bat 批处理教程

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
a.bat
```

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（为什么？？）。

echo 命令的另一种用法二：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\10.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。

有关某个命令的详细信息，请键入 HELP 命令名

ASSOC 显示或修改文件扩展名关联。

AT 计划在计算机上运行的命令和程序。

ATTRIB 显示或更改文件属性。

BREAK 设置或清除扩展式 CTRL+C 检查。

CACLS 显示或修改文件的访问控制列表(ACLs)。

CALL 从另一个批处理程序调用这一个。

CD 显示当前目录的名称或将其更改。

CHCP 显示或设置活动代码页数。

CHDIR 显示当前目录的名称或将其更改。

CHKDSK 检查磁盘并显示状态报告。

CHKNTFS 显示或修改启动时间磁盘检查。

CLS 清除屏幕。

CMD 打开另一个 Windows 命令解释程序窗口。

COLOR 设置默认控制台前景和背景颜色。

COMP 比较两个或两套文件的内容。

COMPACT 显示或更改 NTFS 分区上文件的压缩。

CONVERT 将 FAT 卷转换成 NTFS。您不能转换

当前驱动器。

COPY 将至少一个文件复制到另一个位置。

DATE 显示或设置日期。

DEL 删除至少一个文件。

DIR 显示一个目录中的文件和子目录。

DISKCOMP 比较两个软盘的内容。

DISKCOPY 将一个软盘的内容复制到另一个软盘。

DOSKEY 编辑命令行、调用 Windows 命令并创建宏。

ECHO 显示消息，或将命令回显打开或关上。

ENDLOCAL 结束批文件中环境更改的本地化。

ERASE 删除至少一个文件。

EXIT 退出 CMD.EXE 程序(命令解释程序)。

FC 比较两个或两套文件，并显示

不同处。

FIND 在文件中搜索文字字符串。

FINDSTR 在文件中搜索字符串。

FOR 为一套文件中的每个文件运行一个指定的命令。

FORMAT 格式化磁盘，以便跟 Windows 使用。

FTYPE 显示或修改用于文件扩展名关联的文件类型。

GOTO 将 Windows 命令解释程序指向批处理程序

中某个标明的行。

GRAFTABL 启用 Windows 来以图像模式显示

扩展字符集。

HELP 提供 Windows 命令的帮助信息。

IF 执行批处理程序中的条件性处理。

LABEL 创建、更改或删除磁盘的卷标。

MD 创建目录。

MKDIR 创建目录。

MODE 配置系统设备。

MORE 一次显示一个结果屏幕。

MOVE 将文件从一个目录移到另一个目录。

PATH 显示或设置可执行文件的搜索路径。

PAUSE 暂停批文件的处理并显示消息。

POPD 还原 PUSHD 保存的当前目录的上一个值。

PRINT 打印文本文件。

PROMPT 更改 Windows 命令提示符。

PUSHD 保存当前目录，然后对其进行更改。

RD 删除目录。

RECOVER 从有问题的磁盘恢复可读信息。

REM 记录批文件或 CONFIG.SYS 中的注释。

REN 重命名文件。

RENAME 重命名文件。

REPLACE 替换文件。

RMDIR 删除目录。

SET 显示、设置或删除 Windows 环境变量。

SETLOCAL 开始批文件中环境更改的本地化。

SHIFT 更换批文件中可替换参数的位置。

SORT 对输入进行分类。

START 启动另一个窗口来运行指定的程序或命令。

SUBST 将路径跟一个驱动器号关联。

TIME 显示或设置系统时间。

TITLE 设置 CMD.EXE 会话的窗口标题。

TREE 以图形模式显示驱动器或路径的目录结构。

TYPE 显示文本文件的内容。

VER 显示 Windows 版本。

VERIFY 告诉 Windows 是否验证文件是否已正确

写入磁盘。

VOL 显示磁盘卷标和序列号。

XCOPY 复制文件和目录树。

appwiz.cpl----------添加删除程序

control userpasswords2--------用户帐户设置

cleanmgr-------垃圾整理

CMD----------命令提示符可以当作是 Windows 的一个附件，Ping，Convert 这些不能在图形环境下 使用的功能要借助它来完成。

cmd------jview 察看 Java 虚拟机版本。

command.com------调用的则是系统内置的 NTVDM，一个 DOS 虚拟机。它完全是一个类似 Virtual PC 的 虚拟环境，和系统本身联系不大。当我们在命令提示符下运行 DOS 程序时，实际上也 是自动转移到 NTVDM 虚拟机下，和 CMD 本身没什么关系。

calc----------启动计算器

chkdsk.exe-----Chkdsk 磁盘检查

compmgmt.msc---计算机管理

conf----------启动 netmeeting

control userpasswords2-----User Account 权限设置

devmgmt.msc--- 设备管理器

diskmgmt.msc---磁盘管理实用程序

dfrg.msc-------磁盘碎片整理程序

drwtsn32------ 系统医生

dvdplay--------启动 Media Player

dxdiag----------DirectX Diagnostic Tool

gpedit.msc-------组策略编辑器

gpupdate /target:computer /force 强制刷新组策略

eventvwr.exe-----事件查看器

explorer-------打开资源管理器

logoff---------注销命令

lusrmgr.msc----本机用户和组

msinfo32---------系统信息

msconfig---------系统配置实用程序

net start (servicename)----启动该服务

net stop (servicename)-----停止该服务

notepad--------打开记事本

nusrmgr.cpl-------同 control userpasswords，打开用户帐户控制面板

Nslookup-------IP 地址侦测器

oobe/msoobe /a----检查 XP 是否激活

perfmon.msc----计算机性能监测程序

progman--------程序管理器

regedit----------注册表编辑器

regedt32-------注册表编辑器

regsvr32 /u *.dll----停止 dll 文件运行

route print------查看路由表

rononce -p ----15 秒关机

rsop.msc-------组策略结果集

rundll32.exe rundll32.exe %Systemroot%System32shimgvw.dll,ImageView_Fullscreen----启动一个空白的 Windows 图片和传真查看器

secpol.msc--------本地安全策略

services.msc---本地服务设置

sfc /scannow-----启动系统文件检查器

sndrec32-------录音机

taskmgr-----任务管理器（适用于 2000／xp／2003）

tsshutdn-------60 秒倒计时关机命令

winchat--------XP 自带局域网聊天

winmsd---------系统信息

winver-----显示 About Windows 窗口

wupdmgr----------Windows Update

.bat 是 dos 下的批处理文件

.cmd 是 nt 内核命令行环境的另一种批处理文件

批处理命令

批处理文件或批处理程序是一个包含若干 MS-DOS 命令的正文文件，扩展名为.BAT。当在命令提示符下敲入批处理程序的名称

时，MS-DOS 成组执行此批处理程序中的命令。

任何在命令提示符下可使用的命令都可用在批处理程序中。此外，下面 MS-DOS 命令是专门在批处理程序中使用的。

常用命令

echo、@、call、pause、rem(小技巧：用::代替 rem)是批处理文件最常用的几个命令，我们就从他们开始学起。

==== willsort 编注 ==========

首先, @ 不是一个命令, 而是 DOS 批处理的一个特殊标记符, 仅用于屏蔽命令行回显. 下面是 DOS 命令行或批处理中可能会见到

的一些特殊标记符:

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

"" 字符串界定符

| 命令管道符

< > >> 文件重定向符

@ 命令行回显屏蔽符

/ 参数开关引导符

: 批处理标签引导符

% 批处理变量引导符

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

第一, 除了 :: 之外, 任何以 :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, 只是为了与正常的标号相区

别, 建议使用 goto 所无法识别的标号, 即在 :后紧跟一个非字母数字的一个特殊符号.

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.

==========

echo 表示显示此命令后的字符

echo off 表示在此语句后所有运行的命令都不显示命令行本身

@与 echo off 相象，但它是加在每个命令行的最前面，表示运行时不显示这一行的命令行（只能影响当前行）。

call 调用另一个批处理文件（如果不用 call 而直接调用别的批处理文件，那么执行完那个批处理文件后将无法返回当前文件并执

行当前文件的后续命令）。

pause 运行此句会暂停批处理的执行并在屏幕上显示 Press any key to continue...的提示，等待用户按任意键后继续

rem 表示此命令后的字符为解释行（注释），不执行，只是给自己今后参考用的（相当于程序中的注释）。

此处的描述较为混乱, 不如直接引用个命令的命令行帮助更为条理

ECHO

当程序运行时，显示或隐藏批处理程序中的正文。也可用于允许或禁止命令的回显。

在运行批处理程序时，MS-DOS 一般在屏幕上显示（回显）批处理程序中的命令。

使用 ECHO 命令可关闭此功能。

语法

ECHO [ON|OFF]

若要用 echo 命令显示一条命令，可用下述语法：

echo [message]

参数

ON|OFF

指定是否允许命令的回显。若要显示当前的 ECHO 的设置，可使用不带参数的 ECHO 命令。

message

指定让 MS-DOS 在屏幕上显示的正文。

CALL

从一个批处理程序中调用另一个批处理程序，而不会引起第一个批处理的中止。

语法

CALL [drive:][path]filename [batch-parameters]

参数

[drive:][path]filename

指定要调用的批处理程序的名字及其存放处。文件名必须用.BAT 作扩展名。

batch-parameters

指定批处理程序所需的命令行信息。

PAUSE

暂停批处理程序的执行并显示一条消息，提示用户按任意键继续执行。只能在批处

理程序中使用该命令。

语法

PAUSE

REM

在批处理文件或 CONFIG.SYS 中加入注解。也可用 REM 命令来屏蔽命令（在 CONFIG.SYS

中也可以用分号(;)代替 REM 命令，但在批处理文件中则不能替代）。

语法

REM [string]

参数

string

指定要屏蔽的命令或要包含的注解。

例 1：用 edit 编辑 a.bat 文件，输入下列内容后存盘为 c:\a.bat，执行该批处理文件后可实现：将根目录中所有文件写入 a.txt 中，

启动 UCDOS，进入 WPS 等功能。

批处理文件的内容为:　　命令注释：

@echo off　　不显示后续命令行及当前命令行

dir c:*.* >a.txt　　将 c 盘文件列表写入 a.txt

call c:\ucdos\ucdos.bat　　调用 ucdos

echo 你好　　显示"你好"

pause　　暂停,等待按键继续

rem 准备运行 wps　　注释：准备运行 wps

cd ucdos　　进入 ucdos 目录

wps　　运行 wps

批处理文件的参数

批处理文件还可以像 C 语言的函数一样使用参数（相当于 DOS 命令的命令行参数），这需要用到一个参数表示符"%"。

%[1-9]表示参数，参数是指在运行批处理文件时在文件名后加的以空格（或者 Tab）分隔的字符串。变量可以从%0 到%9，%0 表

示批处理命令本身，其它参数字符串用%1 到%9 顺序表示。

例 2：C:根目录下有一批处理文件名为 f.bat，内容为：

@echo off

format %1

如果执行 C:\>f a:

那么在执行 f.bat 时，%1 就表示 a:，这样 format %1 就相当于 format a:，于是上面的命令运行时实际执行的是 format a:

例 3：C:根目录下一批处理文件名为 t.bat，内容为:

@echo off

type %1

type %2

那么运行 C:\>t a.txt b.txt

%1 : 表示 a.txt

%2 : 表示 b.txt

于是上面的命令将顺序地显示 a.txt 和 b.txt 文件的内容。

==== willsort 编注 ==========

参数在批处理中也作为变量处理, 所以同样使用百分号作为引导符, 其后跟 0-9 中的一个数字构成参数引用符. 引用符和参数之间

(例如上文中的 %1 与 a:) 的关系类似于变量指针与变量值的关系. 当我们要引用第十一个或更多个参数时, 就必须移动 DOS 的参

数起始指针. shift 命令正充当了这个移动指针的角色, 它将参数的起始指针移动到下一个参数, 类似 C 语言中的指针操作. 图示如

下:

初始状态, cmd 为命令名, 可以用 %0 引用

cmd arg1 arg2 arg3 arg4 arg5 arg6 arg7 arg8 arg9 arg10

^ ^ ^ ^ ^ ^ ^ ^ ^ ^

| | | | | | | | | |

%0 %1 %2 %3 %4 %5 %6 %7 %8 %9

经过 1 次 shift 后, cmd 将无法被引用

cmd arg1 arg2 arg3 arg4 arg5 arg6 arg7 arg8 arg9 arg10

^ ^ ^ ^ ^ ^ ^ ^ ^ ^

| | | | | | | | | |

%0 %1 %2 %3 %4 %5 %6 %7 %8 %9

经过 2 次 shift 后, arg1 也被废弃, %9 指向为空, 没有引用意义

cmd arg1 arg2 arg3 arg4 arg5 arg6 arg7 arg8 arg9 arg10

^ ^ ^ ^ ^ ^ ^ ^ ^

| | | | | | | | |

%0 %1 %2 %3 %4 %5 %6 %7 %8

遗憾的是, win9x 和 DOS 下均不支持 shift 的逆操作. 只有在 nt 内核命令行环境下, shift 才支持 /n 参数, 可以以第一参数为基准返

复移动起始指针.

==========

特殊命令

if goto choice for 是批处理文件中比较高级的命令，如果这几个你用得很熟练，你就是批处理文件的专家啦。

一、if 是条件语句，用来判断是否符合规定的条件，从而决定执行不同的命令。 有三种格式:

1、if [not] "参数" == "字符串" 待执行的命令

参数如果等于(not 表示不等，下同)指定的字符串，则条件成立，运行命令，否则运行下一句。

例：if "%1"=="a" format a:

==== willsort 编注

if 的命令行帮助中关于此点的描述为:

IF [NOT] string1==string2 command

在此有以下几点需要注意:

1. 包含字符串的双引号不是语法所必须的, 而只是习惯上使用的一种"防空"字符

2. string1 未必是参数, 它也可以是环境变量, 循环变量以及其他字符串常量或变量

3. command 不是语法所必须的, string2 后跟一个空格就可以构成一个有效的命令行

==========

2、if [not] exist [路径\]文件名 待执行的命令

如果有指定的文件，则条件成立，运行命令，否则运行下一句。

如: if exist c:\config.sys type c:\config.sys

表示如果存在 c:\config.sys 文件，则显示它的内容。

****** willsort 编注 ********

也可以使用以下的用法:

if exist command

device 是指 DOS 系统中已加载的设备, 在 win98 下通常有:

AUX, PRN, CON, NUL

COM1, COM2, COM3, COM4

LPT1, LPT2, LPT3, LPT4

XMSXXXX0, EMMXXXX0

A: B: C: ...,

CLOCK$, CONFIG$, DblBuff$, IFS$HLP$

具体的内容会因硬软件环境的不同而略有差异, 使用这些设备名称时, 需要保证以下三点:

1. 该设备确实存在(由软件虚拟的设备除外)

2. 该设备驱动程序已加载(aux, prn 等标准设备由系统缺省定义)

3. 该设备已准备好(主要是指 a: b: ..., com1..., lpt1...等)

可通过命令 mem/d | find "device" /i 来检阅你的系统中所加载的设备

另外, 在 DOS 系统中, 设备也被认为是一种特殊的文件, 而文件也可以称作字符设备; 因为设备(device)与文件都是使用句柄

(handle)来管理的, 句柄就是名字, 类似于文件名, 只不过句柄不是应用于磁盘管理, 而是应用于内存管理而已, 所谓设备加载也即

指在内存中为其分配可引用的句柄.

==========

3、if errorlevel <数字> 待执行的命令

很多 DOS 程序在运行结束后会返回一个数字值用来表示程序运行的结果(或者状态)，通过 if errorlevel 命令可以判断程序的返回值

，根据不同的返回值来决定执行不同的命令(返回值必须按照从大到小的顺序排列)。如果返回值等于指定的数字，则条件成立

，运行命令，否则运行下一句。

如 if errorlevel 2 goto x2

==== willsort 编注 ==========

返回值从大到小的顺序排列不是必须的, 而只是执行命令为 goto 时的习惯用法, 当使用 set 作为执行命令时, 通常会从小到大顺

序排列, 比如需将返回码置入环境变量, 就需使用以下的顺序形式:

if errorlevel 1 set el=1

if errorlevel 2 set el=2

if errorlevel 3 set el=3

if errorlevel 4 set el=4

if errorlevel 5 set el=5

...

当然, 也可以使用以下循环来替代, 原理是一致的:

for %%e in (1 2 3 4 5 6 7 8...) do if errorlevel %%e set el=%%e

更高效简洁的用法, 可以参考我写的另一篇关于获取 errorlevel 的文章

出现此种现象的原因是, if errorlevel 比较返回码的判断条件并非等于, 而是大于等于. 由于 goto 的跳转特性, 由小到大排序会导致

在较小的返回码处就跳出; 而由于 set 命令的 "重复" 赋值特性, 由大到小排序会导致较小的返回码 "覆盖" 较大的返回码.

另外, 虽然 if errorlevel=<数字> command 也是有效的命令行, 但也只是 command.com 解释命令行时将 = 作为命令行切分符而忽略

掉罢了

==========

二、goto 批处理文件运行到这里将跳到 goto 所指定的标号(标号即 label，标号用:后跟标准字符串来定义)处，goto 语句一般与 if 配

合使用，根据不同的条件来执行不同的命令组。

如:

goto end

:end

echo this is the end

标号用":字符串"来定义，标号所在行不被执行。

==== willsort 编注

label 常被译为 "标签"，但是这并不具有广泛的约定性.

goto 与 : 联用可实现执行中途的跳转, 再结合 if 可实现执行过程的条件分支, 多个 if 即可实现命令的分组, 类似 C 中 switch case

结构或者 Basic 中的 select case 结构, 大规模且结构化的命令分组即可实现高级语言中的函数功能. 以下是批处理和 C/Basic 在语

法结构上的对照:

Batch C / Basic

goto&: goto&:

goto&:&if if{}&else{} / if&elseif&endif

goto&:&if... switch&case / select case

goto&:&if&set&envar... function() / function(),sub()

==========

三、choice 使用此命令可以让用户输入一个字符（用于选择），从而根据用户的选择返回不同的 errorlevel，然后于 if errorlevel 配

合，根据用户的选择运行不同的命令。

注意：choice 命令为 DOS 或者 Windows 系统提供的外部命令，不同版本的 choice 命令语法会稍有不同，请用 choice /?查看用法。

choice 的命令语法（该语法为 Windows 2003 中 choice 命令的语法，其它版本的 choice 的命令语法与此大同小异）：

CHOICE [/C choices] [/N] [/CS] [/T timeout /D choice] [/M text]

描述:

该工具允许用户从选择列表选择一个项目并返回所选项目的索引。

参数列表:

/C choices 指定要创建的选项列表。默认列表是 "YN"。

/N 在提示符中隐藏选项列表。提示前面的消息得到显示，

选项依旧处于启用状态。

/CS 允许选择分大小写的选项。在默认情况下，这个工具

是不分大小写的。

/T timeout 做出默认选择之前，暂停的秒数。可接受的值是从 0

到 9999。如果指定了 0，就不会有暂停，默认选项

会得到选择。

/D choice 在 nnnn 秒之后指定默认选项。字符必须在用 /C 选

项指定的一组选择中; 同时，必须用 /T 指定 nnnn。

/M text 指定提示之前要显示的消息。如果没有指定，工具只

显示提示。

/? 显示帮助消息。

注意:

ERRORLEVEL 环境变量被设置为从选择集选择的键索引。列出的第一个选

择返回 1，第二个选择返回 2，等等。如果用户按的键不是有效的选择，

该工具会发出警告响声。如果该工具检测到错误状态，它会返回 255 的

ERRORLEVEL 值。如果用户按 Ctrl+Break 或 Ctrl+C 键，该工具会返回 0 的 ERRORLEVEL 值。在一个批程序中使用 ERRORLEVEL 参数时，将参数降序排列。

示例:

CHOICE /?

CHOICE /C YNC /M "确认请按 Y，否请按 N，或者取消请按 C。"

CHOICE /T 10 /C ync /CS /D y

CHOICE /C ab /M "选项 1 请选择 a，选项 2 请选择 b。"

CHOICE /C ab /N /M "选项 1 请选择 a，选项 2 请选择 b。"

==== willsort 编注 ==========

我列出 win98 下 choice 的用法帮助, 已资区分

Waits for the user to choose one of a set of choices.

等待用户选择一组待选字符中的一个

CHOICE [/C[:]choices] [/N] [/S] [/T[:]c,nn] [text]

/C[:]choices Specifies allowable keys. Default is YN

指定允许的按键(待选字符), 默认为 YN

/N Do not display choices and ? at end of prompt string.

不显示提示字符串中的问号和待选字符

/S Treat choice keys as case sensitive.

处理待选字符时大小写敏感

/T[:]c,nn Default choice to c after nn seconds

在 nn 秒后默认选择 c

text Prompt string to display

要显示的提示字符串

ERRORLEVEL is set to offset of key user presses in choices.

ERRORLEVEL 被设置为用户键入的字符在待选字符中的偏移值

==========

如果我运行命令：CHOICE /C YNC /M "确认请按 Y，否请按 N，或者取消请按 C。"

屏幕上会显示：

确认请按 Y，否请按 N，或者取消请按 C。 [Y,N,C]?

例：test.bat 的内容如下（注意，用 if errorlevel 判断返回值时，要按返回值从高到低排列）:

```
@echo off
choice /C dme /M "defrag,mem,end"
if errorlevel 3 goto end
if errorlevel 2 goto mem
if errorlevel 1 goto defrag
```

```
:defrag

c:\dos\defrag

goto end


:mem

mem

goto end


:end

echo good bye
```

此批处理运行后，将显示"defrag,mem,end[D,M,E]?" ，用户可选择 d m e ，然后 if 语句根据用户的选择作出判断，d 表示执行标号

为 defrag 的程序段，m 表示执行标号为 mem 的程序段，e 表示执行标号为 end 的程序段，每个程序段最后都以 goto end 将程序跳到

end 标号处，然后程序将显示 good bye，批处理运行结束。

四、for 循环命令，只要条件符合，它将多次执行同一命令。

语法：

对一组文件中的每一个文件执行某个特定命令。

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。

例如一个批处理文件中有一行:

for %%c in (*.bat *.txt) do type %%c

则该命令行会显示当前目录下所有以 bat 和 txt 为扩展名的文件的内容。

WINDOWS 批处理命令详解

批处理命令详解

最近好多猜测弱口令的病毒在网上流行，比如前段时间闹得很厉害的 Worm.Dvldr 蠕虫就是一个典型。这些病毒有个共同点就是利用批处理来进行 ipc$连接，从而来猜测管理员的口令达到控制服务器的目的。病毒由几个文件和几个复杂的批处 理组成。批处理算不上真正意义上的编程，但是它的一些思想和编程比较近似。通过在网上和一些初学的朋友交流，发现他们对于批处理很感兴趣，多多少少了解一 些命令的用法，但缺乏比较系统的了解，所以特意写下这篇教程，好让感兴趣的朋友对批处理有个整体的认识，并能通过该教程举一反三，写出自己的批处理。

该教程一共分为 4 大部分，第一部分是批处理的专用命令，第二部分是特殊的符号与批处理，第三部分是批处理与变量，第四部分是完整案例。因为教程比较长，所 有在杂志上我们将分为两次连载，本期首先刊登一、二两部分，敬请读者注意。

第一部分：批处理的专用命令

批处理文件是将一系列命令按一定的顺序****为一个可执行的文本文件，其扩展名为 BAT。这些命令统称批处理命令，下面我就来给大家介绍一下批处理的命 令。

1、 REM

REM 是个注释命令一般是用来给程序加上注解的，该命令后的内容在程序执行的时候将不会被显示和执行。例：

REM 你现在看到的就是注解，这一句将不会被执行。在以后的例子中解释的内容都 REM 会放在 REM 后面。请大家注意。

2、 ECHO

ECHO 是一个回显命令主要参数有 OFF 和 ON,一般用 ECHO message 来显示一个特定的消息 。例：

Echo off

Rem 以上代表关闭回显即不显示所执行的命令

Echo 这个就是消息。

Rem 以上代表显示“这就是消息”这列字符

执行结果：

C:\\>ECHO.BAT

这个就是消息。

3、 GOTO

GOTO 即为跳转的意思。在批处理中允许以“：XXX”来构建一个标号然后用 GOTO ：标号直接来执行标号后的命令。例

:LABEL

REM 上面就是名为 LABEL 的标号。

DIR C:\\

DIR D:\\

GOTO LABEL

REM 以上程序跳转标号 LABEL 处继续执行。

4、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。例：

批处理 2.BAT 内容如下：

ECHO 这就是 2 的内容

批处理 1.BAT 内容如下：

ECHO 这是 1 的内容

CALL 2.BAT

ECHO 1 和 2 的内容全部显示完成

执行结果如下：

C:\\>1.BAT

这是 1 的内容

这就是 2 的内容

1 和 2 的内容全部显示完成

5、PAUSE

PAUSE 停止系统命令的执行并显示下面的内容。例：

C:\\> PAUSE

请按任意键继续 . . .

6、 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

说明：

[NOT] 将返回的结果取反值即“如果没有”的意思。

ERRORLEVEL 是命令执行完成后返回的退出值

Number 退出值的数字取值范围 0~255。判断时值的排列顺序应该又大到小。返回的值大于或等于指定的值时条件成立。

string1==string2 string1 和 string2 都为字符的数据，英文字符的大小写将看做不同，这个条件中的等于号必须是 2 个（绝对相等），条件想等后即执行后面的 command

EXIST filename 为文件或目录存在的意思。

IF ERRORLEVEL 这条语句必须放在某一个命令后面。执行命令后由 IF ERRORLEVEL 来判断命令的返回值。

例：

1、 IF [NOT] ERRORLEVEL number command

检测命令执行完后的返回值做出判断。

```
echo off
dir z:
rem 如果退出代码为 1（不成功）就跳至标题 1 处执行
IF ERRORLEVEL 1 goto 1
rem 如果退出代码为 0（成功）就跳至标题 0 处执行
IF ERRORLEVEL 0 goto 0
: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

Rem 这里是程序的出口

2、 IF string1==string2 command

检测当前变量的值做出判断

ECHO OFF

IF %1==2 goto no

Echo 变量相等!

Goto exit

:no

echo 变量不相等

goto exit

:exit

大家可以这样看效果 C:\\>test.bat 数字

3、 IF [NOT] EXIST filename command

发现特定的文件做出判断

echo off

IF not EXIST autoexec.bat goto 1

echo 文件存在成功!

goto exit

:1

echo 文件不存在失败!

goto exit

:exit

这个批处理大家可以放在 c 盘和 d 盘分别执行看看效果。

7、 FOR

FOR 这个命令比较特殊是一个循环执行命令的命令，同时 FOR 的循环里面还可以套用 FOR 在进行循环。这篇我们介绍基本的用法就不做套用的循环了，后面再 来讲解套用的循环。在批处理中 FOR 的命令如下：

FOR [%%c] IN (set) DO [command] [arguments]

在命令行中命令如下：

FOR [%c] IN (set) DO [command] [arguments]

常用参数：

/L 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)。

/D 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。

/F 从指定的文件中读取数据作为变量

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

下面来看一个例子：

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

会分析 myfile.txt 中的每一行，忽略以分号打头的那些行，将每行中的第二个和第三个符号传递给 for 程序体；用逗号和/或空格定界符号。请注意，这个 for 程序体的语句引用 %i 来取得第二个符号，引用 %j 来取得第三个符号，引用 %k 来取得第三个符号后的所有剩余符号。对于带有空格的文件名，您需要用双引号将文件名括起来。为了用这种方式来使用双引号，您还需要使用 usebackq 选项，否则，双引号会被理解成是用作定义某个要分析的字符串的。

%i 专门在 for 语句中得到说明，%j 和 %k 是通过 tokens= 选项专门得到说明的。您可以通过 tokens= 一行指定最多 26 个符号，只要不试图说明一个高于字母 \'z\' 或\'Z\' 的变量。请记住，FOR 变量名分大小写，是通用的；而且，同时不能有 52 个以上都在使用中。

您还可以在相邻字符串上使用 FOR /F 分析逻辑；方法是，用单引号将括号之间的 filenameset 括起来。这样，该字符串会被当作一个文件中的一个单一输入行。最后，您可以用 FOR /F 命令来分析命令的输出。方法是，将括号之间的 filenameset 变成一个反括字符串。该字符串会被当作命令行，传递到一个子 CMD.EXE，其输出会被抓进内存，并被当作文件分析。因此，以下例子:

FOR /F "usebackq delims==" %i IN (`set`) DO @echo %i

会枚举当前环境中的环境变量名称。

以下列举一个简单的例子，他将说明参数/L 和没有参数的区别：

删除文件 1.TXT 2.TXT 3.TXT 4.TXT 5.TXT

例：

ECHO OFF

FOR /L %%F IN (1,1,5) DO DEL %%F.TXT

或

FOR %%F IN (1,2,3,4,5) DO DEL %%F.TXT

以上 2 条命令执行的结果都是一样的如下：

C:\\>DEL 1.TXT

C:\\>DEL 2.TXT

C:\\>DEL 3.TXT

C:\\>DEL 4.TXT

C:\\>DEL 5.TXT

8、 SETLOCAL

开始批处理文件中环境改动的本地化操作。在执行 SETLOCAL 之后所做的环境改动只限于批处理文件。要还原原先的设置，必须执行 ENDLOCAL。 达到批处理文件结尾时，对于该批处理文件的每个尚未执行的 SETLOCAL 命令，都会有一个隐含的 ENDLOCAL 被执行。例：

@ECHO OFF

SET PATH /*察看环境变量 PATH

PAUSE

SETLOCAL

SET PATH=E:\\TOOLS　　/*重新设置环境变量 PATH

SET PATH

PAUSE

ENDLOCAL

SET PATH

从上例我们可以看到环境变量 PATH 第 1 次被显示得时候是系统默认路径。被设置成了 E:\\TOOLS 后显示为 E:\\TOOLS 但当 ENDLOCAL 后我们可以看到他又被还原成了系统的默认路径。但这个设置只在该批处理运行的时 候有作用。当批处理运行完成后环境变量 PATH 将会还原。

9、　　SHIFT

SHIFT 命令可以让在命令上的的命令使用超过 10 个（%0~%9）以上的可替代参数例：

ECHO OFF

ECHO %1 %2 %3 %4 %5 %6 %7 %8 %9

SHIFT

ECHO %1 %2 %3 %4 %5 %6 %7 %8 %9

SHIFT

ECHO %1 %2 %3 %4 %5 %6 %7 %8 %9

执行结果如下：

C::\\>SHIFT.BAT 1 2 3 4 5 6 7 8 9 10 11

1 2 3 4 5 6 7 8 9

2 3 4 5 6 7 8 9 10

3 4 5 6 7 8 9 10 11

以上就是基于 WIN2000 下的 9 个批处理命令。

第二部分：特殊的符号与批处理

在命令行下有些符号是不允许使用的但有些符号却有着特殊的意义。

1、 符号(@)

@在批处理中的意思是关闭当前行的回显。我们从上面知道用命令 echo off 可以关掉整个批处理的命令回显但却不能不显示 echo off 这个命令。现在我们在这个命令前加上@这样 echo off 这一命令就被@关闭了回显从而达到所有命令均不回显得要求

2、 符号(>)

>的意思是传递并覆盖。他所起的作用是将运行后的回显结果传递到后面的范围（后面可是文件也可是默认的系统控制台）例：

文件 1.txt 的文件内容为：

1+1

使用命令 c:\\>dir *.txt >1.txt

这时候 1.txt 的内容如下

驱动器 C 中的卷没有标签。

卷的序列号是 301A-1508

C:\\ 的目录

2003-03-11 14:04 1,005 FRUNLOG.TXT

2003-04-04 16:38 18,598,494 log.txt

2003-04-04 17:02 5 1.txt

2003-03-12 11:43 0 aierrorlog.txt

2003-03-30 00:35 30,571 202.108.txt

5 个文件 18,630,070 字节

0 个目录 1,191,542,784 可用字节

>将命令执行的结果覆盖了原始的文件内容。

在传递给控制台的时候程序将不会有任何回显（注意：这里的回显跟 echo off 关掉的回显不是同一概念。Echo off 关掉的是输入命令的回显，这里的回显是程序执行中或后的回显）例：

C:\\>dir *.txt >nul

程序将没有任何显示也不会产生任何痕迹。

3、 符号(>>)

符号>>的作用与符号>相似，但他们的区别在于>>是传递并在文件末尾追加>>也可将回显传递给控制台（用法 同上）例：

文件 1.txt 内同为：

1+1

使用命令 c:\\>dir *.txt >>1.txt

这时候 1.txt 的内容如下

1+1

驱动器 C 中的卷没有标签。

卷的序列号是 301A-1508

C:\\ 的目录

2003-03-11 14:04 1,005 FRUNLOG.TXT

2003-04-04 16:38 18,598,494 log.txt

2003-04-04 17:02 5 1.txt

2003-03-12 11:43 0 aierrorlog.txt

2003-03-30 00:35 30,571 202.108.txt

5 个文件 18,630,070 字节

0 个目录 1,191,542,784 可用字节

>>将命令执行的结果覆加在了原始的文件内容后面。

4、 符号(|)

|是一个管道传输命令意思是将上一命令执行的结果传递给下一命令去处理。例：

C:\\>dir c:\\|find "1508"

卷的序列号是 301A-1508

以上命令的意思为查找 c:\\的所有并发现 1508 字符串。Find 的用法请用 find /?自行查看

在不使用 format 的自动格式化参数的时候我是这样来自动格式化盘片的

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

用过 format 命令的人都知道 format 有一个交互对化过程，要使用者输入 y 来确定当前的命令是否被执行。在这个命令前加上 echo y 并用管道传输符|将 echo 执行的结果 y 传递给 format 从而达到手工输入 y 的目的（这条命令有危害性，测试的时候请谨慎）

5、 符号(^)

^ 是对特殊符号 > 、<、 &、的前导字符。在命令中他将以上的 3 个符号的特殊动能去掉仅仅只吧他们当成符号而不使用他们的特殊意义。例：

c:\\>echo test ^> 1.txt

test > 1.txt

从上面可以看出并没有把 test 写入文件 1.txt 而是将 test >1.txt 当字符串显示了出来。这个符号在远程构建批处理的时候很有效果。

6、 符号(&)

&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将不影响第 2 个命令的执行。例：

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

以上的命令将会连续显示 z: y: c:盘内的内容不理会该盘符是否存在。

7、 符号(&&)

&&符号也是允许在一行中使用 2 个以上不同的命令，当第一个命令执行失败后后续的命令将不会再被执行。例：

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

以上的命令将会提示检查是否存在 z:盘如果存在则执行，如果不存在则停止执行所有的后续命令

8、 符号(" ")

" "符号允许在字符串中包含空格。进入一个特殊的目录可以用如下方法例:

c:\\>cd “Program Files”

c:\\>cd progra~1

c:\\>cd pro*

以上方法都可以进入 Program Files 目录

9、 符号（,）

,符号相当于空格。在某些特殊的情况下可以用,来代替空格使用。例：

c:\\>dir,c:\\

10、 符号(;)

;符号当命令相同的时候可以将不同的目标用;隔离开来但执行效果不变。如执行过程中发生错误则只返回错误报告但程序还是会继续执行。例：

DIR C:\\;D:\\;E:\\F:\\

以上的命令相当于

DIR C:\\

DIR D:\\

DIR E:\\

DIR F:\\

当然还有些特殊的符号但他们的使用范围很小我就不再这里一一的说明了。

第三部分：批处理与变量

在批处理中适当的引用变量将会使你所编制的程序应用面更广。批处理每次能处理的变量从%0~%9 共 10 个。其中%0 默认给批处理的文件名使用。除非在使用 SHIFT 命令后%0 才能被%1 所替代。引用 shift 命令的例子如果把%1 前面多加上一个%0 那么结果如下：

C::\\>SHIFT.BAT 1 2 3 4 5 6 7 8 9 10 11

SHIFT.BAT 1 2 3 4 5 6 7 8 9

1 2 3 4 5 6 7 8 9 10

2 3 4 5 6 7 8 9 10 11

系统是如何区分每个变量的呢，系统区分变量的规则为字符串中间的空格，即只要发现空格就把空格前面的字符当作一个变量而空格后面的字符则作为另一个变量。 如果你的变量是一个当中包含空格的长目录名这时候你需要用上一节特殊符号 8 中所用的引号将他圈起来。例：

批处理内容为：

ECHO %1

ECHO %2

ECHO %3

输入命令：

C:\\>TEST “Program Files” Program Files

Program Files

Program

Files

在一个复杂的批处理中又可能同时使用的变量会超过 10 个这时候会和系统的规则想****那么这个问题怎么解决呢？在系统中还有一种变量称之为环境变量（使 用 SET 命令可以查看当前系统的环境变量）如当前系统目录是%windir%或%SystemRoot%等。当同时使用的参数超过 10 个的时候，我们可以 把某些在后面的程序中还要调用的变量保存为环境变量。具体用法如 SET A=%1 这样我们就命名了一个新的环境变量 A 在调用变量 A 的时候要%A%这样调用，环境变量不受 SHIFT 命令影响。如果要改变一个环境变量需要重新对其设置才能改变。当然也可以进行变量与变量之间 的传递来达到目的。下面我们来看一个例子，批处理如下：

ECHO OFF

SET PASS=%1

SHIFT

SET PASS1=%1

SHIFT

ECHO %PASS% %PASS1% %1 %2 %3 %4 %5 %6 %7 %8 %9

SHIFT

ECHO %PASS% %PASS1% %9

SET PASS=%PASS1%　　变量的传递

SET PASS1=%9

SHIFT

ECHO %PASS% %PASS1% %9

使用命令：C:\\>TEST A B 3 4 5 6 7 8 9 10 K L

A B 3 4 5 6 7 8 9 10 K 注意：这一行显示了 11 个变量

A B L 在使用了 3 次 SHIFT 之后%9 变成了 L

B L 变量的传递后的结果