



请输入关键词

搜索

论坛

搜索

帮助

导航

批处理之家 » BAT教程&资料 » [系列教程]批处理for语句从入门到精通[20101225更新]

[视频教程]批处理基础视频教程

[视频教程]VBS基础视频教程

批处理在线视频分享

发帖

返回列表

1

2

3

4

5

6

7

8

9

10

...

35

下一页

namejm



荣誉版主



帖子	1348
积分	5096
技术	113
捐助	501
注册时间	2007-10-25

发表于 2008-10-16 打印 字体大小: T | T 倒序看帖 跳转到 » 1 楼

[原创] [系列教程]批处理for语句从入门到精通[20101225更新]

版主提醒

为了避免影响技术讨论、提高看帖的舒适性，请大家不要在此帖下跟无实质内容的口水帖，特别是纯顶、纯支持、纯感谢、路过之类的帖子，管理人员将不定期清理此类回帖，请大家多参与讨论少灌水，与人方便，终将给自己带来方便，谢谢合作。

批处理是一门简单的脚本语言，虽然不能独当一面，但是，若作为工作中的辅助工具，绝对会让大家有随用随写、称心如意的畅快感。

和其他语言相比，批处理语言有其先天性的优势：

- 1、系统自带，无需另行安装；
- 2、命令少，语句简洁，上手非常快；
- 3、编写出来的脚本小巧玲珑，随写随用；

但是，因为它以命令行方式工作，操作多有不便，在图形界面大行其道的windows世界里，多多少少会让大众望而却步；就算是对命令行有好感的新手，面对微软有如天书的帮助文件，很多人也会败下阵来，因此，论坛里很多会员也发出了编写系统的批处理教程的呼声。

编写系统的批处理新手教程，一直是论坛管理层讨论的热点问题，但是，各位管理人员大多都有工作在身，而系统的教程涉及的面是如此之广，面对如此浩大的工程，仅凭一两个人的力量，是难以做好的，因此，本人退而求其次，此次发布的教程，以专题的形式编写，日后人手渐多之后，再考虑组织人力编写全面的教程。

之所以选择最难的for，一是觉得for最为强大，是大多数人最希望掌握的；二是若写其他命令教程，如果没有for的基础，展开来讲解会无从下手；三是for也是批处理中最复杂最难掌握的语句，把它攻克了，批处理的学习将会一片坦途。

这次的for语句系列教程，打算按照for语句的5种句式逐一展开，在讲解 for /f 的时候，会穿插讲解批处理中一个最为关键、也是新手最容易犯错的概念：变量延迟，大纲如下：

- 一 前言
- 二 for语句的基本用法
- 三 for /f (含变量延迟)
- 四 for /r
- 五 for /d
- 六 for /l

遵照 yibantiaokuan 的建议，在顶楼放出此教程的txt版本、word版本和pdf版本，以方便那些离线浏览的会员。

由于本人水平有限，本教程尚存在不少错误，承蒙各位热心会员的关注，一些已知的错误在（对“批处理for语句从入门到精通”的找茬行动：<http://bbs.bathome.net/thread-14068-1-1.html>）这个帖子中被指出，并在陆续更新之中，请各位务必注意指正帖中指出的错误之处，并欢迎各位在那个帖子中把本教程中的错误一一挑出，本人将抽空更新教中有误的地方。

下载地址：

[http://pan.baidu.com/share/link?
shareid=3613722265&uk=1124163200](http://pan.baidu.com/share/link?shareid=3613722265&uk=1124163200)

14



zhonglumthgh: 感谢分享 技术 + 1



paladinjin: 感谢分享 技术 + 1

评分人数



rain_cyy: 我找到了我想要的 技术 + 1



小小菜鸟: 感谢分享 技术 + 1



jrx401: 深入浅出啊 技术 + 1



SIGNATURE

**尺有所短寸有所长，学好批处理没商量；
考虑问题复杂化，解决问题简洁化。
心在天山，身老沧州。**

namejm

发表于 2008-10-16 21:10 | 只看该作者

2 楼



荣誉版主



一、前言

在批处理中，for是最为强大的命令语句，它的出现，使得解析文本内容、遍历文件路径、数值递增/递减等操作成为可能；配合if、call、goto等流程控制语句，更是可以实现脚本复杂的自动化、智能化操作；合理使用for语句，还能使代码大为

帖子	1348
积分	5096
技术	113
捐助	501
注册时间	2007-10-25

简化，免除各位编写大量重复语句之苦。而能否熟练使用for语句，已经成为衡量一个人批处理水平高低最主要的标准。

在这个系列教程中，我将通过实际应用中频繁出现的例子，带领大家步入for语句的神奇之门，一步步迈向for语句的魔幻殿堂，使得大家在实际的应用中，能独立写出简洁高效的代码，在批处理的世界里自由驰骋。

注意：以下的讲解，都是基于简体中文版Windows XP Pro SP3的操作系统环境。

SIGNATURE

尺有所短寸有所长，学好批处理没商量；
考虑问题复杂化，解决问题简洁化。
心在天山，身老沧州。

[TOP](#)

namejm

 发表于 2008-10-16 21:10 | 只看该作者

3 楼



荣誉版主



帖子	1348
积分	5096
技术	113
捐助	501
注册时间	2007-10-25

二、for语句的基本用法[2008.11.9更新]

本帖最后由 Hello123World 于 2012-6-16 13:27 编辑

正如色彩缤纷的七彩光芒是由红绿蓝三原色构成的一样，最复杂的for语句，也有其基本形态，它的模样是这样的：

(以下3条代码修改由qzwqzw提供) 在cmd窗口中：

01.	FOR %variable IN (set) DO command [command-parameters] 复制代码
-----	--

在批处理文件中：

01.	FOR %%variable IN (set) DO command [command-parameters], 复制代码
-----	--

具体例子：

01.	For %i in (1 2 3) do @echo %i 复制代码
-----	---

之所以要区分cmd窗口和批处理文件两种环境，是因为在这两种环境下，命令语句表现出来的行为虽然基本一样，但是在细节上还是稍有不同，最明显的一个差异就是：在cmd窗口中，for之后的形式变量I必须使用单百分号引用，即%I；而在批处理文件中，引用形式变量I必须使用双百分号，即%%I。为了方便起见，若不是特别强调，以下的讲解都以批处理文件环境为例。

我们先来看一下for语句的基本要素都有些什么：

- “
- 1、for、in和do是for语句的关键字，它们三个缺一不可；
 - 2、%%I是for语句中对形式变量的引用，就算它在do后的语句中没有参与语句的执行，也是必须出现的；
 - 3、in之后，do之前的括号不能省略；
 - 4、command1表示字符串或变量，command2表示字符串、变量或命令语句；
- ”

现在，你可能已经会写一个简单的for语句了，比如：

[code1]

```
01. @echo off
02. for %%I in (bbs.bathome.net) do echo %%I
03. pause
```

[复制代码](#)

保存为批处理文件并执行，将会在弹出的批处理窗口中看到这样的信息：

[result1]

“ bbs.bathome.net
请按任意键继续... ”

很快地，你会觉得这个for语句是如此的简单，简单到你丝毫感受不出它的强大：这个for语句，和我直接用echo语句没什么两样啊！

是的，演示代码永远都只是演示而已，就像大多数高级语言的教科书一样，在引导新手学习的时候，基本上都是千篇一律地告诉大家如何编写一个能显示 hello world! 的窗口，从这些演示代码中，你看不到它们具有多少实用性，你只是感到有点好奇：咦，居然弹出了一个窗口？片刻之后，你就会觉得索然无味。

那好吧，为了让大家对for更加感兴趣，我们先来分析一下for语句的一些注意事项，之后，再让大家看看更为强大的for语句实例。

- “
- 1、for语句的形式变量I，可以换成26个字母中的任意一个，这些字母会区分大小写，也就是说，%%I和%%i会被认为不是同一个变量；形式变量I还可以换成其他的字符，但是，为了不与批处理中的%0 ~ %9这10个形式变量发生冲突，请不要随意把%%I替换为%%0 ~ %%9中的任意一个；
 - 2、in和do之间的command1表示的字符串或变量可以是一个，也可以是多个，每一个字符串或变量，我们称之为一个元素，每个元素之间，用空格键、跳格键、逗号、分号或等号分隔；
 - 3、for语句依次提取command1中的每一个元素，把它的值赋予形式变量I，带到do后的command2中参与命令
- ”

的执行；并且每次只提取一个元素，然后执行一次do后的命令语句，而无论这个元素是否被带到command2中参与了command2的运行；当执行完一次do后的语句之后，再提取command1中的下一个元素，再执行一次command2，如此循环，直到command1中的所有元素都已经被提取完毕，该for语句才宣告执行结束；

其中，第3点是最为关键的，它描述了for语句的执行过程，是for语句的精髓所在，大家一定要牢记这一条，才能深刻理解更为复杂的for流程。

有了以上的基础，我们再来看一个例子，这个例子修改了[code1]的部分内容，结果将大不一样：

[code2]

```
01. @echo off
02. for %%I in (bbs,bathome,net) do echo %%I
03. pause
```

[复制代码](#)

和[code1]的执行结果[result1]相比，[result2]发生了如下变化：

- 1、显示结果分成了3行（不算最后一行中文提示）；
- 2、每一行都从逗号处被切分；

如果把 bbs.bathome.net 这个字符串中的点号换为空格、跳格或等号，执行结果将和example2的执行结果别无二致。

现在，我们来分析一下[code2]代码中for语句的执行过程：

首先，for语句以逗号为分隔符，把 bbs,bathome.net 这个字符串切分成三个元素：bbs、bathome和cn，由此决定了do后的语句将会被执行3次；

然后，第一次执行过程是这样的：先把 bbs 这个字符串作为形式变量I的值，带入do后的语句中加以执行，也就是执行 echo %%I 语句，此时的I值为bbs，因此，第一次执行的结果，将会在屏幕上显示bbs这个字符串；第二次执行和第一次执行的过程是一样的，只不过此时I的值已经被替换为command1中的第二个元素了，也就是 bathome 这个字符串；如此循环，当第三次echo执行完毕之后，整条for语句才算执行完毕，此时，将执行下一条语句，也就是pause命令。

其实，这个例子只比上一个例子多了一点花样，有趣了那么一点点：一条for语句的执行结果居然被分成了3行！

为了让大家见识一下for的真正威力，本人绞尽脑汁，翻帖无数，不得要领，万般无奈之下，只好亮出了尘封在箱底多年的一段代码：检测当前硬盘都有哪些分区

^_ ^

[code3]

```
01. @echo off
02. set str=c d e f g h i j k l m n o p q r s t u v w x y z
03. echo 当前硬盘的分区有:
```

```

04. for %%i in (%str%) do if exist %%i: echo %%i:
05. pause
    复制代码

```

这段代码能检测硬盘都有哪些分区，包括U盘和移动硬盘的分区，但是，当光驱中有盘的时候，也会被列出来，这是本代码的一个缺憾，在以后的讲解中，我将向大家讲述如何消除这个瑕疵，敬请关注本系列的后续章节。

高级应用：

想知道当前目录下都有哪些文件吗？请用下面的代码：

```

01. @echo off
02. for %%i in (*.*) do echo "%%i"
03. pause
    复制代码

```

想列出当前目录下所有的文本文件吗？请用下面的代码：

```

01. @echo off
02. for %%i in (*.txt) do echo "%%i"
03. pause
    复制代码

```

想列出只用两个字符作为文件名的文本文件吗？(**hello123world注**:实际上这个代码是输出少于或等于两个字符作为文件名的文本文件)请用下面的代码：

```

01. @echo off
02. for %%i in (?.txt) do echo "%%i"
03. pause
    复制代码

```

题外话：

1、列出当前目录下各种文件的方法，最简单的还是用dir命令，但是，从以上代码中，各位可以加深对for语句执行流程的理解（用到了通配符*和?）；

2、注意：以上代码不能列出含有隐藏或系统属性的文件；(**hello123world注**:这里其实有一个很有趣的现象，windows中的系统文件一般具备两种属性——隐藏和系统；但是你如果测试的话就会发现，加上+s属性，但是不加+h的文件是可以被简单的for显示出来的。

例如：

```

01. @echo off
02. attrib +s 1.txt
03. For %%i in (*.txt) do Echo %%i
04. pause
    复制代码

```

这里的1.txt在结果中显示出来了。所以“以上代码不能列出含有隐藏或系统属性的文件”是不准确的，而因该说成“**以上代码不能列出含有隐藏属性的文件**”)

练习：用for语句建立test1.txt、test2.txt和test3.txt三个文本文件。

更全面的练习请看这个帖子：[for语句从入门到精通配套练习题](#)

10



白鹤隐云： 不错啊 技术 + 1



pclq: 6666 技术 + 1

评分人数



1123603165: 乐于助人 技术 + 1



torrent151221: 乐于助人 技术 + 1



康熙爷爷： 技术 技术 + 1

SIGNATURE

尺有所短寸有所长，学好批处理没商量；
考虑问题复杂化，解决问题简洁化。
心在天山，身老沧州。

TOP

namejm

发表于 2008-10-16 21:10 | 只看该作者

4 楼



荣誉版主



帖子 1348
积分 5096
技术 113
捐助 501
注册时间 2007-10-25

前言

for /f 是个十分强大的家伙。

如果说，for语句是批处理中最强大的语句的话，那么，for /f 就是精华中的精华。

for /f 的强大，和它拥有众多的开关密切相关。因为开关众多，所以用法复杂，本章将分成若干小节，为大家逐一介绍强大的 for /f 语句。

(一) 为解析文本而生：for /f 的基本用法

所有的对象，无论是文件、窗体、还是控件，在所有的非机器语言看来，无外乎都是形如"c:\test.txt"、"CWnd"之类的文本信息；而所有的对象，具体的如ini文件中的某条配置信息、注册表中的某个键值、数据库中的某条记录.....都只有转化为具有一定格式的文本信息，方可被代码识别、操控。可以说，编程的很大一部分工作，都是在想方设法绞尽脑汁如何提取这些文本信息。

而提取文本信息，则是for /f的拿手好戏：读取文件内容；提取某几行字符；截取某个字符片段；对提取到的内容再切分、打乱、杂糅.....只要你所能想到的花样，for /f 都会想方设法帮你办到，因为，for /f 就是被设计成专门用于解析文本的。

先来看个例子。

假如有个文本文件test.txt，内容如下：

[txt1]

“ 论坛的目标是：不求最大，但求最好，做最实用的批处理论坛。

论坛地址：bbs.bathome.net。

这里是：新手晋级的福地，高手论剑的天堂。”

那么，将如下代码保存为test.cmd，并放在test.txt同一目录下运行，将会在屏幕上原样显示test.txt的内容：

[code4]

```
01. @echo off
02. for /f %%i in (test.txt) do echo %%i
03. pause
```

[复制代码](#)

这段代码，主要是让你树立这样一种观念：读取文本文件的内容（[qzwqzw](#)
注：改为“逐行分析文本文件的内容”，因为读取文本文件内容的方法命令有很多，比如重定向输入，又比如type/more/find/sort等命令），请使用 for /f 语句！

[进阶话题：for /f 语句是把整个test.txt一次性显示出来的？](#)

在这段代码中，虽然执行结果是把test.txt中的所有内容都显示出来了，貌似 for /f 语句是把整个test.txt一次性显示到屏幕上，实际上并非如此。

无论for语句做何种变化，它的执行过程仍然遵循基本的for流程：依次处理每个元素，直到所有的元素都被处理为止。只不过在for /f语句中，这里的元素是指文件中的每一行，也就是说，for /f 语句是以行为单位处理文本文件的。这是一条极为重要的规则，在上一章中也强调过它的重要性，希望在接下来的学习过程中，你能时刻牢记这一原则，那么，很多问题将会迎刃而解。以下是验证这一说法的演示代码（在[code4]的基础上添加了&pause语句）：

[code5]

```
01. @echo off
02. for /f %%i in (test.txt) do echo %%i&pause
03. pause
```

(二) 切分字符串的利器：delims=

也许你对[code4]这段代码不屑一顾：不就是把test.txt的内容显示出来了么？好像用处不大啊。

好吧，我们来玩个魔术。

还是[txt1]这段文本，把[code4]改造一下：

[code6]

```

01. @echo off
02. for /f "delims=," %%i in (test.txt) do echo %%i
03. pause
复制代码

```

再次运行test.cmd，看到什么变化了吗？！

[result2]

```

01. 论坛的目标是：不求最大
02. 论坛地址：bbs.bathome.net。
03. 这里是：新手晋级的福地
04. 请按任意键继续...
复制代码

```

结果，你惊奇地发现，每行第一个逗号之后的所有内容都不见了（如果有不存在逗号的行，则保留原样），也就说，你成功地提取到了每行第一个逗号之前的所有内容！

试想一下，这段代码会有什么用呢？

如果别人给了你一个软件清单，每行都是“英文软件名（逗号）中文软件名”的格式，而你却只想保留英文名的时候，这段代码将是多么有用啊！再假设，有这么一个IP文件，第一列是数字格式的IP地址，第二列是具体的空间地址，列与列之间用逗号分隔，而你想提取其中数字格式的IP，呵呵，我不说你也知道该怎么办了吧？

要是文本内容不是以逗号分隔，而是以其他符号分隔，那么，把“delims=,”的逗号换成相应的符号就可以了。

在这里，我们引入了一个新的开关：“delims=，”，它的含义是：以逗号作为被处理的字符串的分隔符号。

在批处理中，指定分隔符号的方法是：添加一个形如“delims=符号列表”的开关，这样，被处理的每行字符串都会被符号列表中罗列出来的符号切分开来。

需要注意的是：如果没有指定“delims=符号列表”这个开关，那么，for /f语句默认以空格键或跳格键作为分隔符号。请把[txt1]中不同位置上的标点符号改为空格或跳格，再运行[code4]试试。

[进阶话题：如果我要指定的符号不止一个，该怎么办？](#)

在上面的讲解中，我提到了指定分隔符号的方法：添加一个形如“delims=符号列表”的开关。不知道你注意到没有，我的说法是“符号列表”而非“符号”，这是大有讲究的，因为，你可以一次性指定多个分隔符号！

还是以[txt1]为例，把[code6]再改造一下：

[code7]

```

01. @echo off

```

```

02. for /f "delims=., " %%i in (test.txt) do echo %%i
03. pause
复制代码

```

结果显示：

[\[result3\]](#)

```

01. 论坛的目标是：不求最大
02. 论坛地址：bbs
03. 这里是：新手晋级的福地
04. 请按任意键继续...
复制代码

```

这样，第一个点号或第一个逗号之前的内容都被提取出来了。

[code7]的执行过程是：逐行读取test.txt中的内容，以点号和逗号切分每一行的内容（不存在点号和逗号的行，则不再切分，为了描述的方便，我们把被点号或逗号切分的一个一个的字符串片段，称之为节），然后，for /f 会提取第一节的内容作为最终结果，显示在屏幕上。需要注意的是，在这里，所有行的字符串被切分成了两个以上的节，但是，[code7]的代码只会提取第一节字符串的内容，因为for /f 语句默认只提取第一节的字符串。

(三) 定点提取: tokens=

上一节在讲解 delims= 的时候，我一再强调 for /f 默认只能提取到第一节的内容，现在我们来思考一个问题：如果我要提取的内容不在第一节上，那怎么办？

这回，就该轮到 tokens= 出马了。

tokens= 后面一般跟的是数字，如 tokens=2，也可以跟多个，但是每个数字之间用逗号分隔，如 tokens=3,5,8，它们的含义分别是：提取第2节字符串、提取第3、第5和第8节字符串。注意，这里所说的“节”，是由 delims= 这一开关划分的，它的内容并不是一成不变的。

下面来看一个例子：

[\[txt2\]](#)

```

01. 尺有所短，寸有所长，学好批处理没商量，考虑问题复杂化，解决问题简洁化。
复制代码

```

对[txt2]这段文本，假设它们保存在文件test.txt中，如果我想提取“学好批处理没商量”这句话，该如何写代码呢？

我们稍微观察一下[txt2]就会发现，如果以逗号作为切分符号，就正好可以把“学好批处理没商量”化为单独的一“节”，结合上一节的讲解，我们知道，“delims=,” 这个开关是不可缺少的，而要提取的内容在以逗号切分的第3节上，那么，tokens= 后面的数字就应该是3了，最终的代码如下：

[\[code8\]](#)

```

01. @echo off
02. for /f "delims=, tokens=3" %%i in (test.txt) do echo %%i
03. pause
复制代码

```

如果我们现在要提取的不只一个“节”，而是多个，那又怎么办呢？比如，要提取以逗号切分的第2节和第5节字符串，是写成这样吗？

[\[code9\]](#)

```

01. @echo off
02. for /f "delims=, tokens=2,5" %%i in (test.txt) do echo %%i
03. pause
复制代码

```

运行批处理后发现，执行结果只显示了第2节的内容。

原来，echo 后面的 %%i 只接收到了 tokens=2,5 中第一个数值2所代表的那个字符串，而第二个数值5所代表的字符串因为没有变量来接收，所以就无法在执行结果中显示出来了。

那么，要如何接收 tokens= 后面多个数值所指代的内容呢？

for /f 语句对这种情况做如下规定：

如果 tokens= 后面指定了多个数字，如果形式变量为 %%i，那么，第一个数字指代的内容用第一个形式变量 %%i 来接收，第二个数字指代的内容用第二个形式变量 %%j 来接收，第三个数字指代的内容用第三个形式变量 %%k 来接收……第 N 个数字指代的内容用第 N 个形式变量来接收，其中，形式变量遵循字母的排序，第 N 个形式变量具体是什么符号，由第一个形式变量来决定：如果第一个形式变量是 %%i，那么，第二个形式变量就是 %%j；如果第一个形式变量用的是 %%x，那么，第二个形式变量就是 %%y。

现在回头去看 [code9]，你应该知道如何修改才能满足题目的要求了吧？修改结果如下：

[\[code10\]](#)

```

01. @echo off
02. for /f "delims=, tokens=2,5" %%i in (test.txt) do echo %%i %%j
03. pause
复制代码

```

如果有这样一个要求：显示 [txt2] 中的内容，但是逗号要替换成空格，如何编写代码？

结合上面所学的内容，稍加思索，你可能很快就得出了答案：

[\[code11\]](#)

```

01. @echo off
02. for /f "delims=, tokens=1,2,3,4,5" %%i in (test.txt) do echo %%i %%j
    %%k %%l %%m

```

03. pause

[复制代码](#)

写完之后，你可能意识到这样一个问题：假如要提取的“节”数不是5，而是10，或者20，或者更多，难道我也得从1写到10、20或者更多吗？有没有更简洁的写法呢？

答案是有的，那就是：如果要提取的内容是连续的多“节”的话，那么，连续的数字可以只写最小值和最大值，中间用短横连接起来即可，比如 tokens=1,2,3,4,5 可以简写为 tokens=1-5。

还可以把这个表达式写得更复杂一点：tokens=1,2-5，tokens=1-3,4,5，tokens=1-4,5.....怎么方便就怎么写吧。

大家可能还看到一种比较怪异的写法：

[\[code12\]](#)

```
01. @echo off
02. for /f "delims=, tokens=1,*" %%i in (test.txt) do echo %%i %%j
03. pause
```

结果，第一个逗号不见了，取代它的是一个空格符号，其余部分保持不变。

其中奥妙就在这个星号上面。

tokens=后面所接的星号具备这样的功能：字符串从左往右被切分成紧跟在*之前的数值所表示的节数之后，字符串的其余部分保持不变，整体被*所表示的一个变量接收。

理论讲解是比较枯燥的，特别是为了严密起见，还使用了很多限定性的修饰词，导致句子很长，增加了理解的难度，我们还是结合[\[code12\]](#)来讲解一下吧。

[txt2] 的内容被切分，切分符号为逗号，当切分完第一节之后，切分动作不再继续下去，因为 tokens=1,* 中，星号前面紧跟的是数字1；第一节字符串被切分完之后，其余部分字符串不做任何切分，整体作为第二节字符串，这样，[txt2]就被切分成了两节，分别被变量%%i和变量%%j接收。

以上几种切分方式可以结合在一起使用。不知道下面这段代码的含义你是否看得懂，如果看不懂的话，那就运行一下代码，然后反复揣摩，你一定会更加深刻地理解本节所讲解的内容的：

[\[code13\]](#)

```
01. @echo off
02. for /f "delims=, tokens=1,3-4,*" %%i in (test.txt) do echo %%i %%j
    %%k %%l
03. pause
```

[复制代码](#)

(四) 跳过无关内容，直奔主题：skip=n

很多时候，有用的信息并不是贯穿文本内容的始终，而是位于第N行之后的行内，为了提高文本处理的效率，或者不受多余信息的干扰，for /f 允许你跳过这些无用的行，直接从第N+1行开始处理，这个时候，就需要使用参数 skip=n，其中，n是一个正整数，表示要跳过的行数。例如：

[code14]

```
01. @echo off
02. for /f "skip=2" %%i in (test.txt) do echo %%i
03. pause
```

[复制代码](#)

这段代码将跳过头两行内容，从第3行起显示test.txt中的信息。

(五) 忽略以指定字符打头的行：eol=

在cmd窗口中敲入：for /?，相关的解释为：

“ eol=c - 指一个行注释字符的结尾(就一个) ”

“ FOR /F "eol=; tokens=2,3* delims=, " %i in (myfile.txt) do @echo %i %j %k ”

会分析 myfile.txt 中的每一行，忽略以分号打头的那些行.....

第一条解释狗屁不通，颇为费解：行注释字符的结尾是什么意思？“(就一个)”怎么回事？结合第二条解释，才知道eol有忽略指定行的功能。但是，这两条解释是互相矛盾的：到底是忽略以指定字符打头的行，还是忽略以指定字符结尾的行？

实践是检验真理的唯一标准，还是用代码来检验一下eol的作用吧：

[code15]

```
01. @echo off
02. for /f "eol=;" %%i in (test.txt) do echo %%i
03. pause
```

[复制代码](#)

结果，那些以分号打头的行没有显示出来。

由此可见，第二条解释是正确的，eol= 的准确含义是：忽略以指定字符打头的行。而第一条的“结尾”纯属微软在信口开河。

那么，“(就一个)”又作何解释呢？

试试这个代码：

[code16]

```
01. @echo off
02. for /f "eol=;" %%i in (test.txt) do echo %%i
03. pause
```

[复制代码](#)

此时，屏幕上出现 **此时不应有 ;**。的报错信息。可见，在指定字符的时候，只能指定1个——很多时候，我对这样的设计颇有微词而又无可奈何：为什么只能指定1个而不是多个？要忽略多个还得又是if又是findstr加管道来多次过滤，那效率实在太低下了一一能用到的功能基本上都提供，但是却又做不到更好，批处理，你的功能为什么那么弱？

不知道大家注意到没有，如果test.txt中有以分号打头的行，那么，这些行在代码[code14]的执行结果中将凭空消失。

原来，**for /f** 语句是默认忽略以分号打头的行内容的，正如它默认以空格键或跳格键作为字符串的切分字符一样。**(hello123world注：eol=;这种默认设置，在delims=;时变得无效。)**

很多时候，我们可以充分利用这个特点，比如，在设计即将用for读取的配置文件的时候，可以在注释文字的行首加上分号，例如在编写病毒文件查杀代码的时候，可以通过for语句来读取病毒文件列表，那么，病毒文件列表.ini这个配置文件可以这样写：

“ ;以下是常见的病毒文件，请见一个杀一个^_^
;copyleft:没有
qq.exe
msn.exe
iexplore.exe ”

如果要取消这个默认设置，可选择的办法是：

- 1、为eol=指定另外一个字符；
- 2、使用 for /f "eol=" 语句，也就是说，强制指定字符为空，就像对付delims=一样。

(六) 如何决定该使用 for /f 的哪种句式？(兼谈usebackq的使用)

for /f %%i in (.....) do (.....) 语句有好几种变形语句，不同之处在于第一个括号里的内容：有的是用单引号括起来，有的是用双引号包住，有的不用任何符号包裹，具体格式为：

- “
- 1、for /f %%i in (文件名) do (.....)
 - 2、for /f %%i in ('命令语句') do (.....)
 - 3、for /f %%i in ("字符串") do (.....)
- ”

看到这里，我想很多人可能已经开始犯了迷糊了：如果要解决一个具体问题，面对这么多的选择，如何决定该使用哪一条呢？

实际上，当我在上面罗列这些语句的时候，已经有所提示了，不知道你是否注意到了。

如果你一时无法参透其中奥妙，那也无妨，请听我——道来便是。

1、当你希望读取文本文件中的内容的话，第一个括号中不用任何符号包裹，应该使用的是第1条语句；例如：你想显示test.txt中的内容，那么，就使用 for /f %%i in (test.txt) do echo %%i；

2、当你读取的是命令语句执行结果中的内容的话，第一个括号中的命令语句必须使用单引号包裹，应该使用的是第2条语句；例如：你想显示当前目录下文件名中含有test字符串的文本文件的时候，应该使用 for /f %%i in ('dir /a-d /b *test*.txt') do echo %%i 这样的语句；

3、当你要处理的是一个字符串的时候，第一个括号中的内容必须用双引号括起来，应该是用的是第3条语句；例如：当你想把bbs.bathome.net这串字符中的点号换为短横线并显示出来的话，可以使用 for /f "delims=. tokens=1-3" %%i in ("bbs.bathome.net") do echo %%i-%%%j-%%%k 这样的语句。

很显然，第一个括号里是否需要用符号包裹起来，以及使用什么样的符号包裹，取决于要处理的对象属于什么类型：如果是文件，则无需包裹；如果是命令语句，则用单引号包裹；如果是字符串，则使用双引号括起来。

当然，事情并不是绝对如此，如果细心的你想到了批处理中难缠的特殊字符，你肯定会头大如斗。

或许你头脑中灵光一闪，已经想到了一个十分头痛的问题：在第1条语句中，如果文件名中含有空格或&，该怎么办？

照旧吗？

拿个叫 test 1.txt 的文件来试试。

你很快写好了代码，新建文件-->码字-->保存为批处理，前后费时不到1分钟：

[code17]

01.	@echo off
02.	for /f %i in (test 1.txt) do echo %%i
03.	pause

[复制代码](#)

你兴冲冲地双击批处理，运行后，屏幕上出现了可耻的报错信息：系统找不到文件 test。

当你把 test 1.txt 换成 test&1.txt 后，更怪异的事情发生了：CMD窗口在你眼前一闪而过，然后，优雅地消失了。

你可能觉得自己的代码写错了某些符号，你再仔细的检查了一次，确认没有笔误，然后，你再次双击批处理，结果问题照旧；你开始怀疑其他程序对它可能有影响，于是关掉其他窗口，再运行了一次，问题依旧；你不服气地连续运行了好几次，还是同样的结果。

怪哉！

你一拍大腿，猛然想起了一件事：当路径中含有特殊字符的时候，应该使用引号把路径括起来。对，就是它了！

但是，当你把代码写出来之后，你很快就焉了：for /f %%i in ("test 1.txt") do echo %%i，这不就是上面提到的第3条 for /f 命令的格式吗？批处理会把 test 1.txt 这个文件名识别为字符串啊！

你百无聊赖地在CMD窗口中输入 for /?，并重重地敲下了回车，漫无目的地在帮助信息中寻找，希望能找到点什么。

结果还真让你到了点什么。

你看到了这样的描述：

usebackq - 指定新语法已在下类情况中使用：
在作为命令执行一个后引号的字符串并
且一个单
引号字符为文字字符串命令并允许在
filenameset 中使用双引号扩起文件名称。

但是，通读一遍之后，你却如坠五里雾中，不知所云。

还好，下面有个例子，并配有简单的说明：

FOR /F "usebackq delims==" %i IN (`set`) DO
@echo %i
会枚举当前环境中的环境变量名称。

你仔细对比了for /f语句使用usebackq和不使用usebackq时在写法上的差别，很快就找到了答案：当使用了usebackq之后，如果第一个括号中是一条命令语句，那么，就要把单引号'改成后引号`（键盘左上角esc键下面的那个按键，与~在同一键位上）。

回过头去再看那段关于usebackq的描述，字斟句酌，反复揣摩，终于被你破译了天机：`usebackq` 是一个增强型参数，当使用了这个参数之后，原来的for语句中第一个括号内的写法要做如下变动：如果第一个括号里的对象是一条命令语句的话，原来的单引号'要改为后引号`；如果第一个括号里的对象是字符串的话，原来的双引号"要改为单引号'；如果第一个括号里的对象是文件名的话，要用双引号"括起来。

验证一下，把[code17]改写成如下代码：

[code18]

```
01. @echo off
02. for /f "usebackq" %%i in ("test 1.txt") do echo %%i
03. pause
```

[复制代码](#)

测试通过！

此时，你很可能会仰天长叹：Shit，微软这该死的机器翻译！

至于把[code17]代码中的空格换成&后，CMD窗口会直接退出，那是因为&是复合语句的连接符，CMD在预处理的时候，会优先把&前后两部分作为两条语句来解析，而不是大家想象中的一条完整的for语句，从而产生了严重的语法错误。因为牵涉到预处理机制问题，不属于本节要讨论的内容，在此不做详细讲解。

这个时候，我们会吃惊地发现，区区一条for语句，竟然有多达6种句型：



- 1、for /f %%i in (文件名) do (.....)
- 2、for /f %%i in ('命令语句') do (.....)
- 3、for /f %%i in ("字符串") do (.....)
- 4、for /f "usebackq" %%i in ("文件名") do (.....)
- 5、for /f "usebackq" %%i in (`命令语句`) do (.....)
- 6、for /f "usebackq" %%i in ('字符串') do (.....)



其中，4、5、6由1、2、3发展而来，他们有这样的对应关系：1-->4、2-->5、3-->6。

好在后3种情形并不常用，所以，牢牢掌握好前三种句型的适用情形就可以了，否则，要在这么多句型中确定选择哪一条语句来使用，还真有点让人头脑发懵。

至于 for /f 为什么要增加usebacq参数，我只为第4条语句找到了合理的解

释：为了兼容文件名中所带的空格或&。它在第5、6条语句中为什么还有存在的必要，我也是很明白，这有待于各位去慢慢发现。（[hello123world](#)^注：这种解释虽然有点不靠谱，但也算一种解释，大家将就看看吧。启用usebackq选项的时候，“文件名”取代了“字符串”，那么“字符串”只好改变为“命令语句”，“命令语句”只好用后引号重新表示——简而言之，是“文件名”符号改变引起的蝴蝶效应。言外之意：usebackq除了在处理带空格的文件名时会用到外，根本就没有其它的出场机会和存在价值。）

[本帖最后由 namejm 于 2010-12-25 03:22 编辑]

9



jrx401: 感谢详细的解释。 技术 + 1



chris0912: 感谢分享 技术 + 1

评分人数



L-殇: 已将QQ,MSN,IE删除,现用GG与YY... 技术 + 1



an410398183: 学习学习 技术 + 1



wangx: 对我的帮助很大，谢谢分享 技术 + 1

SIGNATURE

尺有所短寸有所长，学好批处理没商量；
考虑问题复杂化，解决问题简洁化。
心在天山，身老沧州。

TOP

namejm

发表于 2008-10-16 21:11 | 只看该作者

5 楼



荣誉版主



帖子 1348
积分 5096
技术 113
捐助 501
注册时间 2007-10-25

(七) 变量延迟详解[2009.2.12更新]

变量延迟在for语句中起着至关重要的作用，不只是在for语句中，在其他的复合语句中，它也在幕后默默地工作着，为了突出它的重要性，本节内容在单独的楼层中发出来，希望引起大家的重视。

对于批处理新手而言，“变量延迟”这个概念很可能闻所未闻，但是，它却像一堵横亘在你前进道路上的无形高墙，你感受不到它的存在，但当你试图往前冲时，它会把你狠狠地弹回来，让你无法逾越、无功而返；而一旦找到了越过它的方法，你就会发现，在for的世界里，前面已经是一片坦途，而你对批处理的理解，又上升到了一个新的境界。

例如，你编写了这样一个代码：
[\[code19\]](#)

```
01. @echo off
02. set num=0&&echo %num%
03. pause
```

[复制代码](#)

你的本意是想对变量num赋值之后，再把这个值显示出来，结果，显示出来的并不是0，而是显示：ECHO 处于关闭状态。

之所以会出错，是因为“变量延迟”这个家伙在作怪。

在讲解变量延迟之前，我们需要了解一下批处理的执行过程，它将有助于我们深入理解变量延迟。

批处理的执行过程是怎样的呢？

“自上而下，逐条执行”，我想，这个经典的说法大家都已经耳熟能详了，没事的时候倒着念，也还别有一番古韵呢^_^，但是，我想问大家的是，大家真的深刻地理解了这句话的含义了吗？

“自上而下”，这一条和我们本节的讲解关系不大，暂时略过不说，后一条，“逐条执行”和变量延迟有着莫大的干系，它是本节要关注的重点。

很多人往往认为一行代码就是一条语句，从而把“逐条执行”与“逐行执行”等同起来，这就大错特错了。

莫非“逐条执行”里暗藏着玄机？

正是如此。

“逐条”并不等同于“逐行”。这个“条”，是“一条完整的语句”的意思，并不是指“一行代码”。在批处理中，是不是一条完整的语句，并不是以行来论的，而是要看它的作用范围。

什么样的语句才算“一条完整的语句”呢？

1、在复合语句中，整个复合语句是一条完整的语句，而无论这个复合语句占用了多少行的位置。常见的复合语句有：for语句、if.....else语句、用连接符&、||和&&连接的语句，用管道符号|连接的语句，以及用括号括起来的、由多条语句组合而成的语句块；

2、在非复合语句中，如果该语句占据了一行的位置，则该行代码为一条完整的语句。

例如：

[\[code20\]](#)

```

01. @echo off
02. set num=0
03. for /f %%i in ('dir /a-d /b *.exe') do (
04.     set /a num+=1
05.     echo num 当前的值是 %num%
06. )
07. echo 当前目录下共有 %num% 个exe文件
08. dir /a-d /b *.txt|findstr "test">>nul&&(
09.     echo 存在含有 test 字符串的文本本件
10. )||echo 不存在含有 test 字符串的文本文件
11. if exist test.ini (
12.     echo 存在 test.ini 文件

```

```
13. ) else echo 不存在 test.ini 文件
```

```
14. pause
```

[复制代码](#)

上面的代码共有14行，但是只有完整的语句只有7条，它们分别是：

第1条：第1行的echo语句；

第2条：第2行的set语句；

第3条：第3、4、5、6行上的for复合语句；

第4条：第7行的echo语句；

第5条：第8、9、10行上用&&和||连接的复合语句；

第6条：第11、12、13行上的if.....else复合语句；

第7条：第14行上的pause语句。

在这里，我之所以要花这么长的篇幅来说明一行代码并不见得就是一条语句，是因为批处理的执行特点是“逐条”执行而不是“逐行”执行，澄清了这个误解，将会更加理解批处理的预处理机制。

在代码“逐条”执行的过程中，cmd.exe这个批处理解释器会对每条语句做一些预处理工作，这就是批处理中大名鼎鼎的“预处理机制”。

预处理的大致情形是这样的：首先，把一条完整的语句读入内存中（不管这条语句有多少行，它们都会被一起读入），然后，识别出哪些部分是命令关键字，哪些是开关、哪些是参数，哪些是变量引用……如果代码语法有误，则给出错误提示或退出批处理环境；如果顺利通过，接下来，就把该条语句中所有被引用的变量及变量两边的百分号对，用这条语句被读入内存之就已经赋予该变量的具体值来替换……当所有的预处理工作完成之后，批处理才会执行每条完整语句内部每个命令的原有功能。也就是说，如果命令语句中含有变量引用（变量及紧邻它左右的百分号对），并且某个变量的值在命令的执行过程中被改变了，即使该条语句内部的其他地方也用到了这个变量，也不会用最新的值去替换它们，因为某条语句在被预处理的时候，所有的变量引用都已经被替换成字符串常量了，变量值在复合语句内部被改变，不会影响到语句内部的其他任何地方。

顺便说一下，运行代码[code20]之后，将在屏幕上显示当前目录下有多少个exe文件，是否存在含有 test 字符串的文本文件，以及是否存在 test.ini 这个文件等信息。让很多人百思不得其解的是：如果当前目录下存在exe文件，那么，有多少个exe文件，屏幕上就会提示多少次 "num 当前的值是 0"，而不是显示1到N (N是exe文件的个数)。

结合上面两个例子，我们再来分析一下，为什么这两段代码的执行结果和我们的期望有一些差距。

在[code19]中，set num=0&&echo %num%是一条复合语句，它的含义是：把0赋予变量num，成功后，显示变量num的值。

虽然是在变量num被赋值成功后才显示变量num的值，但是，因为这是一条复合语句，在预处理的时候，&&后的%num%只能被set语句之前的语句赋予变量num的具体值来替换，而不能被复合语句内部、&&之前的set语句对num所赋予的值来替换，可见，此num非彼num。可是，在这条复合语句之前，我们并没有对变

量num赋值，所以，&&之后的%num%是空值，相当于在&&之后只执行了echo这一命令，所以，会显示echo命令的当前状态，而不是显示变量num的值（虽然该变量的值被set语句改变了）。

在[code20]中，for语句的含义是：列举当前目录下的exe文件，每发现一个exe文件，变量num的值就累加1，并显示变量num的值。

看了对[code19]的分析之后，再来分析[code20]就不再那么困难了：第3、4、5行上的代码共同构成了一条完整的for语句，而语句"echo num 当前的值是%num%"与"set /a num+=1"同处复合语句for的内部，那么，第4行上set改变了num的值之后，并不能对第5行上的变量num有任何影响，因为在预处理阶段，第5行上的变量引用%num%已经被在for之前就赋予变量num的具体值替换掉了，它被替换成0（是被第2行上的set语句赋予的）。

如果想让代码[code19]的执行结果中显示&&之前赋予num的值，让代码[code20]在列举exe文件的时候，从1到N地显示exe文件的数量，那又该怎么办呢？

对代码[code19]，可以把用&&连接复合语句拆分为两条单独的语句，写成：

```
01. @echo off
02. set num=0
03. echo %num%
04. pause
```

[复制代码](#)

但是，这不是我们这次想要的结果。

对这两段代码都适用的办法是：使用变量延迟扩展语句，让变量的扩展行为延迟一下，从而获取我们想要的值。

在这里，我们先来充下电，看看“变量扩展”有是怎么一回事。

用CN-DOS里批处理达人willsort的原话，那就是：“[在许多可见的官方文档中，均将使用一对百分号闭合环境变量以完成对其值的替换行为称之为“扩展（expansion）”](#)，这其实是一个第一方的概念，是从命令解释器的角度进行称谓的，而从我们使用者的角度来看，则可以将它看作是引用（Reference）、调用（Call）或者获取（Get）。”（见：什么情况下该使用变量延迟？<http://www.cn-dos.net/forum/viewthread.php?tid=20733>）说得直白一点，所谓的“变量扩展”，实际上就是很简单的这么一件事情：用具体的值去替换被引用的变量及紧贴在它左右的那对百分号。

既然只要延迟变量的扩展行为，就可以获得我们想要的结果，那么，具体的做法又是怎样的呢？

[一般说来，延迟变量的扩展行为，可以有如下选择：](#)

- 1、[在适当位置使用 setlocal enabledelayedexpansion 语句；](#)
- 2、[在适当的位置使用 call 语句。](#)

使用 setlocal enabledelayedexpansion 语句，那么，[code19]和[code20]可以分别修改为：

```

01. @echo off
02. setlocal enabledelayedexpansion
03. set num=0&&echo !num!
04. pause
复制代码
```

```

01. @echo off
02. set num=0
03. setlocal enabledelayedexpansion
04. for /f %%i in ('dir /a-d /b *.exe') do (
05.     set /a num+=1
06.     echo num 当前的值是 !num!
07. )
08. echo 当前目录下共有 %num% 个exe文件
09. dir /a-d /b *.txt|findstr "test">>nul&&(
10.     echo 存在含有 test 字符串的文本本件
11. )||echo 不存在含有 test 字符串的文本文件
12. if exist test.ini (
13.     echo 存在 test.ini 文件
14. ) else echo 不存在 test.ini 文件
15. pause
复制代码
```

使用第call语句，那么，[code19]和[code20]可以分别修改为：

```

01.
02. @echo off
03. set num=0&&call echo %%num%%
04. pause
复制代码
```

```

01.
02. @echo off
03. set num=0
04. for /f %%i in ('dir /a-d /b *.exe') do (
05.     set /a num+=1
06.     call echo num 当前的值是 %%num%%
07. )
08. echo 当前目录下共有 %num% 个exe文件
09. dir /a-d /b *.txt|findstr "test">>nul&&(
10.     echo 存在含有 test 字符串的文本本件
11. )||echo 不存在含有 test 字符串的文本文件
12. if exist test.ini (
13.     echo 存在 test.ini 文件
14. ) else echo 不存在 test.ini 文件
15. pause
```

[复制代码](#)

由此可见，如果使用 setlocal enabledelayedexpansion 语句来延迟变量，就要把原本使用百分号对闭合的变量引用改为使用感叹号对来闭合；如果使用call语句，就要在原来命令的前部加上 call 命令，并把变量引用的单层百分号对改为双层。其中，因为call语句使用的是双层百分号对，容易使人犯迷糊，所以用得较少，常用的是使用 setlocal enabledelayedexpansion 语句（set是设置的意思，local是本地的意思，enable是能够的意思，delayed是延迟的意思，expansion是扩展的意思，合起来，就是：让变量成为局部变量，并延迟它的扩展行为）。

通过上面的分析，我们可以知道：

1、为什么要使用变量延迟？因为要让复合语句内部的变量实时感知到变量值的变化。

2、在哪些场合需要使用变量延迟语句？在复合语句内部，如果某个变量的值发生了改变，并且改变后的值需要在复合语句内部的其他地方被用到，那么，就需要使用变量延迟语句。而复合语句有：for语句、if.....else语句、用连接符&、||和&&连接的语句、用管道符号|连接的语句，以及用括号括起来的、由多条语句组合而成的语句块。最常见的场合，则是for语句和if.....else语句。

3、怎样使用变量延迟？

方法有两种：

① 使用 setlocal enabledelayedexpansion 语句：在获取变化的变量值语句之前使用setlocal enabledelayedexpansion，并把原本使用百分号对闭合的变量引用改为使用感叹号对来闭合；

② 使用 call 语句：在原来命令的前部加上 call 命令，并把变量引用的单层百分号对改为双层。

“变量延迟”是批处理中一个十分重要的机制，它因预处理机制而生，用于复合语句，特别是大量使用于强大的for语句中。只有熟练地使用这一机制，才能在for的世界中如鱼得水，让自己的批处理水平更上一层楼。很多时候，对for的处理机制，我们一直是雾里看花，即使偶有所得，也只是只可意会难以言传。希望大家反复揣摩，多加练习，很多细节上的经验，是只有通过大量的摸索才能得到的。
Good Luck!

本节内容在原理上参考了这篇文章：什么情况下该使用变量延迟？

<http://www.cn-dos.net/forum/viewthread.php?tid=20733>，在本论坛中的地址是：<http://bbs.bathome.net/viewthread.php?tid=2899>

特别鸣谢：willsort。

1



白鹤隐云：感谢分享 技术 + 1

评分人数

SIGNATURE

尺有所短寸有所长，学好批处理没商量；
考虑问题复杂化，解决问题简洁化。
心在天山，身老沧州。

namejm

 发表于 2008-10-16 21:11 | 只看该作者

6 楼



荣誉版主



帖子 1348
 积分 5096
 技术 113
 捐助 501
 注册时间 2007-10-25

四、翻箱倒柜遍历文件夹：for /r

(一) for /r 的作用及用法

按照帮助信息里文绉绉的说法，for /r 的作用是“递归”，我们换一个通俗一点的，叫“遍历文件夹”。

更详细的解释就是：在下面的语句中，如果“元素集合”中只是一个点号，那么，这条语句的作用就是：列举“目录”及其之下所有子目录，对这些文件夹都执行“命令语句集合”中的命令语句。其作用与嵌套进 for /f 复合语句的 "dir /ad /b /s 路径" 功能类似。如果省略了“目录”，将在当前目录下执行前面描述的操作。

 for /r 目录 %%i in (元素集合) do 命令语句集合 

先来个代码增强一下印象：

[code21]

```
01. @echo off
02. for /r d:\test %%i in (.) do echo %%i
03. pause
```

[复制代码](#)

执行的结果如下所示：

 d:\test\.
d:\test\1\.
d:\test\2\.
d:\test\3\.



效果就是显示 d:\test 目录及其之下所有子目录的路径，其效果与 dir /ad /b /s d:\test 类似。若要说到两者的区别，可以归纳出3点：

- 1、for /r 列举出来的路径最后都带有斜杠和点号，而 dir 语句则没有，会对获取到的路径进行进一步加工产生影响；
- 2、for /r 不能列举带隐藏属性的目录，而 dir 语句则可以通过指定 /a 后面紧跟的参数来获取带指定属性的目录，更加灵活；
- 3、若要对获取到的路径进行进一步处理，则需要把 dir 语句放入 for /f 语句中进行分析，写成 for /f %%i in ('dir /ad /b /s') do 的形式；由于 for /r 语句是边列举路径边进行处理，所以，在处理大量路径的时候，前期不会感到有停顿，而 for /f 语句则需要等到 dir /ad /b /s 语句把所有路径都列举完之后，再读入内存进行处理，所以，在处理大量路径的时候，前期会感到有明显的停顿。

第2点差别很容易被大家忽视，导致用 for /r 列举路径的时候会造成遗漏；而

第3点则会让大家有更直观的感受，很容易感觉到两者之间的差别。

要是“元素集合”不是点号呢？那又如何？

来看看这个代码：

[\[code22\]](#)

```
01. @echo off
02. for /r d:\test %%i in (a b c) do echo %%i
03. pause
```

[复制代码](#)

运行的结果是：

“
D:\test\1\a
D:\test\1\b
D:\test\1\c
D:\test\2\a
D:\test\2\b
D:\test\2\c
D:\test\3\a
D:\test\3\b
D:\test\3\c
”

原来，它的含义是：列举 d:\test 及其所有的子目录，对所有的目录路径都分别添加a、b、c之后再显示出来。

再来看一个代码：

[\[code23\]](#)

```
01. @echo off
02. for /r d:\test %%i in (*.txt) do echo %%i
03. pause
```

[复制代码](#)

运行结果是：

“
D:\test\test.txt
D:\test\1\1.txt
D:\test\1\2.txt
D:\test\2\1.txt
D:\test\2\2.txt
D:\test\3\1.txt
”

这段代码的含义是：列举 d:\test 及其所有子目录下的txt文本文件（以.txt结尾的文件夹不会被列出来）。

我们再回过头来归纳一下这个语句的作用：

“ for /r 目录 %%i in (元素集合) do 命令语句集合 ”

上面语句的作用是：

1、列举“目录”及该目录路径下所有子目录，并把列举出来的目录路径和元素集合中的每一个元素拼接成形如“目录路径\元素”格式的新字符串，然后，对每一条这样的新

字符串执行“命令语句集合”中的每一条命令；

特别的是：当“元素集合”带以点号分隔的通配符?或*的时候，把“元素集合”视为文件（不视为文件夹），整条语句的作用是匹配“目录”所指文件夹及其所有子文件夹下

匹配的文件；若不以点号分隔，则把“元素集合”视为文件夹（不视为文件）；

2、当省略掉“目录”时，则针对当前目录；

3、当元素集合中仅仅是一个点号的时候，将只列举目录路径；

(二) for /r 还是 dir /ad /b /s? 列举目录时该如何选择

前面已经说过，当列举目录时，for /r 和 dir /ad /b /s 的效果是非常类似的，这就产生了一个问题：当我要获取目录路径并进行进一步处理的时候，两者之间，我该如何选择？

这个问题，前面其实已经有过一些讨论，现在我们再来作详细的分析。

我们来看一下两者各自的优缺点：

1、for /r:

1) 优点：

① 只通过1条语句就可以同时实现获取目录路径和处理目录路径的操作；

② 遍历文件夹的时候，是边列举边处理的，获取到一条路径就处理一条路径，内存占用小，处理大量路径的时候不会产生停顿感；

2) 缺点：

① 不能获取到带隐藏属性的目录，会产生遗漏；

② 不能获取带指定属性的目录

2、dir /ad /s:

1) 优点：

① 能一次性获取带任意属性的目录，不会产生遗漏；

② 能通过指定不同的参数获取带任意属性的目录，更具灵活性。

2) 缺点：

① dir /ad /s 语句仅能获取到目录路径，若要实现进一步的处理，还需要嵌入 for /f 语句中才能实现，写法不够简洁；

② 嵌入 for /f 语句之后，需要写成 for /f "delims=" %%i in ('dir /ad /b /s') do 的格式，受 for /f 语句运行机制的制约，需要先列举完所有的路径放入内存之后，才能对每一条路径进行进一步的处理，处理大量路径时，内存占用量偏大，并且在前期会产生明显的停顿感，用户体验度不够好；

综合上述分析，可以做出如下选择：

1、若仅仅是为了获取某文件夹及其所有子文件夹的路径的话，请选择 dir /ad /b /s 语句；

2、若需要过滤带隐藏属性的文件夹的话，for /r 和 dir 语句都可以实现，但 for /r 内存占用小，处理速度快，是上上之选；

3、若需要获取所有文件夹，则除了 dir /ad /b /s 外，别无选择，因为 for /r 语句会遗漏带隐藏属性的文件夹；

在实际的使用中，我更喜欢使用 for /f 和 dir 的组合，因为它不会产生遗漏，并能给我带来更灵活的处理方式，唯一需要忍受的，就是它在处理大量路径时前期的停顿感，以及在这背后稍微有点偏高的内存占用；在我追求速度且可以忽略带隐藏属性的目录的时候，我会换用 for /r 的方案，不过这样的情形不多——有谁会愿意为了追求速度而容忍遗漏呢？

1



red88: 知识写得简单明了，十分感谢呀！ 技术 + 1

评分人数

SIGNATURE

尺有所短寸有所长，学好批处理没商量；
考虑问题复杂化，解决问题简洁化。
心在天山，身老沧州。

TOP

namejm

发表于 2008-10-16 21:11 | 只看该作者

7 楼



荣誉版主



帖子 1348
积分 5096
技术 113
捐助 501

本帖最后由 Hello123World 于 2011-9-18 13:24 编辑

五、仅仅为了匹配第一层目录而存在：for /d

for /d 中 /d，完整的含义是 /directory，本意是为了处理文件夹，它的完整语句应该是这样的：

注册时间 2007-10-25



for /d %%i in (元素集合) do 命令语句集合



当“元素集合”中包含有通配符?或*时，它会匹配文件夹，但是，相比 for /r 而言，这个时候的for /d，其作用就小得可怜了：它仅能匹配当前目录下的第一级文件夹，或是指定位置上的文件夹，而不能匹配更深层次的子文件夹。

例如：for /d %%i in (d:\test*) do echo %%i 这样的语句，会匹配到形如：d:\test、d:\test1、d:\test2之类的文件夹，若不存在这样的路径，将不会有任何回显。

当“元素集合”中不包含任何的通配符时，它的作用和 "for %%i in (元素集合) do 命令语句集合" 这样的语句别无二致。

因此，for /d 的角色就变得很微妙了：当“元素集合”中包含通配符?或*时，它的作用就是匹配文件夹，此时，它仅能匹配当前目录下的第一级文件夹，或是指定位置上的文件夹，在层次深度上不及 for /r，但和 for /r 一样的坏脾气：不能匹配带隐藏属性的文件夹；在灵活性上不及for /f和dir的组合；当“元素集合”中不包含任何统配符的时候，它完全是 "for %%i in (元素集合) do" 语句的翻版，但是又稍显复杂。

for /d 的作用是如此有限，我使用的次数是如此之少，以至于我一度找不到它的用武之地，认为它食之无味，弃之可惜，完全是鸡肋一块。

某年某月，我在cmd窗口里写下了这样的代码：

[\[code24\]](#)

```
01. for /d %i in (test*) do @echo %i
```

[复制代码](#)

我的本意是想查看在我的临时目录下，长年累月的测试工作到底建立了多少测试文件夹，以便我随后把echo换成rd删除之。这个时候，我发现这条代码是如此的简洁，是 for /r 或 for和 dir /ad /b 的组合所无法替代的（echo换成rd就可以直接删除掉这些测试目录）。

简洁的代码给我带来的喜悦仅仅持续了短短10几秒的时间，我便开始了迷惘——能用到for /d的类似情形，貌似少之又少且乏善可陈啊。

([hello123world](#)注：正如[qzwqzw](#)所言，for /r /d是可以一起使用的；【在for有限的4个参数中，据我所知只有/r /d可以一起使用】)。

例如：

```
01. @echo off
```

```
02. For /r /d %%i in (*) do echo %%i
```

```
03. pause>nul
```

[复制代码](#)

效果：

显示当前目录下所有的文件夹【包括子文件夹】；等价于 "dir /ad /s /b"。

for /r /d 其实是对 /d 参数的扩展，/d参数本身只能处理第一层文件夹，但是加上/r参数后就可以处理所有的子文件夹；

“ for /r /d依然不能处理隐藏文件夹。 ”

“ 这里给出使用for /r /d的一般条件：

- 1.要对文件夹进行操作 (dir /ad /s /b可以显示，但不能对文件夹进行操作) ；
- 2.不处理隐藏文件夹 (说到底，还是for /f 和dir结合的命令更强大些) 。

)

2



Hello123World: 毕竟是2年前的老帖，值得原谅 技术 + 1



qzwqzw: 如果你知道for /d是可以和/r联用，你还觉得 ... 技术 + 1

评分人数

SIGNATURE

**尺有所短寸有所长，学好批处理没商量；
考虑问题复杂化，解决问题简洁化。
心在天山，身老沧州。**

TOP

namejm

发表于 2008-10-16 21:11 | 只看该作者

8 楼



荣誉版主



帖子 1348

积分 5096

技术 113

捐助 501

注册时间 2007-10-25

六、计数循环：for /I

/I者，/loop的缩写是也，从鸟语翻译过来，loop就是循环的意思。实际上，所有的for语句，都可以看成是一种“循环”，只是在/I中，特指按照指定次数进行循环罢了。

for /I 语句的完整格式是这样的：for /I %%i in (x,y,z) do (.....)，在这个语句中，x、y和z都只能取整数，正负皆可，x指代起始值，y指代步长，z为终止值，具体含义为：从x开始计数，以y为步长，直至最接近z的那个整数值为止，这之间有多少个数，do后的语句就执行多少次。

举个具体的例子：

[code25]

```
01. for /I %%i in (1,2,10) do echo bathome
```

[复制代码](#)

在以上的代码中，初始值是1，步长为2，终止值为10，表明计数从1开始，每隔2个数计算一次，直至最接近10的那个整数，罗列出来，就是1,3,5,7,9，再下

一个就是11，超过10了，不再计算在内，所以，do后的语句只执行5次，将连续显示5个bathome。

实际上，x, y和z的值可正可负，甚至为0，限制非常宽松：

- 1、步长y的值不能为0；**
- 2、当步长y的值为正整数时，终止值z不能小于初始值x；**
- 3、当步长y的值为负整数的时候，终止值z不能大于初始值x。**

换而言之，必须保证in和do之间能取到一个有效的数组序列。

例如：

[code26]

```
01. for /l %%i in (-1,2,5) do echo bathome
```

[复制代码](#)

[code27]

```
01. for /l %%i in (5,-2,-1) do echo bathome
```

[复制代码](#)

以上两条代码的功能完全一样，都将显示4次bathome，区别就在于[code26]是正序计算，而[code27]是逆序计数而已。

以下几条代码都是有问题的：

[code28]

```
01. for /l %%i in (1,0,1) do echo bathome
```

[复制代码](#)

[code29]

```
01. for /l %%i in (2,1,1) do echo bathome
```

[复制代码](#)

[code30]

```
01. for /l %%i in (1,-1,2) do echo bathome
```

[复制代码](#)

其中，[code28]违反了步长不能为0的限制，将陷入无限循环中；[code29]和[code30]都犯了同样的错误：无法获得有效的数列元素，导致in和do之间取到的值为空元素，从而使得整条for语句无从执行。

当大家明白了 for /l 的具体功能之后，是否会想到了与它有异曲同工之妙的 goto循环语句呢？似乎，for /l 和 goto 循环语句可以相互替换？

一般而言，for /l语句可以换成goto循环，但是，goto循环并不一定能被 for /l 语句替换掉。具体原因，请大家仔细想想，我在此不再详细解说，只是就大家非常关心的一个问题提供一个简洁的答案，那就是：什么时候该用 for /l 计数循环，而什么时候又该用goto条件循环？

答案非常简单：当循环次数确定的时候，首选 for /I 语句，也可使用goto语句但不推荐；当循环次数不确定的时候，用goto语句将是唯一的选择，因为，这个时候需要用if之类的条件语句来判断何时结束goto跳转。

[本帖最后由 namejm 于 2010-12-25 02:02 编辑]

SIGNATURE

**尺有所短寸有所长，学好批处理没商量；
考虑问题复杂化，解决问题简洁化。
心在天山，身老沧州。**

TOP

namejm



荣誉版主



帖子 1348
积分 5096
技术 113
捐助 501
注册时间 2007-10-25

发表于 2008-10-16 21:12 | 只看该作者

9 楼

后记：

当Windows为我们打开了五彩缤纷的图形窗口的时候
DOS命中注定会陨落
CMD毫无悬念将萎缩
批处理逐渐趋向无声无息
而powershell的到来，无疑会让更多的人忘记批处理
这是一门即将失传的技艺
这是一块行将就木的领域
然而，命令行工具仍然具有批量处理一切的巨大威力
字符界面仍然是高效操作的代名词
曾为批处理的方便灵活而击节赞赏
曾被批处理的简洁快速深深折服
一直以来，总想为批处理的推广做些什么
于是，从在CN-DOS里尽职尽责地为大家解答疑问，到创办了自己的论坛专职
答疑解惑，再到无怨无悔地码字写教程，一步步走来，喜怒哀愁，五味杂陈
直至如今辞去站长等一切管理职务，逐渐淡出批处理圈子
梦依旧在，只是，心有余而力渐有不足
这篇从入门到精通的教学帖，从2008年10月开贴到现在，不知不觉拖拖拉拉
竟然过去了两年有余
每每看到跟帖的会员在问什么时候有更新
心中总有一丝愧疚
今天，终于抽空对它做个了断
只是，年年岁岁花相似，岁岁年年人不同
繁杂的事务使我已不再有当初的心境
for /I 部分总有虎头蛇尾的感觉
只能向各位说声抱歉了
在我彻底淡出批处理圈子之前
我只能尽我所能地向各位倾我所学了
最后，我希望论坛的管理人员能按照顶楼的管理提示经常为这个帖子抽抽水
或者是为了大家阅览的方便而永久锁定这个帖子

[本帖最后由 namejm 于 2010-12-25 02:00 编辑]

1



salad: namejm深入浅出地阐述了bat 技术 + 1

评分人数

SIGNATURE

尺有所短寸有所长，学好批处理没商量；
 考虑问题复杂化，解决问题简洁化。
 心在天山，身老沧州。

TOP

namejm

发表于 2008-10-16 21:12 | 只看该作者

10 楼



占位编辑

荣誉版主



帖子 1348
 积分 5096
 技术 113
 捐助 501
 注册时间 2007-10-25

SIGNATURE

尺有所短寸有所长，学好批处理没商量；
 考虑问题复杂化，解决问题简洁化。
 心在天山，身老沧州。

TOP

Batcher

发表于 2008-10-16 23:30 | 只看该作者

11 楼



期待能学到更多知识

管理员



帖子 8720
 积分 37590
 技术 218
 捐助 510
 注册时间 2008-6-9

SIGNATURE

【扫描二维码捐助论坛的朋友请留言注明论坛账号】 <http://bbs.bathome.net/thread-10403-1-1.html>
 【批处理在线视频分享】 <http://bbs.bathome.net/thread-31727-1-1.html>
 【微信公众号、微信群、QQ群】 <http://bbs.bathome.net/thread-3473-1-1.html>

TOP

newdosuser

发表于 2008-10-25 10:12 | 只看该作者

12 楼

怎么没了,真看得好呢

2018/7/9

[系列教程]批处理for语句从入门到精通[20101225更新] - BAT教程&资料 - 批处理之家 批处理_BAT_CMD_DOS_VBS_Perl_Python_Po...



二级士官



帖子 9
积分 192
技术 0
捐助 0
注册时间 2008-10-24

TOP

AK47

发表于 2008-10-25 22:34 | 只看该作者

13 楼



通俗易懂!非常谢谢楼主

少尉



帖子 104
积分 730
技术 1
捐助 0
注册时间 2008-9-15

SIGNATURE

不管黑猫白猫，会捉老鼠的猫就是好猫！

TOP

zhan5257

发表于 2008-10-26 06:30 | 只看该作者

14 楼



对新手来说真的是太好了。谢谢

五级士官



帖子 15
积分 334
技术 0
捐助 0
注册时间 2008-10-25

SIGNATURE

有些缘分是注定要失去的

TOP

loveruixue 发表于 2008-10-31 16:12 | 只看该作者

15 楼



二级士官



帖子 8

积分 198

技术 0

捐助 0

注册时间 2008-10-25

```
for %%a in ("1.txt" "2.txt" "3.txt") do echo >%%a
```

TOP

[◀ 返回列表](#) [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [... 35](#) [下一页 ▶](#)