批处理特殊符号 通配符与转义符

常用特殊符号

1、@命令行回显屏蔽符
2、%批处理变量引导符
3、> 重定向符
4、>>重定向符
5、<、>、<& 重定向符
6、|命令管道符
7、^转义字符
8、组合命令
9、& 组合命令
10、||组合命令
11、\"\"字符串界定符
12、, 逗号
13、; 分号
14、() 括号
15、! 感叹号

通配符（?和*）

常用来代替未具体指明的文件和数据。

？ 代表单个字符。

* 代表全部字符。

转义符

用在特殊符号之前,取消特殊字符的作用.
比如:
echo 非常^&批处理
连字符&在此命令中只当作一个字符显示到屏幕中
如果不加^ 那么"批处理"将被当作命令执行

| (管道)传递符
当然是传递作用,将前面一条命令的执行结果传递给后面一条命令继续执行.
比如:
echo 0123456 | find "123"
将0123456 传递给find 查找 123

|| 连接符
当前面的命令没有成功执行 则执行后面的命令 可以作判断选择用.
比如:
echo 0123456 | find "789" || echo 字符中没有789
此命令的意思是在0123456中查找字符789,如果没有找到则继续执行后面的命令,
即在屏幕中显示"字符中没有789",如果找到了,也就是如果前面是0123456789的话则停止继续执行||
后面的命令,而是在屏幕中直接显示"0123456789".

&& 连接符
当前面的命令成功执行 再执行后面的命令
比如:
echo 0123456 | find "123" && echo 字符中含有123
这个命令跟||刚好相反.

& 连接符
无论前面的命令是否成功执行 都执行后面的
比如:

echo 0123456 | find "789" & echo 字符中含有123
虽然没有成功到789 但还是会执行echo 命令

> 定向符
将输出的内容重定向到指定(文本中)
比如
echo 123456789>1.txt
输出字符串到1.txt中
如果是 >nul 则是输出到空设备中 起屏蔽屏幕输出的作用
比如
pause>nul 将命令的提示屏蔽掉

>> 定向符(追加)
将输出内容写入指定(文本中)
注意，">"将覆盖文本中原有内容
">>" 则是在文本的最后添加内容，
比如
echo 123>1.txt
echo 456>>1.txt
文本中有两行 分别是 123和456
如果echo 456>1.txt
那么 原有文本中的123将会替换成456

< 输入定向符
从指定的(文本)输入内容
一般用于set /p
比如
echo 456>1.txt
set /p wind=<1.txt
这样 1.txt第一行内容就被赋值给wind了
本来set /p 这个命令是用来设置给用户输入字符的,但是=后用<1.txt,
意思是从1.txt文件中读取信息给SET /P ,而不需要用户输入.

http://wuxiong8665.blog.163.com/blog/static/93512200911623811370/

较详细的一篇

这是一篇针对批处理中常用符号的详细解释，每个符号都有解释及相应的举例，希望通过比较系统的讲述，能让新手尽快入门。

在这篇帖子中，我对常用符号的讲解做如下限定：

1、收集批处理中经常用到的符号；

2、每个常用符号，只讲述最常用的功能；深入的用法留待将来介绍；

这样限定的原因，一是让新手系统地接触最常用符号的常用功能，不至于一开始就陷入技术细节中难以自拔；二是有些符号的用法非常罕见，没有特定的需求可以忽略掉，比如句柄复制符号；三是有些高深的内容本人也没有完全消化，只解说一鳞半爪难免会误人子弟，比如 set /a 中的^、!等符号；

如有遗漏或谬误，请大家及时跟帖，帮忙修正。

————————————————————————————————————

1、@

一般在它之后紧跟一条命令或一条语句，则此命令或语句本身在执行的时候不会显示在屏幕上。请把下面的代码保存为test.cmd文件，然后运行，比较一下两条echo语句在屏幕上的输出差异：

```
1. echo a
2. @pause
3. @echo b
4. @pause
```

复制代码 执行结果如下：

```
C:\Documents and Settings\JM\桌面>echo a
a
请按任意键继续...
b
请按任意键继续...
```

2、%、%%

百分号用在不同的场合，有不同的含义：

① 当百分号成对出现，并且其间包含非特殊字符时，一般做变量引用处理，比如：%var%、%str%。把以下代码保存为批处理文件，运行后观察屏幕显示结果：

```
1. @echo off
2. set str=abc
3. echo 变量 str 的值是： %str%
4. pause
```

复制代码 在屏幕上将显示这样的结果：

```
变量 str 的值是： abc
按任意键继续...
```

另外，百分号作为变量引用还有一种特殊形式，那就是对形式参数的引用，此时，单个百分号后面紧跟0～9这10个数字，如%0、%1，请看演示代码：

```
1. @echo off
2. if defined str goto next
3. set str=
4. set /p str=请把文件拉到本窗口后回车：
5. call "%~0" %str%
6. pause
7. exit
8.
9. :next
10. cls
11. echo 本批处理文件完整路径为："%~0"
12. echo 拖到本窗口的文件完整路径为："%~1"
13. goto :eof
```

复制代码 ② 出现在 set /a 语句中时，表示两数相除取余数，也就是所谓的模运算，它在命令行窗口和批处理文件中的写法略有差异：在命令行窗口中，只需要单个的%，在批处理文件中，需要连续两个百分号，写成%%。

例如：在命令行窗口中，运行 set /a num=4%2 ，则结果将显示0，因为4除以2的余数为0；如果保存为批处理文件，则此语句将略有改变：

```
1. @echo off
2. set /a num=4%%2
3. echo 4除以2的余数为 %num%
4. pause
```

复制代码 ③ 转义符号：如果要显示%本身时，需要在前面用%来转义。例如：

```
1. @echo off
2. echo 一个百分号：%%
3. echo 两个百分号：%%%%
4. echo 三个百分号：%%%%%%
5. pause
```

复制代码 3、:、::

① 以:打头的单个的:表示该行是一个标签，它之后的内容是一个标签段，如:test，则表示:test之下的内容是标签段，而test是这个标签段的名，可以用 goto test 、goto :test 跳转到该标签段或用 call :test 调用该子过程；而连续两个冒号打头表示该行内容为注释内容，实际上，:: 是个无效的标签名，:加上空格同样可以起到注释的作用，此时，::的功能和注释命令rem相同；但是，rem 注释语句中的某些命令符号如重定向符号和管道符号还是会执行，而如果用::来注释的时候，与::同处一行的所有命令或符号直接被命令解释器忽略掉，无形中提高了注释的兼容性和整个程序的执行效率，并且在众多的命令语句中更显得醒目，所以，注释语句推荐使用::的格式。

② 在 set 语句中：和~同时使用时，：起到截取字符串的功能。假设 set str=abcde，那么，set var=%str:~0,1% 表示截取字符串abcde的第一个字符；和=同时使用时，起到替换字符串的功能。假设：set str=abc:de，那么，set var=%str:a=1% 则表示把字符串abc:de中的a替换为1，set var=%str::=2% 则表示把字符串abc:de中的:替换为2；

4、~

① 用在 set 语句中，和:同时使用时，起到截取字符串的功能，请参考上一条的解释；

② 用在 set /a 语句中时，它是一元运算符号，表示将操作数字按位取反，例如，set /a num=~1的执行结果是-2，set /a num=~0的结果是-1

③ 用在for语句中，表示增强for的功能，能够提取到更多的信息。例如：在批处理文件的for语句中：%%~i表示去掉第一对外侧引号，%%~zi表示获取文件的大小(以字节为单位)，%%~ni表示获取文件名，%%~xi表示获取扩展名(带点号)……它们可以组合使用，如%%~nxi表示获取文件名和后缀名。

5、>、>>

一般而言，>表示用新内容覆盖原文件内容，>>表示向原文件追加内容，此时，它们以重定向符号的身份出现；如果用在 set /a 语句中，则>表示分组，>>表示逻辑移位；

6、|

一般而言，它以管道符号的身份出现，表示把在它之前的命令或语句的执行结果作为在它之后的命令或语句的处理对象，简而言之，就是把它之前的输出作为它之后的输入，例如：echo abcd|findstr "b"，表示把echo abcd的执行结果，作为findstr "b" 的执行对象，也就是在字符串abcd中查找b字符；如果test.txt中有abcd字符串，则该语句与 findstr "b" test.txt 具有同样的效果；

7、^

一般而言，^以转义字符的身份出现。因为在cmd环境中，有些字符具备特殊功能，如>、>>表示重定向，|表示管道，&、&&、||表示语句连接……它们都有特定的功能，如果需要把它们作为字符输出的话，echo >、echo | ……之类的写法就会出错——cmd解释器会把它们作为具有特殊功能的字符对待，而不会作为普通字符处理，这个时候，就需要对这些特殊字符做转义处理：在每个特殊字符前加上转义字符^，因此，要输出这些特殊字符，就需要用 echo ^>、echo ^|、echo ^|^|、echo ^^……之类的格式来处理；

8、&

一般而言，&表示两条命令或语句同时执行的意思。如 echo a&echo b，将在屏幕上同时显示a和b字符。当几条语句含义近似或作用相同且没有先后的顺序之别时，启用&符号连接这些语句将会增加程序的可读性；

9、&&、||

这是一对含义截然相反的命令符，&&表示如果它之前的语句成功执行，将执行它之后的语句，而||则表示如果它之前的语句执行失败，将执行它之后的语句；在某些场合，它们能替代 if…… else…… 语句；例如：

```
1. @echo off
2. md test&&echo 成功创建文件夹test||echo 创建文件夹test失败
3. pause
```

复制代码 效果等同于如下代码：

```
1. @echo off
2. md test
3. if "%errorlevel%"=="0" (echo 成功创建文件夹test) else echo 创建文件夹test失败
4. pause
```

复制代码 10、()

小括号对经常出现在for语句和if语句中，还有一些特定场合；在for和if语句中属于语句格式的要求，例如：

① for %%i in (语句1) do (语句2)：在这条语句中，语句1必须用括号对包围，而语句2的括号对则可视情况予以抛弃或保留：如果语句2是单条语句或用&、&&、||等连接符号连接的多条语

句，括号对可以抛弃，如果语句2是有逻辑先后关系的多条语句集合，则必须保留括号对，并且，多条语句必须断行书写；例如：

```
1. @echo off
2. for %%i in (a b c) do echo %%i&echo --------
3. pause
```

复制代码 也可以改写为：

```
1. @echo off
2. for %%i in (a b c) do (
3.     echo %%i
4.     &echo --------
5. )
6. pause
```

复制代码 ② if 条件 (语句1) else (语句2)：如果没有else部分，则语句1的括号对可有可无；如果有else部分，则语句1中的括号对必须保留，此时，语句2中的括号对保留与否，和上一点类似。例如：

```
1. @echo off
2. if exist test.txt echo 当前目录下有test.txt
3. pause
```

复制代码

```
1. @echo off
2. if exist test.txt (echo 当前目录下有test.txt) else echo 当前目录下没有test.txt
3. pause
```

复制代码

```
1. @echo off
2. if exist test.txt (echo 当前目录下有test.txt) else (
3.     echo 当前目录下没有test.txt
4.     pause
5.     cls
6.     echo 即将创建test.txt文件
7.     cd.>test.txt&&echo 成功创建test.txt
8. )
9. pause
```

复制代码 ③ 特定场合下使用括号对，不但可以使代码逻辑清晰，增强可读性，还可能会减少代码量。比如用echo语句构造多行文本内容的时候：

```
1. @echo off
2. (
3. echo 第一行
4. echo 第二行
5. echo 第三行
6. )>test.txt
7. start test.txt
```

复制代码 如果不使用括号对的话，则需要使用如下代码：

```
1. @echo off
```

```
2. echo 第一行>test.txt
3. echo 第二行>>test.txt
4. echo 第三行>>test.txt
5. start test.txt
```

复制代码 11、+、-、*、/

在 set /a 语句中，这些符号的含义分别为：加、减、乘、除。例如：set /a num=1+2-3*4/5。需要注意的是，这些运算符号遵循数学运算中的优先级顺序：先乘除后加减，有括号的先算括号，并且，直接忽略小数点，因此，刚才那个算式的结果是1而不是0或0.6。

另外，有可能会在代码中看到这样的写法：set /a num+=1、set /a num-=1、set /a num*=1 和 set /a num/=1，这些表示累加、累减、累乘、累除，步长都是1，展开后的完整写法为：set /a num=num+1、set /a num=num-1、set /a num=num*1 和 set /a num=num/1(set /a 语句中，变量引用可以忽略百分号对或感叹号对，set /a num=%num%+1 与 set /a num=num+1 等同)

12、equ、neq、lss、leq、gtr、geq

这几个命令符是if语句中常用到的数值比较符号，取自英文的关键字母，具体的含义为：

命令符号	含义	英文解释
EQU	等于	equal
NEQ	不等于	not equal
LSS	少于	less than
LEQ	少于或等于	less than or equal
GTR	大于	greater than
GEQ	大于或等于	greater than or equal

http://www.bathome.net/thread-1205-1-1.html