<https://www.cnblogs.com/SamLXW/p/6196999.html>

**一、认识bash shell**

1、登录取得的shell就记录在/etc/passwd这个文件内

　　可以使用cat /etc/passwd查看

**2、bash shell 功能**

　　a. 命令记忆能力（history），默认1000个，存在~/.bash\_history文件

　　b. 命令与文件补全功能（Tab键的好处）

　　　　【Tab】接在一串命令的第一个字的后面，则为命令补全

　　　　【Tab】接在一串命令的第二个字以后时，则为文件补齐。

　　c. 命令别名设置功能（alias）, alias lm='ls -al'

　　d. 作业控制、前台、后台控制

　　e. 程序脚本（shell script）

　　f.  通配符（Wildcard）

**3、bash shell 的内置命令： type**

　　语法：type [-tpa] name

　　参数：

　　　type: 不加任何参数时， type会显示name是外部命令还是bash内置命令

　　　-t:   当加入-t 参数时， type会将name以下面这些字眼显示出它的意义：

　　　　　　file：表示为外部命令；

　　　　　　alias： 表示该命令为命令别名所设置的名称

　　　　　　builtin : 表示该命令为bash内置的命令功能

　　　-p:  如果后面接的name为外部命令时， 才会显示完整文件名

　　　-a ：会由PATH变量定义的路径中，将所有含name的命令都列出来，包含alias

　　type主要在找出“执行文件”而不是一般文件名。所以，这个type也可以用来作为类似which命令的用途

**二、shell 的变量功能**

　　1、变量的显示与设置： echo， unset

　　　　变量的显示： echo ---》 echo $variable 或 echo ${varible}

　　　　变量的设置规则：

　　　　　　a. 变量与变量内容以一个等号“=”来连接，如下所示： myname=sam

　　　　　　b. 等号两边不能直接接空格符

　　　　　　c. 变量名称只能是英文字母与数字，但是开头字符不能是数字

　　　　　　d. 变量内容若有空格符可使用双引号" 或单引号 ' 将变量内容结合起来，但是双引号内的特殊字符如$等，可以保留原本的特性，单引号内的特殊字符则仅为一般字符（纯文本）

　　　　　　e.可用转义字符“\”将特殊符号变成一般字符

　　　　　　f. 在一串命令中，还需要通过其他的命令提供的信息，可以使用反单引号。`命令` 或 $(命令)，

　　　　　　　　特别注意，那个是键盘上方数字键1左边那个按键，而不是单引号。

　　　　　　g. 若该变量为了增加变量内容时， 则可用$变量名称或￥{变量}累加内容，如下：

　　　　　　　　PATH=$PATH:/home/bin

　　　　　　h. 若该变量需要在其他子进程执行，则需要以export来使变量变成环境变量：

　　　　　　　　"export PATH"

　　　　　　i. 通常大写字符为系统默认变量，自行设置变量可以使用小写字符， 方便判断

　　　　　   j.取消变量： unset 变量名称

　　　　子进程： 就是说在我目前这个shell的情况下，去打开另一个新的shell，新的那个shell就是子进程。在一般的状态下，父进程的自定义变量是无法再子进程内使用的。但是通过export将变量变成环境变量后，就能够在子进程下面应用了。

　　2、 环境变量的功能

　　　　查看环境量： env

　　　　常见的环境变量:

　　　　　　HOME: 代表用户的主文件夹。

　　　　　　SHELL: 它告知我们目前这个环境使用的shell是哪个程序？Linux默认使用/bin/bash的

　　　　　　HISTSIZE: 这个与“历史命令”有关，即是我们曾经执行过的命令可以被系统记录下来，而记录的“条数”则是由这个值来设置的。

　　　　　　MAIL：当我们使用mail这个命令在收信时系统会去读取的邮件信箱文件

　　　　　　PATH: 就是执行文件查找的路径，目录 与目录中间以冒号（:）分隔，由于文件的查找是依序由PATH的变量内的目录来查询，所以目录的顺序也是重要的。

　　　　　　LANG：语系数据

　　　　　　RANDOM: 这是“随机数”的变量。在BASH的环境下，这个RANDOM变量的内容介于0~32767

**set查看所有变量（含有环境变量与自定义变量）**

**a. PS1（提示符的设置）：**

　　　　　　这是PS1(数字的1， 不是英文字母)，这个东西就是我们的“命令提示符”。当我们每次按下【Enter】键去执行某个命令后，最后要在次出现提示符时， 就会主动去读取这个变量值了。上面PS1内显示的是一些特殊符号，这些特殊符号可以显示不同的信息，每个distributions的bash默认的PS1变量内容可能有些区别，你可以用man bash 查询一下PS1的相关说明，以理解下面的一些符号意义。

　　　　\d： 可显示出“星期月日”的日期格式， 如“Mon Feb 2”

　　　　\H: 完整的主机名， 如“www.vbird.tsai”

　　　　\h: 仅去主机名在第一个小数点之前的名字, 如“www”

　　　　\t: 显示时间，为24小时格式的“HH:MM:SS”

　　　　\T: 显示时间， 为12小时格式的“HH:MM:SS”

　　　　\A: 显示时间，为24小时格式的“HH:MM”

　　　　\@：显示时间， 为12小时格式的“am/pm”样式

　　　　\u: 目前用户的账号名称，如“root”

　　　   \v: BASH的版本信息

　　　　\w: 利用basename函数取得工作目录名称，所以仅会列出最后一个目录名

　　　　\#: 执行的第几个命令

　　　　\$: 提示符，如果是root时，提示符为#， 否则就是$

　　　　例： PS1='[\u@\h \w \A #\#]\$ ' --------》 [root@www /home/dmtsai 16:40 #12]#

**b.$(关于本shell的PID)**

　　　　　　“$”本身也是个变量。这个代表的是目前这个Shell的线程号，即是所谓的PID（Process ID）

**c. ？ (关于上个执行命令的回传码)**

**这个变量是上一个执行命令所传回的值， 当我们执行某些命令时， 这些命令都会回传一个执行后的代码。一般来说， 如果成功执行该命令，则会回传一个0值。如果发生错误，则回传一个非0值。**

**d. OSTYPE, HOSTTYPE, MACHTYPE(主机硬件与内核的等级）**

**export: 自定义变量转成环境变量**

**export 变量名称**

**3、影响显示结果的语系变量（locale）**

**4、变量键盘读取、数组与声明： read， array， declare**

**a. read : 读取来自键盘输入的变量**

　　　　read 【-pt】 variable

　　　　-p: 后面可以接提示符

　　　　-t: 后面可以接等待的“秒数”

**b. declare/typeset**

**declare或typeset是一样的功能，就是声明变量的类型。**

**declare [-aixr] variable**

**参数：**

**-a: 将后面名为variable的变量定义成为数组（array）类型**

**-i： 将后面名为variable的变量定义为整数数字（integer）类型**

**-x: 用法与export一样，就是将后面的variable变成环境变量, 注意将-变成+可以进行“取消”操作，将环境变量变成一般变量**

**-r： 将变量设置成为readonly类型，该变量不可被更改内容，也不能重设**

**由于在默认的情况下面， bash对于变量有几个基本的定义：**

**a. 变量类型默认为“字符串”， 所以若不指定变量类型， 则1+2为一个“字符串”而不是“计算式”，**

**b. bash环境中的数值计算， 默认最多仅能到达整数类型，所以1/3结果是0**

**数组（arry）变量类型**

**var[index]=content**

**读取： ${var[index]}**

**5、与文件系统及程序的限制关系： ulimit**

**bash可以限制用户的某些系统资源的，包括可以打开的文件数量、可以使用的CPU时间、可以使用的内存总量等**

**ulimit [-SHacdfltu] [配额]**

**参数：**

**-H： hard limit 严格的设置，必定不能超过这个设置的数值**

**-S： soft limit 警告的设置，可以超过这个设置值，但是若超过则有警告信息**

**-a: 后面不接任何参数，可列出所有的限制额度**

**-c: 当某些进程发生错误时，系统可能会将该进程在内存中的信息写成文件（排错用）， 这种文件就被称为内核文件。此为限制每个内核文件的最大容量。**

**-f: 此shell可以创建的最大文件容量（一般可能设置为2GB）单位为KB**

**-d: 进程可使用的最大断裂内存（segment）容量**

**-l： 可用于锁定（lock）的内存量。**

**-t： 可使用的最大CPU时间（单位为秒）**

**-u: 单以用户可以使用的最大进程数量。**

**注意： 想要复原ulimit的设置最简单的方法就是注销再登陆，否则就是得要重新以ulimit设置才行。不过要注意的是一般身份用户如果以ulimit设置了-f的文件大小，那么他只能继续减少文件的容量，不能增加文件的容量。**

**6、变量内容的删除、替代与替换**

**变量内容的删除与替换**

|  |  |
| --- | --- |
| 变量设置方式 | 说明 |
| ${变量#关键字} | 若变量内容从头开始的数据符合“关键字”， 则将符合的最短数据删除 |
| ${变量##关键字} | 若变量内容从头开始的数据符合“关键字”， 则将符合的最长数据删除 |
| ${变量%关键字} | 若变量内容从尾向前的数据符合“关键字”，则将符合的最短数据删除 |
| ${变量%%关键字} | 若变量内容从尾向前的数据符合“关键字”， 则将符合的最长数据删除 |
| ${变量/旧字符串/新字符串} | 若变量内容符合“旧字符串”， 则第一个旧字符串被新字符串替换 |
| ${变量//旧字符串、新字符串} | 若变量内容符合“旧字符串”， 则全部旧字符串被新字符串替换 |

**变量的测试与内容替换**

**在某些时刻我们经常需要“判断”某个变量是某存在，若变量存在则使用既有的设置，若变量不存在则给予一个常用的设置。**

**new\_var=${old\_var-content}**

**new\_var: 新的变量，主要用来替换旧变量。新旧变量名称其实经常是一样的**

**old\_var: 旧的变量， 被测试的选项**

**content: 变量的“内容”**

**注： new\_var=${old\_var:-content}**

**加上：后若变量内容为空或者为未设置，度能够以后面的内容替换**

**如果想要将旧变量内容也一起替换掉的话，那么就是用等号（=）：**

**如： unset str; var=${str=newvar} --->var=newvar, str=newvar**

**str="oldvar"; var=${str=newvar}--->var=newvar, str=oldvar**

**如果只想知道，如果旧变量不存在时， 整个测试就告知我“有错误”，此时就能够使用问号“？”**

**如： var=${str?无此变量}**

**三、命令别名与历史命令**

**1、命令别名设置：alias, unalias**

**例： alias rm='rm -i'**

**alias 列出所有命令别名**

**unalias rm 解除命令别名**

**2、历史命令：history**

**history [n]**

**history [-c]**

**history [-raw] histfiles**

**参数：**

**n: 数字，是要列出最近的n条命令行的意思**

**-c: 将目前的shell中的所有history内容全部消除。**

**-a: 将目前新增的history命令新增入histfiles中， 若没有加histfiles，默认写入~/.bash\_history**

**-r: 将histfiles的内容读到目前这个shell的history记忆中**

**-w： 将目前的history记忆内容写入histfiles中**

**有关history的命令：**

**！number ：执行第几条命令**

**!command ： 由最近的命令向前搜寻命令串开头为command的那个命令，并执行**

**!! ： 就是执行上一个命令（相当于按up键后，按enter键）**

**四、Bash Shell的操作环境**

**1、路径与命令查找顺序**

**基本上，命令运行的顺序可以这样看：**

**1、以相对/绝对路径执行命令，例如: "/bin/ls"或“./ls”**

**2、由alias找到该命令来执行**

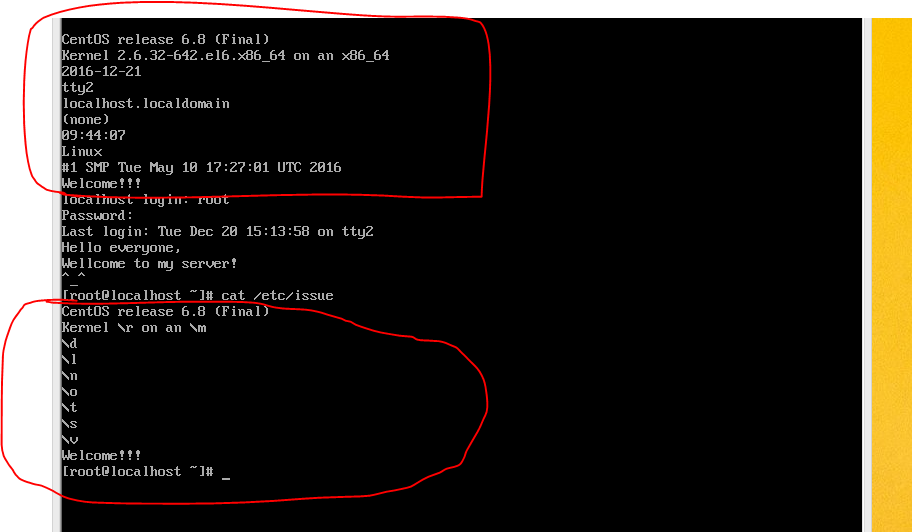
**3、由bash内置的（builtin）命令来执行**

**4、通过$PATH这个变量的顺序找到的第一个命令来执行**

**可以通过type -a command 查看执行顺序**

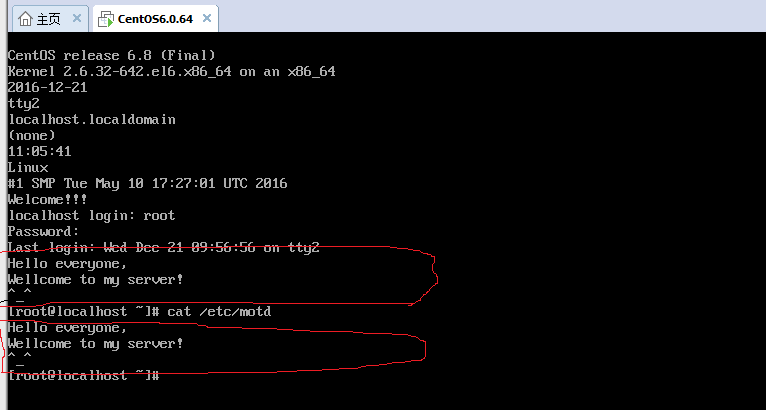
**2、bash的登录与欢迎信息：/etc/issue, /etc/motd**

**a. 终端机接口（tty1~tty6）登录时的登录界面: /etc/issue登录界面提示符设置文件**

****

|  |  |
| --- | --- |
|  | issue内的各个代码意义 |
| \d | 本地端时间的日期 |
| \l | 显示第几个终端机接口 |
| \m | 显示硬件的等级（i386/i486/i586/i686...） |
| \n | 显示主机的网络名称 |
| \o | 显示domain name |
| \r | 操作系统的版本（相当于rname -r） |
| \t | 显示本地端的时间 |
| \s | 操作系统的名称 |
| \v | 操作系统的版本 |

 　　　　　　如果你想要让用户登录后取得一些信息，例如你想要让大家都知道的信息，那么可以将信息加入/etc/motd里面去：如下



**3、bash的环境配置文件**

**（1）、 login与non-login shell**

**login shell:取得bash完整的登录流程中，就称为login shell.**

**non-login shell:**

**login shell 只会读取这两个配置文件：**

**/etc/profile:这是系统整体的设置，最好不要修改此配置文件**

**~/.bash\_profile或~/.bash\_login或~/.profile:　属于用户个人设置**

**/etc/profile（login shell才会读）：**

**这个文件设置的变量主要有：**

**PATH:会依据UID决定PATH变量要不要含有sbin的系统命令目录**

**MAIL： 依据账号设置好用户的mailbox到/var/spool/mail/账号名**

**USER: 根据用户的账号设置此变量内容**

**HOSTNAME: 依据主机的hostname命令决定此变量内容**

**HISTSIZE：历史命令记录条数**

**/etc/profile不止会做这些事， 还会去调用外部的设置数据，下面这些数据会依序被调用进来：**

**a. /etc/inputrc:  这个文件会主动判断用户有没有自定义输入的按键功能，如果没有，/etc/profile就会决定设置“INPUTRC=/etc/inputrc”这个变量。此文件内容为bash的热键、【Tab】键有没有声音等的数据。**

**b. /etc/profile.d/\*.sh:  只要在/etc/profile.d/这个目录内且扩展名为.sh，另外用户能够具有r的权限，那么该文件就会被/etc/profile调用。这个目录下面的文件规定了bash操作接口的颜色、语系、ll与ls命令别名、vi的命令别名、which的命令别名等。如果需要帮所有用户设置一些共享的命令别名时，可以在这个目录下面自行创建扩展名为.sh的文件，并将所需要的数据写入即可。**

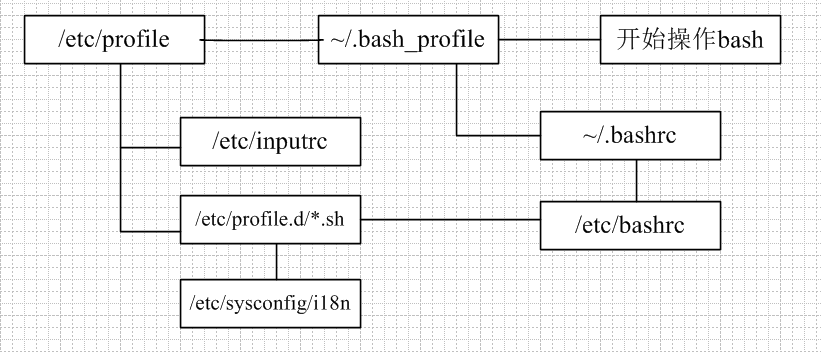
**c. /etc/sysconfig/i18n: 这个文件是由/etc/profile.d/lang.sh调用的。这也是我们决定bash默认使用何种语系的重要配置文件。文件里最重要的就是LANG这个变量的设置。**

**注意：bash的login shell情况下所读取的整体环境配置文件其实只有/etc/profile,但是/etc/profile还会调用其他的配置文件。**

**其实bash在都玩了整体环境设置的/etc/profile并借此调用其他配置文件后，接下来则是会读取用户的个人配置文件。在login shell的bash环境中，所读取的个人偏好配置文件其实主要有三个，依序分别是：**

**~/.bash\_profile, ~/.bash\_login, ~/.profile, 其实bash的loginshell设置只会读取上面三个文件的其中一个，而读取的顺序则是依照上面的顺序。**

**读出顺序如下：**

****

 （2）：source: 读入环境配置文件的命令

　　　　source 配置文件名

　　　　. 配置文件名

　　　　利用source或小数点（.）都可以将配置文件的内容读进目前的shell环境中。

　（3）： non-login shell:

　　　　　　当你取得non-login shell时， 该bash配置文件仅会读取~/.bashrc而已

 　（4）： /etc/bashrc帮我们的bash定义下面的数据：

　　　　　　根据不同的UID规定umask的值

　　　　　　依据不同的UID规定提示符（就是PS1变量）

　　　　　　调用/etc/profile.d/\*.sh的设置

　　（5）、其他相关配置文件

　　　　a、 /etc/man.config: 这个文件的内容规定了使用man的时候man page的路径到哪里去寻找。

　　　　b、~/.bash\_history: 每次登录bash后，bash会先读取这个文件，将所有的历史命令读入内存

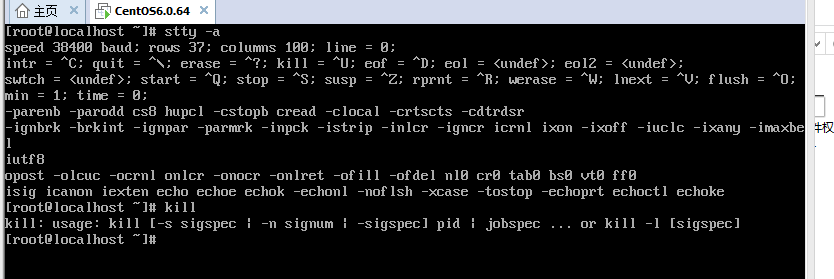
　　　　c、 ~/.bash\_logout: 这个文件记录当我们注销bash后系统再帮我们做完什么操作后才离开

**4、终端机的环境设置： stty， set**

 　　　a、 **查阅目前的一些按键内容： stty [-a]**

　　　　　　参数： -a : 将目前所有的stty参数列出来

**设置热键： 如 stty erase ^h       ^h--->[ctrl]+h**

****

　　eof: End of file 的意思，代表结束输入

　　erase: 向后删除字符

　　intr: 送出一个interrupt（中断）信号给目前正在运行的程序

　　kill： 删除在目前命令行上的所有文字

　　quit： 送出一个quit的信号给目前正在运行的进程

　　start: 在某个进程停止后，重新启动它的输出

　　stop： 停止目前屏幕的输出

　　susp： 送出一个terminal stop的信号给正在运行的进程

　　set 可以帮我们设置整个命令输出/输入的环境。例如记录历史命令、显示错误内容等。

　　参数：

　　　　-u: 默认不启用，若启用后，当使用未设置时，会显示错误信息。

　　　　-v： 默认不启用，若启用后，在讯息被输出前，会先显示信息的原始内容

　　　　-x: 默认不启用，若启用后，在命令被执行前，会显示命令内容（前面有++符号）

　　　　-h: 默认启用，与历史命令有关

　　　　-H： 默认启用，与历史命令有关

　　　　-m: 默认启用，与工作管理有关

　　　　-B：默认启用，与括号[]的作用有关

　　　　-C： 默认不启用，使用>等时， 则若文件存在时，该文件不会被覆盖。

　　b 、显示目前所有的set设置值： echo $- , 默认为himBH

**c、 另外我们还有其他的按键设置功能，就是在前一小节提到的/etc/inputrc这个文件里面设置。**

**cat /etc/inputrc**

 　　d、 bash默认的组合键给它汇整如表：

|  |  |
| --- | --- |
| 组合按键 | 执行结果 |
| Ctrl+C | 终止目前的命令 |
| Ctrl+D | 输入结束（EOF）,例如邮件结束的时候 |
| Ctrl+ M | 就是Enter |
| Ctrl + S | 暂停屏幕的输出 |
| Ctrl+ Q | 恢复屏幕的输出 |
| Ctrl+U | 在提示符下，将整行命令删除 |
| Ctrl+ Z | 暂停目前的命令 |

**5、通配符与特殊符号**

**a、bash中的通配符**

|  |  |
| --- | --- |
| **符号** | **意义** |
| **\*** | **代表0个到无穷多个任意字符** |
| **？** | **代表一定有一个任意字符** |
| **[]** | **同样代表一定有一个在中括号内的字符（非任意字符）。例如[abc]代表一定有一个字符，可能是abc中的任一个** |
| **[-]** | **若有减号在中括号内时，代表在编码顺序内的所有字符，例如[0-9]代表0到9之间的所有数字，因为数字的语系编码是连续的** |
| **[^]** | **若中括号内的第一个字符为指数符号(^)，那表示原向选择，例如[^abc]代表一定有一个字符，只要是非a,b,c的其他字符就接受的意思** |

**特殊字符：**

|  |  |
| --- | --- |
| **符号** | **内容** |
| **#** | **批注符号，这个最常被使用在script当中，视为说明。其后的数据均不执行** |
| **\** | **转义字符，将“特殊字符或通配符”还原成一般字符** |
| **|** | **管道（pipe）， 分隔两个管道命令的界定** |
| **;** | **连续命令执行分隔符，连续性命令的界定（注意，与管道命令不相同）** |
| **~** | **用户的主文件** |
| **$** | **使用变量前导符，即是变量之前需要加的变量替代值** |
| **&** | **作业控制（job control）, 将命令变成背景下工作** |
| **!** | **逻辑运算意义上的“非”的意思** |
| **>,>>** | **数据流重定向，输出导向，分别是“替换”与“累加”** |
| **<,<<** | **数据重定向， 输入导向** |
| **''** | **单引号，不具有变量置换的功能** |
| **""** | **具有变量置换的功能** |
| **``** | **两个“`”中间为可以先执行的命令，也可使用$()** |
| **()** | **在中间为子shell的起始于结束** |
| **{}** | **在中间为命令块的组合** |
|  |  |

**五、数据流重定向**

**数据重定向： 就是将某个命令执行后要出现在屏幕上的数据传输到其他的地方，例如文件或者是设备。**

**1、什么是数据流重定向**

**standard output 与 standard error output:**

**标准输出指的是命令执行所回传的正确的信息， 而标准错误输出可理解为命令执行失败后，所回传的错误信息。**

**数据重定向可以将standard output（简称stdout）与standard error output（简称stderr）分别传送到其他的文件或设备区，分别传送所用的特殊字符则如下所示：**

**a、 标准输入（stdin）: 代码为0，使用<或<<**

**b、 标准输出（stdout）: 代码为1， 使用>或>>**

**c、 标准错误输出（stderr）: 代码为2， 使用2> 或2>>**

**覆盖于累加输出：**

**单个>:覆盖    两个>(>>)： 累加**

**a、 /dev/null垃圾桶黑洞设备与特殊写法：**

**/dev/null可以吃掉任何导向这个设备的信息。**

**如 find /home -name .bashrc 2> /dev/null**

**b、将正确与错误数据通通写入同一个文件的正确写法：**

**find /home -name .bashrc >list 2>&1**

**find /home ianme .bashrc &>list**

**c、 standard input: < 与 <<:**

**<: 将原本需要由键盘输入的数据改由文件内容来替代。**

**例如： cat >file <~/.bashrc**

**<<：代表结束输入的意思**

**2、命令执行的判断依据： ; , &&, ||**

　　　　a、cmd;cmd(不考虑命令相关性的连续命令执行)

 　　　 b、$?(命令回传码)与&& 或 ||

|  |  |
| --- | --- |
|  |  |
| cmd1 && cmd2 | 若cmd1执行完毕且正确执行（$?=0）, 则执行cmd2 |
| 若cmd1执行完毕且为错误（$?！=0）, 则不执行cmd2 |
| cmd1 || cmd2 | 若cmd1执行完毕且正确执行（$?=0）, 则不执行cmd2 |
| 若cmd1执行完毕且为错误（$?!=0）, 则执行cmd2 |

**六、管道命令（pipe）**

　　　　管道命令： “|”仅能出来经由前面一个命令传来的正确信息，也就是standard output的信息，对于stdandard error并没有直接处理的能力。在每个管道后面接的第一个数据必定是“命令”， 而且这个命令必须要能够接收standard input的数据才行，这样的命令才可以是“管道命令”，如less，more， head， tail等都是可以接收standard input的管道命令。

**管道命令主要有两个比较需要注意的地方：**

**管道命令必须要能够接收来自前一个命令的数据成为standard input继续处理才行**

**管道命令仅会处理standard output， 对于standard error output会予以忽略**

**2、选取命令： cut, grep**

　　　　a、cut命令将一段信息的某一段“切”出来，处理的信息以“行”为单位。

　　　　　　cut -d '分隔字符' -f fields

　　　　　　cut -c 字符范围

　　　　　参数：

　　　　　　-d: 后面接分隔字符， 与-f一起使用

　　　　　　-f: 依据-d的分隔字符将一段信息切割成为数段，用-f取出第几段的意思

　　　　　　-c: 以字符的单位取出固定字符区间

　　　　b、 grep：分析一行信息，若当中有我们所需要的信息，就将该行拿出来，简单的语法是这样的

　　　　　　　　grep [-acinv] [--color=auto] '查找字符串' filename

　　　　　　　　参数：

　　　　　　　　　　-a: 将binary文件以text文件的方式查找数据

　　　　　　　　　　-c: 计算找到‘查找字符串’的次数

　　　　　　　　　　-i: 忽略大小写的不同，所以大小写视为相同

　　　　　　　　　　-n: 顺便输出行号

　　　　　　　　　　-v: 反向选择，即显示出没有‘查找符号’内容的那一行

　　　　　　　　　　--color=auto: 可以将找到的关键字部分加上颜色显示

**3、排序命令：sort, wc, uniq**

**a、sort: 帮我们进行排序，而且可以依据不同的数据类型来排序。例如数字与文字的排序就不一样。此外排序的字符与语系的编码有关。**

　　　　sort [-fbMnrtuk] [file or stdin]

　　　　参数：

　　　　　　-f: 忽略大小写的差异， 例如A与a视为编码相同

　　　　　　-b： 忽略最前面的空格符部分

　　　　　　-M： 以月份的名字来排序，例如JAN， DEC等的排序方法

　　　　　　-n： 使用“纯数字”进行排序（默认是以文字类型排序）

　　　　　　-r: 反向排序

　　　　　　-u： 就是uniq，相同的数据中，仅出现一行代表

　　　　　　-t： 分隔符， 默认是用【tab】键来分隔

　　　　　　-k： 以那个区间（field）来进行排序的意思

　　　　例如： /etc/passwd 内容以： 来分隔，我想以第三列来排序

　　　　　　　　cat /etc/passwd | sort -t ':' -k 3

**b、uniq**

　　　　uniq [-ic]

　　　　参数：

　　　　　　-i: 忽略大小写字符的不同

　　　　　　-c: 进行计数

**c、 wc**

　　　　wc [-lwm]

　　　　参数：

　　　　　　-l： 仅列出行

　　　　　　-w： 仅列出多少字（英文单字）

　　　　　　-m: 多少字符

**3、双向重定向： tee**

**tee会同时将数据流送与文件与屏幕（screen）;而输出到屏幕的，其实就是stdout，可以让下个命令继续处理：**

**tee [-a] file**

**参数：**

**-a: 以累加的方式， 将数据加入file当中**

**如： ls -l /home | tee ~/homefile | more**

**将ls 的数据存一份到~/homefile，同时屏幕也有输出信息**

**4、字符转换命令： tr， col， join， paste， expand**

**a、 tr: 可以用来删除一段信息当中的文字， 或者是进行文字信息的替换**

**tr [-ds] SET1 ...**

**参数：**

**-d: 删除信息当中的SET1这个字符串**

**-s: 替换掉重复的字符**

**例： 将last输出的信息中所有的小写字符变成大写字符**

**last | tr '[a-z]' '[A-Z]'**

**b、 col: 可以用来进行简单处理，如将【tab】按键替换成为空格键**

**col [-xb]**

**参数：**

**-x: 将tab键转换成对等的空格键**

**-b： 在文字内有反斜杠（/）时， 仅保留反斜杠最后接的那个字符**

**c、join : 将两个文件当中有相同数据的那一行加在一起。**

**jion [-ti12] file1 file2**

**参数：**

**-t: join默认以空格符分隔数据，并且对比“第一个字段”的数据**

**-i: 忽略大小写的差异**

**-1： 这个是数字的1， 代表第一个文件要用哪个字段来分析的意思**

**-2： 代表第二个文件要用哪个字段来分析的意思**

**例： /etc/passwd的第四个字段是GID， 那个GID记录在/etc/group当中的第三个字段，以GID整合两个文件**

**join -t ':' -1 4 /etc/passwd -2 3 /etc/group**

**注意： 在使用join之前，你所需要处理的文件应该要先经过排序（sort）处理，否则有些对比的项目会被略过。**

**d、 paste： 介质将两行贴在一起，且中间以【tab】键隔开而已**

**paste [-d] file1 file2**

**参数：**

**-d: 后面可以接分隔字符，默认是以【tab】来分隔**

**-： 如果file部分写成-， 表示来自standard iput的数据的意思**

**e、 expand:  将【tab】按键转成空格键， 可以这样做：**

**expand [-t] file**

**参数：**

**-t: 后面可以接数字。一般来说，一个【tab】按键可以用8个空格键替换，我们也可以自行定义一个【tab】代表多少个字符**

**注： unexpand： 将空白转成【tab】命令**

**5、切割命令：split**

**如果你有文件太大，导致一些携带式设备无法复制的问题。split可以将一个大文件依据文件大小或行数来切割成为小文件。**

**split [-bl] file PREFIX**

**参数：**

**-b： 后面可接欲切割的文件大小，可加单位，例如b，k， m等**

**-l： 以行数来进行切割**

**PREFIX： 可作为切割文件的前导文字**

**6、参数代换： xargs**

**产生某个命令的参数的意思。xargs可以读入stdin的数据，并且以空格符或断行字符进行分辨，将stdin的数据分隔成为arguents。**

**xargs [-0epn] command**

**-0: 如果输入的stdini含有特殊字符， 例如`, \, 空格键字符时， 这个参数可以将它还原成一般字符。这个参数可以用于特殊状态**

**-e: 这个是EOF（end of file）的意思。后面可以接一个字符串，当xargs分析到这个字符串时，就会停止继续工作**

**-p: 在执行每个命令的参数时， 都会询问用户的意思**

**-n: 后面接次数，每次command命令执行时， 要使用几个参数的意思**

**当xargs后面没有接任何的命令时， 默认是以echo来进行输出**

**例如：  将/etc/passwd内的第一列取出，仅取三行， 使用finger这个命令将每个账号内容显示出来**

**cut -d ':' -f 1 /etc/passwd | head -n 3 | xargs finger**

**7、关于减号-的用途**

**在管道命令中，经常会使用到前一个命令的stdout作为这次的stdin，某些命令需要用到文件名来进行处理时，该stdin与stdout可以利用减号“-”来替代，举例如下：**

**tar -cvf - /home | tar -xvf -**

**将/home里面的文件打包，但打包的数据不是记录到文件，而是传送到stdout;经过管道后，将tar -cvf - /home传送给后面的tar -xvf -后面的这个-则是取用前一个命令的stdout**