 等价于 "1" "2" "3"（传递了三个参数）。**

#!/bin/bash

# author:菜鸟教程

# url:www.runoob.com

echo "-- \$\* 演示 ---"

for i in "$\*"; do

echo $i

done

echo "-- \$@ 演示 ---"

for i in "$@"; do

echo $i

done

执行脚本，输出结果如下所示：

$ chmod +x test.sh

$ ./test.sh 1 2 3

-- $\* 演示 ---

**1 2 3**

-- $@ 演示 ---

1

2

3

## 十五、Shell 基本运算符

### 1 Shell 和其他编程语言一样，支持多种运算符，包括：

算数运算符

关系运算符

布尔运算符

字符串运算符

文件测试运算符

原生bash不支持简单的数学运算，但是可以通过其他命令来实现，例如 awk 和 expr，expr 最常用。expr 是一款表达式计算工具，使用它能完成表达式的求值操作。

例如，两个数相加(注意使用的是反引号 ` 而不是单引号 ')：

#!/bin/bash

val=`expr 2 + 2`

echo "两数之和为 : $val"

### 2 算术运算符

下表列出了常用的算术运算符，假定变量 a 为 10，变量 b 为 20：

|  |  |  |
| --- | --- | --- |
| **运算符** | **说明** | **举例** |
| + | 加法 | `expr $a + $b` 结果为 30。 |
| - | 减法 | `expr $a - $b` 结果为 -10。 |
| \* | 乘法 | `expr $a \\* $b` 结果为  200。 |
| / | 除法 | `expr $b / $a` 结果为 2。 |
| % | 取余 | `expr $b % $a` 结果为 0。 |
| = | 赋值 | a=$b 将把变量 b 的值赋给 a。 |
| == | 相等。用于比较两个数字，相同则返回 true。 | [ $a == $b ] 返回 false。 |
| != | 不相等。用于比较两个数字，不相同则返回 true。 | [ $a != $b ] 返回 true。 |

实例

#!/bin/bash

# author:菜鸟教程

# url:www.runoob.com

a=10

b=20

val=`expr $a + $b`

echo "a + b : $val"

val=`expr $a - $b`

echo "a - b : $val"

val=`expr $a \\* $b`

echo "a \* b : $val"

val=`expr $b / $a`

echo "b / a : $val"

val=`expr $b % $a`

echo "b % a : $val"

if [ $a == $b ]

then

echo "a 等于 b"

fi

if [ $a != $b ]

then

echo "a 不等于 b"

fi

执行脚本，输出结果如下所示：

a + b : 30

a - b : -10

a \* b : 200

b / a : 2

b % a : 0

a 不等于 b

注意：

乘号(\*)前边必须加反斜杠(\)才能实现乘法运算；

if...then...fi 是条件语句，后续将会讲解。

在 MAC 中 shell 的 expr 语法是：$((表达式))，此处表达式中的 "\*" 不需要转义符号 "\" 。

### 3 关系运算符

关系运算符只支持数字，不支持字符串，除非字符串的值是数字。

下表列出了常用的关系运算符，假定变量 a 为 10，变量 b 为 20：



实例

关系运算符实例如下：

#!/bin/bash

# author:菜鸟教程

# url:www.runoob.com

a=10

b=20

if [ $a -eq $b ]

then

echo "$a -eq $b : a 等于 b"

else

echo "$a -eq $b: a 不等于 b"

fi

if [ $a -ne $b ]

then

echo "$a -ne $b: a 不等于 b"

else

echo "$a -ne $b : a 等于 b"

fi

if [ $a -gt $b ]

then

echo "$a -gt $b: a 大于 b"

else

echo "$a -gt $b: a 不大于 b"

fi

if [ $a -lt $b ]

then

echo "$a -lt $b: a 小于 b"

else

echo "$a -lt $b: a 不小于 b"

fi

if [ $a -ge $b ]

then

echo "$a -ge $b: a 大于或等于 b"

else

echo "$a -ge $b: a 小于 b"

fi

if [ $a -le $b ]

then

echo "$a -le $b: a 小于或等于 b"

else

echo "$a -le $b: a 大于 b"

fi

执行脚本，输出结果如下所示：

10 -eq 20: a 不等于 b

10 -ne 20: a 不等于 b

10 -gt 20: a 不大于 b

10 -lt 20: a 小于 b

10 -ge 20: a 小于 b

10 -le 20: a 小于或等于 b

### 4 布尔运算符

下表列出了常用的布尔运算符，假定变量 a 为 10，变量 b 为 20：



实例

布尔运算符实例如下：

#!/bin/bash

# author:菜鸟教程

# url:www.runoob.com

a=10

b=20

if [ $a != $b ]

then

echo "$a != $b : a 不等于 b"

else

echo "$a != $b: a 等于 b"

fi

if [ $a -lt 100 -a $b -gt 15 ]

then

echo "$a 小于 100 且 $b 大于 15 : 返回 true"

else

echo "$a 小于 100 且 $b 大于 15 : 返回 false"

fi

if [ $a -lt 100 -o $b -gt 100 ]

then

echo "$a 小于 100 或 $b 大于 100 : 返回 true"

else

echo "$a 小于 100 或 $b 大于 100 : 返回 false"

fi

if [ $a -lt 5 -o $b -gt 100 ]

then

echo "$a 小于 5 或 $b 大于 100 : 返回 true"

else

echo "$a 小于 5 或 $b 大于 100 : 返回 false"

fi

执行脚本，输出结果如下所示：

10 != 20 : a 不等于 b

10 小于 100 且 20 大于 15 : 返回 true

10 小于 100 或 20 大于 100 : 返回 true

10 小于 5 或 20 大于 100 : 返回 false

### 5 逻辑运算符

以下介绍 Shell 的逻辑运算符，假定变量 a 为 10，变量 b 为 20:



实例

逻辑运算符实例如下：

#!/bin/bash

# author:菜鸟教程

# url:www.runoob.com

a=10

b=20

if [[ $a -lt 100 && $b -gt 100 ]]

then

echo "返回 true"

else

echo "返回 false"

fi

if [[ $a -lt 100 || $b -gt 100 ]]

then

echo "返回 true"

else

echo "返回 false"

fi

执行脚本，输出结果如下所示：

返回 false

返回 true

### 6 字符串运算符

下表列出了常用的字符串运算符，假定变量 a 为 "abc"，变量 b 为 "efg"：



实例

#!/bin/bash

# author:菜鸟教程

# url:www.runoob.com

a="abc"

b="efg"

if [ $a = $b ]

then

echo "$a = $b : a 等于 b"

else

echo "$a = $b: a 不等于 b"

fi

if [ $a != $b ]

then

echo "$a != $b : a 不等于 b"

else

echo "$a != $b: a 等于 b"

fi

if [ -z $a ]

then

echo "-z $a : 字符串长度为 0"

else

echo "-z $a : 字符串长度不为 0"

fi

if [ -n "$a" ]

then

echo "-n $a : 字符串长度不为 0"

else

echo "-n $a : 字符串长度为 0"

fi

if [ $a ]

then

echo "$a : 字符串不为空"

else

echo "$a : 字符串为空"

fi

执行脚本，输出结果如下所示：

abc = efg: a 不等于 b

abc != efg : a 不等于 b

-z abc : 字符串长度不为 0

-n abc : 字符串长度不为 0

abc : 字符串不为空

### 7 文件测试运算符

文件测试运算符用于检测 Unix 文件的各种属性。

属性检测描述如下：



实例

变量 file 表示文件"/var/www/runoob/test.sh"，它的大小为100字节，具有 rwx 权限。下面的代码，将检测该文件的各种属性：

#!/bin/bash

# author:菜鸟教程

# url:www.runoob.com

file="/var/www/runoob/test.sh"

if [ -r $file ]

then

echo "文件可读"

else

echo "文件不可读"

fi

if [ -w $file ]

then

echo "文件可写"

else

echo "文件不可写"

fi

if [ -x $file ]

then

echo "文件可执行"

else

echo "文件不可执行"

fi

if [ -f $file ]

then

echo "文件为普通文件"

else

echo "文件为特殊文件"

fi

if [ -d $file ]

then

echo "文件是个目录"

else

echo "文件不是个目录"

fi

if [ -s $file ]

then

echo "文件不为空"

else

echo "文件为空"

fi

if [ -e $file ]

then

echo "文件存在"

else

echo "文件不存在"

fi

执行脚本，输出结如下所示：

文件可读

文件可写

文件可执行

文件为普通文件

文件不是个目录

文件不为空

文件存在

## 十六、Shell echo命令

Shell 的 echo 指令与 PHP 的 echo 指令类似，都是用于字符串的输出。命令格式：

echo string

您可以使用echo实现更复杂的输出格式控制。

### 1.显示普通字符串:

echo "It is a test"

这里的双引号完全可以省略，以下命令与上面实例效果一致：

echo It is a test

### 2.显示转义字符

echo "\"It is a test\""

结果将是:

"It is a test"

同样，双引号也可以省略

### 3.显示变量

read 命令从标准输入中读取一行,并把输入行的每个字段的值指定给 shell 变量

#!/bin/sh

read name

echo "$name It is a test"

以上代码保存为 test.sh，name 接收标准输入的变量，结果将是:

[root@www ~]# sh test.sh

OK #标准输入

OK It is a test #输出

### 4.显示换行

echo -e "OK! \n" # -e 开启转义

echo "It is a test"

输出结果：

OK!

It is a test

### 5.显示不换行

#!/bin/sh

echo -e "OK! \c" # -e 开启转义 \c 不换行

echo "It is a test"

输出结果：

OK! It is a test

### 6.显示结果定向至文件

echo "It is a test" > myfile

### 7.原样输出字符串，不进行转义或取变量(用单引号)

echo '$name\"'

输出结果：

$name\"

### 8.显示命令执行结果

echo `date`

注意： 这里使用的是反引号 `, 而不是单引号 '。

结果将显示当前日期

Thu Jul 24 10:08:46 CST 2014

## 十七、Shell printf 命令

### 1 printf含义

printf 命令模仿 C 程序库（library）里的 printf() 程序。

printf 由 POSIX 标准所定义，因此使用 printf 的脚本比使用 echo 移植性好。

printf 使用引用文本或空格分隔的参数，外面可以在 printf 中使用格式化字符串，还可以制定字符串的宽度、左右对齐方式等。默认 printf 不会像 echo 自动添加换行符，我们可以手动添加 \n。

### 2 printf 命令的语法：

printf format-string [arguments...]

参数说明：

format-string: 为格式控制字符串

arguments: 为参数列表。

### 3 实例如下：

$ echo "Hello, Shell"

Hello, Shell

$ printf "Hello, Shell\n"

Hello, Shell

$

### 4 接下来,我来用一个脚本来体现printf的强大功能：

#!/bin/bash

# author:菜鸟教程

# url:www.runoob.com

printf "%-10s %-8s %-4s\n" 姓名 性别 体重kg

printf "%-10s %-8s %-4.2f\n" 郭靖 男 66.1234

printf "%-10s %-8s %-4.2f\n" 杨过 男 48.6543

printf "%-10s %-8s %-4.2f\n" 郭芙 女 47.9876

### 5 执行脚本，输出结果如下所示：

姓名 性别 体重kg

郭靖 男 66.12

杨过 男 48.65

郭芙 女 47.99

%s %c %d %f都是格式替代符

%-10s 指一个宽度为10个字符（-表示左对齐，没有则表示右对齐），任何字符都会被显示在10个字符宽的字符内，如果不足则自动以空格填充，超过也会将内容全部显示出来。

%-4.2f 指格式化为小数，其中.2指保留2位小数。

### 6 更多实例：

#!/bin/bash

# author:菜鸟教程

# url:www.runoob.com

# format-string为双引号

printf "%d %s\n" 1 "abc"

# 单引号与双引号效果一样

printf '%d %s\n' 1 "abc"

# 没有引号也可以输出

printf %s abcdef

# 格式只指定了一个参数，但多出的参数仍然会按照该格式输出，format-string 被重用

printf %s abc def

printf "%s\n" abc def

printf "%s %s %s\n" a b c d e f g h i j

# 如果没有 arguments，那么 %s 用NULL代替，%d 用 0 代替

printf "%s and %d \n"

执行脚本，输出结果如下所示：

1 abc

1 abc

abcdefabcdefabc

def

a b c

d e f

g h i

j

and 0

### 7 printf的转义序列

|  |  |
| --- | --- |
| **序列** | **说明** |
| \a | 警告字符，通常为ASCII的BEL字符 |
| \b | 后退 |
| \c | 抑制（不显示）输出结果中任何结尾的换行字符（只在%b格式指示符控制下的参数字符串中有效），而且，任何留在参数里的字符、任何接下来的参数以及任何留在格式字符串中的字符，都被忽略 |
| \f | 换页（formfeed） |
| \n | 换行 |
| \r | 回车（Carriage return） |
| \t | 水平制表符 |
| \v | 垂直制表符 |
| \\ | 一个字面上的反斜杠字符 |
| \ddd | 表示1到3位数八进制值的字符。仅在格式字符串中有效 |
| \0ddd | 表示1到3位的八进制值字符 |

### 8 printf的转义序列实例

$ printf "a string, no processing:<%s>\n" "A\nB"

a string, no processing:<A\nB>

$ printf "a string, no processing:<%b>\n" "A\nB"

a string, no processing:<A

B>

$ printf "www.runoob.com \a"

www.runoob.com $ #不换行

## 十八、Shell test 命令

**Shell中的 test 命令用于检查某个条件是否成立，它可以进行数值、字符和文件三个方面的测试。**

### 1 数值测试

参数 说明

-eq 等于则为真

-ne 不等于则为真

-gt 大于则为真

-ge 大于等于则为真

-lt 小于则为真

-le 小于等于则为真

### 2 实例演示：

num1=100

num2=100

if test $[num1] -eq $[num2]

then

echo '两个数相等！'

else

echo '两个数不相等！'

fi

输出结果：

两个数相等！

### 3 代码中的 [] 执行基本的算数运算，如：

#!/bin/bash

a=5

b=6

result=$[a+b] # 注意等号两边不能有空格

echo "result 为： $result"

结果为

result 为： 11

### 4 字符串测试

参数 说明

= 等于则为真

!= 不相等则为真

-z 字符串 字符串的长度为零则为真

-n 字符串 字符串的长度不为零则为真

实例演示：

num1="ru1noob"

num2="runoob"

if test $num1 = $num2

then

echo '两个字符串相等!'

else

echo '两个字符串不相等!'

fi

输出结果：

两个字符串不相等!

### 5 文件测试

参数 说明

-e 文件名 如果文件存在则为真

-r 文件名 如果文件存在且可读则为真

-w 文件名 如果文件存在且可写则为真

-x 文件名 如果文件存在且可执行则为真

-s 文件名 如果文件存在且至少有一个字符则为真

-d 文件名 如果文件存在且为目录则为真

-f 文件名 如果文件存在且为普通文件则为真

-c 文件名 如果文件存在且为字符型特殊文件则为真

-b 文件名 如果文件存在且为块特殊文件则为真

实例演示：

cd /bin

if test -e ./bash

then

echo '文件已存在!'

else

echo '文件不存在!'

fi

输出结果：

文件已存在!

## 十九、Shell 流程控制

### 1 和Java、PHP等语言不一样，sh的流程控制不可为空，如(以下为PHP流程控制写法)：

<?php

if (isset($\_GET["q"])) {

search(q);

}

else {

// 不做任何事情

}

在sh/bash里可不能这么写，如果else分支没有语句执行，就不要写这个else。

### 2 if else

**（1）if**

if 语句语法格式：

if condition

then

command1

command2

...

commandN

fi

写成一行（适用于终端命令提示符）：

if [ $(ps -ef | grep -c "ssh") -gt 1 ]; then echo "true"; fi

末尾的fi就是if倒过来拼写，后面还会遇到类似的。

**（2）if else**

if else 语法格式：

if condition

then

command1

command2

...

commandN

else

command

fi

**(3)if else-if else**

if else-if else 语法格式：

if condition1

then

command1

elif condition2

then

command2

else

commandN

fi

以下实例判断两个变量是否相等：

a=10

b=20

if [ $a == $b ]

then

echo "a 等于 b"

elif [ $a -gt $b ]

then

echo "a 大于 b"

elif [ $a -lt $b ]

then

echo "a 小于 b"

else

echo "没有符合的条件"

fi

输出结果：

a 小于 b

**3 for 循环**

与其他编程语言类似，Shell支持for循环。

for循环一般格式为：

for var in item1 item2 ... itemN

do

command1

command2

...

commandN

done

写成一行：

for var in item1 item2 ... itemN; do command1; command2… done;

当变量值在列表里，for循环即执行一次所有命令，使用变量名获取列表中的当前取值。命令可为任何有效的shell命令和语句。in列表可以包含替换、字符串和文件名。

in列表是可选的，如果不用它，for循环使用命令行的位置参数。

例如，顺序输出当前列表中的数字：

for loop in 1 2 3 4 5

do

echo "The value is: $loop"

done

输出结果：

The value is: 1

The value is: 2

The value is: 3

The value is: 4

The value is: 5

顺序输出字符串中的字符：

for str in 'This is a string'

do

echo $str

done

输出结果：

This is a string

**4 while 语句**

while循环用于不断执行一系列命令，也用于从输入文件中读取数据；命令通常为测试条件。其格式为：

while condition

do

command

done

以下是一个基本的while循环，测试条件是：如果int小于等于5，那么条件返回真。int从0开始，每次循环处理时，int加1。运行上述脚本，返回数字1到5，然后终止。

#!/bin/bash

int=1

while(( $int<=5 ))

do

echo $int

let "int++"

done

运行脚本，输出：

使用中使用了 Bash let 命令，它用于执行一个或多个表达式，变量计算中不需要加上 $ 来表示变量，具体可查阅：Bash let 命令

。

while循环可用于读取键盘信息。下面的例子中，输入信息被设置为变量FILM，按<Ctrl-D>结束循环。

echo '按下 <CTRL-D> 退出'

echo -n '输入你最喜欢的网站名: '

while read FILM

do

echo "是的！$FILM 是一个好网站"

done

运行脚本，输出类似下面：

按下 <CTRL-D> 退出

输入你最喜欢的网站名:菜鸟教程

是的！菜鸟教程 是一个好网站

**5 无限循环**

无限循环语法格式：

while

do

command

done

或者

while true

do

command

done

或者

for (( ; ; ))

**6 until 循环**

until 循环执行一系列命令直至条件为 true 时停止。

until 循环与 while 循环在处理方式上刚好相反。

一般 while 循环优于 until 循环，但在某些时候—也只是极少数情况下，until 循环更加有用。

until 语法格式:

until condition

do

command

done

condition 一般为条件表达式，如果返回值为 false，则继续执行循环体内的语句，否则跳出循环。

以下实例我们使用 until 命令来输出 0 ~ 9 的数字：

#!/bin/bash

a=0

until [ ! $a -lt 10 ]

do

echo $a

a=`expr $a + 1`

done

**7 case**

Shell case语句为多选择语句。可以用case语句匹配一个值与一个模式，如果匹配成功，执行相匹配的命令。case语句格式如下：

case 值 in

模式1)

command1

command2

...

commandN

;;

模式2）

command1

command2

...

commandN

;;

esac

case工作方式如上所示。取值后面必须为单词in，每一模式必须以右括号结束。取值可以为变量或常数。匹配发现取值符合某一模式后，其间所有命令开始执行直至 ;;。

取值将检测匹配的每一个模式。一旦模式匹配，则执行完匹配模式相应命令后不再继续其他模式。如果无一匹配模式，使用星号 \* 捕获该值，再执行后面的命令。

**8 跳出循环**

在循环过程中，有时候需要在未达到循环结束条件时强制跳出循环，Shell使用两个命令来实现该功能：break和continue。

break命令

break命令允许跳出所有循环（终止执行后面的所有循环）。

下面的例子中，脚本进入死循环直至用户输入数字大于5。要跳出这个循环，返回到shell提示符下，需要使用break命令。

#!/bin/bash

while :

do

echo -n "输入 1 到 5 之间的数字:"

read aNum

case $aNum in

1|2|3|4|5) echo "你输入的数字为 $aNum!"

;;

\*) echo "你输入的数字不是 1 到 5 之间的! 游戏结束"

break

;;

esac

done

执行以上代码，输出结果为：

输入 1 到 5 之间的数字:3

你输入的数字为 3!

输入 1 到 5 之间的数字:7

你输入的数字不是 1 到 5 之间的! 游戏结束

**9 continue**

continue命令与break命令类似，只有一点差别，它不会跳出所有循环，仅仅跳出当前循环。

对上面的例子进行修改：

#!/bin/bash

while :

do

echo -n "输入 1 到 5 之间的数字: "

read aNum

case $aNum in

1|2|3|4|5) echo "你输入的数字为 $aNum!"

;;

\*) echo "你输入的数字不是 1 到 5 之间的!"

continue

echo "游戏结束"

;;

esac

done

运行代码发现，当输入大于5的数字时，该例中的循环不会结束，语句 echo "游戏结束" 永远不会被执行。

## 二十、Shell 函数

### 1 函数的定义

**[ function ] funname [()]**

**{**

**action;**

**[return int;]**

**}**

说明：

1）、可以带function fun() 定义，也可以直接fun() 定义,不带任何参数。

2）、参数返回，可以显示加：return 返回，如果不加，将以最后一条命令运行结果，作为返回值。 return后跟数值n(0-255

### 2 下面的例子定义了一个函数并进行调用：

#!/bin/bash

# author:菜鸟教程

# url:www.runoob.com

demoFun(){

echo "这是我的第一个 shell 函数!"

}

echo "-----函数开始执行-----"

demoFun

echo "-----函数执行完毕-----"

输出结果：

-----函数开始执行-----

这是我的第一个 shell 函数!

-----函数执行完毕-----

### 3 下面定义一个带有return语句的函数：

#!/bin/bash

# author:菜鸟教程

# url:www.runoob.com

funWithReturn(){

echo "这个函数会对输入的两个数字进行相加运算..."

echo "输入第一个数字: "

read aNum

echo "输入第二个数字: "

read anotherNum

echo "两个数字分别为 $aNum 和 $anotherNum !"

return $(($aNum+$anotherNum))

}

funWithReturn

echo "输入的两个数字之和为 $? !"

输出类似下面：

这个函数会对输入的两个数字进行相加运算...

输入第一个数字:

1

输入第二个数字:

2

两个数字分别为 1 和 2 !

输入的两个数字之和为 3 !

函数返回值在调用该函数后通过 $? 来获得。

注意：所有函数在使用前必须定义。这意味着必须将函数放在脚本开始部分，直至shell解释器首次发现它时，才可以使用。调用函数仅使用其函数名即可。

### 4 函数参数

在Shell中，调用函数时可以向其传递参数。在函数体内部，通过 $n 的形式来获取参数的值，例如，$1表示第一个参数，$2表示第二个参数...

|  |  |
| --- | --- |
| **参数处理** | **说明** |
| $# | 传递到脚本的参数个数 |
| $\* | 以一个单字符串显示所有向脚本传递的参数 |
| $$ | 脚本运行的当前进程ID号 |
| $! | 后台运行的最后一个进程的ID号 |
| $@ | 与$\*相同，但是使用时加引号，并在引号中返回每个参数。 |
| $- | 显示Shell使用的当前选项，与set命令功能相同。 |
| $? | 显示最后命令的退出状态。0表示没有错误，其他任何值表明有错误 |

带参数的函数示例：

#!/bin/bash

# author:菜鸟教程

# url:www.runoob.com

funWithParam(){

echo "第一个参数为 $1 !"

echo "第二个参数为 $2 !"

echo "第十个参数为 $10 !"

echo "第十个参数为 ${10} !"

echo "第十一个参数为 ${11} !"

echo "参数总数有 $# 个!"

echo "作为一个字符串输出所有参数 $\* !"

}

funWithParam 1 2 3 4 5 6 7 8 9 34 73

输出结果：

第一个参数为 1 !

第二个参数为 2 !

第十个参数为 10 !

第十个参数为 34 !

第十一个参数为 73 !

参数总数有 11 个!

作为一个字符串输出所有参数 1 2 3 4 5 6 7 8 9 34 73 !

## 二十一、Shell 输入/输出重定向

### **1 大多数 UNIX 系统命令**

从你的终端接受输入并将所产生的输出发送回到您的终端。一个命令通常从一个叫标准输入的地方读取输入，默认情况下，这恰好是你的终端。同样，一个命令通常将其输出写入到标准输出，默认情况下，这也是你的终端。

### 2 重定向命令列表如下：

|  |  |
| --- | --- |
| **命令** | **说明** |
| command > file | 将输出重定向到 file。 |
| command < file | 将输入重定向到 file。 |
| command >> file | 将输出以追加的方式重定向到 file。 |
| n > file | 将文件描述符为 n 的文件重定向到 file。 |
| n >> file | 将文件描述符为 n 的文件以追加的方式重定向到 file。 |
| n >& m | 将输出文件 m 和 n 合并。 |
| n <& m | 将输入文件 m 和 n 合并。 |
| << tag | 将开始标记 tag 和结束标记 tag 之间的内容作为输入。 |

需要注意的是文件描述符 0 通常是标准输入（STDIN），1 是标准输出（STDOUT），2 是标准错误输出（STDERR）。

### 3 输出重定向

重定向一般通过在命令间插入特定的符号来实现。特别的，这些符号的语法如下所示:

command1 > file1

上面这个命令执行command1然后将输出的内容存入file1。

注意任何file1内的已经存在的内容将被新内容替代。如果要将新内容添加在文件末尾，请使用>>操作符。

实例

执行下面的 who 命令，它将命令的完整的输出重定向在用户文件中(users):

$ who > users

执行后，并没有在终端输出信息，这是因为输出已被从默认的标准输出设备（终端）重定向到指定的文件。

你可以使用 cat 命令查看文件内容：

$ cat users

\_mbsetupuser console Oct 31 17:35

tianqixin console Oct 31 17:35

tianqixin ttys000 Dec 1 11:33

输出重定向会覆盖文件内容，请看下面的例子：

$ echo "菜鸟教程：www.runoob.com" > users

$ cat users

菜鸟教程：www.runoob.com

$

如果不希望文件内容被覆盖，可以使用 >> 追加到文件末尾，例如：

$ echo "菜鸟教程：www.runoob.com" >> users

$ cat users

菜鸟教程：www.runoob.com

菜鸟教程：www.runoob.com

$

**4 输入重定向**

和输出重定向一样，Unix 命令也可以从文件获取输入，语法为：

command1 < file1

这样，本来需要从键盘获取输入的命令会转移到文件读取内容。

注意：输出重定向是大于号(>)，输入重定向是小于号(<)。

实例

接着以上实例，我们需要统计 users 文件的行数,执行以下命令：

$ wc -l users

2 users

也可以将输入重定向到 users 文件：

$ wc -l < users

2

注意：上面两个例子的结果不同：第一个例子，会输出文件名；第二个不会，因为它仅仅知道从标准输入读取内容。

command1 < infile > outfile

同时替换输入和输出，执行command1，从文件infile读取内容，然后将输出写入到outfile中。

**5 重定向深入讲解**

一般情况下，每个 Unix/Linux 命令运行时都会打开三个文件：

标准输入文件(stdin)：stdin的文件描述符为0，Unix程序默认从stdin读取数据。

标准输出文件(stdout)：stdout 的文件描述符为1，Unix程序默认向stdout输出数据。

标准错误文件(stderr)：stderr的文件描述符为2，Unix程序会向stderr流中写入错误信息。

默认情况下，command > file 将 stdout 重定向到 file，command < file 将stdin 重定向到 file。

如果希望 stderr 重定向到 file，可以这样写：

$ command 2 > file

如果希望 stderr 追加到 file 文件末尾，可以这样写：

$ command 2 >> file

如果希望将 stdout 和 stderr 合并后重定向到 file，可以这样写：

$ command > file 2>&1

或者

$ command >> file 2>&1

如果希望对 stdin 和 stdout 都重定向，可以这样写：

$ command < file1 >file2

command 命令将 stdin 重定向到 file1，将 stdout 重定向到 file2。

## 二十二、Shell实例汇总

### 1、编写hello world脚本

#!/bin/bash

# 编写hello world脚本

echo "Hello World!"

### 2、通过位置变量创建 Linux 系统账户及密码

#!/bin/bash

# 通过位置变量创建 Linux 系统账户及密码

#$1 是执行脚本的第一个参数,$2 是执行脚本的第二个参数

useradd "$1"

echo "$2" | passwd ‐‐stdin "$2"

### 3、备份日志

#!/bin/bash

# 每周 5 使用 tar 命令备份/var/log 下的所有日志文件

# vim /root/logbak.sh

# 编写备份脚本,备份后的文件名包含日期标签,防止后面的备份将前面的备份数据覆盖

# 注意 date 命令需要使用反引号括起来,反引号在键盘<tab>键上面

tar -czf log-`date +%Y%m%d`.tar.gz /var/log

# crontab ‐e #编写计划任务,执行备份脚本

00 03 \* \* 5 /root/logbak.sh

### 4、一键部署 LNMP(RPM 包版本)

#!/bin/bash

# 一键部署 LNMP(RPM 包版本)

# 使用 yum 安装部署 LNMP,需要提前配置好 yum 源,否则该脚本会失败

# 本脚本使用于 centos7.2 或 RHEL7.2

yum ‐y install httpd

yum ‐y install mariadb mariadb‐devel mariadb‐server

yum ‐y install php php‐mysql

systemctl start httpd mariadb

systemctl enable httpd mariadb

### 5、监控内存和磁盘容量，小于给定值时报警

#!/bin/bash

# 实时监控本机内存和硬盘剩余空间,剩余内存小于500M、根分区剩余空间小于1000M时,发送报警邮件给root管理员

# 提取根分区剩余空间

disk\_size=$(df / | awk '/\//{print $4}')

# 提取内存剩余空间

mem\_size=$(free | awk '/Mem/{print $4}')

while :

do

# 注意内存和磁盘提取的空间大小都是以 Kb 为单位

if [ $disk\_size -le 512000 -a $mem\_size -le 1024000 ]

then

mail ‐s "Warning" root <<EOF

Insufficient resources,资源不足

EOF

fi

done

### 6、猜数字游戏

#!/bin/bash

# 脚本生成一个 100 以内的随机数,提示用户猜数字,根据用户的输入,提示用户猜对了,

# 猜小了或猜大了,直至用户猜对脚本结束。

# RANDOM 为系统自带的系统变量,值为 0‐32767的随机数

# 使用取余算法将随机数变为 1‐100 的随机数

num=$[RANDOM%100+1]

echo "$num"

# 使用 read 提示用户猜数字

# 使用 if 判断用户猜数字的大小关系:‐eq(等于),‐ne(不等于),‐gt(大于),‐ge(大于等于),

# ‐lt(小于),‐le(小于等于)

while :

do

read -p "计算机生成了一个 1‐100 的随机数,你猜: " cai

if [ $cai -eq $num ]

then

echo "恭喜,猜对了"

exit

elif [ $cai -gt $num ]

then

echo "Oops,猜大了"

else

echo "Oops,猜小了"

fi

done

### 7、检测本机当前用户是否为超级管理员,如果是管理员,则使用 yum 安装 vsftpd,如果不是,则提示您非管理员(使用字串对比版本)

#!/bin/bash

# 检测本机当前用户是否为超级管理员,如果是管理员,则使用 yum 安装 vsftpd,如果不

# 是,则提示您非管理员(使用字串对比版本)

if [ $USER == "root" ]

then

yum ‐y install vsftpd

else

echo "您不是管理员,没有权限安装软件"

fi

### 8、检测本机当前用户是否为超级管理员,如果是管理员,则使用 yum 安装 vsftpd,如果不是,则提示您非管理员(使用 UID 数字对比版本)

#!/bin/bash

# 检测本机当前用户是否为超级管理员,如果是管理员,则使用 yum 安装 vsftpd,如果不

# 是,则提示您非管理员(使用 UID 数字对比版本)

if [ $UID -eq 0 ];then

yum ‐y install vsftpd

else

echo "您不是管理员,没有权限安装软件"

fi

### 9、编写脚本:提示用户输入用户名和密码,脚本自动创建相应的账户及配置密码。如果用户不输入账户名,则提示必须输入账户名并退出脚本;如果用户不输入密码,则统一使用默认的 123456 作为默认密码。

#!/bin/bash

# 编写脚本:提示用户输入用户名和密码,脚本自动创建相应的账户及配置密码。如果用户

# 不输入账户名,则提示必须输入账户名并退出脚本;如果用户不输入密码,则统一使用默

# 认的 123456 作为默认密码。

read -p "请输入用户名: " user

#使用‐z 可以判断一个变量是否为空,如果为空,提示用户必须输入账户名,并退出脚本,退出码为 2

#没有输入用户名脚本退出后,使用$?查看的返回码为 2

if [ -z $user ];then

echo "您不需输入账户名"

exit 2

fi

#使用 stty ‐echo 关闭 shell 的回显功能

#使用 stty echo 打开 shell 的回显功能

stty -echo

read -p "请输入密码: " pass

stty echo

pass=${pass:‐123456}

useradd "$user"

echo "$pass" | passwd ‐‐stdin "$user"

### 10、输入三个数并进行升序排序

#!/bin/bash

# 依次提示用户输入 3 个整数,脚本根据数字大小依次排序输出 3 个数字

read -p "请输入一个整数:" num1

read -p "请输入一个整数:" num2

read -p "请输入一个整数:" num3

# 不管谁大谁小,最后都打印 echo "$num1,$num2,$num3"

# num1 中永远存最小的值,num2 中永远存中间值,num3 永远存最大值

# 如果输入的不是这样的顺序,则改变数的存储顺序,如:可以将 num1 和 num2 的值对调

tmp=0

# 如果 num1 大于 num2,就把 num1 和和 num2 的值对调,确保 num1 变量中存的是最小值

if [ $num1 -gt $num2 ];then

tmp=$num1

num1=$num2

num2=$tmp

fi

# 如果 num1 大于 num3,就把 num1 和 num3 对调,确保 num1 变量中存的是最小值

if [ $num1 -gt $num3 ];then

tmp=$num1

num1=$num3

num3=$tmp

fi

# 如果 num2 大于 num3,就把 num2 和 num3 对标,确保 num2 变量中存的是小一点的值

if [ $num2 -gt $num3 ];then

tmp=$num2

num2=$num3

num3=$tmp

fi

echo "排序后数据(从小到大)为:$num1,$num2,$num3"

### 11、石头、剪刀、布游戏

#!/bin/bash

# 编写脚本,实现人机<石头,剪刀,布>游戏

game=(石头 剪刀 布)

num=$[RANDOM%3]

computer=${game[$num]}

# 通过随机数获取计算机的出拳

# 出拳的可能性保存在一个数组中,game[0],game[1],game[2]分别是 3 中不同的可能

echo "请根据下列提示选择您的出拳手势"

echo "1.石头"

echo "2.剪刀"

echo "3.布"

read -p "请选择 1‐3:" person

case $person in

1)

if [ $num -eq 0 ]

then

echo "平局"

elif [ $num -eq 1 ]

then

echo "你赢"

else

echo "计算机赢"

fi;;

2)

if [ $num -eq 0 ]

then

echo "计算机赢"

elif [ $num -eq 1 ]

then

echo "平局"

else

echo "你赢"

fi;;

3)

if [ $num -eq 0 ]

then

echo "你赢"

elif [ $num -eq 1 ]

then

echo "计算机赢"

else

echo "平局"

fi;;

\*)

echo "必须输入 1‐3 的数字"

esac

### 12、编写脚本测试 192.168.4.0/24 整个网段中哪些主机处于开机状态,哪些主机处于关机状态(for 版本)

#!/bin/bash

# 编写脚本测试 192.168.4.0/24 整个网段中哪些主机处于开机状态,哪些主机处于关机

# 状态(for 版本)

for i in {1..254}

do

# 每隔0.3秒ping一次，一共ping2次，并以1毫秒为单位设置ping的超时时间

ping ‐c 2 ‐i 0.3 ‐W 1 192.168.4.$i &>/dev/null

if [ $? -eq 0 ];then

echo "192.168.4.$i is up"

else

echo "192.168.4.$i is down"

fi

done

### 13、编写脚本测试 192.168.4.0/24 整个网段中哪些主机处于开机状态,哪些主机处于关机状态(while 版本)

#!/bin/bash

# 编写脚本测试 192.168.4.0/24 整个网段中哪些主机处于开机状态,哪些主机处于关机

# 状态(while 版本)

i=1

while [ $i -le 254 ]

do

ping ‐c 2 ‐i 0.3 ‐W 1 192.168.4.$i &>/dev/null

if [ $? -eq 0 ];then

echo "192.168.4.$i is up"

else

echo "192.168.4.$i is down"

fi

let i++

done

### 14、编写脚本测试 192.168.4.0/24 整个网段中哪些主机处于开机状态,哪些主机处于关机状态(多进程版)

#!/bin/bash

# 编写脚本测试 192.168.4.0/24 整个网段中哪些主机处于开机状态,哪些主机处于关机

# 状态(多进程版)

#定义一个函数,ping 某一台主机,并检测主机的存活状态

myping(){

ping ‐c 2 ‐i 0.3 ‐W 1 $1 &>/dev/null

if [ $? -eq 0 ];then

echo "$1 is up"

else

echo "$1 is down"

fi

}

for i in {1..254}

do

myping 192.168.4.$i &

done

# 使用&符号,将执行的函数放入后台执行

# 这样做的好处是不需要等待ping第一台主机的回应,就可以继续并发ping第二台主机,依次类推。

### 15、编写脚本,显示进度条

#!/bin/bash

# 编写脚本,显示进度条

jindu(){

while :

do

echo -n '#'

sleep 0.2

done

}

jindu &

cp -a $1 $2

killall $1

echo "拷贝完成"

### 16、进度条,动态时针版本；定义一个显示进度的函数,屏幕快速显示|  / ‐ \

#!/bin/bash

# 进度条,动态时针版本

# 定义一个显示进度的函数,屏幕快速显示| / ‐ \

rotate\_line(){

INTERVAL=0.5 #设置间隔时间

COUNT="0" #设置4个形状的编号,默认编号为 0(不代表任何图像)

while :

do

COUNT=`expr $COUNT + 1` #执行循环,COUNT 每次循环加 1,(分别代表4种不同的形状)

case $COUNT in #判断 COUNT 的值,值不一样显示的形状就不一样

"1") #值为 1 显示‐

echo -e '‐'"\b\c"

sleep $INTERVAL

;;

"2") #值为 2 显示\\,第一个\是转义

echo -e '\\'"\b\c"

sleep $INTERVAL

;;

"3") #值为 3 显示|

echo -e "|\b\c"

sleep $INTERVAL

;;

"4") #值为 4 显示/

echo -e "/\b\c"

sleep $INTERVAL

;;

\*) #值为其他时,将 COUNT 重置为 0

COUNT="0";;

esac

done

}

rotate\_line

### 17、9\*9 乘法表

#!/bin/bash

# 9\*9 乘法表(编写 shell 脚本,打印 9\*9 乘法表)

for i in `seq 9`

do

for j in `seq $i`

do

echo -n "$j\*$i=$[i\*j] "

done

echo

done

### 18、使用死循环实时显示 eth0 网卡发送的数据包流量

#!/bin/bash

# 使用死循环实时显示 eth0 网卡发送的数据包流量

while :

do

echo '本地网卡 eth0 流量信息如下: '

ifconfig eth0 | grep "RX pack" | awk '{print $5}'

ifconfig eth0 | grep "TX pack" | awk '{print $5}'

sleep 1

done

### 19、使用 user.txt 文件中的人员名单,在计算机中自动创建对应的账户并配置初始密码本脚本执行,需要提前准备一个 user.txt 文件,该文件中包含有若干用户名信息

#!/bin/bash

# 使用 user.txt 文件中的人员名单,在计算机中自动创建对应的账户并配置初始密码

# 本脚本执行,需要提前准备一个 user.txt 文件,该文件中包含有若干用户名信息

for i in `cat user.txt`

do

useradd $i

echo "123456" | passwd ‐‐stdin $i

done

### 20、编写批量修改扩展名脚本

#!/bin/bash

# 编写批量修改扩展名脚本,如批量将 txt 文件修改为 doc 文件

# 执行脚本时,需要给脚本添加位置参数

# 脚本名 txt doc(可以将 txt 的扩展名修改为 doc)

# 脚本名 doc jpg(可以将 doc 的扩展名修改为 jpg)

for i in "ls \*.$1"

do

mv $i ${i%.\*}.$2

done

### 21、使用 expect 工具自动交互密码远程其他主机安装 httpd 软件

#!/bin/bash

# 使用 expect 工具自动交互密码远程其他主机安装 httpd 软件

# 删除~/.ssh/known\_hosts 后,ssh 远程任何主机都会询问是否确认要连接该主机

rm ‐rf ~/.ssh/known\_hosts

expect <<EOF

spawn ssh 192.168.4.254

expect "yes/no" {send "yes\r"}

# 根据自己的实际情况将密码修改为真实的密码字串

expect "password" {send "密码\r"}

expect "#" {send "yum ‐y install httpd\r"}

expect "#" {send "exit\r"}

EOF

### 22、一键部署 LNMP(源码安装版本)

#!/bin/bash

# 一键部署 LNMP(源码安装版本)

menu()

{

clear

echo " ##############‐‐‐‐Menu‐‐‐‐##############"

echo "# 1. Install Nginx"

echo "# 2. Install MySQL"

echo "# 3. Install PHP"

echo "# 4. Exit Program"

echo " ########################################"

}

choice()

{

read -p "Please choice a menu[1‐9]:" select

}

install\_nginx()

{

id nginx &>/dev/null

if [ $? -ne 0 ];then

useradd -s /sbin/nologin nginx

fi

if [ -f nginx‐1.8.0.tar.gz ];then

tar -xf nginx‐1.8.0.tar.gz

cd nginx‐1.8.0

yum -y install gcc pcre‐devel openssl‐devel zlib‐devel make

./configure ‐‐prefix=/usr/local/nginx ‐‐with‐http\_ssl\_module

make

make install

ln -s /usr/local/nginx/sbin/nginx /usr/sbin/

cd ..

else

echo "没有 Nginx 源码包"

fi

}

install\_mysql()

{

yum -y install gcc gcc‐c++ cmake ncurses‐devel perl

id mysql &>/dev/null

if [ $? -ne 0 ];then

useradd -s /sbin/nologin mysql

fi

if [ -f mysql‐5.6.25.tar.gz ];then

tar -xf mysql‐5.6.25.tar.gz

cd mysql‐5.6.25

cmake .

make

make install

/usr/local/mysql/scripts/mysql\_install\_db ‐‐user=mysql ‐‐datadir=/usr/local/mysql/data/

‐‐basedir=/usr/local/mysql/

chown -R root.mysql /usr/local/mysql

chown -R mysql /usr/local/mysql/data

/bin/cp -f /usr/local/mysql/support‐files/mysql.server /etc/init.d/mysqld

chmod +x /etc/init.d/mysqld

/bin/cp -f /usr/local/mysql/support‐files/my‐default.cnf /etc/my.cnf

echo "/usr/local/mysql/lib/" >> /etc/ld.so.conf

ldconfig

echo 'PATH=\$PATH:/usr/local/mysql/bin/' >> /etc/profile

export PATH

else

echo "没有 mysql 源码包"

exit

fi

}

install\_php()

{

#安装 php 时没有指定启动哪些模块功能,如果的用户可以根据实际情况自行添加额外功能如‐‐with‐gd 等

yum -y install gcc libxml2‐devel

if [ -f mhash‐0.9.9.9.tar.gz ];then

tar -xf mhash‐0.9.9.9.tar.gz

cd mhash‐0.9.9.9

./configure

make

make install

cd ..

if [ ! ‐f /usr/lib/libmhash.so ];then

ln -s /usr/local/lib/libmhash.so /usr/lib/

fi

ldconfig

else

echo "没有 mhash 源码包文件"

exit

fi

if [ -f libmcrypt‐2.5.8.tar.gz ];then

tar -xf libmcrypt‐2.5.8.tar.gz

cd libmcrypt‐2.5.8

./configure

make

make install

cd ..

if [ ! -f /usr/lib/libmcrypt.so ];then

ln -s /usr/local/lib/libmcrypt.so /usr/lib/

fi

ldconfig

else

echo "没有 libmcrypt 源码包文件"

exit

fi

if [ -f php‐5.4.24.tar.gz ];then

tar -xf php‐5.4.24.tar.gz

cd php‐5.4.24

./configure ‐‐prefix=/usr/local/php5 ‐‐with‐mysql=/usr/local/mysql ‐‐enable‐fpm ‐‐

enable‐mbstring ‐‐with‐mcrypt ‐‐with‐mhash ‐‐with‐config‐file‐path=/usr/local/php5/etc ‐‐with‐

mysqli=/usr/local/mysql/bin/mysql\_config

make && make install

/bin/cp -f php.ini‐production /usr/local/php5/etc/php.ini

/bin/cp -f /usr/local/php5/etc/php‐fpm.conf.default /usr/local/php5/etc/php‐fpm.conf

cd ..

else

echo "没有 php 源码包文件"

exit

fi

}

while :

do

menu

choice

case $select in

1)

install\_nginx

;;

2)

install\_mysql

;;

3)

install\_php

;;

4)

exit

;;

\*)

echo Sorry!

esac

done

### 23、编写脚本快速克隆 KVM 虚拟机

#!/bin/bash

# 编写脚本快速克隆 KVM 虚拟机

# 本脚本针对 RHEL7.2 或 Centos7.2

# 本脚本需要提前准备一个 qcow2 格式的虚拟机模板,

# 名称为/var/lib/libvirt/images /.rh7\_template 的虚拟机模板

# 该脚本使用 qemu‐img 命令快速创建快照虚拟机

# 脚本使用 sed 修改模板虚拟机的配置文件,将虚拟机名称、UUID、磁盘文件名、MAC 地址

# exit code:

# 65 ‐> user input nothing

# 66 ‐> user input is not a number

# 67 ‐> user input out of range

# 68 ‐> vm disk image exists

IMG\_DIR=/var/lib/libvirt/images

BASEVM=rh7\_template

read -p "Enter VM number: " VMNUM

if [ $VMNUM -le 9 ];then

VMNUM=0$VMNUM

fi

if [ -z "${VMNUM}" ]; then

echo "You must input a number."

exit 65

elif [[ ${VMNUM} =~ [a‐z] ]; then

echo "You must input a number."

exit 66

elif [ ${VMNUM} -lt 1 -o ${VMNUM} -gt 99 ]; then

echo "Input out of range"

exit 67

fi

NEWVM=rh7\_node${VMNUM}

if [ -e $IMG\_DIR/${NEWVM}.img ]; then

echo "File exists."

exit 68

fi

echo -en "Creating Virtual Machine disk image......\t"

qemu‐img create -f qcow2 ‐b $IMG\_DIR/.${BASEVM}.img $IMG\_DIR/${NEWVM}.img &> /dev/null

echo -e "\e[32;1m[OK]\e[0m"

#virsh dumpxml ${BASEVM} > /tmp/myvm.xml

cat /var/lib/libvirt/images/.rhel7.xml > /tmp/myvm.xml

sed -i "/<name>${BASEVM}/s/${BASEVM}/${NEWVM}/" /tmp/myvm.xml

sed -i "/uuid/s/<uuid>.\*<\/uuid>/<uuid>$(uuidgen)<\/uuid>/" /tmp/myvm.xml

sed -i "/${BASEVM}\.img/s/${BASEVM}/${NEWVM}/" /tmp/myvm.xml

# 修改 MAC 地址,本例使用的是常量,每位使用该脚本的用户需要根据实际情况修改这些值

# 最好这里可以使用便利,这样更适合于批量操作,可以克隆更多虚拟机

sed -i "/mac /s/a1/0c/" /tmp/myvm.xml

echo -en "Defining new virtual machine......\t\t"

virsh define /tmp/myvm.xml &> /dev/null

echo -e "\e[32;1m[OK]\e[0m"

### 24、点名器脚本

#!/bin/bash

# 编写一个点名器脚本

# 该脚本,需要提前准备一个 user.txt 文件

# 该文件中需要包含所有姓名的信息,一行一个姓名,脚本每次随机显示一个姓名

while :

do

#统计 user 文件中有多少用户

line=`cat user.txt |wc ‐l`

num=$[RANDOM%line+1]

sed -n "${num}p" user.txt

sleep 0.2

clear

done

### 25、查看有多少远程的 IP 在连接本机

#!/bin/bash

# 查看有多少远程的 IP 在连接本机(不管是通过 ssh 还是 web 还是 ftp 都统计)

# 使用 netstat ‐atn 可以查看本机所有连接的状态,‐a 查看所有,

# -t仅显示 tcp 连接的信息,‐n 数字格式显示

# Local Address(第四列是本机的 IP 和端口信息)

# Foreign Address(第五列是远程主机的 IP 和端口信息)

# 使用 awk 命令仅显示第 5 列数据,再显示第 1 列 IP 地址的信息

# sort 可以按数字大小排序,最后使用 uniq 将多余重复的删除,并统计重复的次数

netstat -atn | awk '{print $5}' | awk '{print $1}' | sort -nr | uniq -c

### 26、对 100 以内的所有正整数相加求和(1+2+3+4...+100)

#!/bin/bash

# 对 100 以内的所有正整数相加求和(1+2+3+4...+100)

#seq 100 可以快速自动生成 100 个整数

sum=0

for i in `seq 100`

do

sum=$[sum+i]

done

echo "总和是:$sum"

### 27、统计 13:30 到 14:30 所有访问 apache 服务器的请求有多少个

#!/bin/bash

# 统计 13:30 到 14:30 所有访问 apache 服务器的请求有多少个

# awk 使用‐F 选项指定文件内容的分隔符是/或者:

# 条件判断$7:$8 大于等于 13:30,并且要求,$7:$8 小于等于 14:30

# 最后使用 wc ‐l 统计这样的数据有多少行,即多少个

awk -F "[ /:]" '$7":"$8>="13:30" && $7":"$8<="14:30"' /var/log/httpd/access\_log |wc -l

### 28、统计 13:30 到 14:30 所有访问本机 Aapche 服务器的远程 IP 地址是什么

#!/bin/bash

# 统计 13:30 到 14:30 所有访问本机 Aapche 服务器的远程 IP 地址是什么

# awk 使用‐F 选项指定文件内容的分隔符是/或者:

# 条件判断$7:$8 大于等于 13:30,并且要求,$7:$8 小于等于 14:30

# 日志文档内容里面,第 1 列是远程主机的 IP 地址,使用 awk 单独显示第 1 列即可

awk -F "[ /:]" '$7":"$8>="13:30" && $7":"$8<="14:30"{print $1}' /var/log/httpd/access\_log

### 29、打印国际象棋棋盘

#!/bin/bash

# 打印国际象棋棋盘

# 设置两个变量,i 和 j,一个代表行,一个代表列,国际象棋为 8\*8 棋盘

# i=1 是代表准备打印第一行棋盘,第 1 行棋盘有灰色和蓝色间隔输出,总共为 8 列

# i=1,j=1 代表第 1 行的第 1 列;i=2,j=3 代表第 2 行的第 3 列

# 棋盘的规律是 i+j 如果是偶数,就打印蓝色色块,如果是奇数就打印灰色色块

# 使用 echo ‐ne 打印色块,并且打印完成色块后不自动换行,在同一行继续输出其他色块

for i in {1..8}

do

for j in {1..8}

do

sum=$[i+j]

if [ $[sum%2] -eq 0 ];then

echo -ne "\033[46m \033[0m"

else

echo -ne "\033[47m \033[0m"

fi

done

echo

done

### 30、统计每个远程 IP 访问了本机 apache 几次?

#!/bin/bash

# 统计每个远程 IP 访问了本机 apache 几次?

awk '{ip[$1]++}END{for(i in ip){print ip[i],i}}' /var/log/httpd/access\_log

### 31、统计当前 Linux 系统中可以登录计算机的账户有多少个

#!/bin/bash

# 统计当前 Linux 系统中可以登录计算机的账户有多少个

#方法 1:

grep "bash$" /etc/passwd | wc -l

#方法 2:

awk -f: '/bash$/{x++}end{print x}' /etc/passwd

### 32、统计/var/log 有多少个文件,并显示这些文件名

#!/bin/bash

# 统计/var/log 有多少个文件,并显示这些文件名

# 使用 ls 递归显示所有,再判断是否为文件,如果是文件则计数器加 1

cd /var/log

sum=0

for i in `ls -r \*`

do

if [ -f $i ];then

let sum++

echo "文件名:$i"

fi

done

echo "总文件数量为:$sum"

### 33、自动为其他脚本添加解释器信息

#!/bin/bash

# 自动为其他脚本添加解释器信息#!/bin/bash,如脚本名为 test.sh 则效果如下:

# ./test.sh abc.sh 自动为 abc.sh 添加解释器信息

# ./test.sh user.sh 自动为 user.sh 添加解释器信息

# 先使用 grep 判断对象脚本是否已经有解释器信息,如果没有则使用 sed 添加解释器以及描述信息

if ! grep -q "^#!" $1; then

sed '1i #!/bin/bash' $1

sed '2i #Description: '

fi

# 因为每个脚本的功能不同,作用不同,所以在给对象脚本添加完解释器信息,以及 Description 后还希望

# 继续编辑具体的脚本功能的描述信息,这里直接使用 vim 把对象脚本打开,并且光标跳转到该文件的第 2 行

vim +2 $1

### 34、自动化部署 varnish 源码包软件

#!/bin/bash

# 自动化部署 varnish 源码包软件

# 本脚本需要提前下载 varnish‐3.0.6.tar.gz 这样一个源码包软件,该脚本即可用自动源码安装部署软件

yum -y install gcc readline‐devel pcre‐devel

useradd -s /sbin/nologin varnish

tar -xf varnish‐3.0.6.tar.gz

cd varnish‐3.0.6

# 使用 configure,make,make install 源码安装软件包

./configure ‐‐prefix=/usr/local/varnish

make && make install

# 在源码包目录下,将相应的配置文件拷贝到 Linux 系统文件系统中

# 默认安装完成后,不会自动拷贝或安装配置文件到 Linux 系统,所以需要手动 cp 复制配置文件

# 并使用 uuidgen 生成一个随机密钥的配置文件

cp redhat/varnish.initrc /etc/init.d/varnish

cp redhat/varnish.sysconfig /etc/sysconfig/varnish

cp redhat/varnish\_reload\_vcl /usr/bin/

ln -s /usr/local/varnish/sbin/varnishd /usr/sbin/

ln -s /usr/local/varnish/bin/\* /usr/bin

mkdir /etc/varnish

cp /usr/local/varnish/etc/varnish/default.vcl /etc/varnish/

uuidgen > /etc/varnish/secret

### 35、编写 nginx 启动脚本

#!/bin/bash

# 编写 nginx 启动脚本

# 本脚本编写完成后,放置在/etc/init.d/目录下,就可以被 Linux 系统自动识别到该脚本

# 如果本脚本名为/etc/init.d/nginx,则 service nginx start 就可以启动该服务

# service nginx stop 就可以关闭服务

# service nginx restart 可以重启服务

# service nginx status 可以查看服务状态

program=/usr/local/nginx/sbin/nginx

pid=/usr/local/nginx/logs/nginx.pid

start(){

if [ -f $pid ];then

echo "nginx 服务已经处于开启状态"

else

$program

fi

stop(){

if [ -! -f $pid ];then

echo "nginx 服务已经关闭"

else

$program -s stop

echo "关闭服务 ok"

fi

}

status(){

if [ -f $pid ];then

echo "服务正在运行..."

else

echo "服务已经关闭"

fi

}

case $1 in

start)

start;;

stop)

stop;;

restart)

stop

sleep 1

start;;

status)

status;;

\*)

echo "你输入的语法格式错误"

esac

### 36、自动对磁盘分区、格式化、挂载

#!/bin/bash

# 自动对磁盘分区、格式化、挂载

# 对虚拟机的 vdb 磁盘进行分区格式化,使用<<将需要的分区指令导入给程序 fdisk

# n(新建分区),p(创建主分区),1(分区编号为 1),两个空白行(两个回车,相当于将整个磁盘分一个区)

# 注意:1 后面的两个回车(空白行)是必须的!

fdisk /dev/vdb << EOF

n

p

1

wq

EOF

#格式化刚刚创建好的分区

mkfs.xfs /dev/vdb1

#创建挂载点目录

if [ -e /data ]; then

exit

fi

mkdir /data

#自动挂载刚刚创建的分区,并设置开机自动挂载该分区

echo '/dev/vdb1 /data xfs defaults 1 2' >> /etc/fstab

mount -a

### 37、自动优化 Linux 内核参数

#!/bin/bash

# 自动优化 Linux 内核参数

#脚本针对 RHEL7

cat >> /usr/lib/sysctl.d/00‐system.conf <<EOF

fs.file‐max=65535

net.ipv4.tcp\_timestamps = 0

net.ipv4.tcp\_synack\_retries = 5

net.ipv4.tcp\_syn\_retries = 5

net.ipv4.tcp\_tw\_recycle = 1

net.ipv4.tcp\_tw\_reuse = 1

net.ipv4.tcp\_fin\_timeout = 30

#net.ipv4.tcp\_keepalive\_time = 120

net.ipv4.ip\_local\_port\_range = 1024 65535

kernel.shmall = 2097152

kernel.shmmax = 2147483648

kernel.shmmni = 4096

kernel.sem = 5010 641280 5010 128

net.core.wmem\_default=262144

net.core.wmem\_max=262144

net.core.rmem\_default=4194304

net.core.rmem\_max=4194304

net.ipv4.tcp\_fin\_timeout = 10

net.ipv4.tcp\_keepalive\_time = 30

net.ipv4.tcp\_window\_scaling = 0

net.ipv4.tcp\_sack = 0

EOF

sysctl –p

### 38、切割 Nginx 日志文件(防止单个文件过大,后期处理很困难)

#mkdir /data/scripts

#vim /data/scripts/nginx\_log.sh

#!/bin/bash

# 切割 Nginx 日志文件(防止单个文件过大,后期处理很困难)

logs\_path="/usr/local/nginx/logs/"

mv ${logs\_path}access.log ${logs\_path}access\_$(date -d "yesterday" +"%Y%m%d").log

kill -USR1 `cat /usr/local/nginx/logs/nginx.pid`

# chmod +x /data/scripts/nginx\_log.sh

# crontab ‐e #脚本写完后,将脚本放入计划任务每天执行一次脚本

0 1 \* \* \* /data/scripts/nginx\_log.sh

### 39、检测 MySQL 数据库连接数量

#!/bin/bash

# 检测 MySQL 数据库连接数量

# 本脚本每 2 秒检测一次 MySQL 并发连接数,可以将本脚本设置为开机启动脚本,或在特定时间段执行

# 以满足对 MySQL 数据库的监控需求,查看 MySQL 连接是否正常

# 本案例中的用户名和密码需要根据实际情况修改后方可使用

log\_file=/var/log/mysql\_count.log

user=root

passwd=123456

while :

do

sleep 2

count=`mysqladmin -u "$user" -p "$passwd" status | awk '{print $4}'`

echo "`date +%Y‐%m‐%d` 并发连接数为:$count" >> $log\_file

done

### 40、根据 md5 校验码,检测文件是否被修改

#!/bin/bash

# 根据 md5 校验码,检测文件是否被修改

# 本示例脚本检测的是/etc 目录下所有的 conf 结尾的文件,根据实际情况,您可以修改为其他目录或文件

# 本脚本在目标数据没有被修改时执行一次,当怀疑数据被人篡改,再执行一次

# 将两次执行的结果做对比,MD5 码发生改变的文件,就是被人篡改的文件

for i in $(ls /etc/\*.conf)

do

md5sum "$i" >> /var/log/conf\_file.log

done

### 41、检测 MySQL 服务是否存活

#!/bin/bash

# 检测 MySQL 服务是否存活

# host 为你需要检测的 MySQL 主机的 IP 地址,user 为 MySQL 账户名,passwd 为密码

# 这些信息需要根据实际情况修改后方可使用

host=192.168.51.198

user=root

passwd=123456

mysqladmin -h '$host' -u '$user' -p'$passwd' ping &>/dev/null

if [ $? -eq 0 ]

then

echo "MySQL is UP"

else

echo "MySQL is down"

fi

### 42、备份 MySQL 的 shell 脚本(mysqldump版本)

#!/bin/bash

# 备份 MySQL 的 shell 脚本(mysqldump版本)

# 定义变量 user(数据库用户名),passwd(数据库密码),date(备份的时间标签)

# dbname(需要备份的数据库名称,根据实际需求需要修改该变量的值,默认备份 mysql 数据库)

user=root

passwd=123456

dbname=mysql

date=$(date +%Y%m%d)

# 测试备份目录是否存在,不存在则自动创建该目录

[ ! -d /mysqlbackup ] && mkdir /mysqlbackup

# 使用 mysqldump 命令备份数据库

mysqldump -u "$user" -p "$passwd" "$dbname" > /mysqlbackup/"$dbname"-${date}.sql

### 43、将文件中所有的小写字母转换为大写字母

#!/bin/bash

# 将文件中所有的小写字母转换为大写字母

# $1是位置参数,是你需要转换大小写字母的文件名称

# 执行脚本,给定一个文件名作为参数,脚本就会将该文件中所有的小写字母转换为大写字母

tr "[a‐z]" "[A‐Z]" < $1

### 44、非交互自动生成 SSH 密钥文件

#!/bin/bash

# 非交互自动生成 SSH 密钥文件

# ‐t 指定 SSH 密钥的算法为 RSA 算法;‐N 设置密钥的密码为空;‐f 指定生成的密钥文件>存放在哪里

rm -rf ~/.ssh/{known\_hosts,id\_rsa\*}

ssh‐keygen -t RSA -N '' -f ~/.ssh/id\_rsa

### 45、检查特定的软件包是否已经安装

#!/bin/bash

# 检查特定的软件包是否已经安装

if [ $# -eq 0 ];then

echo "你需要制定一个软件包名称作为脚本参数"

echo "用法:$0 软件包名称 ..."

fi

# $@提取所有的位置变量的值,相当于$\*

for package in "$@"

do

if rpm -q ${package} &>/dev/null ;then

echo -e "${package}\033[32m 已经安装\033[0m"

else

echo -e "${package}\033[34;1m 未安装\033[0m"

fi

done

### 46、监控 HTTP 服务器的状态(测试返回码)

#!/bin/bash

# 监控 HTTP 服务器的状态(测试返回码)

# 设置变量,url为你需要检测的目标网站的网址(IP 或域名),比如百度

url=http://http://183.232.231.172/index.html

# 定义函数 check\_http:

# 使用 curl 命令检查 http 服务器的状态

# ‐m 设置curl不管访问成功或失败,最大消耗的时间为 5 秒,5 秒连接服务为相应则视为无法连接

# ‐s 设置静默连接,不显示连接时的连接速度、时间消耗等信息

# ‐o 将 curl 下载的页面内容导出到/dev/null(默认会在屏幕显示页面内容)

# ‐w 设置curl命令需要显示的内容%{http\_code},指定curl返回服务器的状态码

check\_http()

{

status\_code=$(curl -m 5 -s -o /dev/null -w %{http\_code} $url)

}

while :

do

check\_http

date=$(date +%Y%m%d‐%H:%M:%S)

# 生成报警邮件的内容

echo "当前时间为:$date

$url 服务器异常,状态码为${status\_code}.

请尽快排查异常." > /tmp/http$$.pid

# 指定测试服务器状态的函数,并根据返回码决定是发送邮件报警还是将正常信息写入日志

if [ $status\_code -ne 200 ];then

mail -s Warning root < /tmp/http$$.pid

else

echo "$url 连接正常" >> /var/log/http.log

fi

sleep 5

done

### 47、自动添加防火墙规则,开启某些服务或端口(适用于 RHEL7)

#!/bin/bash

# 自动添加防火墙规则,开启某些服务或端口(适用于 RHEL7)

#

# 设置变量定义需要添加到防火墙规则的服务和端口号

# 使用 firewall‐cmd ‐‐get‐services 可以查看 firewall 支持哪些服务

service="nfs http ssh"

port="80 22 8080"

# 循环将每个服务添加到防火墙规则中

for i in $service

do

echo "Adding $i service to firewall"

firewall‐cmd --add-service=${i}

done

#循环将每个端口添加到防火墙规则中

for i in $port

do

echo "Adding $i Port to firewall"

firewall‐cmd --add-port=${i}/tcp

done

#将以上设置的临时防火墙规则,转换为永久有效的规则(确保重启后有效)

firewall‐cmd --runtime-to-permanent

### 48、使用脚本自动创建逻辑卷

#!/bin/bash

# 使用脚本自动创建逻辑卷

# 清屏,显示警告信息,创建将磁盘转换为逻辑卷会删除数据

clear

echo -e "\033[32m !!!!!!警告(Warning)!!!!!!\033[0m"

echo

echo "+++++++++++++++++++++++++++++++++++++++++++++++++"

echo "脚本会将整个磁盘转换为 PV,并删除磁盘上所有数据!!!"

echo "This Script will destroy all data on the Disk"

echo "+++++++++++++++++++++++++++++++++++++++++++++++++"

echo

read -p "请问是否继续 y/n?:" sure

# 测试用户输入的是否为 y,如果不是则退出脚本

[ $sure != y ] && exit

# 提示用户输入相关参数(磁盘、卷组名称等数据),并测试用户是否输入了这些值,如果没有输入,则脚本退出

read -p "请输入磁盘名称,如/dev/vdb:" disk

[ -z $disk ] && echo "没有输入磁盘名称" && exit

read -p "请输入卷组名称:" vg\_name

[ -z $vg\_name ] && echo "没有输入卷组名称" && exit

read -p "请输入逻辑卷名称:" lv\_name

[ -z $lv\_name ] && echo "没有输入逻辑卷名称" && exit

read -p "请输入逻辑卷大小:" lv\_size

[ -z $lv\_size ] && echo "没有输入逻辑卷大小" && exit

# 使用命令创建逻辑卷

pvcreate $disk

vgcreate $vg\_name $disk

lvcreate -L ${lv\_size}M -n ${lv\_name} ${vg\_name}

### 49、显示 CPU 厂商信息

#!/bin/bash

# 显示 CPU 厂商信息

awk '/vendor\_id/{print $3}' /proc/cpuinfo | uniq

### 50、删除某个目录下大小为 0 的文件

#!/bin/bash

# 删除某个目录下大小为 0 的文件

#/var/www/html 为测试目录,脚本会清空该目录下所有 0 字节的文件

dir="/var/www/html"

find $dir -type f -size 0 -exec rm -rf {} \;

### 51、查找 Linux 系统中的僵尸进程

#!/bin/bash

# 查找 Linux 系统中的僵尸进程

# awk 判断 ps 命令输出的第 8 列为 Z 是,显示该进程的 PID 和进程命令

ps aux | awk '{if($8 == "Z"){print $2,$11}}'

### 52、提示用户输入年份后判断该年是否为闰年

#!/bin/bash

# 提示用户输入年份后判断该年是否为闰年

# 能被4整除并且并不能被100整除的年份是闰年

# 能被400整除的年份也是闰年

read -p "请输入一个年份:" year

if [ "$year" = "" ];then

echo "没有输入年份"

exit

fi

#使用正则测试变量 year 中是否包含大小写字母

if [[ "$year" =~ [a‐Z] ]];then

echo "你输入的不是数字"

exit

fi

# 判断是否为闰年

if [ $[year % 4] -eq 0 ] && [ $[year % 100] -ne 0 ];then

echo "$year年是闰年"

elif [ $[year % 400] -eq 0 ];then

echo "$year年是闰年"

else

echo "$year年不是闰年"

fi

### 53、生成随机密码(urandom 版本)

#!/bin/bash

# 生成随机密码(urandom 版本)

# /dev/urandom 文件是 Linux 内置的随机设备文件

# cat /dev/urandom 可以看看里面的内容,ctrl+c 退出查看

# 查看该文件内容后,发现内容有些太随机,包括很多特殊符号,我们需要的密码不希望使用这些符号

# tr ‐dc '\_A‐Za‐z0‐9' < /dev/urandom

# 该命令可以将随机文件中其他的字符删除,仅保留大小写字母,数字,下划线,但是内容还是太多

# 我们可以继续将优化好的内容通过管道传递给 head 命令,在大量数据中仅显示头 10 个字节

# 注意 A 前面有个下划线

tr -dc '\_A‐Za‐z0‐9' </dev/urandom | head -c 10

### 54、生成随机密码(字串截取版本)

#!/bin/bash

# 生成随机密码(字串截取版本)

# 设置变量 key,存储密码的所有可能性(密码库),如果还需要其他字符请自行添加其他密码字符

# 使用$#统计密码库的长度

key="0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"

num=${#key}

# 设置初始密码为空

pass=''

# 循环 8 次,生成随机密码

# 每次都是随机数对密码库的长度取余,确保提取的密码字符不超过密码库的长度

# 每次循环提取一位随机密码,并将该随机密码追加到 pass 变量的最后

for i in {1..8}

do

index=$[RANDOM%num]

pass=$pass${key:$index:1}

done

echo $pass

### 55、生成随机密码(UUID 版本,16 进制密码)

#!/bin/bash

# 生成随机密码(UUID 版本,16 进制密码)

uuidgen

### 56、生成随机密码(进程 ID 版本,数字密码)

#!/bin/bash

# 生成随机密码(进程 ID 版本,数字密码)

echo $$

### 57、测试用户名与密码是否正确

#!/bin/bash

# 测试用户名与密码是否正确

#用户名为 tom 并且密码为 123456,则提示登录成功,否则提示登录失败

read -p "请输入用户名:" user

read -p "请输入密码:" pass

if [ "$user" == 'tom' -a "$pass" == '123456' ];then

echo "Login successful"

else

echo "Login Failed"

fi

### 58、循环测试用户名与密码是否正确

#!/bin/bash

# 循环测试用户名与密码是否正确

# 循环测试用户的账户名和密码,最大测试 3 次,输入正确提示登录成功,否则提示登录失败

# 用户名为 tom 并且密码为 123456

for i in {1..3}

do

read -p "请输入用户名:" user

read -p "请输入密码:" pass

if [ "$user" == 'tom' -a "$pass" == '123456' ];then

echo "Login successful"

exit

fi

done

echo "Login Failed"

### 59、Shell 脚本的 fork 炸弹

#!/bin/bash

# Shell 脚本的 fork 炸弹

# 快速消耗计算机资源,致使计算机死机

# 定义函数名为.(点), 函数中递归调用自己并放入后台执行

.() { .|.& };.

### 60、批量下载有序文件(pdf、图片、视频等等)

#!/bin/bash

# 批量下载有序文件(pdf、图片、视频等等)

# 本脚本准备有序的网络资料进行批量下载操作(如 01.jpg,02.jpg,03.jpg)

# 设置资源来源的域名连接

url="http://www.baidu.com/"

echo "开始下载..."

sleep 2

type=jpg

for i in `seq 100`

echo "正在下载$i.$type"

curl $url/$i.$type -o /tmp/${i}$type

sleep 1

done

#curl 使用-o 选项指定下载文件另存到哪里.

### 61、显示当前计算机中所有账户的用户名称

#!/bin/bash

# 显示当前计算机中所有账户的用户名称

# 下面使用3种不同的方式列出计算机中所有账户的用户名

# 指定以:为分隔符,打印/etc/passwd 文件的第 1 列

awk -F: '{print $1}' /etc/passwd

# 指定以:为分隔符,打印/etc/passwd 文件的第 1 列

cut -d: -f1 /etc/passwd

# 使用 sed 的替换功能,将/etc/passwd 文件中:后面的所有内容替换为空(仅显示用户名)

sed 's/:.\*//' /etc/passwd

### 62、制定目录路径,脚本自动将该目录使用 tar 命令打包备份到/data目录

#!/bin/bash

# 制定目录路径,脚本自动将该目录使用 tar 命令打包备份到/data目录

[ ! -d /data ] && mkdir /data

[ -z $1 ] && exit

if [ -d $1 ];then

tar -czf /data/$1.-`date +%Y%m%d`.tar.gz $1

else

echo "该目录不存在"

fi

### 63、显示进度条(回旋镖版)

#!/bin/bash

# 显示进度条(回旋镖版)

while :

do

clear

for i in {1..20}

do

echo ‐e "\033[3;${i}H\*"

sleep 0.1

done

clear

for i in {20..1}

do

echo ‐e "\033[3;${i}H\*"

sleep 0.1

done

clear

done

### 64、安装 LAMP 环境(yum 版本)

#!/bin/bash

# 安装 LAMP 环境(yum 版本)

# 本脚本适用于 RHEL7(RHEL6 中数据库为 mysql)

yum makecache &>/dev/null

num=$(yum repolist | awk '/repolist/{print $2}' | sed 's/,//')

if [ $num -lt 0 ];then

yum -y install httpd

yum -y install mariadb mariadb-server mariadb-devel

yum -y install php php-mysql

else

echo "未配置 yum 源..."

fi

### 65、循环关闭局域网中所有主机

#!/bin/bash

# 循环关闭局域网中所有主机

# 假设本机为 192.168.4.100,编写脚本关闭除自己外的其他所有主机

# 脚本执行,需要提前给所有其他主机传递 ssh 密钥,满足无密码连接

for i in {1..254}

do

[ $i -eq 100 ] && continue

echo "正在关闭 192.168.4.$i..."

ssh 192.168.4.$i poweroff

done

### 66、获取本机 MAC 地址

#!/bin/bash

# 获取本机 MAC 地址

ip a s | awk 'BEGIN{print " 本 机 MAC 地 址 信 息 如 下 :"}/^[0‐9]/{print $2;getline;if($0~/link\/ether/){print $2}}' | grep -v lo:

# awk 读取 ip 命令的输出,输出结果中如果有以数字开始的行,先显示该行的地 2 列(网卡名称),

# 接着使用 getline 再读取它的下一行数据,判断是否包含 link/ether

# 如果保护该关键词,就显示该行的第 2 列(MAC 地址)

# lo 回环设备没有 MAC,因此将其屏蔽,不显示

### 67、自动配置 rsynd 服务器的配置文件 rsyncd.conf

#!/bin/bash

# 自动配置 rsynd 服务器的配置文件 rsyncd.conf

# See rsyncd.conf man page for more options.

[ ! -d /home/ftp ] && mkdir /home/ftp

echo 'uid = nobody

gid = nobody

use chroot = yes

max connections = 4

pid file = /var/run/rsyncd.pid

exclude = lost+found/

transfer logging = yes

timeout = 900

ignore nonreadable = yes

dont compress = \*.gz \*.tgz \*.zip \*.z \*.Z \*.rpm \*.deb \*.bz2

[ftp]

path = /home/ftp

comment = share' > /etc/rsyncd.conf

### 68、修改 Linux 系统的最大打开文件数量

#!/bin/bash

# 修改 Linux 系统的最大打开文件量

# 往/etc/security/limits.conf 文件的末尾追加两行配置参数,修改最大打开文件数量为 65536

cat >> /etc/security/limits.conf <<EOF

\* soft nofile 65536

\* hard nofile 65536

EOF

### 69、设置 Python 支持自动命令补齐功能

#!/bin/bash

# 设置 Python 支持自动命令补齐功能

# Summary:Enable tab complete for python

# Description:

Needs import readline and rlcompleter module

#

import readline

#

import rlcompleter

#

help(rlcompleter) display detail: readline.parse\_and\_bind('tab: complete')

#

man python display detail: PYTHONSTARTUP variable

if [ ! -f /usr/bin/tab.py ];then

cat >> /usr/bin/tab.py <<EOF

import readline

import rlcompleter

readline.parse\_and\_bind('tab: complete')

EOF

fi

sed -i '$a export PYTHONSTARTUP=/usr/bin/tab.py' /etc/profile

source /etc/profile

### 70、自动修改计划任务配置文件

#!/bin/bash

# 自动修改计划任务配置文件

read -p "请输入分钟信息(00‐59):" min

read -p "请输入小时信息(00‐24):" hour

read -p "请输入日期信息(01‐31):" date

read -p "请输入月份信息(01‐12):" month

read -p "请输入星期信息(00‐06):" weak

read -p "请输入计划任务需要执行的命令或脚本:" program

echo "$min $hour $date $month $weak $program" >> /etc/crontab

### 71、使用脚本循环创建三位数字的文本文件(111-999 的文件)

#!/bin/bash

# 使用脚本循环创建三位数字的文本文件(111-999 的文件)

for i in {1..9}

do

for j in {1..9}

do

for k in {1..9}

do

touch /tmp/$i$j$k.txt

done

done

done

### 72、找出/etc/passwd 中能登录的用户,并将对应在/etc/shadow 中第二列密码提出处理

#!/bin/bash

# 找出/etc/passwd 中能登录的用户,并将对应在/etc/shadow 中第二列密码提出处理

user=$(awk -F: '/bash$/{print $1}' /etc/passwd)

for i in $user

do

awk -F: -v x=$i '$1==x{print $1,$2}' /etc/shadow

done

### 73、统计/etc/passwd 中 root 出现的次数

#!/bin/bash

# 统计/etc/passwd 中 root 出现的次数

#每读取一行文件内容,即从第 1 列循环到最后 1 列,依次判断是否包含 root 关键词,如果包含则 x++

awk -F: '{i=1;while(i<=NF){if($i~/root/){x++};i++}} END{print "root 出现次数为"x}' /etc/passwd

### 74、统计 Linux 进程相关数量信息

#!/bin/bash

# 统计 Linux 进程相关数量信息

running=0

sleeping=0

stoped=0

zombie=0

# 在 proc 目录下所有以数字开始的都是当前计算机正在运行的进程的进程 PID

# 每个 PID 编号的目录下记录有该进程相关的信息

for pid in /proc/[1‐9]\*

do

procs=$[procs+1]

stat=$(awk '{print $3}' $pid/stat)

# 每个 pid 目录下都有一个 stat 文件,该文件的第 3 列是该进程的状态信息

case $stat in

R)

running=$[running+1]

;;

T)

stoped=$[stoped+1]

;;

S)

sleeping=$[sleeping+1]

;;

Z)

zombie=$[zombie+1]

;;

esac

done

echo "进程统计信息如下"

echo "总进程数量为:$procs"

echo "Running 进程数为:$running"

echo "Stoped 进程数为:$stoped"

echo "Sleeping 进程数为:$sleeping"

echo "Zombie 进程数为:$zombie"

### 75、从键盘读取一个论坛积分,判断论坛用户等级

#!/bin/bash

# 从键盘读取一个论坛积分,判断论坛用户等级

#等级分类如下:

# 大于等于 90 神功绝世

# 大于等于 80,小于 90 登峰造极

# 大于等于 70,小于 80 炉火纯青

# 大于等于 60,小于 70 略有小成

# 小于 60 初学乍练

read -p "请输入积分(0‐100):" JF

if [ $JF -ge 90 ] ; then

echo "$JF 分,神功绝世"

elif [ $JF -ge 80 ] ; then

echo "$JF 分,登峰造极"

elif [ $JF -ge 70 ] ; then

echo "$JF 分,炉火纯青"

elif [ $JF -lt 60 ] ; then

echo "$JF 分,略有小成"

else

echo "$JF 分,初学乍练"

fi

### 76、判断用户输入的数据类型(字母、数字或其他)

#!/bin/bash

# 判断用户输入的数据类型(字母、数字或其他)

read -p "请输入一个字符:" KEY

case "$KEY" in

[a‐z]|[A‐Z])

echo "字母"

;;

[0‐9])

echo "数字"

;;

\*)

echo "空格、功能键或其他控制字符"

esac

### 77、显示进度条(数字版)

#!/bin/bash

# 显示进度条(数字版)

# echo 使用‐e 选项后,在打印参数中可以指定 H,设置需要打印内容的 x,y 轴的定位坐标

# 设置需要打印内容在第几行,第几列

for i in {1..100}

do

echo -e "\033[6;8H["

echo -e "\033[6;9H$i%"

echo -e "\033[6;13H]"

sleep 0.1

done

### 78、打印斐波那契数列

#!/bin/bash

# 打印斐波那契数列(该数列的特点是后一个数字,永远都是前 2 个数字之和)

# 斐波那契数列后一个数字永远是前 2 个数字之和

# 如:0 1 1 2 3 5 8 13 ... ...

list=(0 1)

for i in `seq 2 11`

do

list[$i]=`expr ${list[‐1]} + ${list[‐2]}`

done

echo ${list[@]}

### 79、判断用户输入的是 Yes 或 NO

#!/bin/bash

# 判断用户输入的是 Yes 或 NO

read -p "Are you sure?[y/n]:" sure

case $sure in

y|Y|Yes|YES)

echo "you enter $a"

;;

n|N|NO|no)

echo "you enter $a"

;;

\*)

echo "error";;

esac

### 80、显示本机 Linux 系统上所有开放的端口列表

#!/bin/bash

# 显示本机 Linux 系统上所有开放的端口列表

# 从端口列表中观测有没有没用的端口,有的话可以将该端口对应的服务关闭,防止意外的攻击可能性

ss -nutlp | awk '{print $1,$5}' | awk -F"[: ]" '{print "协议:"$1,"端口号:"$NF}' | grep "[0‐9]" | uniq

### 81、将 Linux 系统中 UID 大于等于 1000 的普通用户都删除

#!/bin/bash

# 将 Linux 系统中 UID 大于等于 1000 的普通用户都删除

# 先用 awk 提取所有 uid 大于等于 1000 的普通用户名称

# 再使用 for 循环逐个将每个用户删除即可

user=$(awk -F: '$3>=1000{print $1}' /etc/passwd)

for i in $user

do

userdel -r $i

done

### 82、使用脚本开启关闭虚拟机

#!/bin/bash

# 使用脚本开启关闭虚拟机

# 脚本通过调用virsh命令实现对虚拟机的管理,如果没有该命令,需要安装 libvirt‐client 软件包

# $1是脚本的第1个参数,$2是脚本的第2个参数

# 第1个参数是你希望对虚拟机进行的操作指令,第2个参数是虚拟机名称

case $1 in

list)

virsh list --all

;;

start)

virsh start $2

;;

stop)

virsh destroy $2

;;

enable)

virsh autostart $2

;;

disable)

virsh autostart --disable $2

;;

\*)

echo "Usage:$0 list"

echo "Usage:$0 [start|stop|enable|disable] VM\_name"

cat << EOF

#list 显示虚拟机列表

#start 启动虚拟机

#stop 关闭虚拟机

#enable 设置虚拟机为开机自启

#disable 关闭虚拟机开机自启功能

EOF

;;

esac

### 83、调整虚拟机内存参数的 shell 脚本

#!/bin/bash

# 调整虚拟机内存参数的 shell 脚本

# 脚本通过调用 virsh 命令实现对虚拟机的管理,如果没有该命令,需要安装 libvirt‐client 软件包

cat << EOF

1.调整虚拟机最大内存数值

2.调整实际分配给虚拟机的内存数值

EOF

read -p "请选择[1‐2]:" select

case $select in

1)

read -p "请输入虚拟机名称" name

read -p "请输入最大内存数值(单位:k):" size

virsh setmaxmem $name --size $size --config

;;

2)

read -p "请输入虚拟机名称" name

read -p "请输入实际分配内存数值(单位:k):" size

virsh setmem $name $size

;;

\*)

echo "Error"

;;

esac

84、查看 KVM 虚拟机中的网卡信息(不需要进入启动或进入虚拟机)

#!/bin/bash

# 查看 KVM 虚拟机中的网卡信息(不需要进入启动或进入虚拟机)

# 该脚本使用 guestmount 工具,可以将虚拟机的磁盘系统挂载到真实机文件系统中

# Centos7.2 中安装 libguestfs‐tools‐c 可以获得 guestmount 工具

# 虚拟机可以启动或者不启动都不影响该脚本的使用

# 将虚拟机磁盘文件挂载到文件系统后,就可以直接读取磁盘文件中的网卡配置文件中的数据

clear

mountpoint="/media/virtimage"

[ ! -d $mountpoint ] && mkdir $mountpoint

read -p "输入虚拟机名称:" name

echo "请稍后..."

# 如果有设备挂载到该挂载点,则先 umount 卸载

if mount | grep -q "$mountpoint" ;then

umount $mountpoint

fi

# 只读的方式,将虚拟机的磁盘文件挂载到特定的目录下,这里是/media/virtimage 目录

guestmount -r -d $name -i $mountpoint

echo

echo "‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐"

echo -e "\033[32m$name 虚拟机中网卡列表如下:\033[0m"

dev=$(ls /media/virtimage/etc/sysconfig/network‐scripts/ifcfg-\* |awk -F"[/‐]" '{print $9}')

echo $dev

echo "‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐"

echo

echo

echo "+++++++++++++++++++++++++++++++++++++++++++"

echo -e "\033[32m 网卡 IP 地址信息如下:\033[0m"

for i in $dev

do

echo -n "$i:"

grep -q "IPADDR" /media/virtimage/etc/sysconfig/network‐scripts/ifcfg-$i || echo "未配置 IP地址"

awk -F= '/IPADDR/{print $2}' /media/virtimage/etc/sysconfig/network-scripts/ifcfg-$i

done

echo "+++++++++++++++++++++++++++++++++++++++++++"

### 85、不登陆虚拟机,修改虚拟机网卡 IP 地址

#!/bin/bash

# 不登陆虚拟机,修改虚拟机网卡 IP 地址

# 该脚本使用 guestmount 工具,Centos7.2 中安装 libguestfs‐tools‐c 可以获得 guestmount 工具

# 脚本在不登陆虚拟机的情况下,修改虚拟机的 IP 地址信息

# 在某些环境下,虚拟机没有 IP 或 IP 地址与真实主机不在一个网段

# 真实主机在没有 virt‐manger 图形的情况下,远程连接虚拟机很麻烦

# 该脚本可以解决类似的问题

read -p "请输入虚拟机名称:" name

if virsh domstate $name | grep -q running ;then

echo "修改虚拟机网卡数据,需要关闭虚拟机"

virsh destroy $name

fi

mountpoint="/media/virtimage"

[ ! -d $mountpoint ] && mkdir $mountpoint

echo "请稍后..."

if mount | grep -q "$mountpoint" ;then

umount $mountpoint

fi

guestmount -d $name -i $mountpoint

read -p "请输入需要修改的网卡名称:" dev

read -p "请输入 IP 地址:" addr

# 判断原本网卡配置文件中是否有 IP 地址,有就修改该 IP,没有就添加一个新的 IP 地址

if grep -q "IPADDR" $mountpoint/etc/sysconfig/network‐scripts/ifcfg‐$dev ;then

sed -i "/IPADDR/s/=.\*/=$addr/" $mountpoint/etc/sysconfig/network‐scripts/ifcfg‐$dev

else

echo "IPADDR=$addr" >> $mountpoint/etc/sysconfig/network‐scripts/ifcfg‐$dev

fi

# 如果网卡配置文件中有客户配置的 IP 地址,则脚本提示修改 IP 完成

awk -F= -v x=$addr '$2==x{print "完成..."}' $mountpoint/etc/sysconfig/network‐scripts/ifcfg-$dev

### 86、破解虚拟机密码,无密码登陆虚拟机系统

#!/bin/bash

# 破解虚拟机密码,无密码登陆虚拟机系统

# 该脚本使用 guestmount 工具,Centos7.2 中安装 libguestfs‐tools‐c 可以获得 guestmount 工具

read -p "请输入虚拟机名称:" name

if virsh domstate $name | grep -q running ;then

echo "破解,需要关闭虚拟机"

virsh destroy $name

fi

mountpoint="/media/virtimage"

[ ! -d $mountpoint ] && mkdir $mountpoint

echo "请稍后..."

if mount | grep -q "$mountpoint" ;then

umount $mountpoint

fi

guestmount -d $name -i $mountpoint

# 将 passwd 中密码占位符号 x 删除,该账户即可实现无密码登陆系统

sed -i "/^root/s/x//" $mountpoint/etc/passwd

### 87、Shell 脚本对信号的处理,执行脚本后,按键盘 Ctrl+C 无法终止的脚本

#!/bin/bash

# Shell 脚本对信号的处理,执行脚本后,按键盘 Ctrl+C 无法终止的脚本

# 使用 trap 命令可以拦截用户通过键盘或 kill 命令发送过来的信号

# 使用 kill ‐l 可以查看 Linux 系统中所有的信号列表,其中 2 代表 Ctrl+C

# trap 当发现有用户 ctrl+C 希望终端脚本时,就执行 echo "暂停 10s";sleep 10 这两条命令

# 另外用户使用命令:[ kill ‐2 脚本的 PID ] 也可以中断脚本和 Ctrl+C 一样的效果,都会被 trap 拦截

trap 'echo "暂停 10s";sleep 10' 2

while :

do

echo "go go go"

done

### 88、一键部署 memcached

#!/bin/bash

# 一键部署 memcached

# 脚本用源码来安装 memcached 服务器

# 注意:如果软件的下载链接过期了,请更新 memcached 的下载链接

wget http://www.memcached.org/files/memcached-1.5.1.tar.gz

yum -y install gcc

tar -xf memcached‐1.5.1.tar.gz

cd memcached‐1.5.1

./configure

make

make install

### 89、一键配置 VNC 远程桌面服务器(无密码版本)

#!/bin/bash

# 一键配置 VNC 远程桌面服务器(无密码版本)

# 脚本配置的 VNC 服务器,客户端无需密码即可连接

# 客户端仅有查看远程桌面的权限,没有鼠标和键盘的操作权限

rpm --quiet -q tigervnc‐server

if [ $? -ne 0 ];then

yum -y tigervnc‐server

fi

x0vncserver AcceptKeyEvents=0 AlwaysShared=1 \

AcceptPointerEvents=0 SecurityTypes=None rfbport=5908

### 90、关闭 SELinux

#!/bin/bash

# 关闭 SELinux

sed -i '/^SELINUX/s/=.\*/=disabled/' /etc/selinux/config

setenforce 0

### 91、查看所有虚拟机磁盘使用量以及CPU使用量信息

#!/bin/bash

# 查看所有虚拟机磁盘使用量以及CPU使用量信息

virt‐df

read -n1 "按任意键继续" key

virt‐top

### 92、使用 shell 脚本打印图形

#!/bin/bash

# 使用 shell 脚本打印如下图形:

# 打印第一组图片

# for(())为类 C 语言的语法格式,也可以使用 for i in;do ;done 的格式替换

# for((i=1;i<=9;i++))循环会执行 9 次,i 从 1 开始到 9,每循环一次 i 自加 1

clear

for (( i=1; i<=9; i++ ))

do

for (( j=1; j<=i; j++ ))

do

echo -n "$i"

done

echo ""

done

read -n1 "按任意键继续" key

#打印第二组图片

clear

for (( i=1; i<=5; i++ ))

do

for (( j=1; j<=i; j++ ))

do

echo -n " |"

done

echo "\_ "

done

read -n1 "按任意键继续" key

#打印第三组图片

clear

for (( i=1; i<=5; i++ ))

do

for (( j=1; j<=i; j++ ))

do

echo -n " \*"

done

echo ""

done

for (( i=5; i>=1; i‐‐ ))

do

for (( j=1; j<=i; j++ ))

do

echo -n " \*"

done

echo ""

done

### 93、根据计算机当前时间,返回问候语,可以将该脚本设置为开机启动

#!/bin/bash

# 根据计算机当前时间,返回问候语,可以将该脚本设置为开机启动

# 00‐12 点为早晨,12‐18 点为下午,18‐24 点为晚上

# 使用 date 命令获取时间后,if 判断时间的区间,确定问候语内容

tm=$(date +%H)

if [ $tm -le 12 ];then

msg="Good Morning $USER"

elif [ $tm -gt 12 -a $tm -le 18 ];then

msg="Good Afternoon $USER"

else

msg="Good Night $USER"

fi

echo "当前时间是:$(date +"%Y‐%m‐%d %H:%M:%S")"

echo -e "\033[34m$msg\033[0m"

### 94、读取用户输入的账户名称,将账户名写入到数组保存

#!/bin/bash

# 读取用户输入的账户名称,将账户名写入到数组保存

# 定义数组名称为 name,数组的下标为 i,小标从 0 开始,每输入一个账户名,下标加 1,继续存下一个账户

# 最后,输入 over,脚本输出总结性信息后脚本退出

i=0

while :

do

read -p "请输入账户名,输入 over 结束:" key

if [ $key == "over" ];then

break

else

name[$i]=$key

let i++

fi

done

echo "总账户名数量:${#name[\*]}"

echo "${name[@]}"

### 95、判断文件或目录是否存在

#!/bin/bash

# 判断文件或目录是否存在

if [ $# -eq 0 ] ;then

echo "未输入任何参数,请输入参数"

echo "用法:$0 [文件名|目录名]"

fi

if [ -f $1 ];then

echo "该文件,存在"

ls -l $1

else

echo "没有该文件"

fi

if [ -d $1 ];then

echo "该目录,存在"

ls -ld $2

else

echo "没有该目录"

fi

### 96、打印各种格式的时间

#!/bin/bash

# 打印各种时间格式

echo "显示星期简称(如:Sun)"

date +%a

echo "显示星期全称(如:Sunday)"

date +%A

echo "显示月份简称(如:Jan)"

date +%b

echo "显示月份全称(如:January)"

date +%B

echo "显示数字月份(如:12)"

date +%m

echo "显示数字日期(如:01 号)"

date +%d

echo "显示数字年(如:01 号)"

date +%Y echo "显示年‐月‐日"

date +%F

echo "显示小时(24 小时制)"

date +%H

echo "显示分钟(00..59)"

date +%M

echo "显示秒"

date +%S

echo "显示纳秒"

date +%N

echo "组合显示"

date +"%Y%m%d %H:%M:%S"

### 97、使用 egrep 过滤 MAC 地址

#!/bin/bash

# 使用 egrep 过滤 MAC 地址

# MAC 地址由 16 进制组成,如 AA:BB:CC:DD:EE:FF

# [0‐9a‐fA‐F]{2}表示一段十六进制数值,{5}表示连续出现5组前置:的十六进制

egrep "[0‐9a‐fA‐F]{2}(:[0‐9a‐fA‐F]{2}){5}" $1

### 98、统计双色球各个数字的中奖概率

#!/bin/bash

# 统计双色球各个数字的中奖概率

# 往期双色球中奖号码如下:

# 01 04 11 28 31 32 16

# 04 07 08 18 23 24 02

# 02 05 06 16 28 29 04

# 04 19 22 27 30 33 01

# 05 10 18 19 30 31 03

# 02 06 11 12 19 29 06

# 统计篮球和红球数据出现的概率次数(篮球不分顺序,统计所有篮球混合在一起的概率)

awk '{print $1"\n"$2"\n"$3"\n"$4"\n"$5"\n"$6}' 1.txt | sort | uniq -c | sort

awk '{print $7}' 1.txt | sort | uniq -c | sort

### 99、生成签名私钥和证

#!/bin/bash

# 生成签名私钥和证书

read -p "请输入存放证书的目录:" dir

if [ ! -d $dir ];then

echo "该目录不存在"

exit

fi

read -p "请输入密钥名称:" name

# 使用 openssl 生成私钥

openssl genrsa -out ${dir}/${name}.key

# 使用 openssl 生成证书 #subj 选项可以在生成证书时,非交互自动填写 Common Name 信息

openssl req -new -x509 -key ${dir}/${name}.key -subj "/CN=common" -out ${dir}/${name}.crt

### 100、使用awk编写的wc程序

#!/bin/bash

# 使用awk编写的wc程序

# 自定义变量 chars 变量存储字符个数,自定义变量 words 变量存储单词个数

# awk 内置变量 NR 存储行数

# length()为 awk 内置函数,用来统计每行的字符数量,因为每行都会有一个隐藏的$,所以每次统计后都+1

# wc 程序会把文件结尾符$也统计在内,可以使用 cat ‐A 文件名,查看该隐藏字符

awk '{chars+=length($0)+1;words+=NF} END{print NR,words,chars}' $1

## 二十三、工作总结

**grep -rn "CDNCS1" \* (包含CDNCS1当前目录所有文件)**

\* : 表示当前目录所有文件，也可以是某个文件名

-r 是递归查找

-n 是显示行号

-R 查找所有文件包含子目录

-i 忽略大小写

**find / -name 'nohup.out'**

**hostname修改文件：**

sysconfig/network:2:HOSTNAME=CDNCS1

sysconfig/networking/profiles/default/network:1:HOSTNAME=CDNCS1

/usr/local/mysql-tomcat-cms-8081/logs/debug

**tail -f -n 2000 LOG.LOG**

netstat -ano

netstat -aon|findstr "8080"

tasklist|findstr "1220"

taskkill /IM java.exe

TASKKILL /PID 1230

TASKKILL /F(强制结束) /IM（镜像名称） java.exe /T(子进程)

netstat -anp|grep 8081

ps -ef|grep 12006

kill -9 8978

ps -ef|grep iscg

kill -9 8978

ps -ef |grep 5601 |awk '{print $2}'|xargs kill -9

击杀动态进程

fuser -n tcp 5601

kill -9 XXXX

./bin/kibana

vi rrr.txt

esc insert

:wq

:q!

查看各个进程内存使用情况

ps aux --sort -rss

**@@LINUX basic**

@IP地址: 给每个连接在Internet上的主机分配的一个32bit地址。按照TCP/IP协议规定，IP地址用二进制来表示，每个IP地址长32bit，比特换算成字节，就是4个字节。例如一个采用二进制形式的IP地址是“00001010000000000000000000000001”，这么长的地址，人们处理起来也太费劲了。为了方便人们的使用，IP地址经常被写成十进制的形式，中间使用符号“.”分开不同的字节。于是，上面的IP地址可以表示为“10.0.0.1”。IP地址的这种表示法叫做“点分十进制表示法”，这显然比1和0容易记忆得多。

**@网关:**

连接两个不同的网络的设备都可以叫网关设备；网关的作用就是实现两个网络之间进行通讯与控制。

网关设备可以是 交互机（三层及以上才能跨网络）、路由器、启用了路由协议的服务器、代理服务器、防火墙等

网关地址就是网关设备的IP地址。

网络A的IP地址范围为“192.168.1.1~192.168.1.254”，子网掩码为255.255.255.0

网络B的IP地址范围为“192.168.2.1~192.168.2.254”，子网掩码为255.255.255.0

要实现这两个网络之间的通信，则必须通过网关。

如果网络A中的主机发现数据包的目的主机不在本地网络中，就把数据包转发给它自己的网关，再由网关转发给网络B的网关，网络B的网关再转发给网络B的某个主机。

一台主机可以有多个网关。默认网关的意思是一台主机如果找不到可用的网关，就把数据包发给默认指定的网关，由这个网关来处理数据包。现在主机使用的网关，一般指的是默认网关。

**@子网掩码：**

IP地址的网络号和主机号各是多少位呢？如果不指定，就不知道哪些位是网络号、哪些是主机号，这就需要通过子网掩码来实现。什么是子网掩码子网掩码不能单独存在，它必须结合IP地址一起使用。

IP地址 = 网络地址 + 主机地址(又称：主机号和网络号组成)

假设 IP地址为 192.168.1.168，子网掩码为 255.255.255.0， 则网络地址换算步骤如下:

**1)将IP地址和子网掩码分别换算成二进制**

　　　　192.168.1.168 换算成二进制为 11000000.10101000.00000001.10101000

　　　　255.255.255.0 换算成二进制为 11111111.11111111.11111111.00000000

**2)将二者进行与运算**

　　　　　　　　　11000000.10101000.00000001.10101000

　　　　与运算　　11111111·11111111·11111111·00000000

　　　　　　　　----------------------------------------------------------------------------

　　　　　　　　　结果　　 11000000.10101000.00000001.00000000

**3) 将运算结果换算成十进制： 192.168.1.0**

IP地址1 子网掩码1 计算的网络地址1 IP地址2 子网掩码2 计算的网络地址2 是否连通

1 192.168.0.1 255.255.255.0 192.168.0.0 192.168.0.200 255.255.255.0 192.168.0.0 是

2 192.168.0.1 255.255.255.0 192.168.0.0 192.168.1.200 255.255.255.0 192.168.1.0 否

3 192.168.0.1 255.255.255.192 192.168.0.0 192.168.0.200 255.255.255.0 192.168.0.192 否

**@域名与DNS**

我们访问一个网站的时候，往往使用的是域名（相对IP来说更加语义清晰、更加容易记忆，例如 www.baidu.com)。

域名是由一串用点分隔的名字组成的，通常包含组织名，而且始终包括两到三个字母的后缀，以指明组织的类型或该域所在的国家或地区。

然而计算机之间的通信网络通信是通过IP进行的， 因此需要将域名解析为对应的IP，DNS就是进行域名解析的服务器。

DNS 维护着 域名(domain name)和IP地址 (IP address)的对照表表，以解析消息的域名。

1、在浏览器中输入www.qq.com域名，操作系统会先检查自己本地的hosts文件是否有这个网址映射关系，如果有，就先调用这个IP地址映射，完成域名解析。

2、如果hosts里没有这个域名的映射，则查找本地DNS解析器缓存，是否有这个网址映射关系，如果有，直接返回，完成域名解析。

3、如果hosts与本地DNS解析器缓存都没有相应的网址映射关系，首先会找TCP/ip参数中设置的首选DNS服务器，在此我们叫它本地DNS服务器，此服务器收到查询时，如果要查询的域名，包含在本地配置区域资源中，则返回解析结果给客户机，完成域名解析，此解析具有权威性。

4、如果要查询的域名，不由本地DNS服务器区域解析，但该服务器已缓存了此网址映射关系，则调用这个IP地址映射，完成域名解析，此解析不具有权威性。

5、如果本地DNS服务器本地区域文件与缓存解析都失效，则根据本地DNS服务器的设置（是否设置转发器）进行查询，如果未用转发模式，本地DNS就把请求发至13台根DNS，根DNS服务器收到请求后会判断这个域名(.com)是谁来授权管理，并会返回一个负责该顶级域名服务器的一个IP。本地DNS服务器收到IP信息后，将会联系负责.com域的这台服务器。这台负责.com域的服务器收到请求后，如果自己无法解析，它就会找一个管理.com域的下一级DNS服务器地址(qq.com)给本地DNS服务器。当本地DNS服务器收到这个地址后，就会找qq.com域服务器，重复上面的动作，进行查询，直至找到www.qq.com主机

6、如果用的是转发模式，此DNS服务器就会把请求转发至上一级DNS服务器，由上一级服务器进行解析，上一级服务器如果不能解析，或找根DNS或把转请求转至上上级，以此循环。不管是本地DNS服务器用是是转发，还是根提示，最后都是把结果返回给本地DNS服务器，由此DNS服务器再返回给客户机。

[root@localhost tomcat-portalms-8082]#

解析：root：用户名

localhost：主机名称，hostname修改文件：

sysconfig/network:2:HOSTNAME=CDNCS1

sysconfig/networking/profiles/default/network:1:HOSTNAME=CDNCS1

tomcat-portalms-8082：当前目录

#命令开始标志，普通账户是$

[root@localhost ~]#

root@localhost root中第一个root表示当前登陆的为root账户

localhost是主机名称

第二个root表示当前的目录为root目录

#表示当前为root权限

[root@localhost ~]#中的~表示当前在登陆用户的家目录

root用户的家目录为/root,其他用户通常为/home/用户名,但可以改变

家目录主要保存每个用户的登陆信息，个人偏好及一些个人文件等

[zying@localhost zying]$

**@@LINUX安装JDK 1.6**

java -version

echo $JAVA\_HOME

chmod 755 jdk-6u45-linux-x64.bin

./jdk-6u45-linux-x64.bin

#vi /etc/profile

JAVA\_HOME=/root/jdk1.6.0\_45

PATH=$JAVA\_HOME/bin:$PATH

CLASSPATH=.:$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar

export JAVA\_HOME

export PATH

export CLASSPATH

Insert：切换追加和替换

tab：切换编辑和窗口

shift :wq! 保存退出

让/etc/profile文件修改后立即生效 ,可以使用如下命令:

# . /etc/profile

注意: . 和 /etc/profile 有空格.

重启测试

chmod -R 777 tomcat-portalms-8082

参数-R是递归的意思

777表示开放所有权限，当然，题目没有说明是开放什么权限，这里替代一下

**@@LINUX安装tomcat**

1 copy tomcat 到/usr/local/

2 递归授权 chmod -R 777 tomcat-portalms-8082

3 修改3个端口号 vi server.xml 8082

4 打war，扔到/usr/local/tomcat-aaa-8085/webapps下面，上下文pms.war ，http://192.168.11.158:8082/pms

5 看启动日志：

cd tomcat-aaa-8085/logs/

tail -f catalina.out

6 看下端口有没停掉

ps -ef|grep 8085

kill -9 3450

7 到bin下面执行./startup.sh,监控启动日志，访问首页

**@LINUX安装mysql**

1、下载安装包 “mysql-5.6.33-linux-glibc2.5-x86\_64.tar.gz”

2、把下载的安装包移动到/usr/local/下

3、解压 tar zxvf mysql-5.6.33-linux-glibc2.5-x86\_64.tar.gz

复制解压后的mysql目录到系统的本地软件目录

cp mysql-5.6.33-linux-glibc2.5-x86\_64 /usr/local/mysql -r

4、添加系统mysql组和mysql用户

groupadd mysql

useradd -r -g mysql -s /bin/false mysql

5、进入安装mysql软件目录，修改目录拥有者为mysql用户

cd mysql/

chown -R mysql:mysql ./

6、安装数据库，此处可能出现错误。

./scripts/mysql\_install\_db --user=mysql

FATAL ERROR: please install the following Perl modules before executing scripts/mysql\_install\_db:

Data::Dumper

#解决方法：

yum install -y perl-Data-Dumper

7、修改当前目录拥有者为root用户

chown -R root:root ./

修改当前data目录拥有者为mysql用户

chown -R mysql:mysql data

8 添加mysql服务开机自启动

cp support-files/mysql.server /etc/init.d/mysql

# 赋予可执行权限

chmod +x /etc/init.d/mysql

# 添加服务

chkconfig --add mysql

# 显示服务列表

chkconfig --list

9 忽略大小写

**@@linux下安装nginx**

1：从http://nginx.org/download/上下载相应的版本(或者wget http://nginx.org/download/nginx-1.5.9.tar.gz直接在Linux上用命令下载)

2：解压 tar -zxvf nginx-1.5.9.tar.gz

3:启动操作

如果更改了配置就要重启Nginx，要先关闭Nginx再打开？不是的，可以向Nginx 发送信号，平滑重启。

./sbin/nginx -t 查看配置信息是否正确

./sbin/nginx 启动操作

./sbin/nginx -s reload 平滑重启命令

4 停止操作

ps -ef | grep nginx

pkill -9 nginx

5 nginx.conf的配置

6 在工作过程中，部署项目安装的nginx执行报错，找不到库../sbin/nginx: error while loading shared libraries: libcrypto.so.6: cannot open shared object file: No such file or directory

由错误提示可以看出缺少libcrypto.so.6

通过命令：ldd $(which /usr/local/nginx/sbin/nginx) （ldd检测某条命令需要哪些共享库文件的支持）

upstream asg\_pro {

ip\_hash;

server 192.168.11.158:8084;

}

server {

listen 9005;

server\_name 192.168.11.158;

location / {

root /resource;

}

location /css/template {

root /home/mmsp/resource;

}

location /js {

root /home/mmsp/resource;

}

location /poster {

root /home/mmsp;

}

#location /ui {

# alias /home/mmsp/resource/ui/template;

#}

location /portal2\_pfs/generateManage/generateTemplate/root/ {

alias /home/mmsp/resource/;

}

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

root html;

}

location ~ \.(jsp|do)|asg|Play|services/\*|PlayVod|ToOrder|OrderProdVOD|stbsimulator/\*|portalFrontProcessor|userMark|OrderPage|ChannelOrder|PlayByTime|Epg|NpvrServlet|PlayOTTByTime|PlayOTT {

proxy\_pass http://asg\_pro;

proxy\_redirect off;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_set\_header Host $http\_host;

proxy\_connect\_timeout 30;

proxy\_send\_timeout 30;

proxy\_read\_timeout 30;

}

}

} #access\_log logs/host.access.log main;

**@@查看各个经常消耗CPU和内存情况**

**ps aux --sort -rss**



**1.CPU占用最多的前10个进程：**

**ps auxw|head -1;ps auxw|sort -rn -k3|head -10**

**2.内存消耗最多的前10个进程**

**ps auxw|head -1;ps auxw|sort -rn -k4|head -10**

**3.虚拟内存使用最多的前10个进程**

**ps auxw|head -1;ps auxw|sort -rn -k5|head -10**