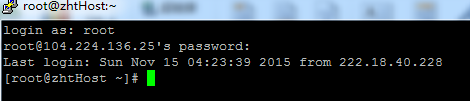
CentOS修改Root密码，先用root登录，输入passwd命令，输入密码，注意此时输入的密码不显示。输入后忽略警告，再次输入密码即可。

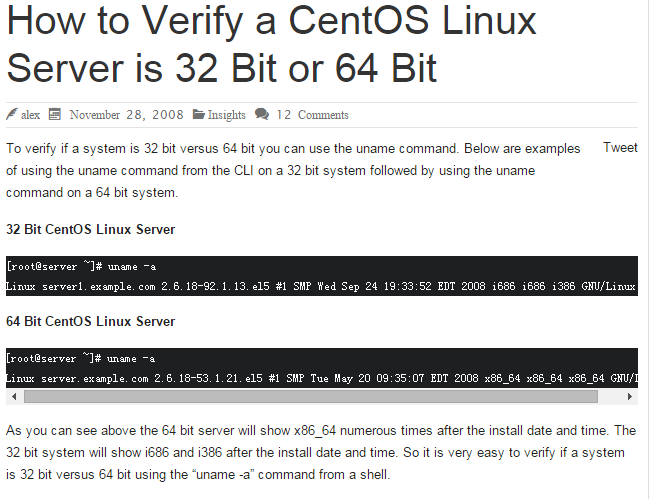
使用puuty远程登录主机后



[root@zhtHost ~]#:root表示当前用户账号，zhtHost表示主机名，~表示当前所在目录，#是提示符。

~代表用户的主文件夹，root用户的主文件夹在/root，使用~表示/root。user用户的的主文件夹在/home/user,所以当使用user账号登录时，~就表示/home/user。

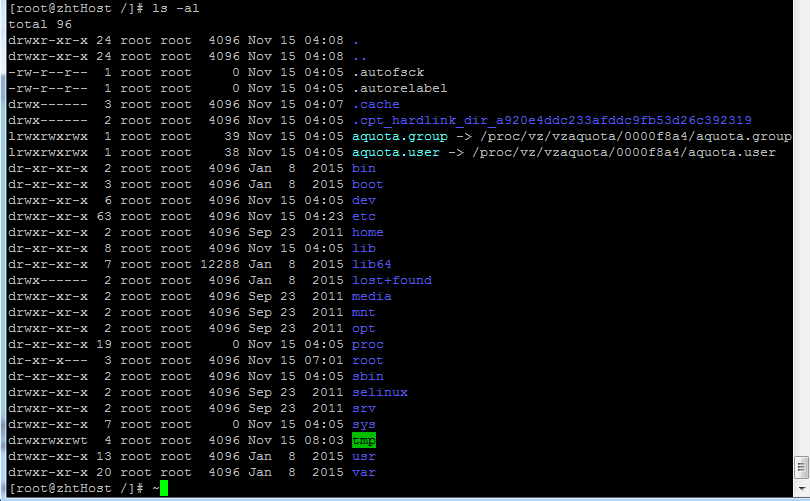
至于提示符，root用户的提示符是#，一般身份用户的提示符是$。exit退出账号。



bash：登录后的命令行操作是在Linux中的bash中进行的，bash是一个命令行终端软件，是Linux下默认的shell。shell的中文是“外壳”，里面是操作系统内核，用户无法直接接触操作系统，必须通过最外层的shell来和系统内核通信。shell的功能是提供给用户一个操作系统的借口，可以调用其他软件，如man，chown，vi,这些独立的应用程序。因此，只要能够操作应用程序的接口都能成为shell。

权限管理：Linux中使用用户组和多用户来进行文件的权限管理。一个系统上可以有多个用户组，一个用户组中可以有多个用户，用户组的作用是在同一用户组的用户们可以共同享有一些文件，其他用户组的成员无法访问。同一用户组的每个用户也可以拥有自己的私有文件，同一组内的其他用户无法访问。任何一个文件都有“user group other”3种身份的权限限制。

文件权限：



如上图所示，每个文件说明从左到右依次是：文件权限——连接——所属用户——所属用户组——文件大小，单位：B——创建或者最后修改日期——文件名（前面带.的表示隐藏文件）。

其中表示文件权限的有10个字母，第一个字母代表文件类型，后面9个分为3组rwx，依次代表user group other的访问权限（所有者、同用户组、非本用户组）。

第一个字母：d：表示文件夹，—：表示文件，l：表示连接文件。b：可供存储接口设备，c：串行接口设备，如键盘、鼠标（一次性读取）。

后面的3组为rwx的组合，r表示可读，read；w可读write，x可以执行。

对文件夹来说，不具备x执行权限就表示无法进入该文件夹。



dev是一个文件夹，属于root用户组中的root用户，root用户对其可读可写可执行，root组的其他用户对其可读可执行，非root组的用户对其可读可执行。

常用命令：

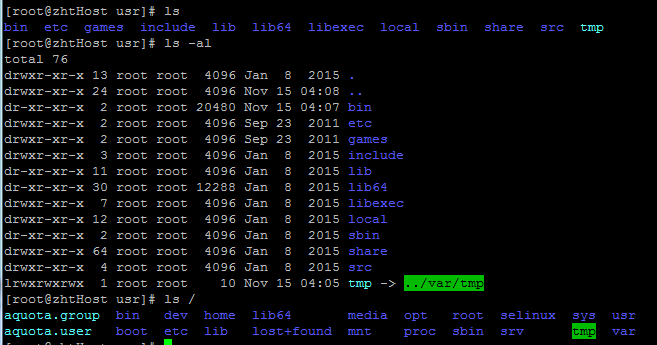
man:帮助命令，如：man ls 输出ls的指令说明，空格翻页，q退出。pgup/pgdn也可以上下翻页。end尾页/home首页。/string（或者?string） 匹配查找string字符，n继续向下查找（next），N继续向上查找。

pwd：查看当前路径



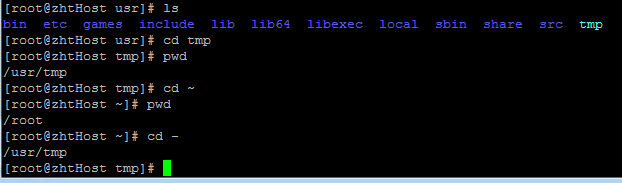
ls:list 列出目录清单，ls 参数 目录，按照相应的设置参数列出文件目录。如：

ls –al / 列出根目录下的所有文件，包括隐藏的文件，其中 –al 可以省略，没有/ 则表示列出当前路径下的文件目录。



ll:ls -l的缩写

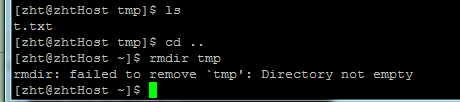
cd:切换当前路径，cd .. 回到上一级目录，cd / 回到系统根目录，cd path 切换到path目录，cd ~ 切换到当前用户主目录，cd – 返回到此目录之前所在目录。



mkdir:新建一个目录

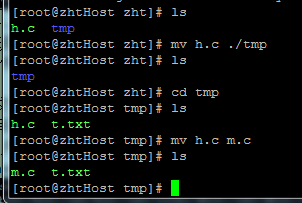


rmdir:删除一个空目录，目录下有文件不能删除。

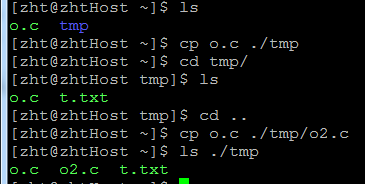


rm:删除文件，如果删除带有文件的文件夹，需要-r 参数，递归删除。-f 强制删除，不会出现警告，-i 删除前询问是否要删除。

mv：移动文件 mv 源文件 目标文件，也可以更改文件名,mv dir1 dir2修改文件夹名。

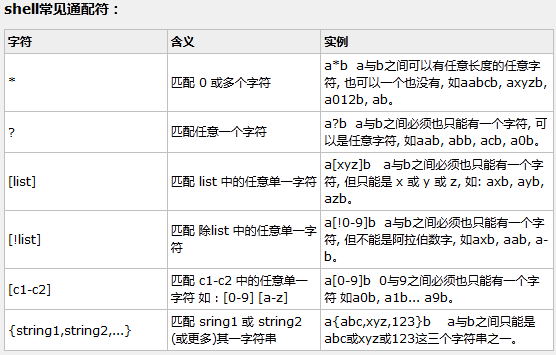


cp:文件复制，cp 源文件 目标文件



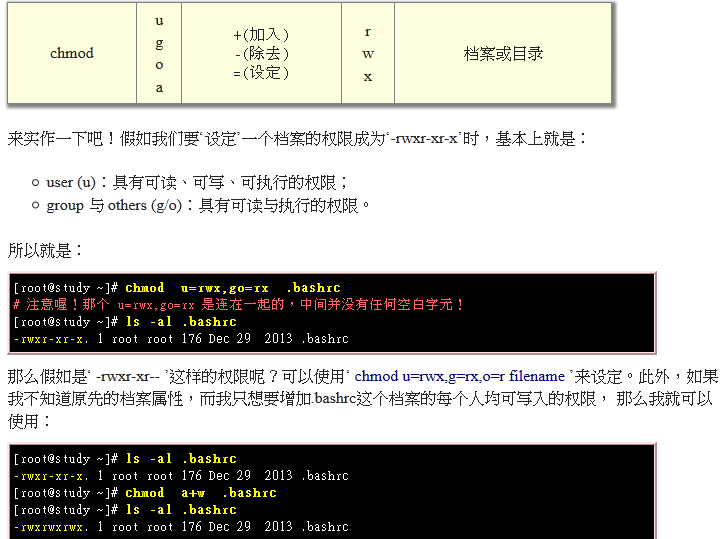
cat:查看文档命令 。

less：查看文档命令，可以上下翻页。



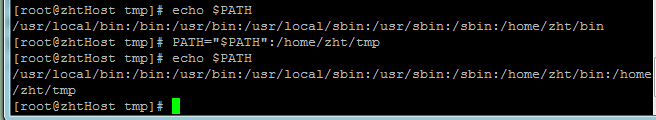
chmod:改变文件权限，如chmod 775 t.txt 将t.txt文件权限修改为所有者同一用户组可读可写可执行，其他用户组可读可执行。快捷的权限修改：

chmod +x t.txt,为文件添加执行权限。 -x ，去除执行权限。

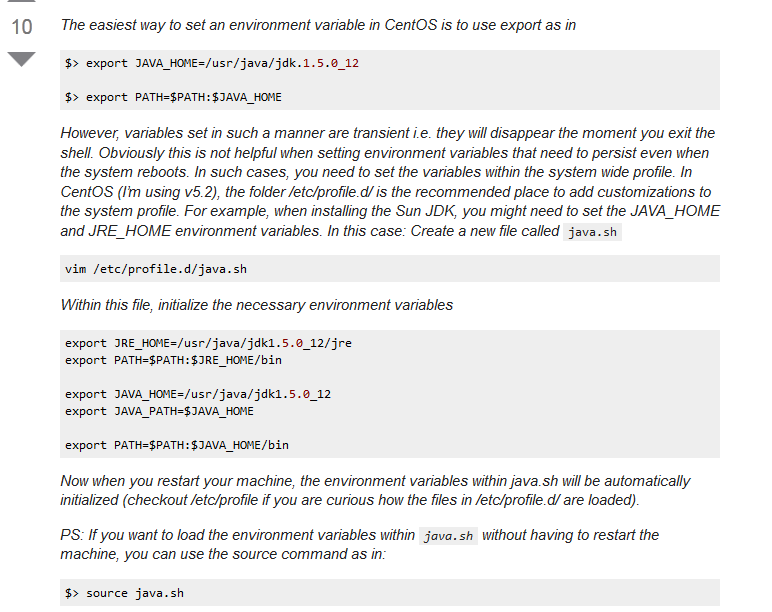


wget:下载文件命令，后接http地址可以下载文件到当前目录下。

PATH:当前用户路径临时设定可以使用echo $PATH 来查看，添加PATH路径PATH=” $PATH”:/home/zht/temp 或者 export PATH=$PATH:/home/zht/tmp。



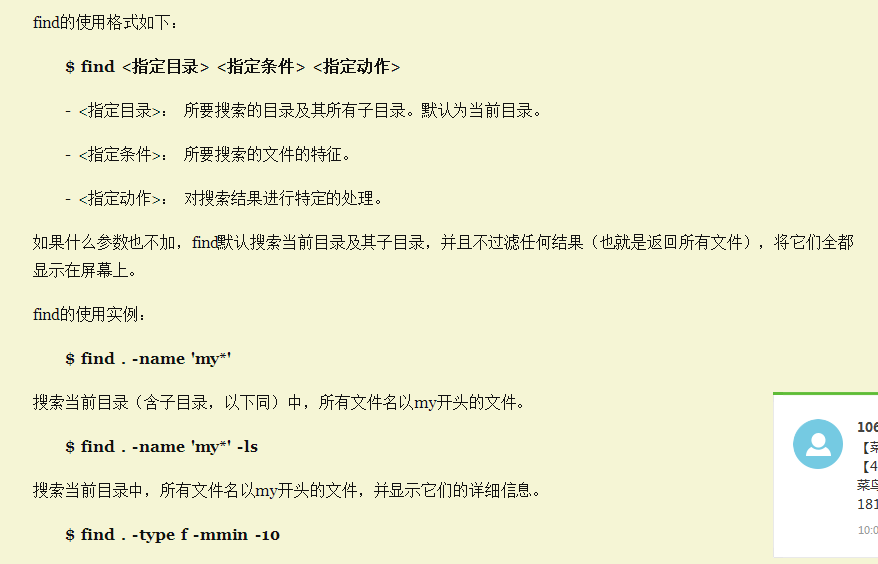
上面的方法只能临时更改path，如果想全局长久更改path，

 查看设定的JAE\_HOME 使用指令：echo $JRE\_HOME

which:查看命令所在目录 ，或者使用type



应用：#whereis  软件名   -->查看软件安装路径  
#which  软件名     -->软件软件的运行路径



find:查找指令。find [指定目录] [指定名字]

如：find / -name “java\*” ：最后一个参数不加“”也可以

查找根目录下所有以java开头的文件，并列出起详细信息。

<http://www.oschina.net/translate/15-practical-linux-find-command-examples>

touch：命令不常用，一般在使用make的时候可能会用到，用来修改文件时间戳，或者新建一个不存在的文件。

实例一：创建不存在的文件命令：touch log2012.log log2013.log

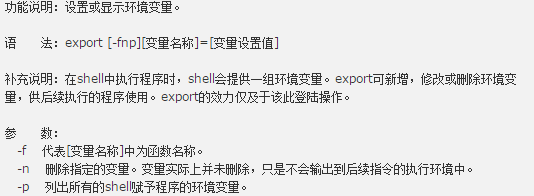
实例二：更新log.log的时间和log2012.log时间戳相同

实例三：设定文件的时间戳命令：touch -t 201211142234.50 log.log

**linux shell中的单引号与双引号的区别**

<http://lspgyy.blog.51cto.com/5264172/1282107>

export



grep是Linux命令行下常用于查找过滤文本文件内容的命令。最简单的用法是：

grep apple fruitlist.txt

grep会返回“fruitlist.txt”中所有包含“apple”的文本行。要注意的是，grep不会返回匹配“Apple”（A字母大写）的文本行，因为grep默认情况下是大小写敏感的。

如果想忽略大小写，可以用-i参数：

grep -i apple fruitlist.txt

如果想搜索目录里所有文件，包括子目录的话，并且在结果中显示行号，可以用以下命令：

grep -nr apple \*

grep的语法支持正则表达式，正则表达式有些复杂，以后再讲解。下面是一些有用的参数：

* -A num, --after-context=num: 在结果中同时输出匹配行之后的num行
* -B num, --before-context=num: 在结果中同时输出匹配行之前的num行，有时候我们需要显示几行上下文。
* -i, --ignore-case: 忽略大小写
* -n, --line-number: 显示行号
* -R, -r, --recursive: 递归搜索子目录
* -v, --invert-match: 输出没有匹配的行

我们可以通过管道操作来让grep变得更强大，管道操作就是把前面一条命令的输出作为后面一条命令的输入，从而把很多简单的命令组合起来完成复杂的功能。例如，如果我们想查找包含apple的行，但又想过滤掉pineapple，可以用下面的命令：

grep apple fruitlist.txt | grep -v pineapple

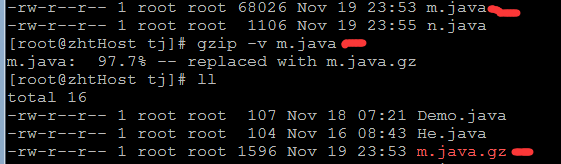
如果我们想把搜索结果保存起来，那么可以把命令的标准输出重定向到文件：

grep apple fruitlist.txt | grep -v pineapple > apples.txt

重定向符号>和管道操作符号|的区别是，重定向后面接的是一个文件，它后面不能再接任何文件或命令了；而管道操作后面接的是命令，可以无限地接下 去。如果想以追加方式写到文件，可以用>>。管道操作是Linux命令行的一种哲学，它是计算机技术中少有的能沿用几十年的技术之一。通过管 道操作，一行命令可以完成Windows下上千行程序也不能完成的文本处理功能。

关于Linux的压缩相关命令：

在Linux中采用gzip和bzig2作为压缩命令，其中bzip2使用的压缩算法相比gzip更加先进，采用gzip命令压缩的文件以.gz为后缀名，采用bzip2压缩的文件以.bz2为文件后缀名。其使用方法是：gzip 参数 待压缩文件名 -v参数常用来显示压缩工程中的压缩比。

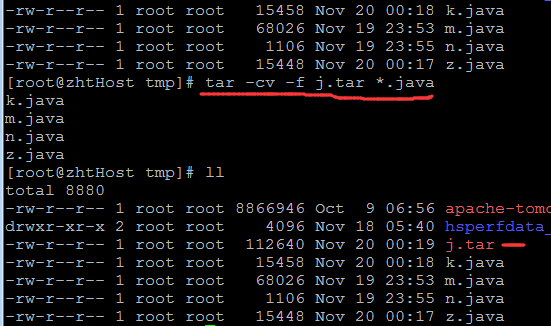
解压的命令：



bzip2的使用方式和gzip是一样的。

由于gzip和bzip2命令只能针对单一文件进行压缩或解压，所以不能同时对多个文件或文件夹进行压缩，这时就需要使用打包命令tar来进行辅助，tar是将多个文件打包成一个文件的命令，同时在打包的过程中也可以使用压缩算法。如果只是打包不 压缩，则文件后缀名为.tar，如果打包后采用gzip命令压缩，则后缀名为.tar.gz，如果采用bzip2命令压缩，则后缀名为.tar.bz2。

1. 只打包：



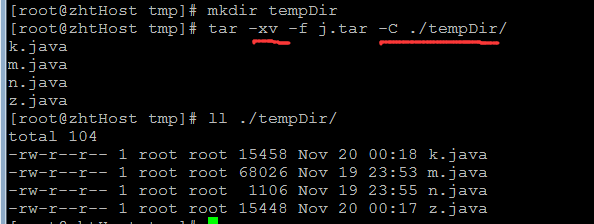
-c:新建打包文件

-v:将正在处理的文件名显示出来。

-f:后面接待处理的文件名，如是打包操作就是打包后生产的新文件名，如果是压缩操作 -f后接的就是压缩后生产的新文件名，如果是解压操作，后面接的是待解压的文件名。很多地方喜欢将-f和前面的参数写在一起，不过还是建议单独写出来。

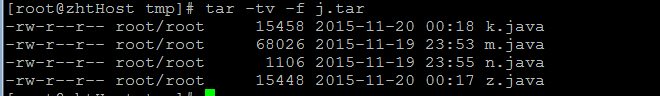
最后 \*.java表示将所有以.java结尾的文件打包成一个j.tar文件。

1. 解打包：

-x：解打包或者解压缩功能

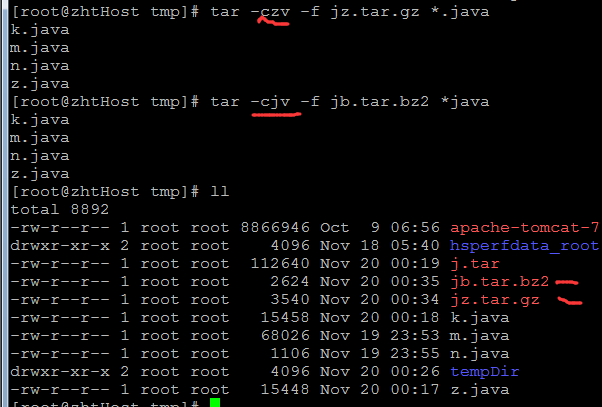
-C：待解打包的目录

1. 查看打包文件名：

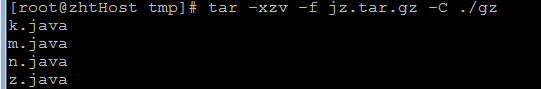
-t:查看打包的文件内容，主要用来查看文件名。

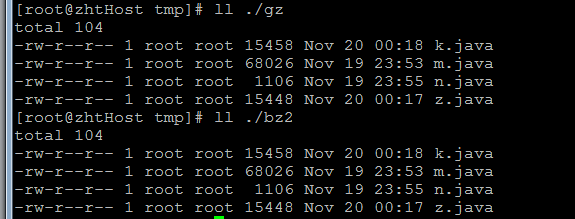
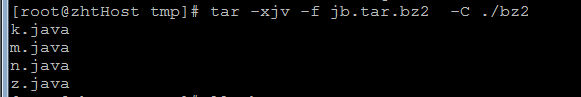
注意，在tar命令，-c -x -t每次只能出现其中的一个。

1. 压缩多个文件：  
   -j：表示采用bzip2命令对打包的文件进行压缩。  
   -z：表示采用gzip命令对打包的文件进行压缩。



1. 解压缩：

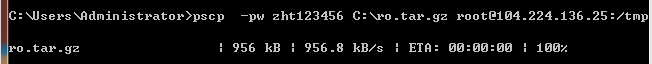




1. 使用-xzv -f参数后为待解压缩文件名。最后面-C接解压后的文件存放目录，如果没有-C参数，则默认解压到当前文件夹。  
   另外同-x -v -t同级别的参数还有-r：向压缩文件中追加文件， -u更新压缩包中的指定文件：

且这5个参数每次只能出现其中一个。

使用pscp 远程给主机传输文件：

相应的从Linux服务器上下载文件只需要将目标和源反过来即可。

 pscp [options] source [source...] [user@]host:target

-p 拷贝文件的时候保留源文件建立的时间。  
-q 执行文件拷贝时，不显示任何提示消息。  
-r 拷贝整个目录  
-v 拷贝文件时，显示提示信息。

-pw passwd 指定登录时所用的口令为 passwd

查看进程命令：

ps命令就是最根本相应情况下也是相当强大地进程查看命令.运用该命令可以确定有哪些进程正在运行和运行地状态、进程是否结束、进程有没有僵死、哪些进程占用了过多地资源等等.总之大部分信息均为可以通过执行该命令得到地.

a：与终端相关的进程；

x：与终端无关的进程；

u：显示用户

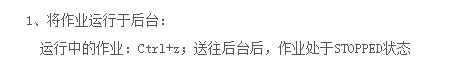
aux：常用组合选项；



-e 显示所有进程,环境变量  
-f 全格式

a   显示终端上地所有进程,包括其他用户地进程  
r   只显示正在运行地进程配色

kill -9 PID 杀死该PID进程。

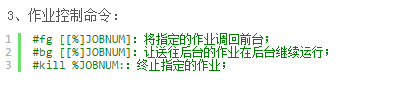


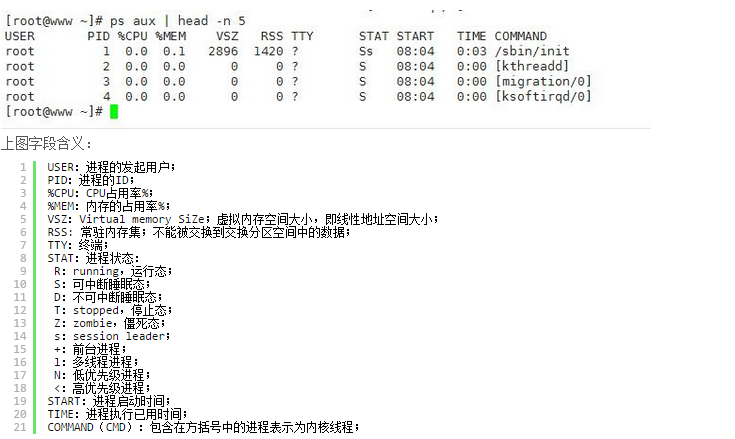
jobs查看当前作业：



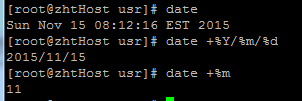
bg 1 或者bg %1均可以将Java

进程调回前台。

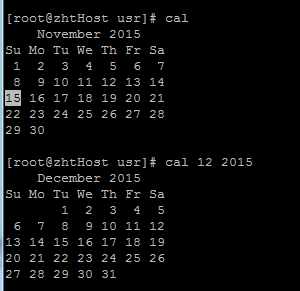




date:查看当前日期，还可以设定输出格式，如：date +%Y/%m/%d，输出：2015/11/15。



cal:输出当前月份的日历，另外cal 2015,输出2015年的日历，cal 11 2015,输出2015年11月份的日历。



tab：按一下，在命令行第一个命令后表示补全命令，第二个命令后，补全文件。在第一个命令未完成时，按2下，则会显示所有已当前输出内容为开头的命令，即提示所有命令，如下：

ctrl+c:停止执行当前命令。即终止命令的执行。强制终止当前进程

ctrl+ｚ:强制当前进程转换为后台，并在后台挂起（暂停）。

ctrl+d:代表EOF（end of file），等同exit。退出

clear:清屏

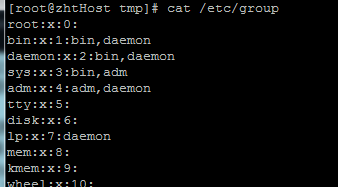
* 关机命令：
  + shutdown -h now（立刻进行关机）
  + shutdown -h 10 10分钟后自动关机
  + halt（立刻进行关机）
  + poweroff（立刻进行关机）
* 重启命令：
  + shutdown -r now（现在重新启动计算机）
  + reboot（现在重新启动计算机）
  + shutdown -r 10 过10分钟自动重启(root用户使用)   
    shutdown -r 20:35 在时间为20:35时候重启(root用户使用)

区别：shutdown -h now和shutdown -r now必须是root用户或者具有root权限的用户才能使用，而halt和reboot是Linux系统中的任何用户都可使用，不需要root权限。

Linux下用户组合用户的管理：

groups:查看当前用户所在的用户组。

使用cat /etc/group 可以查看本系统上所有的用户组合用户信息。



上面每一行的信息表示的分别是：用户组——用户组口令（密码）——GID——用户组用户列表。

groupadd zhts:添加zhts用户组

groupdel zhts:删除zhts用户组

adduser zht:添加zht用户，会自动生成一个zht用户组，且zht的用户主目录在/home/zht 下。如果之间加 –G groupname 就表示将该用户添加到groupname 用户组去。

passwd zht:给zht用户添加密码，这个一定要添加。

su zht：切换到zht用户。

sudo:以root身份执行命令，后面需要输入密码

su为switch user，即切换用户的简写

su - USERNAME切换用户后，同时切换到新用户的工作环境中

su USERNAME切换用户后，不改变原用户的工作目录，及其他环境变量目录

sudo COMMAND 即意为sudo以root用户执行

默认情况下，系统只有root用户可以执行sudo命令。需要root用户通过使用visudo命令编辑sudo的配置文件/etc/sudoers，才可以授权其他普通用户执行sudo命令。

sudo的配置文件为：/etc/sudoers。

sudoers文件中允许指定用户在不需要知道root用户的登陆密码的情况下，可以以root用户身份运行各种命令。此文件必须使用visudo命令编辑配置。

<http://zebralinux.blog.51cto.com/8627088/1369301>

<http://desert3.iteye.com/blog/1663995>

关于Linux下的字符编码格式设定：

使用locale 或者 echo $LANG查看当前系统的编码格式，centos默认是C，C即"POSIX"的别名。所以当我们新安装完一个系统时，默认的locale就是C或POSIX。

此时，系统是无法识别中文的，需要如下设置：

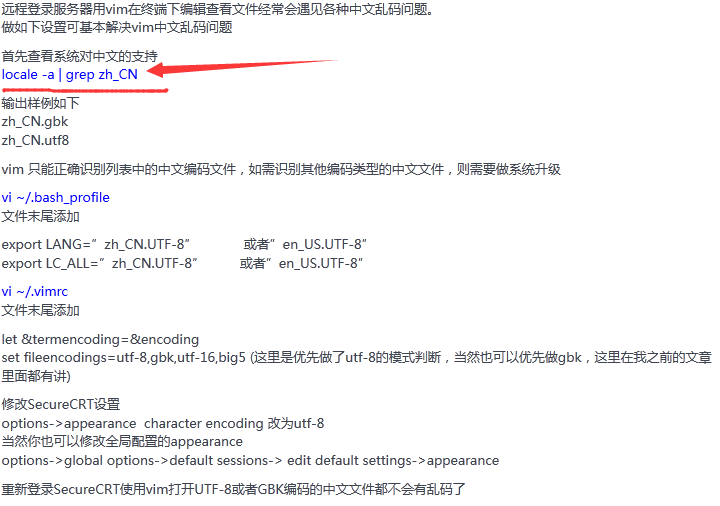




既可以显示以utf-8编码的文件中的中文字符，vim也需要做相应的配置，如下。

let &termencoding=&encoding   
set fileencodings=utf-8,gbk

注意如果将zh\_CN.utf8改为utf8后，系统使用cat命令可以正确显示中文，vim中却不可以。



如果你的LANG环境变量是en\_US.UTF-8，那么系统的菜单、程序的工具栏语言、输入法默认语言就都是英文的。   
如果你的LANG环境变量是zh\_CN.UTF-8，那么系统的菜单、程序的工具栏语言、输入法默认语言就都是中文的。

在Linux中通过locale来设置程序运行的不同语言环境，locale由ANSI C提供支持。locale的命名规则为<语言>\_<地区>.<字符集编码>，如zh\_CN.UTF-8，zh代表中 文，CN代表大陆地区，UTF-8表示字符集。在locale环境中，有一组变量，代表国际化环境中的不同设置：   
1.    LC\_COLLATE   
定义该环境的排序和比较规则   
2.    LC\_CTYPE   
用于字符分类和字符串处理，控制所有字符的处理方式，包括字符编码，字符是单字节还是多字节，如何打印等。是最重要的一个环境变量。   
3.    LC\_MONETARY   
货币格式   
4.    LC\_NUMERIC   
非货币的数字显示格式   
5.    LC\_TIME   
时间和日期格式   
6.    LC\_MESSAGES   
提示信息的语言。另外还有一个LANGUAGE参数，它与LC\_MESSAGES相似，但如果该参数一旦设置，则LC\_MESSAGES参数就会失效。 LANGUAGE参数可同时设置多种语言信息，如LANGUANE="zh\_CN.GB18030:zh\_CN.GB2312:zh\_CN"。   
7.    LANG   
LC\_\*的默认值，是最低级别的设置，如果LC\_\*没有设置，则使用该值。类似于 LC\_ALL。   
8.    LC\_ALL   
它是一个宏，如果该值设置了，则该值会覆盖所有LC\_\*的设置值。注意，LANG的值不受该宏影响。

# 目录详解

/      处于linux系统树形结构的最顶端，它是linux文件系统的入口，所有的目录、文件、设备都在/之下。

/bin   bin是Binary的缩写，存放着linux系统命令。

/dev   dev是Device的缩写。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。（注意：设备文件不是驱动程序。过去，在添加新磁盘或设备后，往往需要手动增加设备文件。现在通常我们不需要手动增加设备文件，运行一下service kudzu start ，系统就会自动配置相应的设备。）

/home  用户的主目录。在liunx系统中，每个用户都有一个自己的目录，一般该目录名是以用户的帐号命名的。

/lib   这个目录里存放着系统最基本的动态链接共享库，包含许多被/bin/和/sbin/中的程序使用的库文件，目录/usr/lib/中含有更多用于用户程序的库文件。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。

/media  linux系统自动识别的一些设备，例如U盘、光驱、移动硬盘等，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，可以将光驱挂载到/mnt/上，然后进入该目录就可以查看光驱里的内容。

/opt   主机额外安装软件所摆放的目录。默认是空的。

/root  这个不用介绍了吧，呵呵。超级管理员的用户主目录。

/selinux  这个目录是RedHat/CentOS所特有的目录，Selinux是一个安全机制，这个比较复杂，这个目录就是存放Selinux相关的文件的，一般我们安装操作系统的时候禁止使用它。

/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。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种被创建。

/usr   包括与系统用户直接有关的文件和目录，例如应用程序及支持它们的库文件。类似于windows下的program files目录。

/boot  包括内核和其它系统启动期间使用的文件。是启动linux时使用的核心文件，有连接文件和镜像文件。

/etc   存放系统配置文件和目录，非常重要，经常会用到它，要牢记。

/lost+found  默认为空，被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）用来放置零散文件（没有名称的文件）。当系统非法关机后，这里就会存放一些文件。

/misc   存放杂项文件或目录，即那些用途或含义不明确的文件或目录可以存放在该目录下。

/proc    操作系统运行时，进程（正在运行中的程序）信息及内核信息（比如cpu、硬盘分区、内存信息等）存放在这里。/proc目录是伪装的文件系统proc的挂载目录，proc并不是真正的文件系统。因此，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。也就是说，这个目录的内容不在硬盘上而是在内存里。

/sbin  大多数涉及系统管理的命令都存放在该目录中，它是超级权限用户root的可执行命令存放地，普通用户无权限执行这个目录下的命令，凡是目录sbin中包含的命令都是root权限才能执行的。

/srv    该目录存放一些服务启动之后需要提取的数据。

/tmp    该目录用于存放临时文件，有时用户运行程序的时候，会产生一些临时文件。/tmp就是用来存放临时文件的。/var/tmp目录和该目录的作用是相似的。

/var    该目录的内容是经常变动的，/var下有/var/log目录用来存放系统日志的目录。/var/www目录用来定义Apache服务器站点存放目录。/var/lib用来存放一些库文件。

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

其它重要系统目录

/usr/bin   这个目录是可执行程序的目录，普通用户就有权限执行。当我们从系统自带的软件包安装一个程序时，他的可执行文件大多会放在这个目录。相似的目录是/usr/local/bin目录。有时/usr/bin中的文件是/usr/local/bin的链接文件。

/usr/sbin  这个目录也是可执行程序的目录，但大多存放涉及系统管理的命令。只有root权限才能执行，相似目录是/sbin或/usr/local/sbin或/usr/X11R6/sbin等。

/usr/src   内核源码默认的放置目录

/proc/cpuinfo  关于处理器的信息，如类似、厂家、型号和性能等。比如cat /proc/cpuinfo

/proc/devices  当前运行内核所配置的所有设备清单。

/proc/filesystems   当前运行内核所配置的文件系统。

/proc/dma 当前正在使用的DMA通道。

/proc/interrupts 正在使用的中断和曾经有多少个中断。

/proc/ioports 当前正在使用的I/O端口。

/etc/init.d 这个目录是用来存放系统或服务器以System V模式启动的脚本，这在以System V模式启动或初始化的系统中常见。比如RedHat Fedora。

/etc/xinetd.d 如果服务器是通过xinetd模式运行的，它的脚本要放在这个目录下。有些系统没有这个目录，比如Slackware，有些老的版本也没有。在Redhat Fedora中比较新的版本中存在。

/etc/rc.d 这是Slackware发行版中有的一个目录，是BSD方式启动脚本的存放地，比如定义网卡，服务器开启脚本等。

/etc/X11 是X-Window相关的配置文件存放地。

# 特定主机系统范围内的[配置文件](http://zh.wikipedia.org/w/index.php?title=%E9%85%8D%E7%BD%AE%E6%96%87%E4%BB%B6&action=edit&redlink=1)。

|  |  |
| --- | --- |
| 目录 | 描述 |
| **/etc/rc /etc/rc.d**  **/etc/rc\*.d** | 启动、或改变运行级时运行的scripts或scripts的目录. |
| /etc/hosts | 本地域名解析文件 |
| **/etc/sysconfig/network** | IP、掩码、网关、主机名配置 |
| **/etc/resolv.conf** | DNS服务器配置 |
| **/etc/fstab** | 开机自动挂载系统，所有分区开机都会自动挂载 |
| **/etc/inittab** | 设定系统启动时Init进程将把系统设置成什么样的runlevel及加载相关的启动文件配置 |
| **/etc/exports** | 设置NFS系统用的配置文件路径 |
| **/etc/init.d** | 这个目录来存放系统启动脚本 |
| **/etc/profile,**/etc/csh.login,  /etc/csh.cshrc | **全局系统环境配置变量** |
| **/etc/issue** | 认证前的输出信息，默认输出版本内核信息 |
| /etc/motd | 设置认证后的输出信息， |
| /etc/mtab | 当前安装的文件系统列表.由scripts初始化，并由mount 命令自动更新.需要一个当前安装的文件系统的列表时使用，例如df 命令 |
| **/etc/group** | 类似/etc/passwd ，但说明的不是用户而是组. |
| **/etc/passwd** | 用户数据库，其中的域给出了用户名、真实姓名、家目录、加密的口令和用户的其他信息. |
| **/etc/shadow** | 在安装了影子口令软件的系统上的影子口令文件.影子口令文件将/etc/passwd 文件中的加密口令移动到/etc/shadow 中，而后者只对root可读.这使破译口令更困难. |
| **/etc/sudoers** | 可以sudo命令的配置文件 |
| **/etc/syslog.conf** | 系统日志参数配置 |

/usr/local 这个目录一般是用来存放用户自编译安装软件的存放目录。一般是通过源码包安装的软件，如果没有特别指定安装目录的话，一般是安装在这个目录中。

/usr/lib 该目录和/lib目录相似，是库文件的存储目录。存放一些常用的共享库。

/usr/share 该目录用于存放系统共用的东西，比如/usr/share/fonts是字体目录，是用户都共用的。

/usr/share/doc 该目录是Linux共享文档的存放地。

/usr/share/man 该目录是共享的帮助文件的存放地。

/var/adm 比如软件包安装信息、日志、管理信息等就存放在该目录下，在Slackware操作系统中是有这个目录的。在Fedora中好象没有。

/var/log 该目录用于存放系统日志。

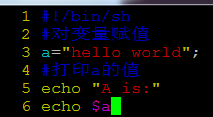
/var/spool 打印机、邮件、代理服务器等假脱机目录存放在该目录下。

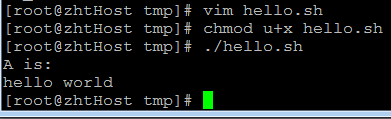
# **Linux 脚本编写**

[鸟哥的私房菜](http://linux.vbird.org/linux_basic/0340bashshell-scripts.php)

第一个例程：







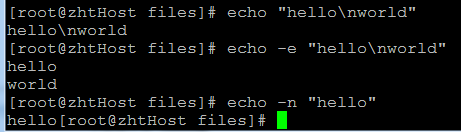
Linux的shell脚本相当于Windows下的.bat批处理文件，并且比批处理更加强大，可以进行变量赋值，流程控制。

第一行：“#!”是一个约定的标记，它告诉系统这个脚本需要什么解释器来执行。/bin/sh默认链接到bin目录下的bash。

#后面接的是注释。

echo命令用来打印信息，是一个最常用的命令。

-n 打印信息之后不换行。

-e 对字符串转义。

扩展名为sh（sh代表shell），扩展名并不影响脚本执行，见名知意就好，如果你用php写shell 脚本，扩展名就用php好了。



这种方式运行的脚本，不需要在第一行指定解释器信息，写了也没用。



有时候变量名很容易与其他文字混淆，比如：

num=2

echo "this is the $numnd"

这并不会打印出"this is the 2nd"，而仅仅打印"this is the "，因为shell会去搜索变量numnd的值，但是这个变量时没有值的。可以使用花括号来告诉shell我们要打印的是num变量：

num=2

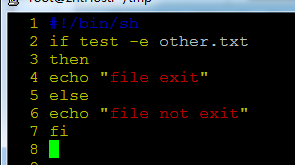
echo "this is the ${num}nd"

这将打印： this is the 2nd

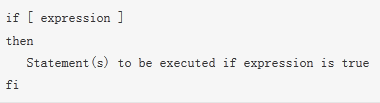
由export关键字处理过的变量叫做环境变量。

<http://www.cnblogs.com/xuejie/archive/2013/01/31/2886552.html>

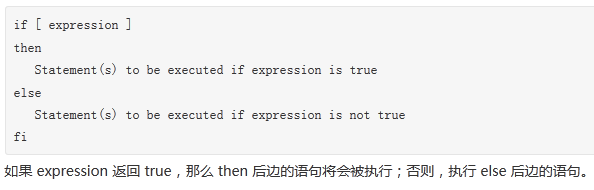
**if条件判断**

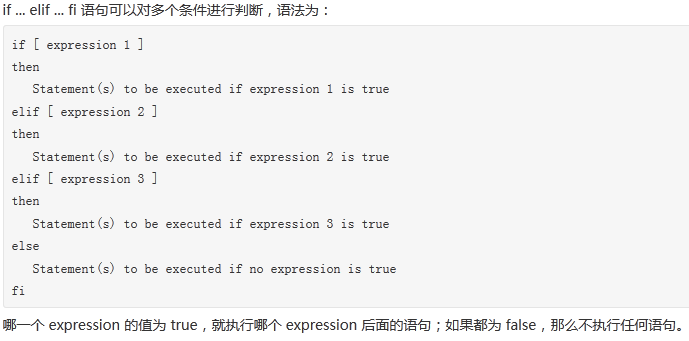


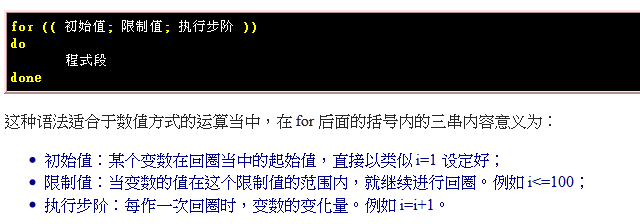
[**test用法：shift+f5返回**](#test用法)

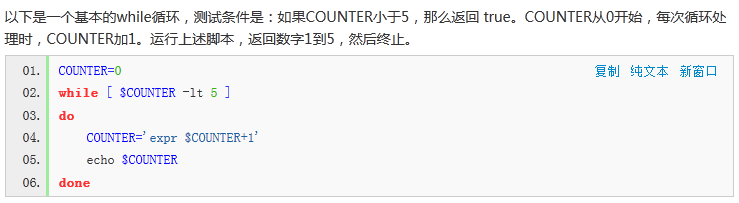


如果 expression 返回 true，then 后边的语句将会被执行；如果返回 false，不会执行任何语句。  
最后必须以 fi 来结尾闭合 if，fi 就是 if 倒过来拼写，后面也会遇见。注意：expression 和方括号([ ])之间必须有空格，否则会有语法错误。









使用脚本切换当前路径：

在linux环境下，常有通过shell脚本进入指定目录的操作，例如有一段脚本chdir.sh：

#!/bin/sh

cd /home/user/Downloads

pwd

在shell环境下通过./chdir.sh执行这段脚本是无法进入Downloads目录的； 这是因为shell在执行脚本时，会创建一个子shell，并在子shell中逐个执行脚本中的指令； 而子shell中从父shell中继承了环境变量，但是执行后不会改变父shell的环境变量；如果想要代码中切换目录的操作生效，只需要通过 source 命令执行即可：

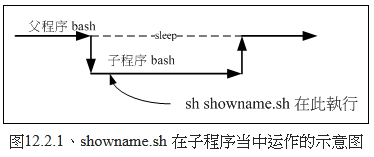
source ./chdir.sh

     source 命令是在当前的shell环境下执行脚本 ，不会创建子shell;

  每次都要输入source是一件烦琐的事情，其实source命令又叫点命令，在需要用到source的情况下，直接换成'.'即可：

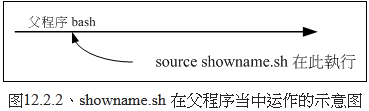
.  ./chdir.sh

注意两点之间的空格。



showname.sh 里面的指令，其实是在下图中的子程序 bash 内执行的。 当 showname.sh 执行完毕后，子程序 bash 内的所有资料便被移除。

source 来执行脚本：在父程序中执行

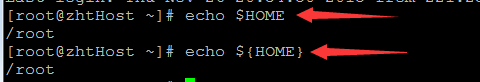


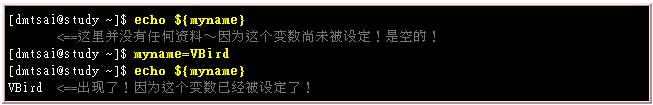
**变量：**‘变量就是以一组文字或符号等，来取代一些设定或者是一串保留的资料！’， 例如：我设定了‘myname’就是‘VBird’，所以当你读取 myname 这个变量的时候，系统自然就会知道！哈！那就是 VBird 啦！ 那么如何‘显示变量’呢？这就需要使用到 echo 这个指令啦！

你可以利用 echo 这个指令来取用变量， 但是，变量在被取用时，前面必须要加上钱字号‘ $ ’才行，举例来说，要知道 PATH 的内容，该如何是好？



请在萤幕上面显示出您的环境变量 HOME 与 MAIL：



那么我要如何‘设定’或者是‘修改’ 某个变量的内容啊？很简单啦！用‘等号(=)’连接变量与他的内容就好啦！举例来说： 我要将 myname 这个变量名称的内容设定为 VBird ，那么：

[**read 用法**](http://www.cnblogs.com/iloveyoucc/archive/2012/04/16/2451328.html)**：**

read命令接收标准输入（键盘）的输入，或其他文件描述符的输入（后面在说）。得到输入后，read命令将数据放入一个标准变量中。下面是 read命令

的最简单形式::

#!/bin/bash

echo -n "Enter your name:"   //参数-n的作用是不换行，echo默认是换行

read  name                   //从键盘输入

echo "hello $name,welcome to my program"     //显示信息

exit 0                       //退出shell程序。

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

由于read命令提供了-p参数，允许在read命令行中直接指定一个提示。

所以上面的脚本可以简写成下面的脚本::

#!/bin/bash

read -p "Enter your name:" name

echo "hello $name, welcome to my program"

exit 0

使用read命令存在着潜在危险。脚本很可能会停下来一直等待用户的输入。如果无论是否输入数据脚本都必须继续执行，那么可以使用-t选项指定一个 计时器。

-t选项指定read命令等待输入的秒数。当计时满时，read命令返回一个非零退出状态;

[**linux shell数据重定向**](http://www.cnblogs.com/chengmo/archive/2010/10/20/1855805.html)**：**

数据流分为三种：标准输入（stdin），标准输出（stdout）和标准错误输出（stderr）。

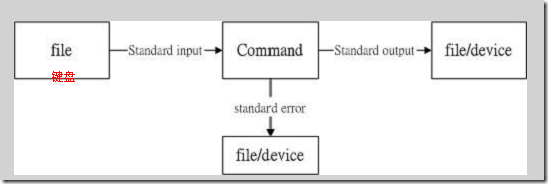
标准输出指的是命令执行所回传的正确信息，而标准错误输出指的是命令执行失败后，所回传的错误信息。这些信息默认是打印在屏幕上的。

1）标准输入（stdin）：代码为0，使用<或者<<

2）标准输出（stdout）：代码为1，使用>或者>>

3）标准错误输出（stderr）：代码为2，使用2>或者2>>

对于任何一条linux 命令执行，它会是这样一个过程：

[](http://images.cnblogs.com/cnblogs_com/chengmo/WindowsLiveWriter/linuxshell_EB66/image_2.png)

一个命令执行了：

先有一个输入：输入可以从键盘，也可以从文件得到

命令执行完成：成功了，会把成功结果输出到屏幕：standard output默认是屏幕

命令执行有错误：会把错误也输出到屏幕上面：standard error默认也是指的屏幕

linux启动后，会默认打开3个文件描述符，分别是：标准输入standard input 0,正确输出standard output 1,错误输出：error output 2

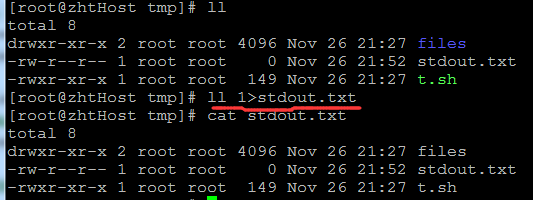
以后打开文件后。新增文件绑定描述符可以依次增加。 一条shell命令执行，都会继承父进程的文件描述符。因此，所有运行的shell命令，都会有默认3个文件描述符。

那么什么时数据流重定向呢？从字面上理解就是改变数据流的流向，使之流向指定的文件或设备。例如，把执行命令所回传的正确信息（标准输出信息）流向一个文 件，而将所回传的错误信息（标准错误输出）流向别一个文件，并把这两个文件的信息保存下来，而在屏幕上不出现任何该命令执行所回传的信息。

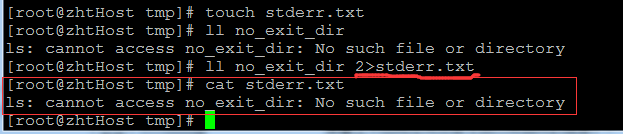
**格式：**

command-line [1~n] > file： file为或文件操作符或设备

上面命令意思是：将一条命令执行结果（标准输出，或者错误输出，本来都要打印到屏幕上面的）  重定向其它输出设备（文件，打开文件操作符，或打印机等等）1,2分别是标准输出，错误输出。

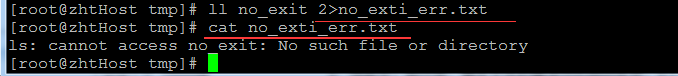
1.将标准输出重定义，>前面的1代表标准输出，是可以省略的，省略后默认为1。

2.将错误输出重定义



注意：

1. shell遇到”>”操作符，会判断右边文件是否存在，如果存在就先删除，并且创建新文件。不存在直接创建。 无论左边命令执行是否成功。右边文件都会变为空。



2、“>>”操作符，数据追加到该文件的末尾。先判断右边文件，如果不存在，先创建。以添加方式打开文件，会分配一个文件描述符[不特别指定，默认为1,2]然后，与左边的标准输出（1）或错误输出（2） 绑定。

3、当命令：执行完，绑定文件的描述符也自动失效。0,1,2又会空闲。

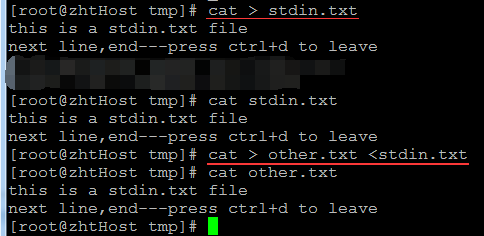
4、一条命令启动，命令的输入，正确输出，错误输出，默认分别绑定0,1,2文件描述符。

5、一条命令在执行前，先会检查输出是否正确，如果输出设备错误，将不会进行命令执行。

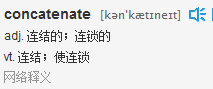
3、标准输入数据流重写向

在标准输入中，<代表将原来需要由键盘输入的数据改由文件内容来替代，<<则代表结束输入。例如：

利用cat来创建一个简单的文件catfile1，如下图所示：



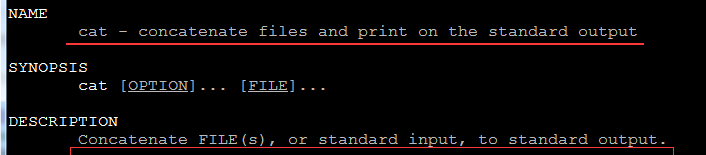
cat命令的用途是连接文件或标准输入并打印。这个命令常用来显示文件内容，或者将几个文件连接起来显示，或者从标准输入读取内容并显示，它常与重定向符号配合使用。

cat主要有三大功能：

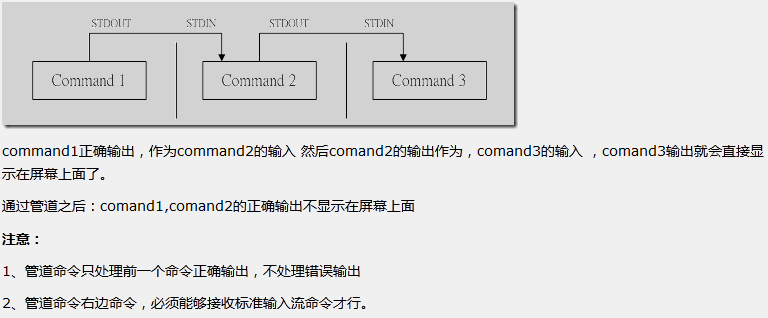
1.一次显示整个文件:cat filename

2.从键盘创建一个文件:cat > filename 只能创建新文件,不能编辑已有文件.

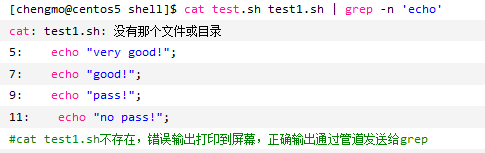
3.将几个文件合并为一个文件:cat file1 file2 > file

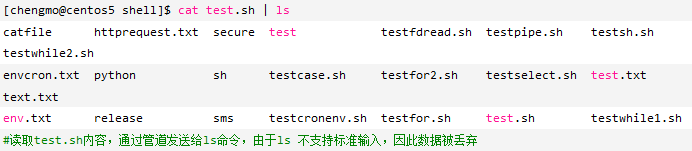


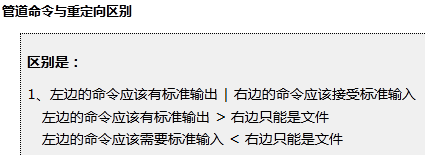
**管道命令**操作符是：”|”,它仅能处理经由前面一个指令传出的正确输出信息，也就是 standard output 的信息，对于 stdandard error 信息没有直接处理能力。然后，传递给下一个命令，作为标准的输入 standard input.



作用接收标准输入的命令才可以用作管道右边。否则传递过程中数据会抛弃。 常用来作为接收数据管道命令有：sed,awk,cut,head,top,less,more,wc,join,sort,split 等等，都是些文本处理命令。









**test的用法：**

test 命令最短的定义可能是评估一个表达式；如果条件为真，则返回一个 0 值。如果表达式不为真，则返回一个大于 0 的值 — 也可以将其称为假值。检查最后所执行命令的状态的最简便方法是使用 $? 值。

1. 关于某个文件名的『类型』侦测(存在与否)，如 test -e filename    
-e 该『文件名』是否存在？(常用)    
-f 该『文件名』是否为文件(file)？(常用)    
-d 该『文件名』是否为目录(directory)？(常用)    
-b 该『文件名』是否为一个 block device 装置？    
-c 该『文件名』是否为一个 character device 装置？    
-S 该『文件名』是否为一个 Socket 文件？    
-p 该『文件名』是否为一个 FIFO (pipe) 文件？    
-L 该『文件名』是否为一个连结档？

2. 关于文件的权限侦测，如 test -r filename    
-r 侦测该文件名是否具有『可读』的属性？    
-w 侦测该文件名是否具有『可写』的属性？    
-x 侦测该文件名是否具有『可执行』的属性？    
-u 侦测该文件名是否具有『SUID』的属性？    
-g 侦测该文件名是否具有『SGID』的属性？    
-k 侦测该文件名是否具有『Sticky bit』的属性？    
-s 侦测该文件名是否为『非空白文件』？

3. 两个文件之间的比较，如： test file1 -nt file2    
-nt (newer than)判断 file1 是否比 file2 新    
-ot (older than)判断 file1 是否比 file2 旧    
-ef 判断 file2 与 file2 是否为同一文件，可用在判断 hard link 的判定上。 主要意义在判定，两个文件是否均指向同一个 inode 哩！

4. 关于两个整数之间的判定，例如 test n1 -eq n2    
-eq 两数值相等 (equal)    
-ne 两数值不等 (not equal)    
-gt n1 大于 n2 (greater than)    
-lt n1 小于 n2 (less than)    
-ge n1 大于等于 n2 (greater than or equal)    
-le n1 小于等于 n2 (less than or equal)

5. 判定字符串的数据    
test -z string 判定字符串是否为 0 ？若 string 为空字符串，则为 true    
test -n string 判定字符串是否非为 0 ？若 string 为空字符串，则为 false。   
注： -n 亦可省略    
test str1 = str2 判定 str1 是否等于 str2 ，若相等，则回传 true    
test str1 != str2 判定 str1 是否不等于 str2 ，若相等，则回传 false

6. 多重条件判定，例如： test -r filename -a -x filename    
-a (and)两状况同时成立！例如 test -r file -a -x file，则 file 同时具有 r 与 x 权限时，才回传 true。    
-o (or)两状况任何一个成立！例如 test -r file -o -x file，则 file 具有 r 或 x 权限时，就可回传 true。    
! 反相状态，如 test ! -x file ，当 file 不具有 x 时，回传 true