$0 这个程式的执行名字

$n 这个程式的第n个参数值，n=1..9

$\* 这个程式的所有参数,此选项参数可超过9个。

$# 这个程式的参数个数

$$ 这个程式的PID(脚本运行的当前进程ID号)

$! 执行上一个背景指令的PID(后台运行的最后一个进程的进程ID号)

$? 执行上一个指令的返回值 (显示最后命令的退出状态。0表示没有错误，其他任何值表明有错误)

$- 显示shell使用的当前选项，与set命令功能相同

$@ 跟$\*类似，但是可以当作数组用



获取变量值的一部分



使用basename $tt也可达到

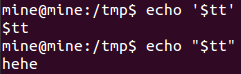


 `command`

等同于



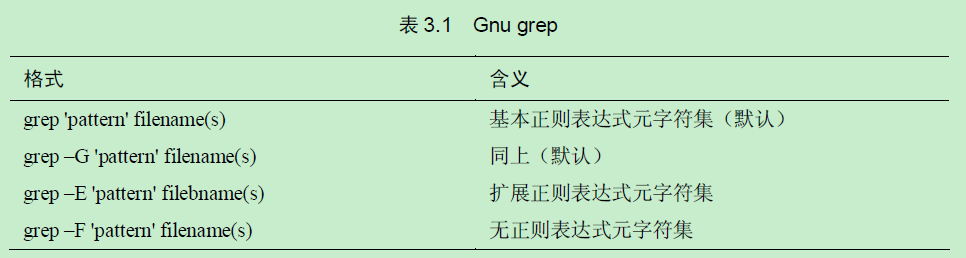
如果双引号引用，除$,倒引号``，反斜杠\之外均作为普通字符，而且一旦赋给变量多个单词，Shell 翻译时就会自动取消空格；如果单引号引用，全部作为普通字符出现。不加引号默认替换成一行



倒引号`shell语句` 解释执行，语句结果替换对应位置



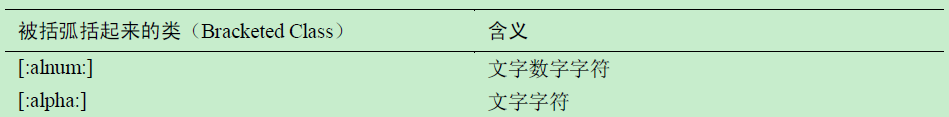
**Grep**：

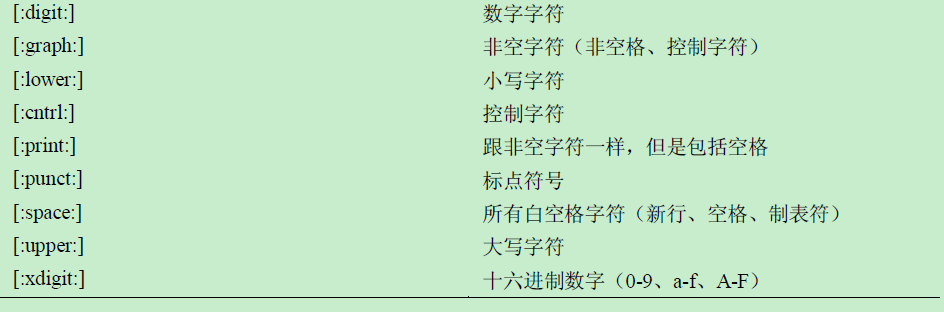


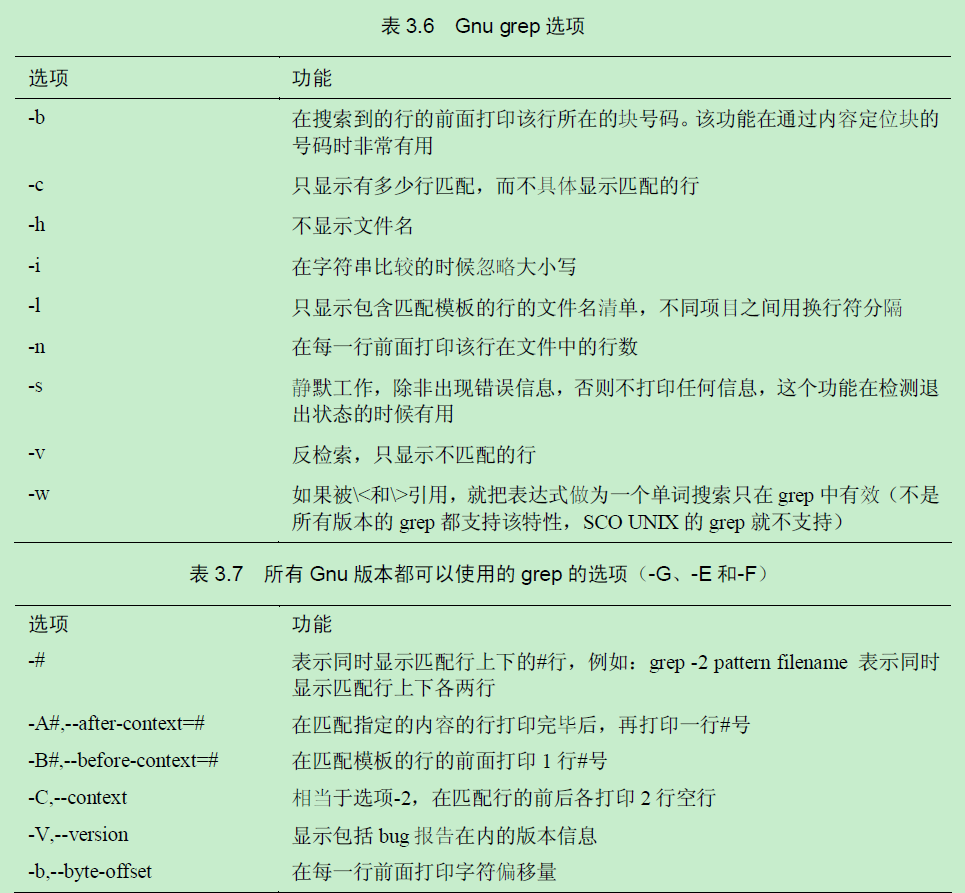
Pattern可为正则表达式

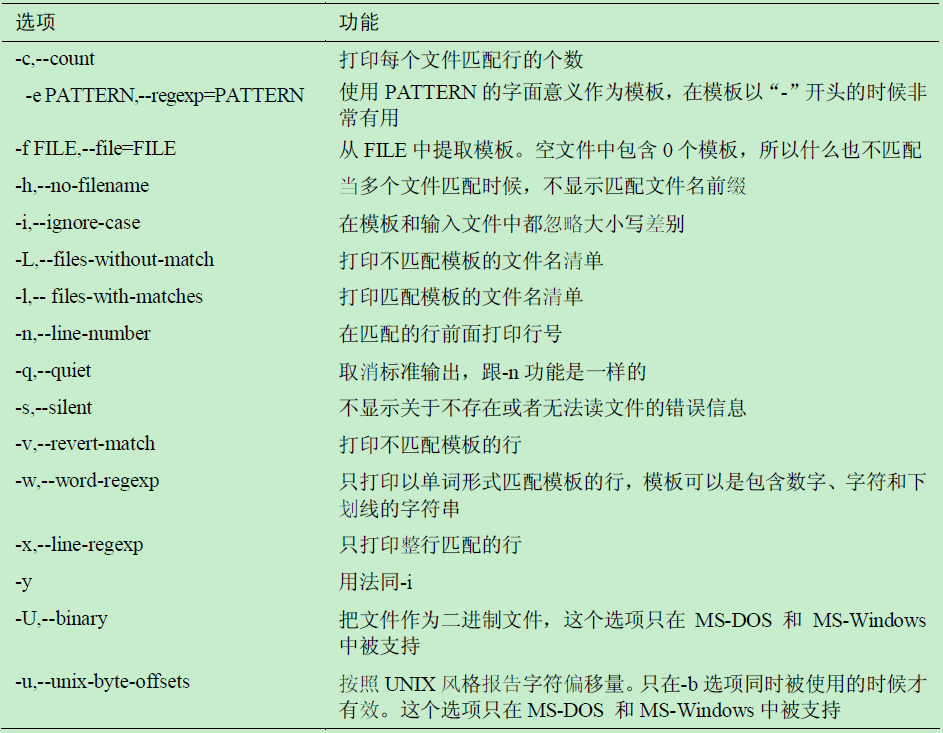
如果模板找到，grep 返回状态值0，表示成功；如果无法找到模

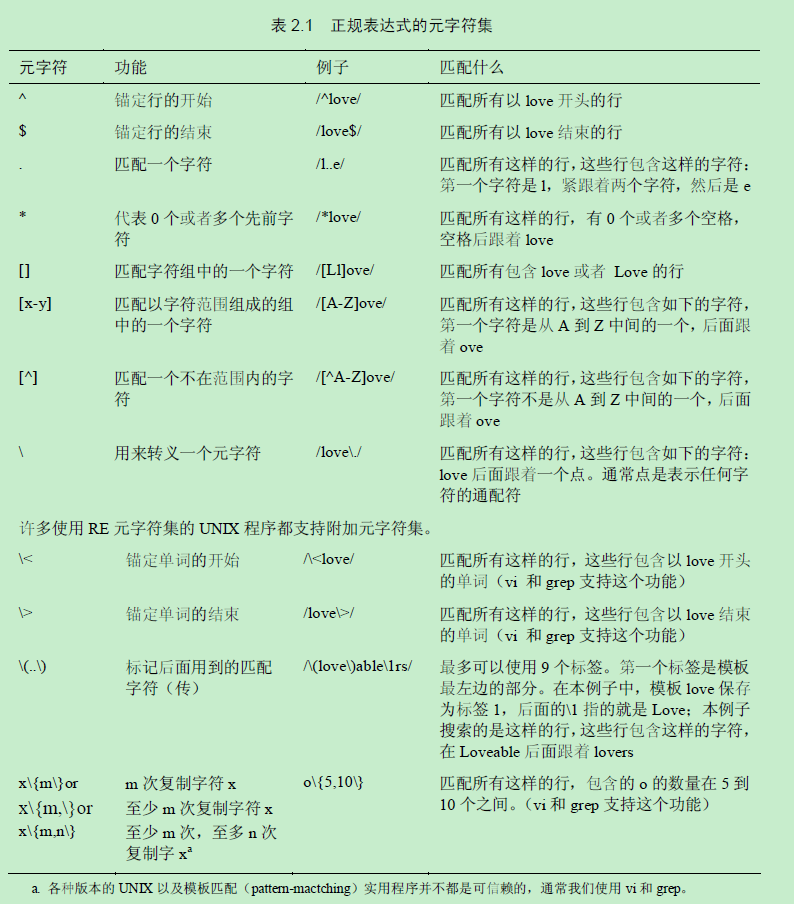
板，则返回退出状态值1；如果文件无法找到，就返回退出状态值2











X？ 0或1个X . 任意字符

X\* 0或多个X [abc] 其中之一

X+ 1或多个X [^abc] 除此之外

X{n} n个x \s 空格字符 \S 非空格字符

X{n，}至少n个x \d 数字 \D 非数字

X{,n}至多n个x \w 单词字符 \W 非单词字符

^a 以a开头 \b单词分界线 \B单词内的空字符串

a$ 以a结尾

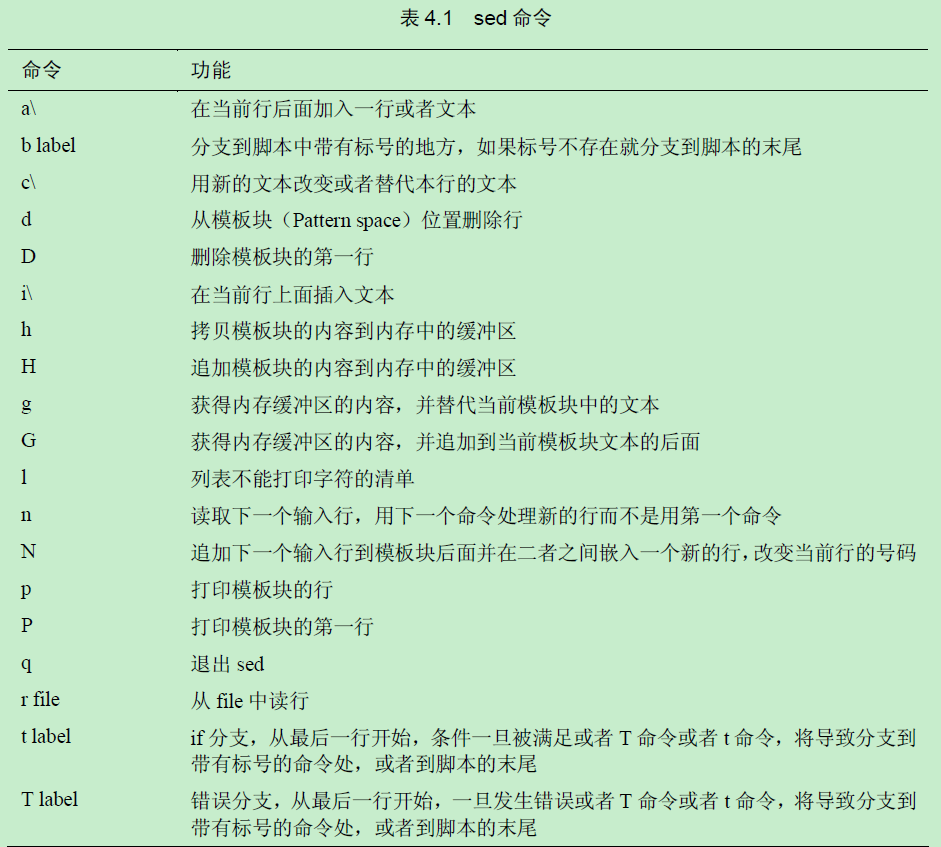
Vim中：

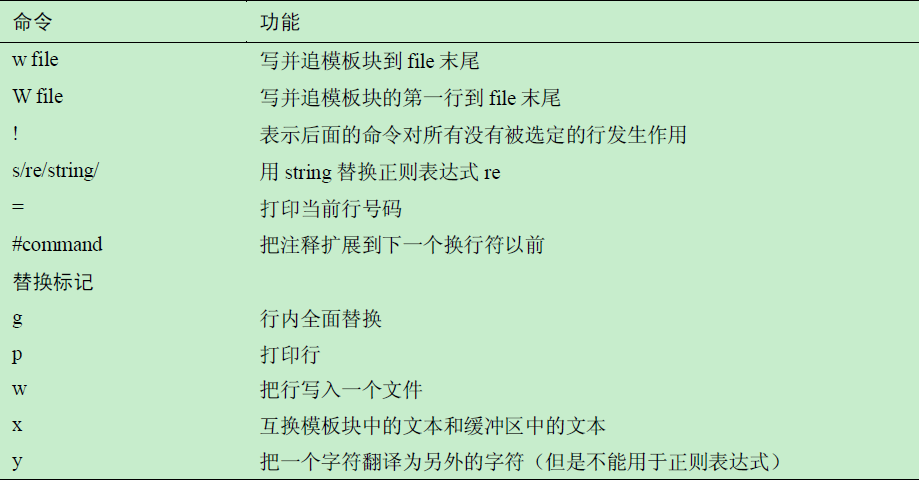
替换，注意转义

**Sed**

**sed [OPTION]... {script-only-if-no-other-script} [input-file]...**

**Script：'/Tom/d' 删除包含模板Tom的行**





**替换s命令：s/old/new/g**

**读入文件r命令：/pattern/r newfile 将newfile的内容添加到匹配行下面**

**写入文件w命令：/pattern/w newfile 将匹配到的行写入newfile**

**追加a命令：/pattern/a\text text加到匹配行后**

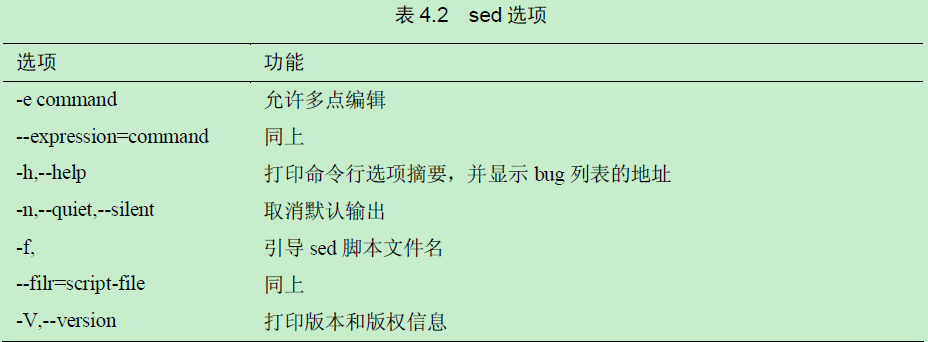
**插入i命令：/pattern/i\text text加到匹配行前**

**逗号表示范围搜索：**

**sed ‘pattern1，pattern2Command’file**

**搜索匹配pattern1的行与匹配pattern2的行之间的所有行。**

**使用脚本编辑，-f sedfile 执行**



Sed新增正则表示式元字符



**AWK**

awk ‘pattern{action}’filename

awk ‘pattern’filename

awk ‘{action}’filename

pattern可为关系表达式 如 $1 == 555

eq： **awk '!($2 < 100 && $3 < 20)' filename**

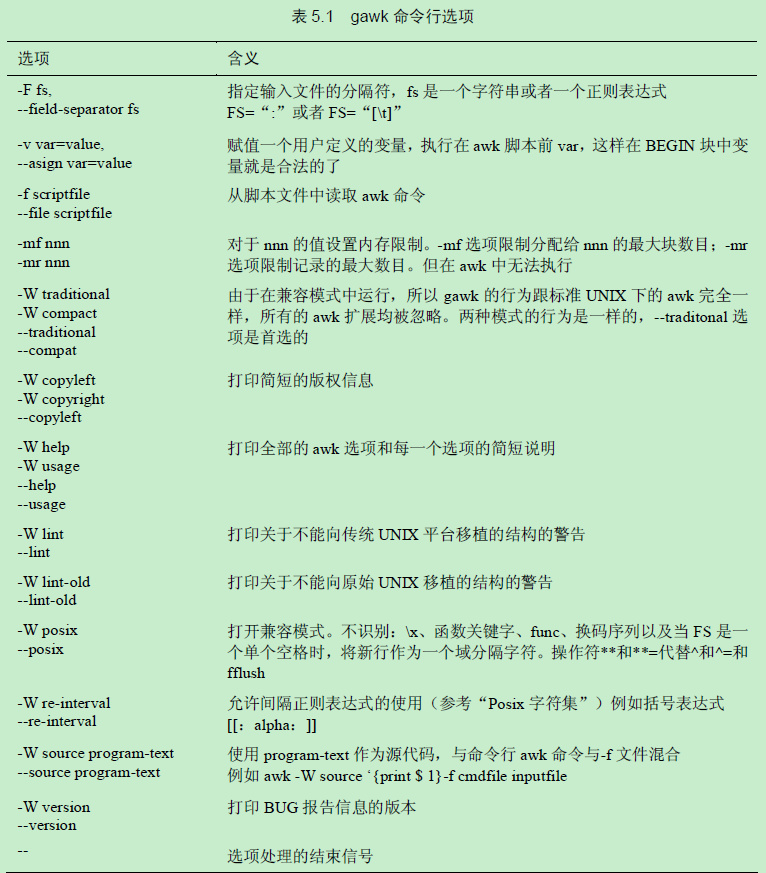


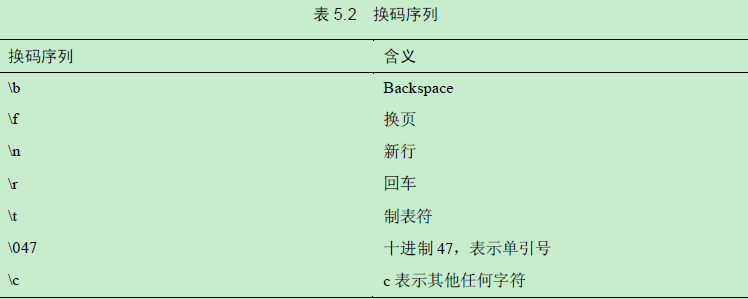
\B 匹配单词内的空字符串

~匹配操作符：

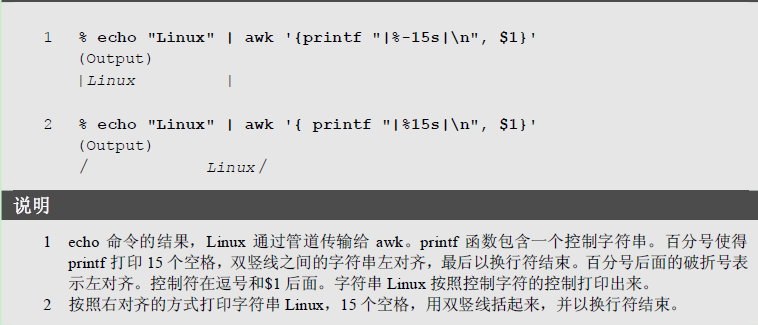
‘$1 ~ /bili/’ 第一个域中匹配bili的行

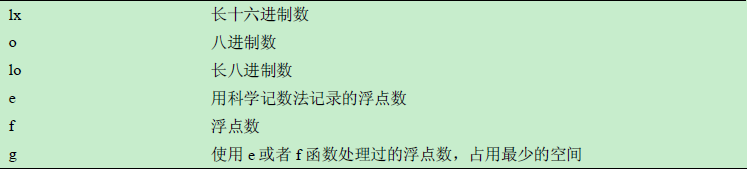
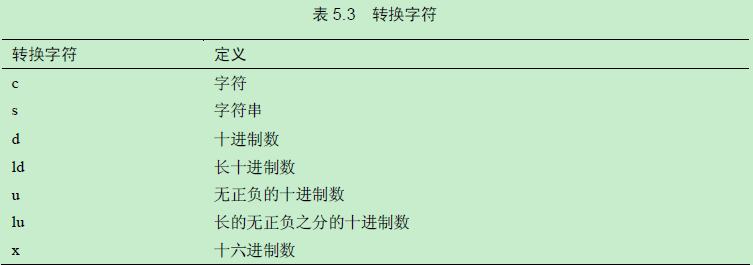
‘$1 !~ /bili/’第一个域中不匹配bili的行

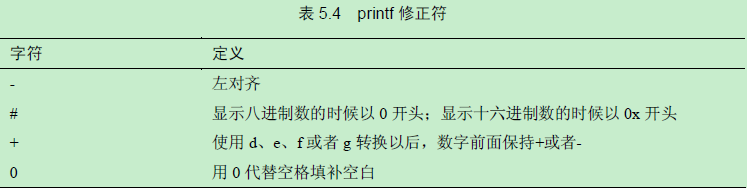




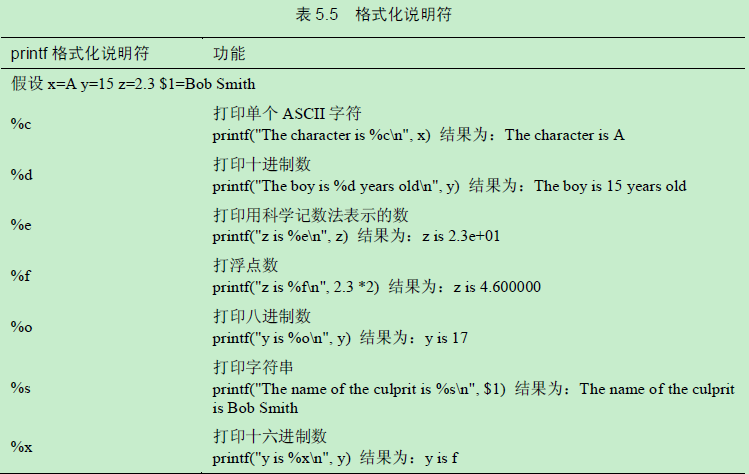
**Printf命令**

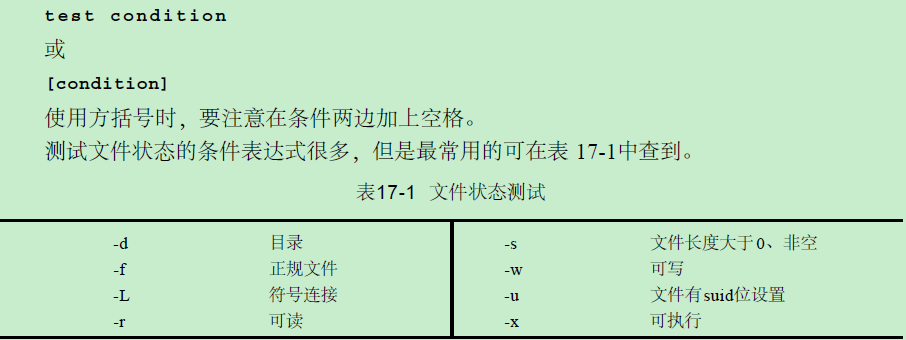






示例如下：

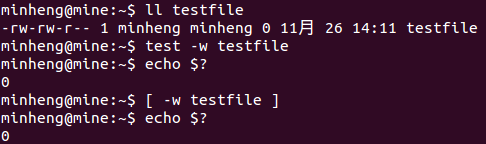
测试文件



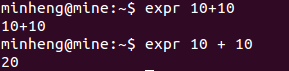
-z 字符串是否为空 –n非空字符串

-a 逻辑与 –o逻辑或 ！逻辑否

Eq：



Expr num1 operator num2 （使用\*时需转义 \\*）



隐藏输入 stty –echo