<"文件"  
<<"here文档"

cat << BROWSERS

mozilla

links

lynx

konqueror

opera

netscape

BROWSERS

<<<"here string"  
< <(命令替换) **< <(ls /tmp/)**

如果重定向操作符是<<-，输入的行（包含分隔符）将被清除所有开头的制表符.简单点说,就是删除前面的tab

read -ra array <<< "$element"

read -t 5 -p "please enter your name:" name

**=========================================================================**

while read line ;do echo ${line};done < <(ls /tmp/)

ls /tmp |while read line;do echo $line;done

for i in `ls /tmp` ;do echo $i ;done

grep -o local < /etc/hosts  
echo  $(< /etc/hosts)

grep -o local < /etc/hosts  
echo  $(< /etc/hosts)

< /etc/hosts grep -o local

总结:   The command substitution $(cat file) can be replaced by the equivalent but faster $(< file). $(<file) 比$(cat file)要快

**Unix/Linux Shell编程实战：使用嵌入文档Here Documents**

一、Here Documents（嵌入文档）  
Here Documents作为重定向的一种方式，指示shell从源文件的当前位置开始读取输出，直到遇到只包含一个单词的文本行时结束。在该过程中读到的所有文本行都将作为某一个命令的标准输入而使用。  
       here-documents的使用形式:

command <<[-] limit\_string  
         msg\_body  
limit\_string

**如果用双引号或单引号将”limit\_string”引用起来或用转义符\将其转义，则here-document中的文本将不被扩展，即参数替换被禁用**。否则，here-document中的所有文本都将进行常规的参数扩展、命令替换、表达式计算。在后一种情况下，字符序列\<newline>将被忽略, 并且必须对元字符\, $, 和`使用\进行转义。

[root@centos test]# cat << "a"

> kj\fdf

> dkfj

> a

kj\fdf

dkfj

[root@centos test]# cat << a

> dkf\ =🡺\当成是续行

> fjkdf

> abc

> a

dkffjkdf

abc

**如果用<<而不是<<-，则后面的limit\_string必须位于行首，否则，若here\_document用在函数内部，则会报语法错误；用在函数外面，其后面的所有内容均会被当做here\_document的内容。**

**如果重定向操作符是<<-, 则msg\_body和limit\_string行中的所有前缀TAB字符都将被忽略（但空格不会被忽略）。这样源代码中的here-documents就可以按照优雅的嵌入方式进行对齐。**

二、技术的应用  
         下面我们考察一下在Shell脚本中使用Here Documents的必要性所在。Shell脚本对于文本文件的处理是非常方便的，因此实现本文所要完成的工作还是比较简单的，只需对所有文件进行一次循环过程。但是问题在于，从①-②，②-③的每一部，实际上都是使用cut、echo、cat等进行流处理，每个处理过程的输出作为下一步骤的输出。这里面自然的要使用重定向的方法。在以前对BASH用的不熟的时候，往往笨拙的使用临时文件存储这些中间结果。比如： 00000004.REG -> 00000004.cut -> 00000004.line -> all.txt。这样处理的过程中，代码显的非常杂乱，而且有时候不得不写出多个脚本来处理重定向，这种情况使得问题更为加重。  
        因此，这次下决心要将代码写的精简一些。在BASH内部可以使用Here Documents进行重定向，使流的输入输出都是在同一脚本内进行的，不用拆开成多个脚本，也省略了大量的中间文件，使代码非常清晰、紧凑。

Here Documents的一般形式如下：

0001 commands <<ID   
0002     here documents   
.......     ......   
000n ID<\n>   
        在上面这个语法描述中，0001行的commands是任何SHELL命令。ID是一个字符串标识，标志着嵌入文档的开始（习惯上，常用的标识是EOF，等等）。从0002行可以写任何文本，这里写入的内容都将作为标准输入传送给commands。在最后一行写标识符ID，代码嵌入文本结束。

在使用Here Documents编写脚本的时候注意以下两点：

在000n行上，必须只有一个ID，也就是说，ID前面不能有空格，ID后面必须是回车符。否则SHELL处理时会出错。这一点在实际时要尤为注意。

当使用 <<- 时， 输入行和000n行上的ID前的任何TAB字符将会被忽略；

嵌入文档内可以写普通文本（比如说，写程序的使用Usage），也可以是标准的SHELL语句。如果是SHELL语句，要使用反引号（`）括起来，这时候语句的标准输出也将作为嵌入文档送给commands。这个特性非常有用。

**1.**cat用法  
[exam@Exam-Server user]$ cat aa.sh  
     1 #!/bin/bash  
     2   
     3 cat <<EOF # cat > file << EOF，则输出到新文件中，而不是stdout  
     4    Now:  
     5    `date`  
     6 EOF  
[exam@Exam-Server user]$ sh aa.sh  
Now:

       可以将here\_document的输出保存到文件或变量中

 variable=$(cat << EOF

This varibale

runs over multiple lines.

EOF

)

       echo $variable

**This varible runs over mutiple lines.**

      可以用匿名的命令来接收here\_document的输出，该技术可以用来注释掉代码块。

: << EOF

${HOSTNAME?}${USER?}${MAIL?} # 如果其中一个变量没被设置, 那么就打印错误信息.

EOF

三 7月 26 19:23:24 CST 2006

        上面是一个最简单的例子，但是已经基本概括了Here Documents的使用方法。在aa.sh中，EOF作为嵌入文档的标识ID。4、5行中的Here Documents是一行标准文本“Now”和一个命令date的输出。两部分输出送给了cat命令去执行。

**2．shell编辑文件**

              shell编辑文件最常用的方法就是echo 字符串 >> 文件。但是要删除一行怎么办？Here Document就搞定了。

    经Jeremiah测试，在Here Document中使用vi是不行的。替代方法是使用ed命令。在命令行执行以下：

$ touch base.txt   
$ ed base.txt   
a   
this is line1.   
this is line2.   
this is line3.   
this is line4.   
.   
wq

    先新建一个文件base.txt，然后ed这个文件，输入a表示最后追加，输入了四行文字。.表示结束输入。wq表示保存退出。

    那我们再用ed命令，在shell脚本里面对这个文件再次进行操作。如下。

#!/bin/sh   
ed base.txt << EOF   
3 # 到第3行  
d # 删除该行  
i # 本行增加  
this is line 3 new.  # 新行内容  
.  # 结束  
w   
q # wq表示保存  
EOF

执行的结果如下所示。

$ sh ed\_file.sh && cat base.txt      
60   
this is line3.   
65   
this is line1.   
this is line2.   
this is line 3 new.   
this is line4.

    关于ed的操作和参数，可以查看linux帮助或去搜索相关的资料。

**3. shell控制数据库**

    假设执行下面的操作访问[**数据库**](http://lib.csdn.net/base/14)。

$ [**MySQL**](http://lib.csdn.net/base/14) -u root   
Welcome to the MySQL monitor.    Commands end with ; or \g.   
Your MySQL connection id is 1257   
Server version: 5.1.35-community MySQL Community Server (GPL)   
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.   
  
mysql> use mysql   
Reading table information for completion of table and column names   
You can turn off this feature to get a quicker startup with -A   
  
mysql> select \* from user;   
mysql> exit   
Bye

如果我们要用shell脚本访问，则可以编写如下的脚本。

#!/bin/sh   
  
mysql -u root << EOF   
use mysql;  
select \* from user;   
exit   
EOF

执行如下。

sh mysql\_access.sh

可以看到结果相同。