**bat/cmd 常见特殊字符**

| **字符** | **说明** | **举例** |
| --- | --- | --- |
| @ | 取消命令[回显](https://so.csdn.net/so/search?q=%E5%9B%9E%E6%98%BE&spm=1001.2101.3001.7020) | 其后**紧跟**一条命令/语句，该命令/语句在执行时，  不会在屏幕上显示   1. :: 关闭命令回显 2. @echo off 3. echo a 4. @echo a 5. echo pause 6. @echo pause |
| % | 1、引用变量  2、引用形式参数  3、取余符号  4、[转义字符](https://so.csdn.net/so/search?q=%E8%BD%AC%E4%B9%89%E5%AD%97%E7%AC%A6&spm=1001.2101.3001.7020)，转义% | **1、引用变量，此时是成对出现**   1. :: 关闭命令回显 2. @echo off 3. set a=1 4. echo %a%   **2、引用形式参数**   1. %0：脚本名 2. %1 ~ %9：获取脚本后的参数值 3. %10以后：%1连接数字值，即参数1的值连接0   **3、取余符号**  注意：在脚本里面，需要两个%%才能发挥取余的作用   1. @echo off 2. :: 此时，输出的是5参数2 3. set a=5%2 4. echo %a% 5. set /A b=5%%2 6. echo %b%   **4、转义%**   1. :: 关闭命令回显 2. @echo off 3. set a=%% 4. echo %a% |
| **::** | 注释 | 该符号后的内容，仅作注释使用 |
| : | 1、标签符  2、:~  3、:= | 1、标签符   1. :: 关闭命令回显 2. @echo off 3. :label 4. :: 下面两种写法，命令行窗口不会退出 5. :: echo a||pause||exit 6. :: echo a||pause&&exit 7. :: 会退出 8. echo a||pause&exit 9. goto label   2、截取字符串   1. :: 关闭命令回显 2. @echo off 3. set str=hello world 4. set cut=%str:~2,4% 5. echo %str%   3、替换字符串   1. :: 关闭命令回显 2. @echo off 3. set str=hello world 4. set replace=%str:he=HE% 5. echo %replace% |
| ~ | 1、:~，截取字符串    2、~：一元运算符    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 | 2、一元运算符，对数字按位取反   1. :: 关闭命令回显 2. @echo off 3. set /a num1=~1 4. set /a num2=~0 5. echo %num1% 6. echo %num2% |
| > | **1、重定向符，覆盖原内容**    2、与set一起使用，表示分组 |  |
| >> | **1、重定向符，向原文件追加内容**    2、与set一起使用，表示逻辑移位 |  |
| **|** | 管道符。 | 把它之前的输出，作为它之后的输入   1. :: 关闭命令回显 2. @echo off 3. echo abcd|findstr "b" |
| **^** | 转义字符。 | 若要输出特殊字符，在前面加上^即可   1. :: 关闭命令回显 2. @echo off 3. echo ^> 4. :: echo >，该语句会报错 5. echo ^| 6. echo ^^ 7. echo ^|^| |
| **&** | 表示两条命令或语句同时执行。 | 当几条语句含义近似或作用相同，且没有先后顺序之分时，  使用&符号连接这些语句，将会增加程序的可读性。    将在屏幕上同时显示a和b   1. :: 关闭命令回显 2. @echo off 3. echo a&echo b |
| && | 逻辑与。 | 如果它之前的语句成功执行，将执行它之后的语句。  在某些场合，它们能替代 if…else… 语句 |
| || | 逻辑或。 | 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 |
| **()** | ① 用在for循环中。  ② 用在if语句中。  ③ 对语句进行分组。 | ① for %%i in (语句1) do (语句2)。  语句1必须使用括号对，语句2的括号对可视情况而定：  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   1. :: 关闭命令回显 2. @echo off 3. for %%i in (a b c) do echo %%i&echo -------- 4. pause 5. :: 也可以改写为： 6. for %%i in (a b c) do ( 7. echo %%i 8. echo -------- 9. ) 10. pause   ② 用在if语句中。  if 条件 (语句1) else (语句2)。  如果没有else，语句1的括号对可有可无；  如果有else部分，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。   1. :: 关闭命令回显 2. @echo off 3. if exist test.txt echo 当前目录下有test.txt 4. pause 5. if exist test.txt (echo 当前目录下有test.txt) else echo 当前目录下没有test.txt 6. pause 7. if exist test.txt (echo 当前目录下有est.txt) else ( 8. echo 当前目录下没有test.txt 9. pause 10. cls 11. echo 即将创建test.txt文件 12. cd.>test.txt&&echo 成功创建test.txt 13. ) 14. pause   ③ 对语句进行分组。  在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。   1. :: 关闭命令回显 2. @echo off 3. ( 4. echo 第一行 5. echo 第二行 6. echo 第三行 7. )>test.txt 8. start test.txt 9. :: 如果不使用括号对的话，则需要使用如下代码： 10. @echo off 11. echo 第一行>test.txt 12. echo 第二行>>test.txt 13. echo 第三行>>test.txt 14. start test.txt |
| +、+=    -、-=    \*、\*=    /、/= | 算术运算符 |  |
| equ  neq  lss  leq  gtr  geq | 比较运算符。 | 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 |