**历史命令，命令别名，常用快捷键，输入输出重定向，多命令顺序执行及管道符，通配符和其他特殊符号**

# history 历史命令

功能：帮助用户回忆执行过的命令。

    -c 清除所有历史命令

    -w    把缓存中的历史命令写入历史命令保存文件~/.bash\_history

     !命令号   执行这条历史命令

     !字符串    执行以该字符串开头的最近的一条历史命令

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

**使用 Ctrl+R 搜索历史**

对命令历史进行搜索，按上下键能查找，按左右键能修改，按回车键就可以执行该命令。

**快速重复执行上一条命令**

1,使用上方向键，并回车执行。

2,按 !! 并回车执行。

3,输入 !-1 并回车执行。

4,按 Ctrl+P 并回车执行。

# 命令别名

alias  别名="命令名"        临时生效

eg:alias  copy =cp   alias xrm=”rm -r”

#alias 查看别名信息

unalias 别名      删除别名

eg： unalias copy

**让别名永久生效**    #vi   /root/.bashrc     添加别名

**命令执行时顺序**

1，第一顺位执行用绝对路径或相对路径执行的命令。

2，第二顺位执行别名。

3，第三顺位执行bash的内部命令。

4，第四顺位执行按照$PATH环境变量定义的目录查找顺序找到的第一个命令。

# 输入输出重定向

**同标准I/O一样，Shell对于每一个进程预先定义3个文件描述字（0,1,2）。**

**分别对应于：0：STDIN 标准输入    1：STDOUT 标准输出   2：STDERR 标准错误输出。**

默认地，标准的输入为键盘，但是也可以来自文件或管道（pipe |）。设备文件名为/dev/stdin。  
默认地，标准的输出为终端（terminal)，但是也可以重定向到文件，管道或后引号（backquotes `）。设备文件名为/dev/stdout。

默认地，标准的错误输出到终端，但是也可以重定向到文件。设备文件名为/dev/stderr。   
标准的输入，输出和错误输出分别表示为STDIN,STDOUT,STDERR，也可以用0，1，2来表示。

还有3~9也可以作为文件描述符。3~9你可以认为是执行某个地方的文件描述符，常被用来作为临时的中间描述符。

**>或>>  输出重定向  >会替换；>>会累加，不会替换之前的内容**

eg: #ls -l /tmp > /tmp.msg   (替换)

eg:#date >> /tmp.msg   （追加）

可以这么创建文件： echo "hehe" >> /home/test1.txt

**< 输入重定向**

eg：wall < /etc/issue.net  把issue.net里的内容广播出去

eg：tr "a-z" "A-Z" < /etc/passwd        将小写转为大写（输入重定向）,在屏幕输出

**自动创建文件**

cat > log.txt << EXIT  (这里的EXIT只是一个标志，可变动，再输入这个标志或ctrl+d 即可结束写入)

> ccc

> ddd

> EXI

按ctrl+d 结束输出

**2> 错误输出重定向**

eg:cp -R /usr /backup/usr.bak 2> /bak.error  
（0和1的输入输出重定向把数字省略了）

**正确输出和错误输出同时保存**

命令 > 文件 2>&1    覆盖方式，把正确输出和错误输出都保存到同一个文件当中。

命令 >> 文件 2>&1    追加方式，把正确输出和错误输出都保存到同一个文件当中。

命令 &> 文件    覆盖方式，把所有输出都保存到同一个文件当中。

命令 &>> 文件    追加方式，把所有 输出都保存到同一个文件当中。

命令 >> 文件1  2>>文件2    正确输出追加到文件1中，错误输出追加到文件2中。

& 把程序放到后台执行，但是当终端关闭时，程序可能关闭

nohup  能保证 终端关闭时，进程依然存在

----------------------------------------------

**分析 2>&1       以>/dev/null  2>&1  为例:**

分解这个组合：  
            1：> 代表重定向到哪里，例如：echo '123' > /home/123.txt  
            2：/dev/null 代表空设备文件  
            3：2> 表示重定向stderr标准错误  
            4：& 表示等同于的意思，2>&1，表示2的输出重定向等同于1  
            5：1 表示stdout标准输出，系统默认值是1，所以'>/dev/null'等同于 '1>/dev/null'  
因此，>/dev/null 2>&1也可以写成“1> /dev/null 2> &1”

那么>/dev/null 2>&1语句执行过程为：  
   1>/dev/null ：首先表示标准输出重定向到空设备文件，也就是不输出任何信息到终端，说白了就是不显示任何信息。  
   2>&1 ：接着，标准错误输出重定向到标准输出，因为之前标准输出已经重定向到了空设备文件，所以标准错误输出也重定向到空设备文件。

在Linux下最常用的方式有两种：

         command > file 2>file  与command > file 2>&1

首先command > file 2>file 的意思是将命令所产生的标准输出信息,和错误的输出信息送到file 中。command  > file 2>file 这样的写法,stdout和stderr都直接送到file中, file会被打开两次,这样stdout和stderr会互相覆盖,这样写相当使用了FD1和FD2两个同时去抢占file 的管道。  
而command >file 2>&1 这条命令就将stdout直接送向file, stderr 继承了FD1管道后,再被送往file,此时,file 只被打开了一次,也只使用了一个管道FD1,它包括了stdout和stderr的内容。  
从IO效率上,前一条命令的效率要比后面一条的命令效率要低,所以在编写shell脚本的时候,较多的时候我们会command > file 2>&1 这样的写法。

扩展一下

        另外一个非常有用的重定向操作符是 '-'，请看下面这个例子：  
         $ (cd /source/directory && tar cf - . ) | (cd /dest/directory && tar xvfp -)

该命令表示把 /source/directory 目录下的所有文件通过压缩和解压，快速的全部移动到/dest/directory 目录下去，这个命令在/source/directory 和 /dest/directory 不处在同一个文件系统下时将显示出特别的优势。  
  
下面还几种不常见的用法：  
           n<&- 表示将 n 号输入关闭  
           <&- 表示关闭标准输入（键盘）  
           n>&- 表示将 n 号输出关闭  
           >&- 表示将标准输出关闭

**wc [option] filename   统计文件字数**

功能：统计文件中的文件行数、字数和字符数。 

 选项：

    -l    lines   统计文件的行数   
    -w  words  统计文件的单词数 (英文单字)  
    -c   bytes   统计文件的字节数

    -m  chars  统计文件字符数

注释：若缺省文件名则指标准输入

eg:#wc

输入字符，按ctrl+d结束，输出统计结果

eg:# wc /etc/services

10774  58108 641020 /etc/services

依次为：行数   单词数   字符数   文件名

eg:#wc << hello

输入字符，直到输入hello才结束并统计

eg:cat test.txt |wc -l        用wc命令做到只打印统计数字不打印文件名

注意事项：

a：一个汉字占三个字节，一个回车符等不可见字符也占一个字节  
b：一行的末尾如果没有回车符，则不算是一行，也就是说，如果一个文件的最后一行末尾没有换行符，wc命令统计的行数会比实际行数少一，所谓实际行数是你所看到的行数，实际上，没有回车符，确实不能算作一行  
c：所谓一个字就是连续的字符，即不被空字符分割的连续的字母或者汉字算作是一个字

**统计文件中，字符串出现的行数**

eg:统计aaa.txt文件中，unix出现的行数

# grep -o "unix" aaa.txt | wc -l

# cat aaa.txt | grep unix|wc -l

# grep -c "unix" aaa.txt

# 管道符

**格式： 命令1 | 命令2**将命令1的正确输出作为命令2的操作对象。

eg: ls -l /etc|more          more下查看/etc 下的文件

eg:ls -l /etc|grep init      查看etc目录下与init相关的文件

eg:wc -l /etc/services     查看 /etc/services文件有多少行

# 命令连接符

     ;      命令1;命令2    用分号;间隔的各命令按顺序依次执行，命令之间没有逻辑联系      eg:pwd;ls;date

  &&    命令1&&命令2    逻辑与关系，命令1执行成功后，命令2才被执行，命令1失败，命令2不执行

    ||      命令1||命令2    逻辑或关系，命令1执行失败，命令2才被执行；命令1执行正确，命令2不执行

dd if=输入文件 of=输出文件  bs=字节数   count=个数

选项：

if=输出文件     指定源文件或源设备

of=输出文件    指定目标文件或目标设备

bs=字节数       指定一次输入/输出多少字节，即把这些字节看做一个数据块

count=个数     指定输入/输出多少个数据块

eg:date;dd if=/dev/zero of=/root/testfile bs=1k count=100000 ;date

创建文件，并查看复制使用的时间

eg:mkdir test && echo "aaa" >test/a.txt && more a,txt

 创建test目录 ，并且向test目录下文件a.txt写入aaa,然后查看a.txt的文件内容

eg:#命令 && echo yes || echo no        判断一条命令是否正确执行了

# 命令替换符

将一个命令的输出作为另外一个命令的参数。

格式为： 命令1 `命令2`(数字键1左边的符号)

eg：ls -l `which touch`

把which touch  的输出作为 ls的参数，执行结果为：-rwxr-xr-x. 1 root root 52656 6月 22 2012 /bin/touch

# 通配符

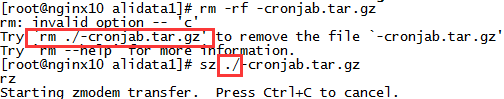
|  |  |
| --- | --- |
| 通配符 | 作用 |
| ？ | 匹配一个任意字符 |
| \* | 匹配0个或任意多个任意字符 |
| [abc] | 匹配括号中任意一个字符 |
| [a-z] | 匹配范围内的任意一个字符 |
| [^] | 逻辑非，匹配不是中括号内的一个字符 |

**其他特殊符号**

|  |  |
| --- | --- |
| 符号 | 作用 |
| '' | 单引号，单引号中所有的特殊符号，如$等都没有特殊含义 |
| "" | 双引号，双引号中特殊符号没有特殊含义，但是"$","`"和"\"是例外，  拥有"调用变量的值"，"引用命令"和"转义符"的特殊含义。 |
| `` | 反引号，反引号括起来的内容是系统命令，在bash中会先执行。 和$()的作用一样，不过推荐使用$()，因为反引号容易看错 |
| $() | 和反引号作用一样，用来引用系统命令 |
| # | 在shell脚本中，#开头的行代表注释 |
| $ | 用于调用变量的值，如需要调用变量name的值，  使用$name就能得到变量的值 |
| \ | 转义符，跟在\之后的特殊符号将失去特殊含义，变为普通字符。  如\$将输出$符号，而不当做是变量引用 |

如果打压缩文件的时候，不小心把文件名弄成以-开头的了，再使用sz 传输文件时，会报错，因为linux把-视作命令的开头，怎么解决呢？

使用sz ./-file-name



**快捷键**

<Tab> 命令补全

CTRL +  I 等同于按制表符<TAB>键

Ctrl + a 切换到命令行开头    同Shift +Home

Ctrl + x  按两次，切换到行首，能从行首到行尾来回切换

Ctrl + e 切换到命令行末尾    同Shift +End

Ctrl + l 清除屏幕内容，同clear

Ctrl + u 清除或者剪切光标之前的所有内容（直到行首）

Ctrl + w 清除或剪切光标之前的字符（遇到空格停止，多次操作==ctrl+u  ）

Ctrl + k 清除剪切光标之后的所有内容,（直到行尾）

Ctrl + y 粘贴刚才所删除的字符,适用Ctrl + w，Ctrl + u 和Ctrl + k

Ctrl + r 在历史命令中查找 （这个非常好用，输入关键字就调出以前的命令了）

Ctrl + c 终止命令

Ctrl + d 删除提示符当前的字符，如果命令行为空，则为退出shell，等同logout

Ctrl + h 将光标前的一个字符删除

Ctrl + b 前移一个字符

Ctrl + f 后移一个字符

Ctrl + z 暂停，转入后台运行，不要经常使用

Ctrl + t 将光标处的字符和光标前一个字符替换位置

Ctrl + s 暂停屏幕输出

Ctrl + q 恢复屏幕输出

!! 重复执行最后一条命令     显示执行命令和执行结果

↑(Ctrl+p) 显示上一条命令  
↓(Ctrl+n) 显示下一条命令

ctrl+p：回显使用的历史命令