

批處理中 setlocal enabledelayedexpansion 的作用詳細整理

設定本地為延遲擴充套件。其實也就是：延遲變數，全稱延遲環境變數擴充套件，想進階，變數延遲是必過的一關！所以這一部分希望你能認真看。

为了更好的說明問題，我們先引入一個例子。

例 1:

```
@echo off
set a=4
set a=5&echo %a%
pause
```

結果：4

解說：為什麼是 4 而不是 5 呢？在 echo 之前明明已經把變數 a 的值改成 5 了？

讓我們先了解一下批處理執行命令的機制：批處理讀取命令時是按行讀取的

（另外例如 for 命令等，其後用一對圓括號閉合的所有語句也當作一行），在處理之前要完成必要的預處理工作，這其中就包括對該行命令中的變數賦值。我們現在分析一下例 1，批處理在執行到這句“set a=5&echo %a%”之前，先把這一句整句讀取並做了預處理——對變數 a 賦了值，那麼%a%當然就是 4 了！

（沒有為什麼，批處理就是這樣做的。）而為了能夠感知環境變數的動態變化，批處理設計了變數延遲。簡單來說，在讀取了一條完整的語句之後，不立即對該行的變數賦值，而會在某個單條語句執行之前再進行賦值，也就是說“延遲”了對變數的賦值。那麼如何開啟變數延遲呢？變數延遲又需要注意什麼呢？

舉個例子說明一下：

例 2:

```
@echo off
setlocal enabledelayedexpansion
set a=4
```

```
set a=5&echo !a!  
pause
```

結果：5

解說：由於啟動了變數延遲，得到了正確答案。變數延遲的啟動語句是“setlocal enabledelayedexpansion”，並且變數要用一對歎號“!!”括起來（注意要用英文的歎號），否則就沒有變數延遲的效果。分析一下例 2，首先“setlocal enabledelayedexpansion”開啟變數延遲，然後“set a=4”先給變數 a 賦值為 4，“set a=5&echo !a!”這句是給變數 a 賦值為 5 並輸出（由於啟動了變數延遲，所以批處理能夠感知到動態變化，即不是先給該行變數賦值，而是在執行過程中給變數賦值，因此此時 a 的值就是 5 了）。再舉一個例子鞏固一下。
例 3:

```
@echo off  
setlocal enabledelayedexpansion  
for /l %%i in (1,1,5) do ( set a=%%i echo !a! )  
pause
```

結果：12345

解說：本例開啟了變數延遲並用“!!”將變數擴起來，因此得到我們預期的結果。如果不用變數延遲會出現什麼結果呢？結果是這樣的：ECHO 處於關閉狀態。ECHO 處於關閉狀態。ECHO 處於關閉狀態。ECHO 處於關閉狀態。ECHO 處於關閉狀態。即沒有感知到 for 語句中的動態變化。

batman 的說明

我來簡要說一下吧：

set：設定

local：本地（環境變數）

enable：能夠

delayed：延遲

expansion：擴充套件

setlocal enabledelayedexpansion 就是擴充套件本地環境變數延遲，
比較下面兩段程式碼：

```
@echo off  
for /l %%i in (1,1,10) do (  
set "str=%%i"
```

```
echo %str%  
)  
pause>nul
```

```
@echo off&setlocal enabledelayedexpansion  
for /l %%i in (1,1,10) do (  
set "str=%%i"  
echo !str!  
)  
pause>nul
```

第一段程式碼只會顯示 10 行“ECHO 處於關閉狀態。”，而第二段程式碼則會正確顯示 1-10 的 10 行數字。這是為什麼呢？因為在兩段程式碼的 for 迴圈前 str 都是沒有被定義的，而由於第一段程式碼沒有開啟變數延遲，所以 str 值一直是沒有定義，因而顯示出了 10 行報錯資訊；而第二段程式碼開啟了變數延遲，在 for 迴圈中每次賦予 str 的值被傳遞下去，因而會正確顯示 10 行數字，但這裡的 str 變數符必須要寫成!str!，這是沒有道理可講的，只要記住就好了。

setlocal enabledelayedexpansion 是什麼意思？

是：設定本地為延遲擴充套件。其實也就是：延遲變數，全稱“延遲環境變數擴充套件”，在 cmd 執行命令前會對指令碼進行預處理，其中有一個過程是變數識別過程，在這個過程中，如果有兩個%括起來的如%value%類似這樣的變數，就會對其進行識別，並且查詢這個變數對應的值，再而將值替換掉這個變數，這個替換值的過程,就叫做變數擴充套件，然後再執行命令。

在解釋之前，先看幾個例子的區別：

例一：

```
set value=kkkkkkkk  
echo %value%
```

將這段程式碼儲存到一個字尾為 bat 的文字檔案中。然後開啟 dos，進到對應目錄下，執行這個檔案，結果如下：

```
C:\Documents and Settings\Administrator\桌面\ln\temp\bat>set value=kkkkkkkk  
C:\Documents and Settings\Administrator\桌面\ln\temp\bat>echo kkkkkkkk  
kkkkkkkk
```

最後一行是結果，但是在結果之前，還有兩句，set value=kkkkkkk 和 echo kkkkkkk，但是在語句中，我們並沒有寫 echo kkkkkkk 的語句，這表明至少在執行到 echo %value% 這句時，對變數進行的值的替換。這就是變數的擴充套件。

那麼什麼是變數的延遲擴充套件呢？

如果大家知道 C 的“靜態變數”概念，那就應該知道，c 編譯的時候，會對靜態變數進行值的替換，但這個替換是基於靜態的前提下，那麼進行變數擴充套件時，也是這樣，但如果出現動態的情況會怎樣？在 cmd 執行中，發生動態的一種情況是在 for 語句中進行變數賦值，例如：

例二：

```
@echo off
for /l %%i in (1,1,3) do (
set k=%%i ::對 k 進行迴圈賦值
echo %k% %%i
)
```

執行這樣的指令碼，出現如下結果：

```
_1
_2
_3
```

結果出現這三句話。_ 表示空格

注：k 沒有賦初值，則替換為空。

例三：

```
@echo off
set k=yyy
for /l %%i in (1,1,3) do (
set k= %%i ::對 k 進行迴圈賦值
echo %k% %%i
)
```

結果:

```
yyy 1
yyy 2
yyy 3
```

注：k 有賦初值，則都替換為 yyy。、

例項四：

```
@echo off
setlocal enabledelayedexpansion
set k= 3
for /l %%i in (1,1,3) do (
set k=%%i
echo %k% %%i
)
```

結果：

3 1

3 2

3 3

這裡已經是用到了延遲變數，為什麼還會出現這種情況呢？再看例項五：

例項五：

```
@echo off
setlocal enabledelayedexpansion
set k= 3
for /l %%i in (1,1,3) do (
set k=%%i
echo !k! %%i
)
```

結果：

1 1

2 2

3 3

原來在延遲變數擴充套件中，要使用！來引用變數。