[cmd的变量总结](https://www.cnblogs.com/feiquan/p/10170203.html)

转自：https://blog.csdn.net/flyoutsan/article/details/52811095

cmd变量通过set设置变量，通过可以使用set /?查看有关变量的帮助文档。

接下来谈set的用法：

**1.set 变量名=值**

值可以包含空格、一直到命令结束，也可以是Ctrl+G这种代表警报声的字符(echo输出会发出警报声“滴～”），与echo类似

**2.set 变量名**

在系统中预定义了一批环境变量（所有的环境变量将附加在博客尾），如最常见的path变量，记录了系统应用程序的默认路径，如果仅仅使用set 变量名，那么将会打印所有以这个变量名开头的变量为的值。例如如果set p，将会打印ProgramFiles, ProgramFIles(X86), path等以p开头的变量的值。如果用户通过set设置了新的变量，如set pt=point那么新变量在这个cmd中将加入环境变量，因此set p也将会打印出pt的值，与环境变量不同的是，该变量只在当前cmd有效，同时对环境变量的更改也只是在当前cmd有效。

**3.set /P 变量=提示**

变量将通过用户输入接收值，提示作为提示信息输出，例如set /p v=输入v的值:，那么将会出现

输入v的值

（等待输入）

**4.set /A 变量=表达式**

表达式将被视为算术表达式，变量赋值为算术表达式的值，算术运算符参照帮助文档或者官方网站文档（可以发现，与C语言运算符基本相同）：

如果要使用其他变量的值，需要使用%变量名%（或者 !变量名!)来表示该变量。同时0X与0分别表示十六进制与八进制数字。

[复制代码](javascript:void(0);)

+ Add set /a "\_num=\_num+5"

+= Add variable set /a "\_num+=5"

- Subtract (or unary)set /a "\_num=\_num-5"

-= Subtract variable set /a "\_num-=5"

\* Multiply set /a "\_num=\_num\*5"

\*= Multiply variable set /a "\_num\*=5"

/ Divide set /a "\_num=\_num/5"

/= Divide variable set /a "\_num/=5"

% Modulus set /a "\_num=5%%2"

%%= Modulus set /a "\_num%%=5"

! Logical negation 0 (FALSE) ⇨ 1 (TRUE) and any non-zero value (TRUE) ⇨ 0 (FALSE)

~ One's complement (bitwise negation)

& AND set /a "\_num=5&3" 0101 AND 0011 = 0001 (decimal 1)

&= AND variable set /a "\_num&=3"

| OR set /a "\_num=5|3" 0101 OR 0011 = 0111 (decimal 7)

|= OR variable set /a "\_num|=3"

^ XOR set /a "\_num=5^3" 0101 XOR 0011 = 0110 (decimal 6)

^= XOR variable set /a "\_num=^3"

<< Left Shift. (sign bit ⇨ 0)

>> Right Shift. (Fills in the sign bit such that a negative number always remains negative.)

Neither ShiftRight nor ShiftLeft will detect overflow.

<<= Left Shift variable set /a "\_num<<=2"

>>= Right Shift variable set /a "\_num>>=2"

( ) Parenthesis group expressions set /a "\_num=(2+3)\*5"

, Commas separate expressions set /a "\_num=2,\_result=\_num\*5"

[复制代码](javascript:void(0);)

**5.%变量名:str1=str2%**

表示将变量的值中包含的str1使用str2替换后获得的变量，例如

set a="minecraft great!"

echo %a: = is so %

echo %a

此时将会输出:

minecraft is so great!

minecraft great!

这里将空格替换为( is so )，因此输出minecraft is so great!，而%a%依然如故，说明原变量并没有发生变化

**6.%变量名（已定义）:~start[, length]%**

表示从start出开始（包括start，第一个计数为0)，取length长的子串，如果length省略，则表示取到串尾，start可以为负数，最后一个字符为-1，从后往前依次为-2、-3、-4……

**7.setlocal [Enable|Disable]DelayedExpansion**

执行cmd或者bat文件是从头向后执行，变量有一个扩展规则，即在执行一个语句块时变量将会被变量的值代替，例如

set a=hello& echo a

这个语句通过&连接成为一个语句块，又如

(set a=steve

if defined a echo hello %a%)

是通过()组合的一个语句块

在执行语句块时，a将会被a的值所代替，而语句块中的set语句将会在该语句块执行完后才有效，例如

set a=hello

(set a=steve

if defined a echo hello %a%)

将会输出hello而不是steve，可以理解为当前语句块中所有的set都在下一句语句块执行时才有效，在当前语句块不做任何事情

这样将会对程序的逻辑性产生很大的影响，为了解决这个问题，我们可以使用setlocal EnableDelayExpansion。

EnableDelayedExpansion，直接翻译为“启用扩展延迟”，这个我们可以理解为原来的情况是在执行语句块时所有的变量都被扩展为值了，此时还未执行set命令，而启用后，扩展被延迟到语句块执行结束时，此时set命令已经执行，原来变量的值已经被set为了新的值，因此set已经生效。注意，启用后变量的调用将会变为!变量名!，如果使用%变量名%，调用的将会是未使用扩展延迟的状态的变量（即与原来没什么区别），与之相对的还有setlocal DisableDelayedExpansion，为相反的效果。

当执行这个命令后，在执行endlocal之前，setlocal将会一直生效，因此EndLocal就是将setlocal的效果“终结”的命令，使用后，setlocal（无论enable还是disable），效果都将end，一个setlocal对应一个endlocal。

预处理机制：批处理读取命令时是按行读取的（另外例如 for 命令等，其后用一对圆括号闭合的所有语句也当作一行），在处理之前要完成必要的预处理工作，这其中就包括对该行命令中的变量赋值。在不启用变量延迟，也不对变量动态捕获其扩展变化时，变量在预处理阶段不作改变  
  
setlocal enabledelayedexpansion ，就是启用变量延迟，我们可以形象的认为是启用了“对变量动态捕获扩展变化”。而 ! 括起来的变量，就是要动态捕获扩展的目标变量，如果不需要，可以继续使用 % 括变量。  
  
示例:

[复制代码](javascript:void(0);)

@echo off

SETLOCAL ENABLEDELAYEDEXPANSION

set var=0

for /l %%i in (1 1 10) do (

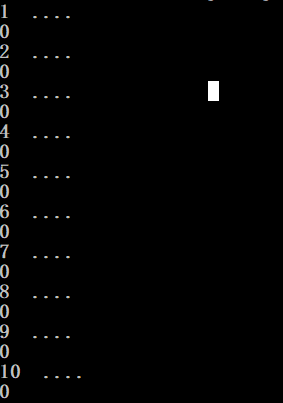
set var=%%i  
  
rem 启用延缓环境变量

echo !var! ....  
  
rem 未启用延缓环境变量

echo %var%

)

[复制代码](javascript:void(0);)



**8.setlocal [Enable|Disable]Extensions**

启用或者禁用Extension，部分cmd命令拥有启用Extension后的“额外”功能，如果没有启用，那么功能不可用，如if。

**9.setlocal**

setlocal将表示setlocal后直到endlocal，所有的变量都是局部变量，例如

@echo off  
set b=1  
setlocal  
set b=2  
setlocal   
set b=3  
set c=hi  
echo %b% %c%  
endlocal  
echo %b%   
if defined c echo %c%  
endlocal  
if defined c echo %b% %c%

输出为

3 hi

2

又如：

[复制代码](javascript:void(0);)

@echo off

set a=1

setlocal

set a=2

setlocal

set a=3

set c=4

echo %a% %c%

endlocal

echo %a%

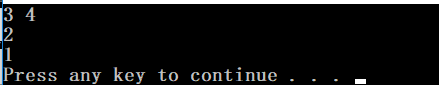
endlocal

echo %a%

pause

[复制代码](javascript:void(0);)

输出：



**10.for %i in (set) do**

在文件（bat或者cmd）中需要将%i使用%%i来代替，%%i不会出现变量扩展的问题，在循环中同步更新%%i的值（谢天谢地，一切正常）

**11.bat/cmd文件 + 参数1 + 参数2 + 参数3 + ……**

通过%1,%2,%3表示对参数1、2、3……的引用，%0表示bat文件的绝对路径+文件名，如C:\Users\Administrator\Desktop\test.bat

**12.系统预定义环境变量表直接输入 set 后可以获得**