for循环

```
for var in item1 item2 ... itemN
do
    command1
    command2
    ...
    commandN
done
for i in 1 2 3 hello abc
do
    echo $i
done
```

输出当前路径下所有文件名

```
for i in `ls`
do
    echo $i
done
```

输出序列1-10

```
for i in $(seq 1 10)
do
    echo $i
done
```

```
for i in {1..10}
do
    echo $i
done
```

输出字母a-z

```
for i in {a..z}
do
    echo $i
done
```

for ((...;...;...)) do...done

```
for ((i=1;i<=10;i+=2))
do
    echo $i
done
```

while循环

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。

```
while condition
do
    command
done
```

偶数和

```
sum=0
i=0

while [ $i -le 50 ]
do
    let sum+=i
    let i+=2
done

echo "0-50的偶数和为：$sum"
```

循环输入名字，文件结束符为Ctrl+d，输入文件结束符后read指令返回false。

```
while read -p "Please input your name: " name
do
    echo $name
done
```

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：

```
a=0

until [ ! $a -lt 10 ]
do
    echo $a
    a=`expr $a + 1`
done
```

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。

```
while true
do
    read -p "请输入1-5虹之间的数字：" n

    case $n in
        1|2|3|4|5) echo "你输入的数字为$n"
        ;;
        *) echo "请输入正确的数字范围"
           break
        ;;
    esac
done
```

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：

```shell
while true
do
    read -p "请输入1-5之间的数字：" n

    case $n in
        1|2|3|4|5) echo "你输入的数字为$n"
        ;;
        *) echo "请输入正确的数字范围"
            continue
            echo "游戏结束，这句话永远不会被执行"
        ;;
    esac
done
```