目录

前言

目前网上批处理教程良莠不齐且不够系统，让批处理爱好者学习难度很大。本教程系统整理了批处理知识，讲解了绝大部分的 **DOS** 命令，程序力求最简，让从没有接触过 **DOS** 命令的朋友也可以看懂。另外，在每个命令后面有高级技巧篇，讲解命令的特殊（旁门左道）用法。

批处理定义

批处理**(Batch)**，也称为批处理脚本，具有**.bat** 或者**.cmd** 的扩展名。顾名思义，批处理就是对某对象进行批量的处理，属于脚本语言的一种。

DOS 批处理是基于 **DOS** 命令的，用来自动地批量执行 **DOS** 命令，以实现特定操作的脚本。批处理是一种简化的脚本语言，它应用于 **DOS** 和 **Windows** 系统中，它是由 **DOS** 或者 **Windows** 系统内嵌的命令解释器**(**通常是 **Command.com** 或 **CMD.exe)**解释运行，类似于 **Unix** 中的 **Shell** 脚本。

简单的批处理，是逐行书写在命令行中的各种命令；复杂点的，需要用 **if**，**for**，**goto**，**call** 等命令控制程序的运行过程，如同 **C**，**Basic** 等中高级语言一样；如果需要实现更复杂的应用，则要借助外部程序，包括系统本身提供的外部命令和第三方提供的工具或者软件。

为什么要学习 **DOS** 批处理？

1、与计算机语言相比，批处理不需要编译器，编写、执行代码非常方便。

2、批处理可以完成很多直接使用 **Windows** 操作很麻烦的工作。例如清理系统垃圾，假如没有安装任何软件，每次清理都需要一个个手工清理，估计没几个人有耐心做到，用批处理就非常便捷了。大批量删除、修改、搜索文件，使用批处理也是最佳选择。再比如我们可以将一个个复杂的 **Windows** 操作制作成一个个批处理文件，需要时只需要双击运行就行了，大大减轻我们的记忆负担。例如隐藏文件，如果是在 **Windows** 下操作，则首先打开 **regedit** 注册表编辑器，然后按照路径

HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced\Folder\Hidden\SHOWALL
一级级展开，找到 **CheckedValue** 键，将键值改为 0 就实现彻底隐藏文件，将键值改为 1 就恢复显示文件。如果我们经常需要隐藏和显示文件，每次这么操作就异常繁琐，而且对记忆也是不小的负担。我们可以编辑批处理程序：
隐藏代码：
```
reg add "HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced\Folder\Hidden\SHOWALL" /v CheckedValue /t REG_DWORD /d 0 /f
```
显示代码：
```
reg add "HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced\Folder\Hidden\SHOWALL" /v CheckedValue /t REG_DWORD /d 1 /f
```
分别保存为两个 **bat** 文件，需要哪个就双击哪个就行了，这样就不必记忆大量路径。当然，我们也可以将这两个功能合并到一个 **bat** 文件里，这样就更加方便了。

3、在某些方面，使用批处理编程比使用高级计算机语言编程简洁得多。
例如将资料隐藏在图片的代码：
```
copy /b photo1.jpg+secret.rar=photo2.jpg
```
简简单单一条代码就可以将 **photo1.jpg** 和 **secret.rar** 合并为一张内涵图**(**具体操作见 **copy** 命令专题**)**，如果用 **C** 语言实现就复杂多了。

4、批处理还可以做 **Windows** 中不能做的事情。
例如在 **WINDOWS** 下，系统只提供了文件隐藏和只读属性，只有 **DOS** 才能把文件的属性改成系统文件属性。
电脑维护方面，**DOS** 批处理是个利器，例如我们的 **C:\Program Files\Internet Explorer**，**Internet Explorer** 文件夹被伪装成了回收站，这个用在 **Windows** 下无法解决**(**重装 **IE** 也不行，重装系统另当别论**)**，用很多软件也无法解决，而用 **DOS** 批处理命令就很简单了，简简单单几行代码就行：
```
C:
cd\
cd    program files\internet explorer
attrib desktop.ini -s -h
del desktop.ini
```
将上面代码保存成批处理文件运行就行了。

5、计算机安全的攻防，**DOS** 批处理是个不能避开的工具，像 **ping ,ipconfig /all,net,telnet** 等命令，在网络中的应用非常得力，无论是测试还是攻击！黑客多少都懂一些 **DOS** 命令。

结论：想对计算机操作和维护更加便利、想充分使用计算机的更多功能，操作一些 **Windows** 下无法实现的操作、想成为计算机高手、想成为黑客或者防范黑客等，就必须学习 **DOS** 批处理！
批处理也有劣势，毕竟它不能算一门计算机语言，有些方面用批处理无法实现或者实现起来太复杂。在执行方面，批处理的速度远远比不上 **C** 语言编译出来的 **exe** 文件运行速度，这个在对数字的计算方面尤为明显。

学习批处理究竟难不难？

学习批处理，几乎等于学习 **DOS** 命令。早期的计算机使用的操作系统基本都是 **DOS** 系统，一般用户学习批处理很容易。后来的计算机使用的操作系统基本都是 **Windows** 操作系统，几乎不用输入什么命令，大多数人对 **DOS** 命令一窍不通，学习起来会困难些。

编译器

学过计算机高级语言的人都知道，程序代码写好后，还需要编译器来编译后才能运行。批处理代码不需要编译器，最常见的方式是将代码直接写在记事本上，然后命名为**.bat** 或**.cmd** 格式文件就可以双击运行了。如果程序非常简短，也可以在命令行窗口直接写代码运行，调出命令行窗口的方法：“开始-->运行，输入 **cmd** 调出命令行窗口”。在 **cmd** 窗口输入 **“edit”**，也可以调出代码编辑器，不过似乎不太好用。网上也有一些可以用于编辑批处理的程序。

反编译：

批处理程序很容易被反编译，只需修改文件后缀名就可以知道程序代码，这是个严重的缺点。网上很多软件可以将 **bat** 文件转换成为 **exe** 文件防止反编译，可是好多编译好后就被杀毒软件杀掉了，原因是网上的这些软件无非是进行打包加壳后变成的 **EXE**，一旦执行，必然会释放出原始文件，而且加壳后很多杀毒软件会很敏感，报毒查杀。不过有一个软件可以做到真正将 **bat** 转换为 **exe**，还可以加图标，附加程序等，那就是 **QuickBFC**。

预备知识

DOS 一些常识

1、DOS 程序运行完后都有返回码，可以帮助我们诊断程序运行情况。
2、编辑批处理命令，所有字符必须在英文格式和半角状态下编辑。
3、按下 **Ctrl+C** 组合键可以强行终止批处理运行。
4、批处理认行不认命令数目。这是说批处理对断行很敏感，而对于一行有多少命令倒是无所谓，只要用**&**、**&&**、|、||等连接好就行了。
5、“-”和“/”是等价的。例如“**shutdown /s**”等价于“**shutdown -s**”。

DOS 操作系统名字命名规则：

DOS 下的文件名一般都是 8.3 格式，表示文件名为 8 个字符，扩展名为 3 个字符。文件名不能使用以下字符：< > ? / \ " . * : | 和空格，扩展名决定文件类型。当文件名超过 8 个字符时，就出现了 **Tempor~1** 这种形式的名字，仍然是 8 个字符。这个命名规则延续到了 **Windows** 操作系统，但 **Windows** 操作系统对命名限制放宽了，允许文件名超过 8 个字符，扩展名虽然大多都是 3 个字符，但也可以是 4 个字符甚至更多，例如 **Word**2007 的扩展名为**.docx**。对于点“.”，可以出现在文件名里，计算机以最后一个“.”后面的字符作为该文件的扩展名。名字也允许有空格，例如 **C:\program files** 这个文件夹的名字就存在空格。但其余的符号< > ? / \ " * : |这九个符号仍旧不允许出现在文件名中。

以 **C** 盘的应用程序文件夹 **program files** 的名字为例：

program files	全名文件名
progra~1	短名文件名
pro*	含有通配符的文件名

说明：

dos 文件名模式下的缩写长于 8 个字符的文件名和文件夹名，都被简化成前面 6 个有效字符，后面~1，如果有重名的文件就用~2,~3,..... 来区分，这就是短名。而含有统配符的文件名，就更加简化，假如我们当前 **C** 盘没有以 **P** 开头名字的文件夹，那么我们可以命名为 **p***，这样也可以打开文件夹。假如，我们的 **C** 盘里有“**program files**”文件夹和“**progra**”文件夹，那么我们输入 **C:\>cd pro***，会进入哪一个文件夹呢？答案是进入了名字排名靠前的“**progra**”文件夹。

查看命令帮助信息的三种方法：

要学批处理，离不开 **DOS** 命令，**DOS** 的命令太多，各条命令的用法规则就更多了，对记忆是个不小的负担，所以，**DOS** 每一条命令都会有帮助信息。

方法一：使用 **help** 命令查看相关的帮助信息。 例如 **help dir**，　　这种方法有时候是不管用的。
方法二：使用 **x** 查看相关的帮助信息。 例如 **shutdown x**　　这种方法有时候也不管用。
方法三：使用**/?**查看相关的帮助信息。 例如 **for /?**　　这种方法最常用。

如果我们要将调出的帮助信息存储到文件里，方便随时查看，可以这么操作（以 **shutdown** 为例）：

shutdown /?> shutdown.txt

将上面代码保存到记事本后，改为 **bat** 格式，双击运行就行了，我们就可以在当前目录下得到一个名为《**shutdown.txt**》的 **shutdown** 命令帮助文件。

再比如，要获得 **DOS** 帮助信息：

help /?>help.txt

注意：不要过度迷信 **DOS** 下的帮助信息，命令的帮助信息往往很难看懂或者有一小部分是错的。

例如 **FOR** 命令的帮助信息，初学者阅读起来简直就是在看天书。又例如我们查看**“set”**命令的帮助命令，就会发现它没有把**“**按位‘与’**”**的符号列出来。所以，一切都以实际操作通过为准。

命令语法格式解析

我们调出的 **DOS** 命令帮助信息，都采用标准的语法格式写出来，下面详细解释：

一、在 **DOS** 中，命令使用格式的一般形式

用中文表达的形式为：

[路径**]**关键字**[**盘符**][**路径**]**文件名**[**扩展名**](**参数**)[**参数**]**

用符号表达的形式为：

[Path]关键字**[D:][Path]filename(,ext)[pam]**

关键字
在命令中，关键字是必不可少的。
例如 **DIR** 命令中的“**DIR**”必须有，但其他项可有可无，根据需要选定。
盘符和路径
表示驱动器号，它可以是 **A:** ，**B:**， **C:**， **D:**， **E:**等等。如果省略了这个参数，就选择默认的驱动器。
路径
用\分隔。
盘符和路径连在一起，组成完全的路径。在命令格式中分开来写，是因为很多时候，只使用盘符而没有子目录。
[路径**]** 项放在关键字前面时，是用来指出外部命令文件所在的路径。当然，如果此外部命令文件就在当前路径中（即在当前子目录下），或者在由 **PATH** 命令设置的其他默认目录中，这个**[**路径**]** 项就可以省略。

二、在 **DOS** 中表示命令用法的标准语法格式。
下面以 **shutdown** 关机命令为例详细说明：
shutdown 标准语法格式**:**
shutdown [/i | /l | /s | /r | /a | /p | /h | /e] [/f] [/m\\ComputerName] [/txxx]　[/d [p|u:]xx:yy [/c "comment"]]
也有人喜欢这么写：
shutdown [-i | -l | -s | -r | -a | -p | -h | -e] [-f] [-m \\ComputerName] [-t XXX] [-d [p:]　XX:YY-c"Comment"]
两种写法，除了“-”变成“/”之外，其余都是相同的。也就是说，批处理里，“-”和“/”是等价的，两种书写都可以。
说明：
　　1、中括号**[]**里的内容是可选项目，可选可不选。
　　2、中括号里面的内容用“|”分隔，表示可选择的几个项不可以同时选定，只可以选择其中一项或者全都不选。例如：上面第一个中括号里有“**/s**”和“**/a**”，如果这样写代码：“**shutdown /s /a**”，执行无效，计算机不知道我们究竟是要关机还是取消关机。又如：**ECHO [ON | OFF]**中的“**ON | OFF**”表示可以输入 **ON**，也可以输入 **OFF**，根据需要来选定，但就不能既 **ON** 又 **OFF** 互相矛盾。
　　3、尖括号**<>**用来表明可选参数下的子参数。
　　4、大括号**{}**表示其中的项必须选一项。
　　5、省略号**...**指可以输入重复的项，知道需要的数目。例如 **path** 的语法格式为：**PATH [[drive:]path[;...][;%PATH%]**，表示关键字“**path**”后面可带上多个路径，具体多少完全由用户决定。

6、中括号[]、分隔符和省略号只在说明命令格式时使用，在实际使用命令中是不出现的。例如：**shutdown [/s] [/t** 3600]是无效代码。应该写成 **shutdown /s /t** 3600。除了这三种符号在实际代码中不出现，其余在语法格式中的符号在实际使用中都要输入。包括：逗号、分号、等号、问号、冒号、引号、斜杆和反斜杠等，另外，连空格都不能忘记！

7、参数项，每个参数一般由一个斜杠加一个字母组成，对命令起辅助作用。有的命令可以选用多个参数项。

8、中括号与中括号之间的命令没有次序之分。例如：

shutdown /s /t 3600 **/c** "计算机看你不顺眼，准备关机"

shutdown /c "计算机看你不顺眼，准备关机" **/t** 3600 **/s**

两条代码是等价的，执行后，计算机会在一个小时后关机，并且显示“计算机看你不顺眼，准备关机”字样。

特殊字符的输入

开始->运行->输入 cmd 回车　(打开 cmd 窗口)

输入 edit 打开编辑窗口

ctrl+p　　(意思是允许输入特殊字符)

按 ctrl+a 将会出现笑脸图案。每次需要输入特殊字符时，都先按 ctrl+p 然后再 ctrl+字母输入。

温馨提示

本《批处理标准教程》文件推荐用“Adobe Acrobat 9 Pro”软件打开阅读，用别的 PDF 阅读器，在文件里复制的代码可能无法正确运行。

另外，复制下来的代码没有了缩进，也没有空格，致使好多代码运行结果格式效果不对。所以建议读者们将文中的代码手动打出来并运行。

第一章　基础篇

第一个批处理程序：

```
@echo off
echo Hello World!
pause
```

将上面三条指令保存到记事本(txt 格式文件)里，然后将后缀名重命名为**.bat** 格式或**.cmd** 格式，双击运行，在屏幕上就会显示：

```
Hello World!
请按任意键继续...
```

@

关闭当前命令行回显

放在命令前，它的作用是让执行窗口中，无论 **echo** 是否为打开状态，都不显示它后面这一行的命令本身。

---

echo

回显命令

语法：

```
ECHO [ON | OFF]
ECHO [message]
```

参数说明：

echo on

打开回显

一般系统默认都是 **echo on**，所以这个代码不常用，除了执行过 **echo off** 后，想再次回显才会用到 **echo on**。

echo off
关闭回显
执行 **echo off** 可以关闭掉后面所有批处理命令的回显，只显示执行后的结果，除非再次执行 **echo on** 命令。但 **echo off** 无法关掉 **echo off** 这个命令本身。我们可在其前面添加**@**，就可以达到所有命令都不显示的目的，直到执行 **echo on** 为止。所以，我们一般都用**@echo off** 作为批处理的行首，取消所有的批处理命令回显，可以说，**@echo off** 放在程序首行，已经成为批处理程序的标志。有的人习惯不以**@echo off** 作为行首，而改为在每行命令前加“**@**”，这个太繁琐，应该用**@echo off** 简洁。

echo+空格
查询当前计算机的回显状态，也就是看看当前回显是打开的还是关闭的。

echo+信息
显示信息
注意，**echo** 后必须紧跟一个空格或特殊字符，以区分 **echo** 命令和信息，该空格或特殊字符不会作为信息被显示出来。
例如：
echo Hello World!
echo.Hello World!
echo/Hello World!
三个都是等效的。

echo.
显示一个空行，相当于一个回车。
注意：“**echo**”与“**.**”之间不能有空格，否则“**.**”将被当做提示信息输出到屏幕。另外“**.**”可以用 **, : ; / [\ ] + (=** 等任意一个符号替代，不过几乎所有人都只用“**echo.**”这一种形式。

echo 文件内容**>**文件名
将文件内容输出到指定文件中。如果指定的文件中原有别的信息，原先的信息将被清空。
echo 文件内容**>>**文件名
将文件内容追加到指定文件中。如果指定的文件中原有别的信息，原先的信息不会被清空。

程序例子：
echo Hello World!>ABC.txt
这个没有指定 **ABC.txt** 文件的路径，那么执行后，会在当前目录下生成一个名为 **ABC.txt** 的文件，里面的内容为：
Hello World!
注意格式细节：**echo** 是关键字，后面必须紧跟一个空格。
内容 **Hello World!**后面没有有空格，那么生成的“**Hello World!**”末端也不会有空格，并且光标到下一行了，等于输出“**Hello World!**”后并输出一个回车。假如 **Hello World!**后面加多一个空格，那么这个空格也会被输出到文件里，变为“**Hello World! **”，同样，光标也移到下一行了，等效于输出“**Hello World! **”后再输出一个回车。
>符号后面也不必有空格。有无空格效果一样。

程序程序例子：
echo Hello World!>D:\ABC.txt
echo Hello World!>D:ABC.txt
这两种都是指定输出文件路径的格式，两种写法都可以在 **D** 盘根目录里生成 **ABC.txt**，但第二种不符合我们一般的路径书写规范，所以不建议采用。
对于**>>**追加形式的输出到文件里的格式，与**>**一样。

---

echo 输出高级技巧

一：输出字符：
情况 1：输出“**+ - * / \ ( ) ? ; . , " ' : :: ~ @ `**”
@echo off
echo + - * / \ ( ) ? ; . , " ' : :: ~ @ `
pause
输出“**+ - * / \ ( ) ? ; . , " ' : :: ~ @ `**”这些字符直接 **echo** 输出即可。

情况 2：输出“**^ > >> & && | ||**”
echo ^^
echo ^>
echo ^>^>
echo ^&
echo ^&^&
echo ^|
echo ^|^|
pause

输出“^ > >> & && | ||”这些字符，就需要转义字符^的帮助。
有人说不用转义字符^也可以：**echo ">>"**，这个是可以，但多输出了双引号，不能令人满意。不用转义字符的输出应该是这样的：

```
@echo off
set var="||"
for %%i in (%var%) do echo %%~i
pause
```

不过，这样一来，代码复杂了许多，所以还是借助转义字符输出最直接。

情况 3：输出%

```
@echo off
echo %%%%%%%
pause
```

输出为

```
%%%
```

为什么呢？原来，用 **echo** 输出**%**，输出的个数是代码个数除以 2，另外，由于批处理不支持浮点数，那么像 0.5 个就直接舍弃。
于是像 **echo %**，没有输出什么内容。**echo %%** 和 **echo %%%**效果一样，都是输出一个**%**。这个无论是否延迟变量，效果都一样。

情况 4：输出!
在没有延迟变量的情况下，非常好办。

```
@echo off
echo !
echo !!
echo !!!
pause
```

分别输出 1、2、3 个感叹号。
如果有延迟变量，那么情况就不一样了，因为感叹号的含义发生了转变！它是变量标识符。输出一个感叹号应该这么输出：

```
@echo off
setlocal enabledelayedexpansion
echo ^^!
pause
```

可以理解为第一次预处理时，!只是一个普通字符，第一个^转义第二个^。有第二次预处理是因为开启了延迟，会把第一次预处理留下的^当作转义字符，用来转义!。简单来说就是，第一次预处理，!只是普通字符，第二次预处理，!变成了特殊字符。

二：输出大量数据

如果我们输出大量的数据，用 **echo** 命令每行只能输出一个数据，这样会浪费大量的显示界面，不符合我们的习惯。如何将大量数据在同一行输出呢？这个用 **echo** 命令无法实现，要用 **set** 实现，具体实现办法参照 **set** 命令。

三：用 **echo** 来作为参数：

```
@echo off
set /p var=
echo %var%|findstr /be [0-9.]* >nul && echo Yes||echo No
pause
```

这里的 **echo %var%**不用于显示功能，而用作参数功能。因为纯粹的**%var%**不可以被执行，我们又找不到合适的命令来作为**%var%**的开头，所以选用 **echo**。后面的“**>nul**”，是因为我们只需要参数功能就可以了，不需要其显示功能，所以用“**>nul**”阻止信息显示。其余的一些需要带参数的命令，也可以参考这个用法。

四：用 **echo** 发声

```
echo ^G
```

使主板喇叭发生一次鸣响，需要多次鸣响，可以多输入几个**^G**。有些主板不支持声响，执行 **echo ^G** 后没有任何效果。需要说明的是，“**^G**”不是两个字符，而是一个符号，对应的十进制 **ASCII** 码是 7，在 **DOS** 窗口中用 **Ctrl+G** 或者 **Alt+7** 输入（7 为小键盘上的 7）。那么，在批处理文件里如何写这个符号呢？我们可以从命令行里利用组合键输入这个命令，然后输出到文件里，我们就可以调用或者复制这个蜂鸣符号了。 例如在命令行窗口执行 **echo ^G > D:\test.txt**，我们就会在 **D** 盘新建的 **test** 文件里看到一个不可显示的符号，将这个 **txt** 文件改为 **bat** 文件后执行会响两次声，也就每个符号会响两次。为什么呢？原来第一次蜂鸣，是因为计算机要回显本符号，可本符号无法显示，于是计算机给出执行错误的警报。第二次响声是执行本符号的效果，也就是发出一声蜂鸣。于是，我们如果只想发出一声鸣响，阻止本符号回显就行了：

```
@echo •
```

这是只响一次的代码，在 **echo** 前加**@**的目的就是阻止**^G** 回显。当然，不加**@**而改为在程序头添加**@echo off** 或 **echo off** 都是可以的。

rem 和::

两个都可以用于批处理注解。

REM

在批处理文件或 **CONFIG .SYS** 里加上注解或说明。

语法：**REM [comment]**

注解批处理时，注解的标准写法是写在被注解程序的上一行。

例如：

::设置颜色

color 9f

REM 和**::**比较

1 相同之处

两个都是注释命令，在批处理脚本中**::**和 **rem** 命令等效，它们后面的内容在执行时不显示，也不起任何作用，只是增加了脚本的可读性。

2 不同之处

当关闭回显时，**rem** 和**::**后的内容都不会显示，但是当打开回显时，**rem** 后的内容会显示出来，然而**::**后的内容仍然不会显示。

由于我们在执行程序时，一般都不需要将程序注释显示出来，所以一般情况下我们都采用**::**注释，这样当打开 **echo on** 后，注释内容仍然不会显示啦。

实际上，批处理中可以用于注释的命令只有 **rem**，**::**不算是命令。之所以**::**也可以用于注释，是利用了批处理的符号规则：

任何以冒号开头的行，在批处理中都被视为标号而直接忽略其后的所有内容。

有效标号：冒号后跟一个以字母或数字开头的字符串，goto 和 call 语句可以识别。
无效标号：冒号后紧跟一个特殊符号，goto 和 call 语句无法识别，所有内容作废。

利用这个，我们可以用**::**这个无效标号来注释批处理，类似**:+**也可以注释批处理，但以**::**最为常见。

注释高级技巧：
%注释内容%
这种用两个百分号包括起来的注释不常用，它是利用了**%**这个符号的特殊性。这样的邪恶用法应该摒弃，因为有可能使程序发生不可预见的错误。

---

pause

暂停批处理程序，并显示以下消息：
请按任意键继续. . .

暂停高级技巧：
pause>nul
只暂停，不显示任何信息，并且光标移到下一行。

如果我们不想用默认的提示语：“请按任意键继续...”，想用自己自定义信息提示，可以这么写：
@echo off
echo 按 N 键退出当前程序
pause > nul
执行后程序会显示：“按 **N** 键退出当前程序”的字样。这个可以用来显示其他提示语后退出。
整合起来写为：
echo 按 N 键退出当前程序 & pause>nul

---

title

设置命令提示窗口的窗口标题。

语法：
TITLE [string]
　参数 **string**　　指定命令提示窗口的标题。

程序例子：
title 批处理教程
执行后，我们可以看到 **cmd** 窗口的标题变成“批处理教程”了。

标题高级技巧：
我们编写的程序如果运行速度比较慢，那就有必要显示当前程序的运行进度给用户看，否则用户不知道程序是否还在运行？如果还在运行，需要再等待多长时间？
显示进度较少采用 **echo** 和 **cls** 的组合来刷新进度显示，因为 **cls** 清除的是整个屏幕，会造成显示一闪一闪的结果。最佳选择是用 **title** 来显示进度。

例如：
```
@echo off
title  圆周率计算程序
echo  按任意键开始计算  & pause>nul
echo  计算进行中，进度请看标题栏...
set i=0
:loop
if %i% lss 10000 (
    set /a i+=1
    title  圆周率计算程序
当前计算到第%i%位
    ::这里是为了更明显点看到显示的效果，所以添加一个时间延迟。
    ping /n 1 127.1>nul
    goto :loop
)
Pause
```

本程序没有实际计算圆周率，只是演示一下显示进度的效果而已。

---

color

设置默认的控制台前景和背景颜色。

语法：
COLOR [attr]

attr	指定控制台输出的颜色属性

attr 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（文字颜色）。数值对应的颜色如下表所示：

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

程序例子：
color 12

上面两个数字都是十六进制数字，前一个数字是背景颜色，后一个数字是文字颜色，即前景颜色。上面代码的意思是：设置背景色为蓝色，文字颜色为绿色。
注意：1、两个数字之间不能有空格！如果执行 **color** 1 2，就达不到我们想要的设置颜色的效果啦。
　　2、背景颜色和文字颜色不能一样！也就是说，两个数字不能一样，否则该颜色设置不成功，并且 **color** 命令会将 **errorlevel** 的值设置为 1，这个我们可以用 **echo %errorlevel%**来查看。

```
@echo off
color 24
echo Hello world!
echo "color 24" The errorlevel number is%errorlevel%
color 00
echo "color 00" The errorlevel number is%errorlevel%
pause
```

这个程序后面的 **color** 00 对前面的 **color** 24 不会改变，返回值是 1。

用 **color** 指令，只能设置控制台的全部前景（文字）颜色和全部背景颜色，无法对文字作部分设置，也就是说，同一个窗口无法显示彩色字符。

颜色设置高级技巧：
用 **color** 无法在同一窗口将前景或背景设置成多种颜色。
批处理里可以用 **findstr** 带**/a** 参数实现，不过实现的彩色效果依旧不能令人非常满意，因为它只对文件名彩色。具体实现方法见 **findstr** 命令。
另外，用第三方也可以实现：**wbat** 或 **ANSI.SYS** 等，在此不再详述。

mode

配置系统设备

功能一：设置 **cmd** 窗口大小
这是 **mode** 最常用的功能。
格式：**mode con [:] [cols=c][lines=n]**

> **WindowsXP 默认 CMD 窗口值为：**
> **大小为：宽×高=80×25**
> **颜色 07**

程序例子：
mode con cols=113 **lines**=15
　　宽　　高
此命令设置 **dos** 窗口大小为 113 列，15 行。
注意，**cols** 的最小值为 14，**lines** 的最小值为 1，否则设置不成功。另外，经过 **mode** 设置过窗口大小后，不会出现滚动条。如果内容过多，窗口又不够大，那么就会造成无法全部显示的结果。

功能二：显示代码页：
代码页，通俗的说就是 **DOS** 中显示的语言。具体代码页可以查看代码页这个百科词条。
比如，想让 **DOS** 下显示的语言为美式英语，则输入 **mode con cp select**=437 即可，输出为：
Status for device CON:

Lines: 40
Columns: 100
Keyboard rate: 31
Keyboard delay: 1
Code page: 437
如果输入的是 **mode con cp select**=936（936 表示简体中文），则输出：
设备状态　**CON:**

行:　　40
列:　　100
键盘速度: 31

键盘延迟:　1
代码页: 936

mode 还有其它的一些功能，但大多都用处不大，这里就不再讲述了。

---

goto

无条件跳转命令

语法:
GOTO label
　label　指定批处理程序中用作标签的文字字符串。

标签必须单独一行，并且以冒号打头。所以，**goto** 和**:**分不开，当程序运行到 **goto** 时，将自动跳转到**:**定义的部分去执行了。

可以用 **goto** 和 **if** 的组合来实现循环，程序例子:

```
@echo off
:begin
set /a var+=1
echo %var%
if %var% leq 3 goto begin
pause
```

运行显示

```
1
2
3
4
请按任意键继续...
```

goto :eof
这是一整条命令，可用于退出批处理。
注意书写：**geto** 后面有空格，**:**和 **eof** 之间没有空格。
它的确切含义是: 在不定义标签的情况下将控制传送到当前批处理脚本文件的末端。当脚本的控制到达脚本的末端，再执行的话，脚本就自行退出了，**goto :eof** 就是利用这一点来退出脚本的。这个的退出原理和没有暂停语句的脚本的退出原理是一样的，例如:

```
@echo off
echo Hello World!
```

上面程序没有设置暂停，那么程序显示完 **Hello World** 后就自行关闭了。我们看到的执行效果是：程序闪了一下就没了。

一个细节，**goto** 后面的标签前要不要带冒号？答案是带不带都可以，标准格式是带的：

```
@echo off
:loop
set var=1
goto loop
pause

@echo off
:loop
set var=1
goto :loop
pause
```

上面两种写法都可以，但是标准的写法要带冒号，建议采用标准格式。

跳转高级技巧：

用 **call** 也可以实现跳转

例如：

```
@echo off
:begin
set /a var+=1
echo %var%
if %var% leq 3 call :begin
pause
```

用 **goto** 和 **call** 实现跳转是有区别的：

1、书写格式，**goto** 后面的标签前可以有冒号也可以没有冒号，**call** 后面标签前必须要有冒号。

2、**goto** 跳转到目标程序段后，就顺序执行下去，直到程序执行完毕。

call 跳转到目标程序段后，会将目标程序执行完，然后回到 **call** 原来的地方，然后继续执行下去。

关于 **call** 调用程序的详细规则，请看 **call** 命令详细用法。

退出命令

不要小看了批处理的退出命令！小小的一个退出功能实现，也是挺复杂滴！它们之间有细微的差别，彻底理清它们的区别，可以让我们对批处理的认识更加深刻。

三个退出命令：

exit
exit /b
goto :eof

这三个命令都可以退出批处理，但它们之间还是有细微差别的：
(1) 运行 **GOTO :EOF** 后，**CMD** 返回并将等待下一条命令。
(2) 运行 **EXIT** 后，　　**CMD** 将直接关闭并返回到曾启动 **cmd.exe** 的程序或返回到”资源管理器”。
(3) 运行 **EXIT /B** 后，　**CMD** 将直接关闭并返回到曾启动 **cmd.exe** 的程序或返回到”资源管理器”。

goto :eof
的确切含义是：在不定义标签的情况下将控制传送到当前批处理脚本文件的末端。当脚本的控制到达脚本的末端，再执行的话，脚本就自行退出了，**goto :eof** 就是利用这一点来退出脚本的，这个的退出原理和没有暂停语句的脚本的退出原理是一样的。
例如：

```
@echo off
echo Hello World!
```

上面程序没有设置暂停(**pause**)，那么程序显示完 **Hello World** 后就自行关闭了。我们看到的执行效果是：程序闪了一下就没了。

exit
退出 **CMD.EXE** 程序(命令翻译程序)或当前批处理脚本。
语法：**EXIT [/B] [exitCode]**
　/B　　指定要退出当前批处理脚本而不是 **CMD.EXE**。如果从一个批处理脚本外执行，则会退出 **CMD.EXE**
　exitCode 指定一个数字号码。如果指定了 **/B**，将 **ERRORLEVEL** 设成那个数字。如果退出 **CMD.EXE**，则用那个数字设置过程退出代码。

exit 命令和 **goto :eof** 不同，它就是确确实实的退出 **CMD.exe** 指令，具备退出功能。

exit /b
这个指令的确切含义是退出当前的批处理脚本而不退出 **CMD.EXE**，**exit /b** 指令具备关闭功能。但如果从一个批处理脚本外执行，则会退出 **CMD.EXE**。
例如我们在 **CMD** 窗口执行 **exit /b**，**CMD** 窗口会关闭。由于 **exit /b** 是退出当前脚本而不退出 **CMD**，所以显现出来的效果和 **goto :eof** 功能一模一样。

exit /b 与 **goto :eof** 的细微差别：
1、**exit /b** 具备退出功能。**goto :eof** 不具备退出功能，它是利用批处理执行完毕后会自动退出这个特性达到退出脚本的功能。

2、在批处理以外执行 **exit /b**，它还是会退出 **CMD** 窗口的。例如在 **CMD** 窗口执行 **exit /b**，窗口就关闭了。但执行 **goto :eof**，窗口没有关闭，光标继续等待用户输入。还是那句话：**goto :eof** 不具备退出功能。

3、**exit /b** 还具备设置 **errorlevel** 数值的功能。
具体如下：
假如我们在 **D** 盘的一个名为《**abc.bat**》的脚本，里面的代码为：
@echo off
exit /b 1234567
我们点击开始-->运行，打开 **CMD** 窗口，输入 **D:\abc.bat**，具体执行和运行结果如下：
C:\Documents and Settings\Administrator>D:\abc.bat
C:\Documents and Settings\Administrator>echo %errorlevel%
1234567
C:\Documents and Settings\Administrator>
此时的 **errorlevel** 值变成 1234567 了。
退出 **CMD** 窗口后再查看 **errorlevel** 值，发现它恢复回原来的数值 0。

上面之所以要在 **D** 盘弄一个《**abc.bat**》脚本，是因为如果直接在 **CMD** 窗口执行代码 **exit /b** 1234567，**errorlevel** 值是设置成功了，但同时 **CMD** 窗口也被 **exit /b** 关闭了，此时我们再次打开 **CMD** 窗口，**errorlevel** 值又恢复为 0，看不到什么效果。

三个退出命令实际操作过程实例：
@echo off
echo Hello World!
goto :eof
将这个文件命名为《**abc.bat**》，保存路径 **D:\abc.bat**
双击《**abc.bat**》运行，程序闪了一下就关闭了。
我们从“开始-->运行，打开 **cmd** 窗口”，切换到 **D** 盘根目录，然后输入“**start abc.bat**”，具体操作过程和显示如下：
C:\Documents and Settings\Administrator>D:
D:\>start abc.bat
D:\>
这是原来打开的 **CMD** 窗口的执行情况，窗口的标题栏显示：**C:\WINDOWS\system32\cmd.exe**，也就是此窗口的路径，和刚开始我们打开时的路径一样。程序打开了一个新的窗口，新窗口显示如下：
Hello World !

D:\>
这个窗口并没有退出，光标在 **D:\>**后闪烁，等待用户继续输入命令。我们注意到此时这个新的窗口的标题也是 **C:\WINDOWS\system32\cmd.exe**，说明原来的 **abc.bat** 已经关闭，返回到 **CMD** 窗口了，就是上面打开的 **CMD** 窗口，只不过重复打开而已。

我们从开始-->运行，打开 **cmd** 窗口，切换到 **D** 盘，然后输入“**abc.bat**”，具体操作过程和显示如下：
C:\Documents and Settings\Administrator>D:
D:\>abc.bat
Hello World!

D:\>
程序并没有打开新窗口，光标继续等待用户输入。

程序修改为：
@echo off
echo Hello World!
exit
双击《**abc.bat**》运行，程序闪了一下就关闭了。
我们从“开始-->运行，打开 **cmd** 窗口”，切换到 **D** 盘，然后输入“**start abc.bat**”，具体操作过程和显示如下：

C:\Documents and Settings\Administrator>D:
D:\>start abc.bat
D:\>
我们可以看到，新打开的窗口闪了一下就关闭了。
我们从开始-->运行，打开 **cmd** 窗口，切换到 **D** 盘，然后输入**“abc.bat”**，具体操作过程和显示如下：
C:\Documents and Settings\Administrator>D:
D:\>abc.bat
运行到这里，没有打开新窗口，**CMD** 窗口就直接退出了。其实 **CMD** 也不是直接退出，中间有显示一下**“Hello Worle!”**，只不过速度太快，难以觉察而已。

程序修改为：
@echo off
echo Hello World!
exit /b
程序运行情况和 **goto :eof** 情况完全一样。

上面是分别在批处理文件运行和 **CMD** 窗口调用运行的情况。
在 **CALL** 命令子标签中使用情况如下：
(1) 使用 **GOTO :EOF,** 将返回到 **CALL** 命令**, FOR** 循环也将继续迭代**.**
(2) 使用 **EXIT,** 将直接从批处理文件**(TEST5.bat)**退出**,** 不能返回到 **CALL** 命令**, FOR** 循环迭代被终止**.**
(3) 使用 **EXIT /B,** 将返回到 **CALL** 命令**, FOR** 循环也将继续迭代**.**

例外情况
一些情况下**,** 在 **CMD** 窗口中运行 **EXIT** 并不是退出 **CMD.**
(1) 在 **FTP** 子系统中**,** 运行 **!** 从 **FTP** 子系统临时退出到命令提示符下**,** 这是运行 **EXIT,** 将返回到 **FTP** 子系统**.**
下面是实际操作的一个过程：
C:\Documents and Settings\Administrator>ftp
ftp> !
Microsoft Windows XP [版本 5.1.2600**]**
(C) 版权所有 1985-2001 **Microsoft Corp.**
C:\Documents and Settings\Administrator>exit
ftp> quit
C:\Documents and Settings\Administrator>
(2) 在命令提示符下嵌套新实例 **cmd.exe** 时**,** 运行 **EXIT** 将关闭嵌套的命令解释器**,** 而不是父命令解释器**.**
C:\Documents and Settings\Administrator>cmd

Microsoft Windows XP [版本 5.1.2600**]**
(C) 版权所有 1985-2001 **Microsoft Corp.**
C:\Documents and Settings\Administrator>exit
C:\Documents and Settings\Administrator>
我们一开始在运行 **cmd** 命令后，注意到标题栏多了一个**-cmd**，说明进入了 **cmd** 嵌套。

---

start

启动另一个窗口运行指定的程序或命令。

语法：
START ["title"] [/Dpath] [/I] [/MIN] [/MAX] [/SEPARATE | /SHARED] [/LOW | /NORMAL | /HIGH | /REALTIME | /ABOVENORMAL | /BELOWNORMAL] [/WAIT] [/B] [command/program] [parameters]

参数说明：
"title"　　在窗口标题栏中显示的标题。
path　　起始目录
B　　在不创建新窗口的情况下开始应用程序。 除非启动 **^C** 处理，否则该应用程序会忽略 **^C** 处理；**^Break** 是唯一可以中断该应用程序的方式
I　　新环境是传递给 **cmd.exe** 的原始环境，而不是当前环境
MIN　　开始时窗口最小化
MAX　　开始时窗口最大化
SEPARATE　在分开的空间内开始 16 位 **Windows** 程序
SHARED　在分共享的空间内开始 16 位 **Windows** 程序
LOW　在 **IDLE** 优先级类别开始应用程序　低优先级
NORMAL　在 **NORMAL** 优先级类别开始应用程序　一般优先级
HIGH　在 **HIGH** 优先级类别开始应用程序　高优先级
REALTIME　在 **REALTIME** 优先级类别开始应用程序 实时优先级
ABOVENORMAL 在 **ABOVENORMAL** 优先级类别开始应用程序　超出常规优先级
BELOWNORMAL 在 **BELOWNORMAL** 优先级类别开始应用程序　低出常规优先级
WAIT　　启动应用程序并等候它结束
command/program
　　如果是内部 **cmd** 命令或批文件，那么该命令处理器是用 **/K** 命令行开关运行 **cmd.exe** 的。这表示该窗口在命令运行后仍然存在。
　　如果不是内部 **cmd** 命令或批文件，则是一个程序，并作为窗口应用程序或控制台应用程序运行。
parameters　这些为传送到命令/程序的参数

start 批处理调用外部程序，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情况，等外部程序执行完后才继续执行剩下指令。

程序例子：

```
@echo off
start D:\Tencent\QQ\Bin\QQ.exe
start C:\批处理教程.txt
::打开 E 盘
start E:
exit
```

上面程序会按顺序依次执行打开腾讯软件-->打开《批处理教程》-->打开 **E** 盘，完成后批处理文件自动自我关闭。
注：由于 **DOS** 命令对于调用外部程序的优先级别不同，**.exe** 可以省略，但**.txt** 就不能省略了。这里提倡不要区分要不要写，统一将后缀全写上的良好风格，确保程序万无一失。

"title"
这个参数可以设置标题，测试了一下，只发现对命令行窗口有效。
例如我们的 **D** 盘根目录有一个《**MHL.bat**》的批处理文件，执行代码：

```
@echo off
start "执行" D:\MHL.bat
pause
```

我们可以看到，打开的 **MHL.bat** 窗口的标题栏为“执行-**MHL.bat**”，标题多了“执行”两个字。

打开带有空格的路径
假如我们的路径带有空格，那么就必须用双引号将路径括起来，但是 **start** 命令很奇怪，双引号代表设置标题，我们对路径添加双引号，程序会认为那是标题设置，程序还是无法执行。
例如：**start "D:\Photoshop CS\photoshop.exe"**这个命令是无法执行的。
解决的办法就是在 **start** 后面多添加一对空双引号，这样程序就认为第一对双引号的内容为标题，第二对双引号的内容为路径，程序就可以正确执行了：

```
start "" "D:\Photoshop CS\photoshop.exe"
```

/max 打开程序时窗口最大化
/min 打开程序时窗口最小化
这两个命令不一定每次都奏效，程序例子：
start　/max "" "D:\Photoshop CS\photoshop.exe"　打开 **photoshop** 时窗口最大化
start　/min "" "D:\Photoshop CS\photoshop.exe"　打开 **photoshop** 时窗口最小化
这里，**/max** 和**/min** 对于 **photoshop** 有效，但对一些软件就不一定奏效了，例如千千静听，是不是用**/max** 或**/min**，打开的窗口状态都是上一次打开窗口的状态。

优先级参数：

LOW	在 **IDLE** 优先级类别开始应用程序	低优先级
NORMAL	在 **NORMAL** 优先级类别开始应用程序	一般优先级
HIGH	在 **HIGH** 优先级类别开始应用程序	高优先级
REALTIME	在 **REALTIME** 优先级类别开始应用程序	实时优先级
ABOVENORMAL	在 **ABOVENORMAL** 优先级类别开始应用程序	超出常规优先级
BELOWNORMAL	在 **BELOWNORMAL** 优先级类别开始应用程序	低出常规优先级

这个根据需要选择，设置为高优先级或者超出常规优先级，在入侵他人计算机时非常有用。

/wait
启动应用程序并等候它结束。假如批处理程序中没有 **pause** 暂停语句，那么用 **start** 启动某个程序后，批处理自身会马上关闭退出。有了**/wait** 语句，批处理打开程序后会一直保持打开状态，等到我们将打开的程序关闭，批处理会提示：
^C 终止批处理操作吗**(Y/N)?**
我们输入 **Y**，批处理才退出。
例如
start /wait D:\abc.exe
打开 **abc.exe** 后，批处理本身会一直打开，直到 **abc.exe** 关闭，批处理才提示“**^C** 终止批处理操作吗**(Y/N)?**”，我们根据需要作出选择。

打开网页
start http:\\www.baidu.com
执行命令后会打开百度首页。

call

调用另一个批处理程序或自身程序段，调用完后，程序会回到原来的位置继续执行下去。**call** 也可以调用自身程序段。 如果在脚本或批处理文件外使用 **call** 命令，则其不会在命令行起作用。

语法

call [[Drive:][Path] FileName [BatchParameters]] [:Label [Arguments]]

参数说明：

[Drive:][Path] FileName

指定要调用的批处理程序的位置和名称。**FileName** 参数必须有 **.bat** 或 **.cmd** 扩展名。

BatchParameters

指定批处理程序所需的任何命令行信息，包括命令行选项、文件名、批处理参数（即从 **%0** 到 **%9**）或变量（例如，**%baud%**）。

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 **goto:eof** 结尾，调用完后程序会回到原来的地方继续按顺序执行下去。假如被调用命令段(子程序)以 **exit** 结尾，那么调用完子程序后，程序就直接退出，不会回到原来的地方了。假如被调用的子程序没有 **goto :eof** 或 **exit** 等退出标志，那么程序在调用子程序后，会一直执行到整个程序末尾，然后再回到原来的地方继续执行下去。

Arguments

对于以 **:Label** 开始的批处理程序，指定要传递给其新实例的所有命令行信息，包括命令行选项、文件名、批处理参数（即从 **%1** 到 **%9**）或者变量（例如，**%baud%**）。

注意事项：

不要在 **call** 命令中使用管道和重定向符号。

可以创建调用自身的批处理程序，但是，必须提供退出条件。否则，父和子批处理程序循环上千次。

在启用命令扩展的情况下（即默认情况下），**call** 将接受 **Label** 参数作为调用目标。语法如下：

call:Label Arguments

[Drive:][Path] FileName 程序例子：
CALL "C:\Documents and Settings\Administrator桌面\圆周率**.bat"**
程序会执行桌面的《圆周率**.bat**》批处理程序。

程序例子：
我们在同一个文件夹下，有 **a.bat** 和 **b.bat** 两个批处理文件，
a.bat 里面的代码为：

```
@echo off
set /p var=请选择(Y/N):
if /i "%var%"=="Y" call b.bat
echo a.bat 执行完毕。
pause >nul
```

b.bat 里面的代码为：

```
@echo off
echo Hello World!
start explorer.exe
echo b.bat 执行完毕。
pause >nul
```

我们双击执行 **a.bat**，执行结果如下

```
请选择(Y/N):Y
Hello World!
b.bat 执行完毕。
a.bat 执行完毕。
```

并且，计算机实在打开 **explorer** 后才显示 **a.bat** 执行完毕的。

子程序语法：

```
:label
command1
command2
...
commandn
```

在子程序段中，参数**%0** 指标签**":label"**。
在主程序中，要注意用 **goto**、**exit**、**goto :eof** 等等跳转语句，避免误进入子程序过程。
子程序和主程序中的变量都是全局变量，其作用范围都是整个批处理程序。
主程序传递至子程序的参数在 **call** 语句中指定，在子程序中用

%1--%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。

程序例子：

```
@echo off
set /p a=请输入一个数字：
if /i %a%==0 (call :o ) else (call :t)
:o
echo Hello World!
pause>nul & exit
:t
echo This is a bat program.
pause>nul & exit
----------------------
```

用 **goto** 和 **call** 都可以在程序中实现跳转。
但是，**goto** 和 **call** 还是有区别的，就是 **goto** 说到底还只是跳转而已，跳转到目的地后，程序按顺序执行下去，不会返回原来的地方。**call** 调用完程序段后，还会返回原来的地方继续执行下去。另外 **call** 调用程序段，在调用时，可以给参数，**goto** 则不具备带参数功能。

请看两个程序：
程序 **A**

```
@echo off
echo  早上好!
goto noon
echo  晚安!
pause
:noon
      echo  中午好!
:night
      echo  晚上好!
pause
```

程序 **B**

```
@echo off
echo  早上好!
call :noon
echo  晚安!
pause
:noon
```

```
    echo 中午好!
:night
    echo 晚上好!
pause
```

程序 **A** 执行结果为：

```
早上好!
中午好!
晚上好!
请按任意键继续. . .
```

程序 **B** 执行结果为：

```
早上好!
中午好!
晚上好!
请按任意键继续. . .
晚安!
请按任意键继续. . .
中午好!
晚上好!
请按任意键继续. . .
```

从这里就很明显看出两个命令的区别了：

goto 将流程跳转到标签处后，程序从标签处开始执行到结束，整个流程执行完毕；**call** 将流程跳转到标签处后，程序从标签处开始执行到结束，然后会跳转回执行 **call** 的下一条语句，直到整个流程执行完毕；所以 **call** 并不会像 **goto** 那样打乱了流程，它只是暂时中断了当前流程而跳到目标程序段执行，目标程序段执行完毕后，仍回到原来 **call** 的地方按原顺序继续进行下去。所以，在 **call** 后面经常会有 **goto :eof** 或 **exit** 等退出的语句，例如：

```
@echo off
set var=123
call :loop
pause>nul & goto :eof
:loop
set /a var+=1
echo var=%var%
pause>nul
```

假如上面程序没有 **goto :eof** 退出语句，那么程序会显示两次数据，一次是 124，一次是 125。

call 带参数的情况
程序例子：
```
@echo off
call :an Hello 世界!
:an
echo %1
echo %2
pause>nul
```
输出为：
```
Hello
世界!
```

程序例子：
```
@echo off
call :sub return 你好!
echo 子程序返回值：%return%
echo %0
pause>nul
:sub
set %1=%2
echo %0
```

将上面程序保存在桌面《**abc.bat**》，运行输出为：
```
:sub
子程序返回值：你好!
"C:\Documents and Settings\Administrator\桌面\abc.bat"
```
可以看出，主程序里的**%0** 代表主程序的完整路径，就是 **"C:\Documents and Settings\Administrator\桌面\abc.bat"**，而子程序的**%0** 也代表子程序的完整路径，它就是“**:sub**”。对于子程序中的 **return** 参数，子程序运行结束后，主程序可以直接用，**return** 在全局都有效。注意，主程序里，不可以用**%1--%9** 的形式调用子程序参数。例如：
```
@echo off
call :sub return 你好!
```

```
echo %1
echo %2
pause>nul
:sub
set %1=%2
```

主程序这样的调用无效。

再看一个求多个数据在子程序相加的例子：

```
@echo off
set sum=0
call :sub sum 10 20 35
echo  数据求和结果: %sum%
pause>nul
goto :eof
:sub
set /a %1=%1+%2
shift /2
if not "%2"=="" goto sub
```

程序输出为 65。

自我调用的例子：

本脚本名为 **a.bat**，里面只有一行代码：

```
call a.bat
```

双击运行 **a.bat**，脚本被执行了 1241 次。

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:

%~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

要理解上面的知识，下面的例子很重要！

```
@echo off
call :sub tmp.txt
pause & exit
: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 扩充的路径只含有短名: %~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 查找列在 PATH 环境变量的目录，并将第一个参数扩充到找到的第一个完全合格的名称。%~$PATH :1
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
```

在 **C:\windows** 目录下新建一个文件 **linshi.txt**，将上面代码保存在里面并改名为 **linshi.bat**，双击执行。
情况一：桌面上没有 **tmp.txt** 文件，执行结果如下：

删除引号: **tmp.txt**
扩充到路径: **C:\Documents and Settings\Administrator**桌面**\tmp.txt**
扩充到一个驱动器号: **C:**
扩充到一个路径:**\Documents and Settings\Administrator**桌面****
扩充到一个文件名: **tmp**
扩充到一个文件扩展名: **.txt**
扩充的路径只含有短名: **C:\DOCUME~1\ADMINI~1**桌面**\tmp.txt**
扩充到文件属性:
扩充到文件的日期/时间:
扩充到文件的大小:
查找列在 **PATH** 环境变量的目录，并将第一个参数扩充到找到的第一个完全合格的名称。
扩展到驱动器号和路径: **C:\Documents and Settings\Administrator**桌面****
扩展到文件名和扩展名: **tmp.txt**
扩展到类似 **DIR** 的输出行: **C:\Documents and Settings\Administrator**桌面**\tmp.txt**
请按任意键继续. . .

情况二：桌面有一个名为《**tmp.txt**》内容为空的文件，执行结果如下：

删除引号: **tmp.txt**
扩充到路径: **C:\Documents and Settings\Administrator**桌面**\tmp.txt**
扩充到一个驱动器号: **C:**
扩充到一个路径:**\Documents and Settings\Administrator**桌面****
扩充到一个文件名: **tmp**
扩充到一个文件扩展名: **.txt**
扩充的路径只含有短名: **C:\DOCUME~1\ADMINI~1**桌面**\tmp.txt**
扩充到文件属性: **--a------**
扩充到文件的日期/时间: 2013-12-30 10:06
扩充到文件的大小: 0
查找列在 **PATH** 环境变量的目录，并将第一个参数扩充到找到的第一个完全合格的名称。
扩展到驱动器号和路径: **C:\Documents and Settings\Administrator**桌面****
扩展到文件名和扩展名: **tmp.txt**
扩展到类似 **DIR** 的输出行: **--a------** 2013-12-30 10:06 0 **C:\Documents**

and Settings\Administrator桌面**\tmp.txt**
请按任意键继续. . .

情况三：桌面没有名为 **tmp.txt** 的文件，倒是在 **D** 盘根目录有一个名为 **tmp.txt** 的文件，即：**D:\tmp.txt**，现在我们将 **call** 语句改为 **call :sub D:\tmp.txt**，其余全都不变，代码如下

```
@echo off
call :sub D:\tmp.txt
pause & exit
: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 扩充的路径只含有短名: %~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 查找列在 PATH 环境变量的目录，并将第一个参数扩充到找到的第一个完全合格的名称。%~$PATH :1
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
```

执行结果如下：

```
删除引号: D:\tmp.txt
扩充到路径: D:\tmp.txt
扩充到一个驱动器号: D:
扩充到一个路径:\
扩充到一个文件名: tmp
扩充到一个文件扩展名: .txt
扩充的路径只含有短名: D:\tmp.txt
扩充到文件属性: --a------
扩充到文件的日期/时间: 2013-12-30 10:06
扩充到文件的大小: 0
查找列在 PATH 环境变量的目录，并将第一个参数扩充到找到的第一
```

个完全合格的名称。**D:\tmp.txt**
扩展到驱动器号和路径**: D:**
扩展到文件名和扩展名**: tmp.txt**
扩展到类似 **DIR** 的输出行**: --a------ 2013-12-30 10:06 0 D:\tmp.txt**
请按任意键继续**. . .**

用 **call** 带参数的实际例子------斐波那契数列

```
@echo off
setlocal enabledelayedexpansion
echo. & echo                                                    斐波那契数列
echo.
echo  请输入数列总项数
set /p num=(DOS 计算能力不够强大，输入的数字不要超过 46):
echo. & echo ------------------------------------------------------------
if %num% gtr 46 echo. & echo  输入数据太大，按任意键退出  &
pause >nul & exit
set /a e=!num!-2
set a=1
set b=1
set c=2
if %num% geq 1 call :shuchu !a!
if %num% geq 2 call :shuchu !b!
for /l %%i in (1,1,%e%) do (
    set /a tem=!a!+!b!
    set /a c+=1
    set /a d=!c!%%5
    call :shuchu !tem!
    if !d!==0 echo.
    set a=!b!
    set b=!tem!
)
echo. & echo. & echo  输出结束，按任意键退出  & pause>nul
::这里要注意退出，否则程序会继续执行后面的:shuchu 程序段
goto :eof

:shuchu
set f=                                  %1
set /p pr=!f:~-11!<nul
```

这里用 **call**，是因为程序有多个地方用到输出语句，而一个输出需要用到 **set f=　　%1** 和 **set /p pr=!f:~-11!<nul** 两句，比较繁琐，所以采用调用函数的办法简化重复程序段。

call 高级技巧

一、嵌套和递归：
下面以汉诺塔程序作为例子：

```
@echo off
title 汉诺塔(Hanoi)
echo. & echo.
echo                                                    汉诺塔程序
setlocal enabledelayedexpansion
echo. & set /p plate=请输入盘子数量：
echo. & echo ----------------------------
call :move !plate! A B C
echo ----------------------------
echo. & echo 运算结束，按任意键退出。 & pause>nul
:move
    if %1==1 (echo 盘子从 %2 柱子移动到 %4 柱子) else (
        set i=%1
        set /a i-=1
        call :move !i! %2 %4 %3
        echo 盘子从 %2 柱子移动到 %4 柱子
        set j=%1
        set /a j-=1
        call :move !j! %3 %2 %4
    )
```

问题：用 **call** 来递归，嵌套层级最大是多少？
答案：1240 级
测试代码：

```
@echo off
call :loop 1
:loop
echo %1>>tem.txt
set /a n=%1+1
call :loop %n%
```

问：不用 **call**，能否实现嵌套呢？
答案是肯定的！批处理有一个很特殊的参数**%0**，它本身还可以带参数运算，这一点和 **call** 命令很类似。

看下面两个例子：

```
@echo off
set /a var=%1+1
echo %var%
%0 %var%
```

显示从 1 开始，每次增加 1，不断显示下去。

如果是 **call** 命令，代码就变成：

```
@echo off
:loop
set /a var=%1+1
echo %var%
call :loop %var%
```

显示效果和上面完全一样。

但是 **call** 和%0 还是有区别的：

call 有迭代层级限制，而%0 没有迭代层级限制。

call 调用完程序后还会回到原来的地方继续执行下去，所以 **call** 可以实现递归。但%0 仅仅重复调用自身而已，所以%0 不可以实现递归。

call 可以只调用程序中的某一小段程序，而%0 只能调用的是自身整个程序，所以在有延迟变量的情况下，%0 调用没几次就达到极限，无法执行下去了。另外，因为这个，%0 无法实现复杂的程序调用。

二、用 **call** 实现延迟功能：

代码一：

```
@echo off
set str=www.cn-dos.net
set n=4
echo %%str:~%n%%%
pause>nul
```

代码二：

```
@echo off
set str=www.cn-dos.net
set n=4
call echo %%str:~%n%%%
pause>nul
```

运行代码一，我们并不能得到想要的结果“**cn-dos.net**”，而是会显示为“**%str:~4%**”，这是为什么呢？因为在 **cmd** 中存在预处理机制，

在读取 **echo %%str:~%n%%%**这句的时候，先是迫不及待地脱去最外层的%符，变为了 **echo　%str:~%n%%**，在这里第二层的%就变成了字符%了，自然就显示为“**%str:~4%**”了，而要如何解决这个问题呢？我们就可用到 **call** 的延时作用了，通过 **call** 延时后，**cmd** 就会一层层脱去%并解释%里面的变量，如运行代码二就能正确显示为“**cn-dos.net**”了。

三、**call** 邪门用法

```
@echo off
set a=dos
set b=a
set c=b
set d=c
set e=d
echo %a%
call echo %%%b%%%
call call echo %%%%%%%c%%%%%%%
call call call echo
%%%%%%%%%%%%%%%d%%%%%%%%%%%%%%
%
pause>nul
```

问：有 **n** 个 **call** 时，需要多少个%号？
答：(2^(n+1))-1 个，2 的 **n**+1 次方再减 1 个%号。

```
@echo off
set a=dos
call call call call call call echo
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%a%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%
pause>NUL
```

输出为：**dos**

这样的语法挺繁琐的，尚不清楚有什么实际应用。

shift

更改批处理文件中可替换参数的位置。

语法格式
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...**，并且此时不影响 **%0** 和 **%1**。
看到上面说明，不甚理解是吧？看下面例子就明白了：
求 1 到 9 相加的和：

```
@echo off
set sum=0
call :sub    sum 1 2 3 4 5 6 7 8 9
echo  数据求和结果:%sum%
pause>nul
:sub
set /a %1=%1+%2
shift /2
if not "%2"=="" goto sub
```

程序输出结果为：
数据求和结果：45

这里引出了几个问题：
1：子函数的参数**%1-%9** 不可以在主程序中直接使用。
我们上面的“**echo** 数据求和结果**:%sum%**”，如果改为“**echo** 数据求和结果**:%1**”，则没有任何显示。

2：子函数中的到最后，**%2-%9** 全都空值。对于**%1**，它此时的值等于 45，但如果我们此时在子函数里用 **echo %1** 看结果时，显示出来的是“**sum**”，也就是变量名而非变量值。
　　批处理不允许这种**%%1%**的格式，所以用**%1** 这个参数，我们无法显示出其数值，必须用**%sum%**的格式显示数值。

3：**set** 命令不允许 **set /a %1=%2+%3** 这种将数值直接赋值到参数的计算，当然也不允许 **set /a** 3=4+5 这种将数值赋值到数值的计算。

如果我们非用 **set /a %1=%2+%3** 这种形式不可的计算，那么就应该从一开始将变量赋值给**%1** 参数。上面的程序 **sum** 就是这个作用。

4：程序运行到最后，所有参数的值都空，但我们如果用**%***显示参数数值，可以发现所有参数数值不变！这是为什么呢？

这里重点讲一下 **shift**。
程序的运算以及数值过程如下：

一开始，共有 10 个参数，分别是

%1	%2	%3	%4	%5	%6	%7	%8	%9	无
sum=0	1	2	3	4	5	6	7	8	9

第一次计算

set /a %1=%1+%2　　也就是：**sum=sum**+1=1

此时是个参数分别是：

%1	%2	%3	%4	%5	%6	%7	%8	%9	无
sum=1	1	2	3	4	5	6	7	8	9

然后，我们执行 **shif** /2，那么是个参数分别是：

%1	%2	%3	%4	%5	%6	%7	%8	%9	无
sum=1	2	3	4	5	6	7	8	9	空

第二次计算

set /a %1=%1+%2　　也就是：**sum=sum**+2=3

此时是个参数分别是：

%1	%2	%3	%4	%5	%6	%7	%8	%9	无
sum=3	2	3	4	5	6	7	8	9	空

然后，我们执行 **shif** /2，那么是个参数分别是：

%1	%2	%3	%4	%5	%6	%7	%8	%9	无
sum=3	3	4	5	6	7	8	9	空	空

第三次计算

set /a %1=%1+%2　　也就是：**sum=sum**+3=6

此时是个参数分别是：

%1	%2	%3	%4	%5	%6	%7	%8	%9	无
sum=6	3	4	5	6	7	8	9	空	空

然后，我们执行 **shif** /2，那么是个参数分别是：

%1	%2	%3	%4	%5	%6	%7	%8	%9	无
sum=6	4	5	6	7	8	9	空	空	空

第四次计算

set /a %1=%1+%2　　也就是：**sum=sum**+4=10

此时是个参数分别是：

%1	%2	%3	%4	%5	%6	%7	%8	%9	无
sum=10	4	5	6	7	8	9	空	空	空

然后，我们执行 **shif** /2，那么是个参数分别是：

%1	%2	%3	%4	%5	%6	%7	%8	%9	无
sum=10	5	6	7	8	9	空	空	空	空

第五次计算

set /a %1=%1+%2　　也就是：**sum=sum**+5=15

此时是个参数分别是：

%1	%2	%3	%4	%5	%6	%7	%8	%9	无
sum=15	5	6	7	8	9	空	空	空	空

然后，我们执行 **shif** /2，那么是个参数分别是：

%1	%2	%3	%4	%5	%6	%7	%8	%9	无
sum=15	6	7	8	9	空	空	空	空	空

第六次计算

set /a %1=%1+%2　　也就是：**sum=sum**+6=21

此时是个参数分别是：

%1	%2	%3	%4	%5	%6	%7	%8	%9	无
sum=21	6	7	8	9	空	空	空	空	空

然后，我们执行 **shif** /2，那么是个参数分别是：

%1	%2	%3	%4	%5	%6	%7	%8	%9	无
sum=21	7	8	9	空	空	空	空	空	空

第七次计算

set /a %1=%1+%2　　也就是：**sum=sum**+7=28

此时是个参数分别是：

%1	%2	%3	%4	%5	%6	%7	%8	%9	无
sum=28	7	8	9	空	空	空	空	空	空

然后，我们执行 **shif** /2，那么是个参数分别是：

%1	%2	%3	%4	%5	%6	%7	%8	%9	无
sum=28	8	9	空	空	空	空	空	空	空

第八次计算

set /a %1=%1+%2　　也就是：**sum=sum**+8=36

此时是个参数分别是：

%1	%2	%3	%4	%5	%6	%7	%8	%9	无
sum=36	8	9	空	空	空	空	空	空	空

然后，我们执行 **shif** /2，那么是个参数分别是：

%1	%2	%3	%4	%5	%6	%7	%8	%9	无
sum=36	9	空	空	空	空	空	空	空	空

第九次计算

set /a %1=%1+%2　　也就是：**sum=sum**+9=45

此时是个参数分别是：

%1	%2	%3	%4	%5	%6	%7	%8	%9	无
sum=36	9	空	空	空	空	空	空	空	空

然后，我们执行 **shif** /2，那么是个参数分别是：

%1	%2	%3	%4	%5	%6	%7	%8	%9	无
sum=36	空	空	空	空	空	空	空	空	空

从本例看出，这种形式的程序，输入的参数的个数不局限于 9 个，可以比 9 个更多。

---

errorlevel

程序返回码

用法：**echo %errorlevel%**

查看当前程序返回值，以知道程序或该命令行是否执行成功。**DOS** 程序在运行完后都有返回码，返回码为 0 表明程序执行正确，执行错误会返回非 0 数值，虽然一般都是返回 1，但不能认为错误的返回值就是 1，具体返回什么值，要看具体命令，**errorlevel** 返回值可以参看批处理变量专题。

批处理的选择结构：

if

执行批处理程序中的条件处理。

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

参数说明：

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　　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。

以上是帮助信息里的内容，下面是程序详解：
if 命令是一个判断命令，根据得出的每一个结果，它都可以对应一个相应的操作。

IF 的三个格式
（1）结果判断：**IF [NOT] ERRORLEVEL number do command**
（2）比对判断：**IF [NOT] string1==string2 do command**
（3）存在判断：**IF [NOT] EXIST filename do command**

(1)IF [NOT] ERRORLEVEL number do command
if errorlevel 这个句子必须放在某一个命令的后面，执行该命令后才由 **if errorlevel** 来判断命令的返回值。返回值 **number** 的数字取值范围 0~255，**(**具体各种命令返回值参见 **errorlevel** 返回值一览表**)**。返回值大于等于指定值时，条件成立。

即：
if　errorlevel 0　如果返回值大于或等于 0（代表前面程序执行成功）
if　errorlevel 1　　如果返回值大于或等于 1（代表前面程序执行失败）失败的原因有多重，不同的失败有不同的失败代码。
if not　errorlevel 0　如果返回值小于 0（代表前面程序执行失败）不常用，由于 **errorlevel** 值不可能小于 0，所以无论如何程序都不执行后面语句。
if not　errorlevel 1　　如果返回值小于 1（代表前面程序执行成功）不常用，这个虽然代表成功，但不如 **if　errorlevel** 0 来得直接点，所以一般没有人用。

很多人认为 **errorlevel** 里，成功用 0 表示，失败用 1 表示，实际上，**errorlevel** 的值可以是 0~255 中的任意一个数值。看下面的例子就明白了（假如我们的计算机没有 **Z** 盘）：
程序例子：

```
@echo off
dir c:
if errorlevel 1 goto 1
if errorlevel 0 goto 0
exit
:0
echo 命令执行成功!
goto exit
:1
echo 命令执行失败!
goto exit
:exit
pause
```

上面这段程序是正确的，可行的，执行成功就会显示成功，执行失败就会显示失败。

程序稍微改动一下：

```
@echo off
dir Z:
if errorlevel 0 goto chenggong
if errorlevel 1 goto shibai
exit
:chenggong
echo 命令执行成功!
goto exit
```

```
:shibai
echo 命令执行失败!
goto exit
:exit
pause
```

这个程序无论执行成功与否，都显示“命令执行成功!”，很多人对此困惑不已，为什么呢？
原因出在 IF [NOT] ERRORLEVEL number do command 这个语句。if errorlevel 0 goto chenggong 的确切含义是：如果 errorlevel 返回值大于或等于 0，就 goto chenggong。由于 errorlevel 的取值范围是 0~255，所以无论程序执行结果如何，“if errorlevel 0”总是成立，都会 goto chenggong。那么，为什么上面程序只是“if errorlevel 0 goto chenggong”和“if errorlevel 1 goto shibai”交换了一下位置，就不会出现这个程序错误呢？原来，如果先执行“if errorlevel 1 goto shibai”表示“errorlevel 返回值大于或等于 1，那么就 goto shibai”，假如 errorlevel 返回值等于 1，那么就直接 goto shibai 了，程序执行结果正确；假如 errorlevel 返回值等于 0，0 不大于 1 也不等于 1，于是程序没有执行 gogo shibai ，继续执行下一语句“if errorlevel 0 goto chenggong”，此时的判断是“如果 errorlevel 返回值大于或等于 0，就 goto chenggong”。上面说过，无论程序返回值是什么，“if errorlevel 0”总是成立，当然就执行 goto chenggong 了，程序执行结果也是正确的。

运用 **IF [NOT] ERRORLEVEL number do command**，由于其确切含义与我们的思维习惯有点不一样，容易犯错，所以我们可以改用
IF %ERRORLEVEL% == number do command
这种语句形式，确切比对两个数字是否一样，就不会出现这种问题了。
程序修改如下：

```
@echo off
dir c:
if %errorlevel% == 0 goto 0
if %errorlevel% == 1 goto 1
:0
echo 命令执行成功!
goto exit
:1
echo 命令执行失败!
goto exit
:exit
pause
```

这里，两个 **IF** 语句无论怎么交换位置，程序总能判断准确。
IF %ERRORLEVEL% == number do command 和 **IF [NOT] string1==string2 do command** 两个语句是一样的，就是比较确切的两个数字或字符串是否完全一致。
上面的所有程序例子，用 **if ... else...**形式，可以简洁点。用上面形式，是为了方便读者理清思路。

(2)IF [NOT] string1==string2 do command

注意：比对字符时，大小写也会严格比对。
if string1==string2 do command
如果 **string1** 和 **string2** 相同，则执行 **command**
if not string1==string2 do command
如果 **string1** 和 **string2** 不同，则执行 **command**
这里的 **string1** 和 **string2** 两个字符串，比对时，大小写也会严格比对，除非用“**/i**”开关忽略大小写才会认为“**A=a**”。
if /i string1==string2 do command
忽略大小写，如果 **string1** 和 **string2** 相同，则执行 **command**
if /i not string1==string2 do command
忽略大小写，如果 **string1** 和 **string2** 不同，则执行 **command**

上面的等号必须是两个==，表示绝对等于的意思，否则就是赋值符号。“==”前面和后面可以有空格，也可以没有空格。
特别要注意的是，当 **string1** 或者 **string2** 中任何一个字符串出现了空格，那么比较就不能继续下去，程序会自动退出。怎么办呢？可以采用这种形式：
if [not] "string1"=="string2" command
对两个字符串都加上双引号后，即使是空格也会被当成符号来进行比较了。所有的比对应当采用这种添加双引号的形式，确保程序万无一失。
但假如我们要比较两个纯数字的大小时，就不宜添加双引号了，否则计算机会认为比对的是字符串而非数值，就会出现“9>12”这种情况。
上面的所有 **string** 仅代表格式而已，实际上我们对于变量 **string1** 和 **string2** 书写的形式是：
if [not] %string1%==%string2% command
if [not] "%string1%"=="%string2%" command

例如：

```
@echo off
SET /P a=请输入 string1
set /p b=请输入 string2
if %a%==%b% goto qingkuang1
if not %a%==%b% goto qingkuang2
exit
:qingkuang1
echo  两个字符串相同
pause
exit
:qingkuang2
echo  两个字符串不同
pause
exit
```

这个程序，我们假如输入“**abcd**”和“**efga**”两个字符串，程序可以顺利执行，并显示“两个字符串不同”。
假如输入“**abc de**”和“**ae**”，程序会停止执行并自动关闭，因为“**abc de**”字符串有空格，比对无法进行，程序自动关闭。我们更改如下：

```
@echo off
SET /P a=请输入 string1
set /p b=请输入 string2
if "%a%"=="%b%" goto qingkuang1
if not "%a%"=="%b%" goto qingkuang2
:qingkuang1
echo  两个字符串相同
pause
exit
:qingkuang2
echo  两个字符串不同
pause
exit
```

那么输入“**abc de**”和“**ae**”后，程序可以顺利执行并显示：两个字符串不同。

比较两个字符串时，保险起见，一定要记得添加双引号！

程序例子：
@echo off
set string1=abcdefg
set string2=1234567
if not %string1%==%string2% (echo 不同) else (echo 相同)
pause
输出为“不同”。

(3)IF [NOT] EXIST filename do command
用于判断存在情况。
if exist filename do command　　如果存在，执行 **command**
if not exist filename do command　　如果不存在，执行 **command**

程序例子：
if exist "C:\abc.txt" echo 文件存在!
if not exist "C:\abc.txt" echo 文件不存在!

程序例子：
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。
if not exist C:\Progra~1\Tencent\AD*.gif exit
如果不存在那些 **gif** 文件则退出脚本。

IF 增强的用法：
IF [/I] string1 compare-op string2 command
IF CMDEXTVERSION number command
IF DEFINED variable command
其中**/I** 是忽略大小写，有了**/I**，程序会认为 **A=a**
IF [/I] string1 compare-op string2 command

因为我们常用的数学符号(大于号、小于号等)在批处理里已经有特殊意义与用途了，所以批处理不得不重新定义大小符号：

equ	**等于**
neq	**不等于**
lss	**少于**
leq	**少于或等于**
gtr	**大于**
geq	**大于或等于**
/i	**忽略大小写**

例如：

```
@echo off
set /p a=请输入 string1
set /p b=请输入 string2
if %a% equ %b% goto qingkuang1
if %a% lss %b% goto qingkuang2
if %a% gtr %b% goto qingkuang3
exit
:qingkuang1
echo string1 等于 string2
pause
goto end
:qingkuang2
echo string1 小于 string2
pause
goto end
:qingkuang3
echo string1 大于 string2
pause
:end
exit
```

我们输入 923 和 4567 后，会显示：**string1** 小于 **string2**
输入 **abc**912 和 **abb**123 后，会显示：**string1** 大于 **string2**
也就是说，如果我们使用不带双引号的格式，**if %a% equ %b% goto qingkuang1**，那么，计算机在比较两个纯数字的字符串时，会自动将字符转换为数值进行比较。对于包含字母数字的字符串则不转换，并且从左到右，只对比第一个不同的字符的大小。
如果我们采用 **if "%a%" equ "%b%" goto qingkuang1** 带双引号的写法，即使输入的两个字符串都是纯数字，计算机也不会自动将字符转换为数值进行比较。于是假如我们输入的两个字符串为 9 和 12，计算机会判断 9 大于 12 的结论。所以说，究竟要不要带双引号，要根据具体情况而定。

我们有时候需要忽略大小写进行比较，可以这么写：

```
if /i "%a%"=="%b%" echo  两个字符串相同
```

这时，计算机会认为 **abc** 和 **ABC** 是相同的。

@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

IF CMDEXTVERSION number command
看看程序例子就明白了：
@echo off
xcopy C:\wepkeys.txt D:
if cmdextversion 1 **echo** 文件拷贝失败
if cmdextversion 0 **echo** 文件拷贝成功
pause
这里，**cmdextversion** 条件的作用跟 **errorlevel** 一样。
++
常用 **errorlevel** 返回值：
backup
0 备份成功
1 未找到备份文件
2 文件共享冲突阻止备份完成
3 用户用 **ctrl-c** 中止备份
4 由于致命的错误使备份操作中止

diskcomp
0 盘比较相同
1 盘比较不同
2 用户通过 **ctrl-c** 中止比较操作
3 由于致命的错误使比较操作中止
4 预置错误中止比较
diskcopy
0 盘拷贝操作成功
1 非致命盘读/写错
2 用户通过 **ctrl-c** 结束拷贝操作
3 因致命的处理错误使盘拷贝中止
4 预置错误阻止拷贝操作
format
0 格式化成功
3 用户通过 **ctrl-c** 中止格式化处理
4 因致命的处理错误使格式化中止
5 在提示“**proceed with format（y/n）?**”下用户键入 **n** 结束
xcopy
0 成功拷贝文件
1 未找到拷贝文件
2 用户通过 **ctrl-c** 中止拷贝操作
4 预置错误阻止文件拷贝操作 5 拷贝过程中写盘错误

IF DEFINED variable command
如果变量已定义运行 **command** 的命令
或者说，如果变量存在，则运行 **command** 命令。
常用来判断变量是否存在。

程序例子一：
```
@echo off
set str1=ok
set str2=no
if defined str1 (echo str1 已经被定义) else echo str1 没有被定义
if defined str2 (echo str2 已经被定义) else echo str2 没有被定义
if defined str3 (echo str3 已经被定义) else echo str3 没有被定义
pause
```

程序例子二：
读入任意字符串，计算并输出字符串的长度：

```
@echo off
set num=0
set /p str=请输入任意长度的字符串:
if not defined str (
    echo  您没有输入任何内容
    pause>nul & exit
)
:len
set /a num+=1
set str=%str:~0,-1%
if defined str goto len
echo  字符串长度为：%num%
pause>nul
```

循环结构
if 和 **goto** 可以合起来构成循环结构
程序例子：

```
@echo off
set num=1
:begin
if not %num%==5 (
    echo %num%
    set /a num+=1
    goto begin
)
pause
```

输出为：

```
1
2
3
4
按任意键继续.....
```

从上面例子可以看出，**if** 是先判断条件，条件为真才执行。如果一开始条件不成立，那么程序一次也不执行。

IF ELSE 语句

因为 **if** 才是批处理命令，而 **else** 不是，我们无法单独执行 **else**。而批处理的执行是认行不认命令数目，所以 **else 必须和 if 在同一行上！**

左括号“**(**”不是批处理命令，我们无法单独执行“**(**”，所以左括号必须和上面命令同行。而右括号“**)**”是回应前面离它最近的左括号，所以它可以位于程序的任何地方。

注意：**if else** 语句必须用括号，例如：

```
if %var%==A (echo YES) else (echo NO)
```

测试了一下，前面一对括号是必须的，后面那对括号可有可无，但为了程序的美观性，建议后面括号也写上。

程序例子：

设计一个程序，判断我们的输入的是不是字母 **a**

```
@echo off
set /p word=请输入一个字母：
if /i "%word%" == "a" (echo a) else (echo  非字母 a)
pause
```

这是一个最简单的 **if else** 程序例子，下面是注解。

```
@echo off
::注意，word 与后面的等号之间不能有空格！否则赋值不成功！
set /p word=请输入一个字母：
if /i "%word%" == "a" (echo a) else (echo  非字母 a)
pause
```

程序例子：

```
if exist filename (
    del filename
) else (
    echo filename missing
)
```

else 子句必须在 **if** 之后出现在同一行上。

判断输入是不是字母 **a** 或 **b**：

```
@echo off
set /p word=请输入一个字母：
if /i "%word%" == "a" (echo a) else (if /i "%word%" == "b" (echo
b) else echo  非 A 也非 B)
pause
```

程序中，**else** 里面还嵌套一个 **if** 判断语句，我们可以用括号括起来。一般不提倡这种写法，因为如果嵌套多了的话，程序就不太容易阅读。正规的书写格式如下：

```
@echo off
set /p word=请输入一个字母：
if /i "%word%" == "a" (echo a) else (
    if /i "%word%" == "b" (echo b) else echo not A an not B
)
pause
```

注意：**else** 后面一定要跟一个左括号，提示程序后面内容是包含在 **else** 里面的，否则左括号放到下一行程序头，程序就出错了。

来一个复杂点的例子：

```
@echo off
set /p word=请输入一个字母：
if /i "%word%" == "a" (echo a) else (if /i "%word%" == "b" (echo
b) else (if "%word%" == "c" (echo c) else (if "%word%" == "d"
(echo d) else echo  输入非 ABCD 字母。  )))
pause
```

写成下面为：

```
@echo off
set /p word=请输入一个字母：
if /i "%word%" == "a" (echo a) else (
    if /i "%word%" == "b" (echo b) else (
        if /i "%word%" == "c" (echo c) else (
            if /i "%word%" == "d" (echo d) else echo not ABCD
word.
        )
    )
)
pause
```

使用 **if** 需要注意的地方：

在 **DOS** 中，**if** 每次只能判断一个条件，无法判断多条件，也就是无法进行与、或、异或等判断。
如果要用到“与”、“或”条件判断，程序可以这么写：

与：**if** 条件 **A**　**if** 条件 **B doing something**
或：**if** 条件 **A**　**doing something**　**if** 条件 **B**　**doing something**

这个对于“与”还问题不大，但对于“或”，如果 **do something** 代码比较长，要重复写多遍就挺麻烦了，可是没办法。

注意，**if** 的条件后面一定要紧跟一个空格！

```
if 2 neq 3(
    echo Hello!
    echo world!
)

if 2 neq 3 (
    echo Hello!
    echo world!
)
```

这是两种写法，初学者一粗心就写成上面写法，在条件 2 **neq** 3 后面没有空格，导致程序语法错误，而且因为是括号的缘故，还愣是查不出原因！第二种写法才正确，切记切记！

---

if 语句不会进行运算，所以下面程序：
if %num%%%5== 0 echo %num%可以被 5 整除
这种写法不可以，运行错误。必须分开写。

```
set /a panduan=%num%%%5
if %panduan% == 0 echo %num%
```

可以被 5 整除
分成这两段写就行了。

下面代码这样写是不科学的，就是当我们输入回车的时候，程序会发生错误而自动退出。

```
@echo off
set /p var=(Y/N):
if /i %var%==Y (echo Yes) else (echo No)
pause
```

代码应该这样写：

```
@echo off
set /p var=(Y/N):
if /i "%var%"=="Y" (echo Yes) else (echo No)
pause
```

那么输入回车，也会正确执行，输出 **NO**。

另外，带双引号还有一个好处，就是避免对比的字符串双方出现空格，没有双引号就无法比对下去的情况。

例如：

```
@echo off
set str1=ABC DEF
set str2=ABC DEF
if /i "%str1%"=="%str2%" (echo Yes) else (echo No)
pause
```

在这里，因为两个字符串存在至少一个空格，所以双引号是必须的，否则程序执行出错，自动关闭。

```
if /i "%var%"=="abc" (commandA) else (commandB)
```

这种情况下一定要注意，**commandA** 一定要用括号！否则程序出错。而 **commandB** 则可有可无，不过我们还是用括号括起来，美观而且便于阅读。

批处理的延迟变量

要想进阶，延迟变量是必过的一关，稍微复杂点的批处理程序几乎都会用到延迟变量！

命令：

setlocal enabledelayedexpansion

中文翻译：扩展本地环境变量延迟

解释：

set：设置	**local**：本地（环境变量）	
enable：能够	**delayed**：延迟	**expansion**：扩展
setlocal enabledelayedexpansion 就是扩展本地环境变量延迟。		

点评：数了一下共 30 个字母，对记忆是个不小的负担！晕死，这么长的单词组合体，用每个单词的第一个字母代替**(SLEDE)**不好吗？！

注意：使用了“**setlocal enabledelayedexpansion**”后，如果在复合语句之外引用变量，则使用**%var%** 或 **!var!**都是可以的。若想在复合语句中引用复合语句即时得到的变量，则必须使用**!var!**。如果在复合语句中还是使用**%var%**变量，那么得到的变量将是复合语句之前 **var** 的值，此时如果 **var** 在复合语句之前没有定义，那么值为空值。

延迟变量应用实例：

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

这个程序输出是 4 而不是 5。批处理读取命令时是按行读取的，在处理之前要完成必要的预处理工作，这其中就包括对该命令行中的变量赋值。在运行到“**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** 赋值，那么**%a%**当然就是 4 了！（没有为什么，批处理就是这么做的）解决这个问题的办法就是批处理变量延迟：

```
setlocal enabledelayedexpansion
```

程序例子：

```
@echo off
setlocal enabledelayedexpansion
set a=4
set a=5 & echo !a!
pause
```

这个就可以输出 5 了。注意：变量要用一对感叹号括起来。看了这个是不是晕晕的？你肯定会认为，代码这样写不就行了：

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

输出为 5，正确。

确实，这样写是没什么问题，程序也很简洁。实际上，延迟变量不是可有可无的，在某些情况下，没有延迟变量，程序就无法正确运行。接下来的问题就是：变量延迟有什么作用？什么情况下要使用变量延迟？

详细解释：
在什么时候需要延迟变量？该如何引用延迟变量？这是大多数新手很纳闷，迫切想要知道的问题。
要想了解延迟变量，首先要明白什么是“复合语句”，所谓“复合语句”就是指一对()里的所有命令。比如 **for** 的 **do** 后面，如：

```
for /f "delims=" %%i in (a.txt) do (
    set var=%%i
    echo %%i
    set num=%%i
)
```

这里 **do** 后面的三句命令，在一对( )里面，这就叫“复合语句”，当然不止 **for** ，**if** 等等也有，如：

```
if "%var%"=="abc" (
    echo ok
    set lis=123
    echo %lis%
)
```

没有命令，纯粹的括号里的语句也是复合语句：

```
(
set var=1949
set tem=0
set /a var/=2
set /a tem+=1
)
```

上面四个语句用括号括起来了，之前没有类似 **for** 或 **if** 等命令，它们也是复合语句！如果要实时得到 **var** 和 **tem** 的值，就必须使用**!var!**和**!tem!**形式。反正就是凡是()里的所有语句，就叫“复合语句”
这里还要注意一下，括号里的语句无论只有一句还是多句，都算是复合语句。

例如：

```
@echo off
setlocal enabledelayedexpansion
set num=100
for /l %%i in (1,1,%num%) do (set S%%i=1)
for /l %%i in (1,1,%num%) do (echo !S%%i!)
pause
```

这里的**(echo !S%%i!)**，虽然只有一句，但它仍然是复合语句，那么用**!S%%i!**才能回显实时的数值，否则回显**%S%%i%**的值全都是空值，无法显示。

另外：这也是复合语句 **set abc=123 & echo %abc%**
即通过管道命令**&**连接起来的命令，也是复合语句。

复合语句：一对括号里( )的所有命令是复合语句；通过管道命令&、&&、|、||连接起来的命令也是复合语句。

批处理有个很吐血的缺点：计算机无法实时获取复合语句内的变量！%var%获得的是复合语句前变量的值，要想实时获得复合语句里变量的动态值，就要使用延迟变量，并且符号要要用!var!，所以变量延迟适用范围是：如果在复合语句中，并且要实时获得变量的动态值，就要使用延迟变量。

cmd 在处理“复合语句”的时候，如果“复合语句”中用到了变量，会把变量的值当作复合语句之前变量的值来引用。如果在此之前变量没有被赋值，就把它当成空值。

例子 1：

```
@echo off
for /l %%i in (1 1 10) do (
    set var=%%i
            echo %var%
)
pause
```

运行上面的代码，显示什么？显示 10 个 **echo** 处于关闭状态。按照逻辑，**var** 的值应该依次是 1、2、3.....10 才对啊！这就是因为没有开启 延迟变量的缘故，**cmd** 把 **var** 的值当作复合语句之前的值来引用， 而在本例中，复合语句之前并没有给 **var** 定义，所以 **var** 的值是空的，所以会显示 10 个 **echo** 处于关闭状态。

例子 2：

```
@echo off
set var=abc
for /l %%i in (1 1 10) do (
      set var=%%i
      echo %var%
)
pause
```

运行上面的代码，会显示什么？全部显示 **abc**，因为复合语句之前给 **var** 赋值 **abc**，所以没有延迟变量的情况下，**var** 值一律为复合语句之前的 **abc**。

例子 3：

```
@echo off
set var=abc
for /l %%i in (1 1 5) do (
	set var%%i=%%i
	echo %var%
)
echo %var1% %var2% %var3% %var4% %var5%
pause
```

运行上面代码，显示结果如下：

```
abc
abc
abc
abc
abc
1 2 3 4 5
```

这说明什么呢？说明，在复合语句中，并不是没有给变量赋值，只是你若没有开启延迟变量，你就没法在复合语句中提取到它，要等复合语句运行完毕后，才能提取到。也就是说，在复合语句中提取变量值，变量值是复合语

句前的值，等到复合语句运行完毕后，我们此时可以正确提取变量在复合语句中所赋的值。

变量的表示方法：两种：　1、%var%　2、!var!
第一种表示方法是普通情况下的表达法，第二种就是引用延迟的变量。
在开启了延迟变量的情况下，如果在复合语句之外，用哪种方法表示都可以。 但是你若要在复合语句中引用复合语句即时得到的变量，就要用第二种方法。

```
@echo off
setlocal enabledelayedexpansion
set var=abc
for /l %%i in (1 1 5) do ( set var%%i=%%i )
echo %var1% %var2% %var3% %var4% %var5%
echo !var1! !var2! !var3! !var4! !var5!
pause
```

运行结果显示：

```
1 2 3 4 5
1 2 3 4 5
```

说明使用 **setlocal enabledelayedexpansion** 延迟变量后，在复合语句之外使用**%var%** 和 **!var!**是等效的。但在复合语句之内，要延迟变量，就必须使用**!var!**符号，否则使用**%var%**符号，得到的只是复合语句之前给变量 **var** 所赋的值。

例子 4：

```
@echo off
setlocal enabledelayedexpansion
set var=abc
for /l %%i in (1 1 10) do (
	set var=%%i
	echo %var%
	echo !var!
)
pause
```

注意：例子中有两个 **echo** 一个是显示 **%var%** 一个是显示 **!var!**，结果很明白了，**%var%** 显示的结果是复合语句之前变量 **var** 的值，而 **!var!** 显示的就是复合语句中即时得到的值。

延迟变量高级技巧：不用中间变量，交换两个变量值

```
@echo off
set var1=abc
set var2=123
echo 交换前：var1=%var1%  var2=%var2%
set var1=%var2% & set var2=%var1%
echo 交换后：var1=%var1%  var2=%var2%
pause
```

上面程序，如果中间的 **set var1=%var2%&set var2=%var1%**分开成两段来写，就达不到交换数据的效果了。

如果被交换的两个变量都是数据，那么用“异或”位运算也可以实现：

```
@echo off
set a=10
set b=1
set /a a=a^^b
set /a b=a^^b
set /a a=a^^b
echo a=%a% b=%b%
pause
```

如果两个数据都是数字，还可以用数学原理来实现：

```
@echo off
set var1=123
set var2=456
echo var1=%var1%
echo var2=%var2%
set /a var1=%var1%+%var2%
set /a var2=%var1%-%var2%
set /a var1=%var1%-%var2%
echo var1=%var1%
echo var2=%var2%
pause
```

延迟变量的一个实际应用例子：

读入一个起始数字和一个终止数字，连续输出从起始数字到终止数字之间的所有数字，数字与数字之间要有空格，程序如下：

```
@echo off
setlocal enabledelayedexpansion
set /p beg=请输入起始数字：
set /p end=请输入最终数字：
for /l %%i in (%beg%,1,%end%) do (
    set num=             %%i
    set /p var=!num:~-7!<nul
    set /a var=%%i%%10
    if !var!==0 echo.
)
pause>nul
```

输出为：

```
 1   2   3   4   5   6   7   8   9  10
11  12  13  14  15  16  17  18  19  20
21  22  23  24  25  26  27  28  29  30
31  32  33  34  35  36  37  38  39  40
41  42  43  44  45  46  47  48  49  50
```

这个程序没有开启变量延迟是无法实现的。本程序不仅上下数字会靠右对齐，而且每输出 10 个数字就自动换行。

批处理里，还有一个命令 **call** 也可以用来延迟变量：
例子：

```
@echo off
set num=1
set num=2 & call echo %%num%%
pause
```

从上面可见：
如果使用 **setlocal enabledelayedexpansion** 语句来延迟变量，变量符号就要改用感叹号；
如果使用 **call** 语句，就要在原来命令的前部加上 **call** 命令，并把变量引用的单层百分号对改为双层。
因为 **call** 语句使用的是双层百分号对，容易使人犯迷糊，而且每个命令前要添加 **call**，比较繁琐，所以用得较少，常用的是使用 **setlocal enabledelayedexpansion** 语句来延迟变量。

批处理中的变量与参数

批处理中的变量，可以分为两类，分别为“系统变量”和“自定义变量”。

一、系统变量
它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它们定义了值，不需要我们来给它赋值，我们只需要调用就可以了。它们全部如下：

%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。也就是获得当前路径，并将其转换为字符串。
%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	返回计算机名称。
%COMSPEC%	系统	返回命令行解释器可执行程序的准确路径。也就是返回 **cmd.exe** 的路径，一般在 **C:\WINDOWS\system32\cmd.exe**。
%DATE%	系统	返回当前日期字符串。和使用 **date/t** 效果一样。
%ERRORLEVEL%	系统	返回上一条命令的错误代码。通常用 0 表示正确，非零表示错误。
%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数目(所有 **CPU** 的总核心数)。
%OS%	系统	返回操作系统名称。
%PATH%	系统	指定可执行文件的搜索路径。也就是在这些目录下的可执行文件（不仅仅是**.exe**，可以用 echo %PATHEXT%查看哪些属于可执行文件。）可以直接在开始-->运行里直接执行，当然也可以再命令提示符、批处理中直接执行。例如记事本文件位于 **C:\WINDOWS\NOTEPAD.EXE**，那么我们点击“开始-->运行，输入 **NOTEPAD**”就可以打开记事本了。或者我们打开 **CMD** 窗口，直接输入 **NOTEPAD** 也可以打开记事本。
%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。返回值为 **x86** 或 **IA64** 或 **RISC**。这些都是常见的架构，或者称作指令集。**Windows** 操作系统都是基于 **x86** 架构开发的，国产 **CPU** 不是采用 **x86** 指令集，所以无法运行 **Windows**。
%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型号。
%PROCESSOR_REVISION%	系统	返回处理器版本号。
%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 operation system** 根目录(即系统根目录）的驱动器。
%SYSTEMROOT%	系统	返回 **Windows server operation system** 根目录位置。
%TEMP%和%TMP%	系统 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	返回当前时间字符串。使用与 **time /t** 命令相同的格式。
%USERDOMAIN%	本地	返回包含用户账户的域的名称。
%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。

测试代码：

```
@echo %ALLUSERSPROFILE%
pause>nul
```

文件名为《**ABC.bat**》，放在桌面上运行结果如下：

%ALLUSERSPROFILE%
　　C:\Documents and Settings\All Users
%APPDATA%
　　C:\Documents and Settings\Administrator\Application Data
这个文件夹存储的是某些软件的 **MSI** 安装文件，一般不需要删除它们，因为有些软件运行时会对此有需要。如果你确实需要删除也可以，因为这里面的数据一般来说是用不到的，只是一个自动备份。（这个仁者见仁智者见智，有说可以删除的，有说不可以删除的，有待进一步研究）。
%CD%
　　C:\Documents and Settings\Administrator桌面
　　也就是当前的路径。
%CMDCMDLINE%
　　cmd /c ""C:\Documents and Settings\Administrator桌面**\ABC.bat" "** 也就是返回当前启动 **CMD** 的命令行的完整路径（包括文件名），假如我们是在开始-->运行-->输入 **CMD** 后再输入 **echo %CMDCMDLINE%** ，返回的值为：**"C:\WINDOWS\system**32**\cmd.exe"**

%CMDEXTVERSION%
　　2
%COMPUTERNAME%
　　NXNVOC3**PHMAIKF**7
%COMSPEC%
　　C:\WINDOWS\system32**\cmd.exe**
%DATE%
　　2013-08-30 星期五
%ERRORLEVEL%
　　0
%HOMEDRIVE%
　　C:
%HOMEPATH%
　　\Documents and Settings\Administrator

%HOMESHARE%
　　ECHO 处于关闭状态。
%LOGONSERVER%
　　\\NXNVOC3PHMAIKF7

%NUMBER_OF_PROCESSORS%
　　2
%OS%
　　Windows_NT　仅仅一个 **Windows_NT**，我们还是无法判断具体是什么操作系统，目前微软已推出 14 个 **Windows NT** 操作系统，分别是：

Microsoft Windows NT 3.1 **(**1993**)**
Microsoft Windows NT 3.5 **(**1994**)**
Microsoft Windows NT 3.51 **(**1995**)**
Microsoft Windows NT 4.0 **(**1996**)**
(从 5.0 版开始，**Windows NT** 只是简单地称为 **Windows** 了，下面的版本是后来的版本**)**
Microsoft Windows 2000 **(Windows NT** 5.0**) (**1999**)**
Microsoft Windows XP (Windows NT 5.1**) (**2001**)**
Microsoft Windows Server 2003 **(Windows NT** 5.2**) (**2003**)**
Microsoft Windows Server 2003 **R**2 **(Windows NT** 5.2**) (**2006**)**
Microsoft Windows Vista (Windows NT 6.0**) (**2006**)**
Microsoft Windows Server 2008 **(Windows NT** 6.0**) (**2008**)**
Microsoft Windows 7 **(Windows NT** 6.1**) (**2009**)**
Microsoft Windows Server 2008 **R**2 **(Windows NT** 6.1**) (**2009**)**
Microsoft Windows 8 **(Windows NT** 6.2**) (**2012**)**
Microsoft Windows Phone (WP8**) (Windows NT** 6.2**) (**2012**)**
Microsoft Windows Server 2012 **(Windows NT** 6.2**) (**2012**)**
Microsoft Windows Codename "blue" (Windows 8.1**) (Windows NT** 6.3**) (**2013**)**
Microsoft Windows Codename "9**" (Windows ?)(**目前微软有开发该系统的计划，预计 2016 年推出**)**
Windows NT 4.0 **Workstation** 的启动画面是晚上。
）

%PATH%
　　C:\WINDOWS\system32**;C:\WINDOWS;C:\WINDOWS\System**32**\Wbem;C:\Program Files\Microsoft SQL Server**100**\Tools\Binn\;C:\Program Files\Microsoft SQL Server**100**\DTS\Binn\;D:\QuickTime\QTSystem**

%PATHEXT%
.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH
%PROCESSOR_ARCHITECTURE%
x86
%PROCESSOR_IDENTFIER%
ECHO 处于关闭状态。
%PROCESSOR_LEVEL%
6
%PROCESSOR_REVISION%
2**a**07
%PROMPT%
PG
%RANDOM%
15150
%SYSTEMDRIVE%
C:
%SYSTEMROOT%
C:\WINDOWS
%TEMP%和**%TMP%**
C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp
%TIME%
11:22:06.64
%USERDOMAIN%
NXNVOC3PHMAIKF7
%USERNAME%
Administrator
%USERPROFILE%
C:\Documents and Settings\Administrator
%WINDIR%
C:\WINDOWS
以上所有变量，都是系统预定好的，需要用时，直接调用就行了。

应用例子：利用**%PROCESSOR_ARCHITECTURE%**判断当前计算机是几位的：
@echo off
if "%PROCESSOR_ARCHITECTURE:~0,3%" equ "x86**" (echo** 32 位计算机**) else echo** 64 位计算机
pause>nul
这里有个小小的问题，就是中文不能是“32 位”，“64 位”，否则在 **txt** 保存运行后，代码“位”会变成希腊字母“λ”。并且，如果你再

次改动代码的任何一个字符，或者将代码的任何一个字符删除后原原本本打回去，程序永远都无法再正确运行了。这个是记事本和中文编码兼容的问题而不是 DOS 的 BUG。

%ERRORLEVEL%值一览表

ATTRIB.EXE	
(a) Target file/folder not found	= ERRORLEVEL 1
(b) Invalid switch	= ERRORLEVEL 1
(c) Sharing violation (target file in use)	= ERRORLEVEL 1
(d) Invalid drive specification	= ERRORLEVEL 1
(e) Drive not ready (either Abort or Fail reply)	= ERRORLEVEL 1
(f) Invalid/wrong number of parameters	= ERRORLEVEL 1

CHOICE.COM	
(a) With reply list, and reply nn in list (This reply list position = return code is main use of CHOICE)	= ERRORLEVEL nn
(b) Invalid switch	= ERRORLEVEL 255
(c) Invalid switch syntax	= ERRORLEVEL 255
(d) Timeout default not in reply list	= ERRORLEVEL 255
(e) choice /?	= ERRORLEVEL 255

CSCRIPT.EXE	
(a) Cannot find script file	= ERRORLEVEL 1
(b) No script engine for file extension	= ERRORLEVEL 1
(c) No file extension in script file	= ERRORLEVEL 1
(d) Drive not ready (There is no Abort, Retry, Fail stall)	= ERRORLEVEL 1

DELTREE.EXE	
(a) Required parameter missing	= ERRORLEVEL 1
(b) Invalid switch	= ERRORLEVEL 1
(c) deltree /?	= ERRORLEVEL 1
(d) Not ready reading drive (Abort reply) (Note: Fail reply returns ERRORLEVEL 0)	= ERRORLEVEL 18

EXTRACT.EXE	
(a) Invalid switch	= ERRORLEVEL 1

FC.EXE	
(a) Insufficient number of filespecs	= ERRORLEVEL 1
(b) Too many filenames on command line	= ERRORLEVEL 1
(c) Sharing Violation + Abort reply (Note: Fail reply returns ERRORLEVEL 0)	= ERRORLEVEL 5
(d) Drive not ready (Note: Fail reply returns ERRORLEVEL 0)	= ERRORLEVEL 18

FIND.EXE	
(a) Target string found (=f0und)	= ERRORLEVEL 0
(b) Target string missing (=m1ssing)	= ERRORLEVEL 1
(c) find /?	= ERRORLEVEL 1
(d) Parameter format not correct	= ERRORLEVEL 2
(e) Specified file to search not found	= ERRORLEVEL 2
(f) Specified file in use + Fail reply	= ERRORLEVEL 2
(g) Drive not ready + Fail reply	= ERRORLEVEL 2
(h) Specified file in use + Abort reply	= ERRORLEVEL 5
(i) Drive not ready + Abort reply	= ERRORLEVEL 5

FORMAT.COM	
(a) Drive not ready (There is no Abort, Retry, Fail stall)	= ERRORLEVEL 4

FTP.EXE	
(a) Brief help (use: ftp -h for Brief help)	= ERRORLEVEL 2
(b) Error opening script file (file missing)	= ERRORLEVEL 2
(c) Invalid switch	= ERRORLEVEL 2

KEYB.COM	
(a) Invalid switch	= ERRORLEVEL 1
(b) Invalid keyboard code specified	= ERRORLEVEL 1
(c) keyb /?	= ERRORLEVEL 1

MEM.EXE	
(a) Invalid switch	= ERRORLEVEL 1

MODE.COM	
(a) Invalid parameter	= ERRORLEVEL 1
(b) Invalid switch	= ERRORLEVEL 1

MORE.COM	
(a) Invalid switch (MORE doesn't accept switches)	= ERRORLEVEL 1
(b) Drive not ready (Abort reply) (Note: Fail reply returns ERRORLEVEL 0)	= ERRORLELEL 5

MOVE.EXE	
(a) Required parameter missing	= ERRORLEVEL 1
(b) Unable to create destination	= ERRORLEVEL 1
(c) Unable to open source (you see this when trying to MOVE a folder from one drive to another. You need to use XCOPY /S followed by DELTREE, since MOVE won't handle folder moves across drives)	= ERRORLEVEL 1
(d) Sharing violation + Fail reply (Note: file is nevertheless COPIED, not moved, in this case)	= ERRORLEVEL 1
(e) Sharing violation + Abort reply (Note: file is nevertheless COPIED, not moved, in this case)	= ERRORLEVEL 5
(f) Drive not ready (Abort reply) (Note: Fail reply - unusually - returns ERRORLEVEL 1)	= ERRORLEVEL 18

PING.EXE	
(a) Unknown host (usually=name not found on DomainNameServer)	= ERRORLEVEL 1
(b) Brief help (with no parameter) Note: for PING Brief help with /? switch, ERRORLEVEL is 0	= ERRORLEVEL 1
(c) Invalid switch (and displays the Brief help as well)	= ERRORLEVEL 1
(d) Interrupted with [Ctrl-C] & nbsp;	= ERRORLEVEL 255

SORT.EXE	
(a) Invalid switch	= ERRORLEVEL 1
(b) Drive not ready (Abort reply) (Note: Fail reply returns ERRORLEVEL 0)	= ERRORLEVEL 15

START.EXE	
(a) start /? (real mode)	= ERRORLEVEL 1
(b) start /? (GUI)	= ERRORLEVEL 255
(c) Can't find file specified for START	= ERRORLEVEL 255
(d) No file association for specified file	= ERRORLEVEL 255
(e) Drive not ready (There is no Abort, Retry, Fail stall)	= ERRORLEVEL 255

SUBST.EXE	
(a) Invalid parameter	= ERRORLEVEL 1
(b) Invalid switch	= ERRORLEVEL 1
(c) Path not found	= ERRORLEVEL 1
(d) Drive not ready (Abort reply) (Note: Fail reply - unusually - returns ERRORLEVEL 1)	= ERRORLEVEL 21

TRACERT.EXE	
(a) Unable to resolve target system name (usually=name not found on DomainNameServer)	= ERRORLEVEL 1
(b) Invalid switch	= ERRORLEVEL 1
(c) Brief help (no parameter)	= ERRORLEVEL 1
(d) Interrupted with [Ctrl-C] Note: for TRACERT Brief help, type command without parameters	= ERRORLEVEL 255

XCOPY.EXE	
(a) File not found	= ERRORLEVEL 1
(b) Invalid date in /d switch	= ERRORLEVEL 4
(c) Invalid number of parameters	= ERRORLEVEL 4
(d) Invalid parameter	= ERRORLEVEL 4
(e) Device not ready	= ERRORLEVEL 4
(f) Unable to create directory	= ERRORLEVEL 4
(g) System can't find file	= ERRORLEVEL 5

几个常用命令的返回值及其代表的意义：

backup

0 备份成功

1 未找到备份文件

2 文件共享冲突阻止备份完成

3 用户用 **ctrl-c** 中止备份

4 由于致命的错误使备份操作中止

diskcomp

0 盘比较相同

1 盘比较不同

2 用户通过 **ctrl-c** 中止比较操作

3 由于致命的错误使比较操作中止

4 预置错误中止比较

diskcopy

0 盘拷贝操作成功

1 非致命盘读/写错

2 用户通过 **ctrl-c** 结束拷贝操作

3 因致命的处理错误使盘拷贝中止

4 预置错误阻止拷贝操作

format

0 格式化成功

3 用户通过 **ctrl-c** 中止格式化处理

4 因致命的处理错误使格式化中止

5 在提示“**proceed with format（y/n）?**”下用户键入 **n** 结束

xcopy

0 成功拷贝文件

1 未找到拷贝文件

2 用户通过 **ctrl-c** 中止拷贝操作

4 预置错误阻止文件拷贝操作

5 拷贝过程中写盘错误

上面表只是给出常见的一些错误代码，在实际编程中，还可能遇到上面表没给出的代码，具体 **errorlevel** 返回值是多少，我们可以用 **echo %errorlevel%**实际查看。

应用例子：

利用系统变量实现清理系统垃圾：

```
@echo off
title 清除 WindowsXP 系统垃圾
del /f /s /q %systemdrive%\*.tmp
del /f /s /q %systemdrive%\*._mp
del /f /s /q %systemdrive%\*.log
::帮助的临时文件
del /f /s /q %systemdrive%\*.gid
::丢失簇的恢复文件
del /f /s /q %systemdrive%\*.chk
::旧备份文件
del /f /s /q %systemdrive%\*.old
del /f /s /q %systemdrive%\recycled\*.*
::临时备份文件
del /f /s /q %windir%\*.bak
::删除上网预读信息
del /f /s /q %windir%\prefetch\*.*
::直接删除 C:\Windows\temp 文件夹，并新建 temp 文件夹。
rd /s /q %windir%\temp & md %windir%\temp
::删除用户名下的 cookies 文件夹内容。
del /f /q "%userprofile%\cookies\*.*"
::删除最近浏览文件的快捷方式
del /f /q "%userprofile%\recent\*.*"
::清除用户临时页面文件
del /f /s /q "%userprofile%\Local Settings\Temporary
InternetFiles\*.*"
::清除用户临时文件
del /f /s /q "%userprofile%\Local Settings\Temp\*.*"
del /f /s /q "f:\Temporary Internet Files\*.*"
del /f /s /q "%userprofile%\recent\*.*"
::删除 C 盘安装信息文件
del /f /s /q "C:\Program Files\Installshield Installation
Information\*.*"
::删除升级补丁信息
del /f /s /q "C:\WINDOWS\SoftwareDistribution\Download\*.*"
::删除历史记录
rd    /s /q "%userprofile%\Local Settings\History"
dir %SystemRoot%\$*$ /ad/b >%SystemRoot%\vTmp.txt
for /f %%a in (%SystemRoot%\vTmp.txt) do rd /s /q
"%SystemRoot%\%%a"
echo    清除系统垃圾完成！ & pause
```

用%RANDOM%获取随机数。

%random%可以随机获得 0 到 32767 之间的任意一个十进制数字。（32767 用二进制表示为 15 个 1）

例如：

```
@echo off
echo %random%
pause
```

注意，下面的程序只获得一个随机数：

```
@echo off
for /l %%i in (1,1,10) do echo %random%
pause
```

每次执行程序，获得的十个随机数都是一样的。我们应该用延迟变量：

```
@echo off
setlocal enabledelayedexpansion
for /l %%i in (1,1,10) do echo !random!
pause
```

这样每次就可以获得十个不同的随机数。

假如我们要产生指定区间的随机数，可以用这种格式：

%random%%%(max-min+1)+min

产生**[min,max]**区间里的随机数，注：批处理中求模得用两个**%%**符号

程序例子，获得 1 到 10 之间的随机数字：

```
@echo off
set /a num=%random%%%10+1
echo %num%
pause
```

在这里，表达式**%random%**的**%**符号不再是可有可无的了。

程序例子，在当前目录下的 **test.txt** 文本中随机取一行数据

```
@ECHO OFF
SETLOCAL ENABLEDELAYEDEXPANSION
FOR /F "TOKENS=*" %%i IN (TEST.TXT) DO (SET /A h+=1 &
SET r!h!=%%i)
SET /A s=%RANDOM% %% h+1
ECHO !r%s%!
PAUSE
```

随机取值程序

```
@echo off
mode con cols=84 lines=36
::设置窗口大小
title  随机取值程序
::设置窗口标题
color 9f
::设置窗口颜色
set num=0
::取值次数计数器归零
:begin
set k=
echo. & echo.
echo %k%                                                随机取值程序
echo. & echo.
set /p a=%k%请输入目标数字：
set /p min=%k%请输入取值范围最小值：
set /p max=%k%请输入取值范围最大值：
set /a jishu=max-min+1
::计算取值范围大小
if %max% lss %min% (
                        echo.
                        echo %k%取值范围最大值小于最小值，输入错
误，请重新输入。
                        pause > nul
                        cls
                        goto begin
)
echo.
echo %k%您输入的数字是                  %a%
echo %k%取值范围是                      [ %min% , %max% ]
if %a% lss %min% goto tuichu
if %a% gtr %max% goto tuichu
::如果目标数字在取值范围之外，返回并重新输入
echo. & echo.
echo %k%按任意键开始随机取值
echo.
pause > nul
```

```
echo
--------------------------------------------------------------------------------
:xun
set /a b=%RANDOM%%%(max-min+1)+min
::在取值范围内随机取值
set /a panduan=%num%%%10
if %panduan% == 0 echo.
::每输出 10 个数字，就回车换行
set /a num+=1
::计数器加 1
set c=            %b%
set /p=%c:~-7% <nul
if %b% == %a% (
                        echo. & echo. & echo.
                        echo
--------------------------------------------------------------------------------

                        echo %k%本次从 [%min% , %max%] 共
%jishu%个数字中随机取值%num%次，出现了目标数字%a%
                        echo. & echo.
                        echo     按任意键退出
                        pause > nul
                        exit
) else (goto xun)
:tuichu
echo.& echo.
echo %k%目标数字在取值范围之外，无法取值，请重新输入
pause > nul
cls
goto begin
```

系统参数：

%0，%1，%2，%3，%4，%5，%6，%7，%8，%9，%*

不可以改变 **for** 里的循环变量的值。不可以改变参数**%0-%9** 的值。**for** 里的循环变量**%%i** 和参数**%0-%9**，就代表它是变量了，在引用它时，不可以再对齐增加变量符号，即**%%%i%** 、**!%%i!**、**%%1%**、**!%1!**都是不可以的。

对于**%1--%9**

%1　　批处理的第一个参数
%2　　批处理的第二个参数
%3　　批处理的第三个参数
%4　　批处理的第四个参数
%5　　批处理的第五个参数
%6　　批处理的第六个参数
%7　　批处理的第七个参数
%8　　批处理的第八个参数
%9　　批处理的第九个参数

批处理有且只有这九个参数，不存在**%10** 以及以上的参数。

看到这里，令人产生很大的疑问：哪个批处理？本身批处理吗？批处理参数在哪里？究竟怎么用？

要明白参数的具体作用，先看在 **call** 应用的例子，因为 **call** 是允许使用参数的命令。

```
@echo off
call :loop Hello World!
pause>nul & goto :eof
:loop
echo %1
echo %2
```

输出为：

```
Hello
World!
```

上面参数 **Hello World!**，计算机认为是两个参数，如果想让 **hello world** 是一个参数，应该用英文格式双引号引起来。

再看一个用 **call** 带参数递归算阶乘的程序：

```
@echo off
title 阶乘--递归算法
echo. & echo. & echo                                                    阶乘--
递归算法
setlocal enabledelayedexpansion
echo. & echo.
set /p n=请输入一个数：
set result=1
if !n!==0 (echo 结果等于 1 & pause>nul & goto eof) else
(call :loop !n!)
echo. & echo 结果等于!result!
pause>nul
:loop
if not %1==1 (
                        set /a result=!result!*%1
                        set /a x=%1
                        set /a x-=1
                        call :loop !x!
)
```

当然，参数也不仅仅用在 **call** 上，请看下面例子：

在同一个目录下，有两个 **bat** 文件：

123**.bat** 里面的代码分别为：

```
@echo off
start abc.bat 1,2,3,4
call abc.bat a b c d
pause
```

abc.bat 里面的代码为：

```
@echo off
echo %1
echo %2
echo %3
echo %4
pasue>nul
```

假如我们双击运行 **abc.bat**，则执行结果为：
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。

而我们双击执行 123**.bat**，执行结果打开了两个窗口，一个窗口输出
1
2
3
4
另一个窗口输出
a
b
c
d
说明，上面赋予参数时，参数与参数之间用逗号或空格隔开都是可以的。

也许你还会问：知道参数是这么回事，但这有什么实际用途呢？

例 1：**C:**根目录下有一批处理文件名为 **f.bat**，内容为：
@echo off
format %1

如果在 **cmd** 窗口执行
C:\>f.bat 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:**，即格式化 **A** 盘！
从这个例子也看不出**%1-%9** 参数有什么好处，因为我们的代码可以这么改：
@echo off
set /p a=请输入一个驱动器号
format %a%
pause>nul
执行这个文件，程序会提示输入，我们输入 **a:**，程序就会格式化 **a** 盘。
这里，用 **set** 读入的方法比用**%1-%9** 的方法更加简便。

例 2：**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。
这个例子同样不够好，我们同样可以不用**%1-%9** 参数来实现显示文本文件。

看下面一个程序**:**
del /f /q %1
rd /q /s %1
将这段代码保存为《删除**.bat**》，我们随便将一个文件或文件夹拖到其上面放，该程序就会将其彻底删除，从这里可以看出参数的小小作用了吧，嘿嘿。
这里用两个删除命令**del**和**rd**，是为了保证可以删除文件以及文件夹，加上参数，是为了删除前不必经过确认就删除。有关其用法，详见 **del** 和 **rd** 命令专题。

%* 它的作用不是很大，只是返回参数而已，不过它是一次返回全部参数的值，不用输入**%1--%9** 来一个个确定。

程序例子**:**
@echo off
echo %*
将上面程序保存为 **test.bat** 后，放在 **C** 盘根目录下
进入 **CMD**，输入 **cd C:**进入 **C** 盘根目录
输入**:**
test.bat 第 1 个参数 第 2 个参数 第 3 个参数 第 4 个参数
输出结果为
第 1 个参数 第 2 个参数 第 3 个参数 第 4 个参数

注意：连空格也一起输出。

上面说到，参数最多只可以九个，测试程序如下：
看程序：
@echo off
echo %1
echo %10
echo %11
pause>nul
将上面代码保存到 **D** 盘根目录《**a.bat**》，我们在命令行窗口，转到 **D** 盘根目录，输入：
a.bat 1 2 3 4 5 6 7 8 9 10 11
程序输出为：
1
10
11
从这里猛一看，参数似乎可以 10 以及以上。其实非也，我们变化一下输入就知道了，我们输入：
a.bat A B C D E F G H I J K
程序输出为：
A
A0
A1
这里就可以看出，**%**10 以上是无效参数啦。

但我们如果不指定返回的参数，用**%***输出全部参数，那么参数的个数就没有限制啦。
代码例子：

@echo off
echo %*
pause >nul

将上面代码保存为 **E** 盘根目录下《**shuchu.bat**》，然后在 **CMD** 窗口，转到 **E** 盘根目录，输入：
shuchu A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
输出为：
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

%0 这个不是返回参数的值了，有两层意思：
第一层：如果**%0** 前有 **echo** 等命令，则代表返回该批处理所在的绝对路径。
第二层：如果**%0** 前没有任何参数，则代表自身批处理。会无限循环执行 **bat**（也就是调用自身 **bat** 程序）。

程序例子：
@echo off
echo %0
pause
保存为 **test.bat** 后放在桌面运行，会显示如下结果：
"C:\Documents and Settings\Administrator\桌面\test.bat"
注意，上面的显示结果有双引号。不想要双引号，应该这么写：
@echo off
for %%i in (%0) do echo %%~i
pause

程序例子：
@echo off
net user
%0

执行后，它会无限循环执行 **net user** 这条命令，也就是不断显示用户账户信息，直到你手动停止为止。

copy %0 d:\wind.bat
将自身复制到 **D** 盘根目录，并命名为 **wind.bat**，里面的代码就是
copy %0 d:\wind.bat

@echo off
echo Hello World!
copy %0 d:\wind.txt
则将自身复制到 **D** 盘根目录，并命名为 **wind.txt**，里面的代码是
@echo off
echo Hello World!
copy %0 d:\wind.txt

%0 高级用法：
1、最简单的死循环程序，同时也是最简单的批处理程序：
%0
将上面这个代码保存后执行，程序会不断重复显示自身路径，直到手动关闭程序为止。

2、用**%0** 实现嵌套与递归

好多人不知道，不仅仅是 **call** 调用函数的时可以使用参数，**%0** 同样也可以办到！

```
@echo off
set /a var=%1+1
echo %var%
if %var% lss 100 (%0 %var%) else pause
```
显示从 1 开始，显示到 100。

如果是 **call** 命令，代码就变成：
```
@echo off
:loop
set /a var=%1+1
echo %var%
if %var% lss 100 call :loop %var%
pause
```
和上面程序显示效果一样。

但是 **call** 和**%0** 还是有区别的，具体区别详见 **call** 命令专题。

二、自定义变量

一般用 **set** 读入的方法定义和赋值。
程序里设置变量
```
set var=Hello World!
echo %var%
pause>nul
```

读入的方法设置变量：
```
set /p var=请输入一个变量:
echo %var%
pause >nul
```

上面两个例子就是我们常用的设置自定义变量。

提到变量，就不能不说两个符号**%**和**!**
%非常特殊，用法复杂，初学者往往对它理解很迷糊。在批处理里，我们对变量不用像 **C** 语言一样预先定义就可以使用，非常自由。但恰恰是这个看似自由的机制，却常常使我们陷入痛苦之中！那就是，我们没有预先说明哪些符号是变量，计算机如何识别呢？答案就是：在编程过程中，我们对我们所要使用的变量用**%**标明，在延迟变量情况下则用**!**标明。

例子一：

```
@echo off
set num=12
echo %num%
pause
```

例子二：

```
@echo off
setlocal enabledelayedexpansion
set num=12
for /l %%i in (1,1,5) do (
                        set /a num+=10
                        echo !num!
)
pause
```

上面两个例子就是关于常规情况下变量符号**%**和延迟变量情况下变量符号**!**的使用，它们的功能就是告诉计算机该串代码是变量。

不过，**%0-%9** 和**%***这十一个符号中的**%**不是代表变量，而是参数的意思。参数和普通变量不同，不可以改变参数的值。

在 **for** 里的**%%i**，两个**%**号标出的不是普通变量，而是循环变量。循环变量和普通变量不同，不可以改变循环变量的值。

for 里的循环变量**%%i** 和参数**%0-%9**，就代表它是变量了，在引用它时，不可以再对齐增加变量符号，即**%%%i%** 、**!%%i!**、**%%1%**、**!%1!**都是不可以的。

在计算里，**%**表示取余数，例如：

```
@echo off
set num=100
set /a result=num%%7
echo %result%
pause
```

输出 2

在输出格式里，**%**表示输出格式的意思**%str:~m,n%**，这个在 **set** 命令专题里有详细介绍。

总结一下：%应用范围是变量、参数、循环变量、余数计算、格式。

SET

set 不允许变量名含有等号。

一：用 **set** 命令赋值自定义变量
标准格式
SET [variable=[string]]
注意：等号左边不允许有空格，否则出错。等号右边全部都会赋值到 **variable** 变量上，包括空格。比如 **set var=** 123，我们回显 **var**，其值为“ 123”，123 前有一个空格。

程序例子：

```
@echo off
set monkeyking=sunwukong
echo %monkeyking%
pause
```

程序输出为：**sunwukong**

假如有几个变量需要赋的值都是一样的，我们当然可以分别用 **set var=a** 这种形式的式子进行赋值。
还有一种简便的方法，借助**/a** 参数，可以统一进行赋值：**set /a A=B=C=D=**100，执行这条命令后，**A**、**B**、**C**、**D** 四个变量的值都赋值 100。

注意：批处理对变量名大小写不敏感！但对变量值的大小写敏感！
例如：

```
@echo off
set var=123
echo %Var%
echo %VAr%
echo %VAR%
pause
```

输出三个 123。这一点，我们切忌在代码用了 **A** 变量，然后又使用 **a** 变量，这样会造成程序错误，因为计算机认为 **A** 和 **a** 两个变量是相同的。再举一例：

```
@echo off
set /a A=123,set B=456
if %a% lss %b% (echo A 小于 B) else (echo a 大于 b)
pause
```

输出结果为 **A** 小于 **B**。
对变量值大小写敏感的例子：

```
@echo off
set /p A=
if %a% == Y (echo Yes) else (echo No)
pause
```

我们输入 **y** 和 **Y**，计算机判断的结果是不一样的，说明计算机对变量值敏感。但是，用户输入字母时，认为 **A** 等于 **a**，不会去区分大小写。于是，我们就需要关掉大小写敏感开关了：

```
@echo off
set /p A=
if /i %a% == Y (echo Yes) else (echo No)
pause
```

这样，无论我们输入 **Y** 还是 **y**，结果都输出 **Yes**。

但在 **for** 语句里又不一样了，**for** 对循环变量大小写敏感！它区分大小写！
例如：**for /l %%B in** (1,1,10) **echo %%b**，这样写是不行的。

建议我们在用到变量时，严格区分大小写，也就是自始至终同一变量的大小写保持一致，并且用过的大写后就不要再用小写，反之用过小写后就不再用大写。

二：用 **set** 命令读入并赋值到自定义变量。
格式：

```
SET /P variable=[promptString]
```

/P　命令行开关允许将变量数值设成用户输入的一行输入。
注意：等号左边不允许有空格，否则出错。等号右边我们输入的所有字符全部都会赋值到 **variable** 变量上，包括空格。比如我们输入“ 123”，回显 **var**，其值为“ 123”，123 前有一个空格。

程序例子：

```
@echo off
set /p monkeyking=请输入猴王名字：
echo  猴王的名字是%monkeyking%
pause
```

程序在执行时，会提示：“请输入猴王的名字”，我们输入 **sunwukong** 后，显示“猴王的名字是 **sunwukong**”。

三：用 **set** 实现计算

语法：**SET /A expression**
/A 命令行开关指定等号右边的字符串为待计算的数字表达式。该表达式的各种计算符号优先权顺序从高到低为：

符号	注释	优先级	备注
( )	分组	**1**	
!　~　-	一元运算符	**2**	-是负号，表示负数
***　/　%**	算术运算符	**3**	%是求余数运算，例如 100%3=1
+　-	算术运算符	**4**	
<<　>>	二进制逻辑位移	**5**	输入表达式时，符号需要添加双引号
&	二进制按位“与”	**6**	输入表达式时，符号需要添加双引号
^	二进制按位“异”	**7**	输入表达式时，符号需要添加双引号
|	二进制按位“或”	**8**	输入表达式时，符号需要添加双引号
=　*=　/=　%=　+=　-=	算术赋值	**9**	输入表达式时，符号需要添加双引号
&=　^=　|=　<<=　>>=	二进制算术赋值	**10**	输入表达式时，符号需要添加双引号
,	表达式分隔符	**11**	

set 的**/A** 参数就是让 **set** 可以支持数学符号进行运算。所有需要进行计算的表达式，都必须添加**/a** 开关。

注意：

DOS 计算只能精确到整数，小数采用四舍五入。
DOS 计算的有效值范围是-2147483647 至 2147483647，超出无效。

运算符号说明：
( )　　最优先运算，例如(1+2)*3 等于 9，没有括号则等于 7。
!　　逻辑非，例如“!1=0”，“!8=0”，“!0=1”。除了 0 之外的任何数字，取“!”都等于 0，而“!0”等于 1（在 **DOS** 里，字母和符号取非也都等于 1）。
~　　取反，但批处理的操作都是对数字先加 1 后再取反数。例如：“~1=-2”，“~0=-1”，“~-8=7”，批处理简直是胡闹。
-　　负号，表示负数。如果两头都有数字的话，就是减号。
+　-　*　/　数学最基本的加、减、乘、除，具体应用不解释。
%　　取余数，这是一个非常特殊的运算符！
　　我们在用“**set /p a=**请输入一个表达式：”，这种读取用

户输入的形式时，我们在命令行窗口可以输入 **m%n** 形式的表达式。
如果是在批处理文本里，**“set /a num=”**形式的表达式，则要用 **set /a num=m%%n** 形式输入，否则运算无法进行，即使用转义字符也不行！

程序例子：

```
@echo off
set /p var=请输入一个表达式：
set /a result=%var%
echo %result%
pause
```

我们在命令行窗口输入“1984%7”，输出结果为 3，意思是 1984 除以 7，余数是 3。
程序例子：

```
@echo off
set /a result=1984%%7
echo %result%
pause
```

输出为 3。

以上符号，我们在输入计算机时，正常输入就行，不必加双引号。
下面所有符号，无论是用户手动输入到命令行窗口的形式，还是批处理文件里数学表达式的形式，都必须添加双引号，否则计算机无法识别与计算。

<< >>　　位运算符号，我们输入“7">>"1”结果是 3，解释：十进制的 7 用二进制表示为 111，向右移动一位变成 11，11 换算成十进制就是 3。再比如“-3"<<"2”，计算结果为-12。

&　　二进制按位与，两个数从个位对齐，上下两个数字都是 1，结果为 1，否则为 0。例如“77"&"21”结果是 5。解释：77 换算成二进制是 1001101，21 换算成二进制是 10101，按位与后结果为 101，换算成十进制就是 5。

^　　二进制按位异，两个数从个位对齐，上下两个数字相同结果为 0，两个数字不同，结果为 1。例如“141"^"215”计算结果为 90。解释：141 换算成二进制是 10001101，215 换算成二进制是 11010111，将两个数字从个位开始对齐，然后对照上下数字，两个数字相同时就等于 0，两个数字不同时就等于 1，结果就是 01011010，换算成十进制就是 90。

|　　二进制里按位或，两个数从个位对齐，上下两个数字只要有一个等于 1，结果就为 1，否则为 0。例如“73"|"58”，结果为

123。解释：73 换算成二进制是 1001001，58 换算成二进制是 111010，按位或后等于 1111011 换算成十进制就是 123。

=　　*=　　/=　　%=　　+=　　-=　　这六个都是赋值符号，可以简化表达式。例如 **set /a var = %var% + 1** 可以表达为 **set /a var+=1**，这种形式常用于计数器。 例如“**set num=2**”，意思是将 **num** 的值赋值为 2，此时我们“**echo %num%**”，结果为 2。如果我们继续运行“**set num+=5**”，结果为 7。意思是我们将 **num** 的值加上 5，结果再赋值到 **num**，因为一开始 **num** 等于 2，加上 5 后等于 7,7 赋值到 **num** 上去，于是 **num** 的值等于 7。

&=　　^=　　|=　　<<=　　>>=　　这五个符号都是二进制运算赋值符号，在批处理里一般运用不多，用法和上面的算数赋值差不多，都是先运算后赋值。例如“**set num"&="2**”，假如 **num** 的值一开始是 2，那么运算结果是 8。

,　　逗号运算符，分隔不同运算组。

例如运行 **set /a a=1+1,b=2+1,c=3+1** 后，我们 **echo %a%** 显示 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，只是没显示出来而已。

十六进制有 **0x** 前缀， 八进制有 0 前缀，其余数字值都是十进制数字。因此， **0x12** 等于 18 等于 022。请注意八进制公式可能很容易搞混: 08 和 09 是无效的数字，因为 8 和 9 不是有效的八进制位数。凡是位计算，计算机都会换算成二进制运算后还原成十进制显示出来。

set /a var=num+1

在 **set /a** 后面的表达式 **var=num+1**，左边是被赋值变量，不可以使用变量符号，即不可以是**%var%**或**!var!**形式。对于右边的计算式 **num+1**，则变量 **num** 使用变量符号即**%var%**或**!var!**，或不使用变量符号都是可以的。

问：在延迟变量情况下，右边计算表达式不使用变量符号，即不使用**!var!**形式，那么变量 **var** 值是否是实时的动态值？

答：是的，即使不使用**!var!**形式，得到的值和使用**!var!**形式得到的值一样，都是动态值。但是，如果开启了延迟变量，而 **var** 使用**%var%**形式，那么得到的值就不是动态值了。

测试代码：

```
@echo off
setlocal enabledelayedexpansion
set var=0
for /l %%i in (1,1,10) do (
                        set /a var=%%i*2+1
                        set /a tem1=!var!+2
                        set /a tem2=var+2
                        set /a tem3=%var%+2
                        echo tem1=!tem1!    tem2=!tem2! tem3=!tem3!
)
pause
```

上面输出 **tem1** 和 **tem2** 值相同，而 **tem3** 恒等于 2。

一个建议：对于 **set /a** 的计算式，最好用双引号引起来，有些表达式会变得简单。
例如：

```
@echo off
set /p N=请输入一个数字：
set /a num=1^<^<N
echo %num%
pause
```

上面是不用双引号的例子，我们表达 **num=1<<N** 式子时，就需要用转义符号^，否则<管道符号的意思是从某地方读入数据。
如果用双引号，式子就变得简单了：

```
@echo off
set /p N=请输入一个数字：
set /a num="1<<N"
echo %num%
pause
```

所以，为了表达式简便，我们建议将 **set /a** 后面表达式用双引号括起来。

有一种情况，就是 **set /a** 表达式如果含有括号，而 **set /a** 语句外又有括号包含着 **set** 语句时，那么此时的 **set /a** 表达式里的括号就必须用转义字符转义或者整个表达式用双引号。
原因是批处理太死板了，括号一般表达的是分隔符效果，但在表达式里，表达的是计算效果，不转义的话，批处理认为括号早早的就括回去，程序就出错了。例如：

```
@echo off
setlocal enabledelayedexpansion
for /l %%i in (1,1,10) do (
                        set /a var=^(%%i+1^)*3
                        echo var=!var!
                        echo ^(%%i^)
)
pause
```

当然，使用一个转义字符也是可以的：

```
@echo off
setlocal enabledelayedexpansion
for /l %%i in (1,1,10) do (
                        set /a var=(%%i+1^)*3
                        echo var=!var!
                        echo (%%i^)
)
pause
```

改为使用双引号如下：

```
@echo off
setlocal enabledelayedexpansion
for /l %%i in (1,1,10) do (
                        set /a var="(%%i+1)*3"
                        echo var=!var!
                        echo "(%%i)"
)
pause
```

使用双引号后，**echo "(%%i)"**会输出双引号，比较讨厌。

另外，**echo** 后面回显的内容有括号，而还有括号包含着 **echo**，这种情况，**echo** 回显的括号也要转义：

```
@echo off
(
echo (123^)
)
pause
```

程序例子：

```
@echo off
set /p var=请输入表达式：
set /a result=%var%
echo 计算结果是%result%
pause
```

运行过程如下：

请输入表达式：2"<<"1
计算结果是 4
请按任意键继续. . .
程序例子：
@echo off
set /a result=2"<<"1
echo 计算结果是**%result%**
pause
运行结果：
计算结果是 4
请按任意键继续. . .
注意，上面两个例子，无论是用户输入的模式还是批处理文件表达式的形式，<<符号都必须添加双引号，否则计算无法进行。上面所有需要添加双引号的运算符，与此一样。

下面是一些计算例子：
程序例子：
@echo off
set /p input=请输入计算表达式：
::等号左右两边都可以有空格或没有空格。也就是 1、**set /a var=%input%**　2、 **set /a var = %input%**　3、 **set /a var =%input%**　4、**set /a var= %input%**这四种写法都是可以的。
set /a var=%input%
echo 计算结果：**%input%=%var%**
pause
请输入计算表达式：**(**25+75**)***2**/(**15+5**)**
计算结果：**(**25+75**)***2**/(**15+5**)**=10

请输入计算表达式：10/3
计算结果：10/3=3
请输入表达式：8/9
计算结果：8/9=0
（**DOS** 计算精确到整数，小数直接舍弃，不会四舍五入）

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=-2147483648（这个结果当然是错的）
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，也就是 DOS 的计算范围是 -2147483647 到 2147483647。

逗号计算过程：
所有逗号隔开的式子都会被分别进行计算。
例如：**set /a a=1+1,b=2+1,c=3+1** 后，**echo %a%**会显示 2。我们用 **echo %a% %b% %c%,**可以查看三个变量计算结果分别为 2，3，4。

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。这个常用于计数器。

另外我们在 **CMD** 里输入 **set /a var=1 & 1 "**与运算**",**他并不会显示为 1,而是报错。我们需要把它用双引号引起来,也可以用转义字符^，看例子 **set /a var= 1 "&" 1** ，也可以 **set /a var=1 ^& 1**，其他操作符用法
set /a var= 1 "^" 1 异运算
set /a var= 1 %% 1 取模（余数）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
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

这里说一下，对于等号右边的表达式，如果是变量时，可以用两个**%**包围，也可以不包围。
set /a var=num1+num2　　等价于 **set /a var=%num1%+%num2%**
set /a var=num1/num2　　等价于 **set /a var=%num1%/%num2%**
set /a var=num1%%num2　等价于 **set /a var=%num1%%%%num2%**

阶乘程序：

```
@echo off
set a=1
set /p n=请输入一个数字：
if %n%==0 (
                    echo 0!=1 & pause>nul & exit
) else (
```

```
        if %n% lss 0 (
            echo  负数不可以阶乘！ & pause>nul & exit
        )
)
for /l %%i in (1,1,%n%) do set /a a*=%%i
echo %n%!=%a%
pause>nul
```

注：上面阶乘程序没有用高精度计算，限于批处理计算能力有限，不可以计算太大的数字的阶乘，最多只能计算到 12 的阶乘。

思考题：求 2 的 **n** 次方。
利用位运算的<<符号，每左移一次，数字就会增大一倍，这样计算效率最高。参考代码：

```
@echo off
set /p n=请输入 2 的几次方:
set /a num=1^<^<N
echo %num%
pause
```

用 **set** 命令显示诗句前面的多个空格。
我们在显示一首诗的时候，在每一句前面需要打出很多空格，使得整首诗占据窗口的中间。但是，一开始对每行空格输入的数量，不一定复合要求，需要进行增删修改。如果每一行都需要增删空格，操作就非常繁琐。用 **set** 就可以简单处理，而且删减空格也很方便。
程序例子：

```
@echo off
::下面 k 值是 16 个空格。
set k=
echo %k%水光潋滟晴方好，
echo %k%山色空蒙雨亦奇。
echo %k%欲把西湖比西子，
echo %k%淡妆浓抹总相宜。
pause
```

用 set 命令进行字符串处理（这个不应只属于 **set** 的内容，应该归入格式内容，在没有 **set** 的情况，格式仍旧适用）

1、字符串替换
%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，由于 **str1** 可以是整个 **PATH** 的值，也可以是部分 **PATH** 的值，所以用这个语法更多的是部分替换掉 **PATH** 里的内容。
要替换掉全部 **PATH** 的内容，我们直接用 **set PATH=string** 就行了。
需要说明的是，**%PATH:str1=str2%**不依赖 **set** 而存在，和 **set** 一起使用，只不过可以将替换后的内容赋值到某个变量上而已。

程序例子：
```
@echo off
set a= bbs. verybat. cn
::一开始的 a 变量里包含有四个空格符号。
echo  替换前的值: "%a%"
set var=%a: =%
::对 a 变量里的所有内容，空格都替换掉（在这里没有替换什么内容，表示删除）
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.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

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**% ”，a** 是要进行字符替换的变量,"."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"。

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** 和 **n** 可以取正数，也可以取负数，有不同的含义。

需要说明的是，**%a:~[m[,n]]%** 不依赖 **set** 可以单独使用，例如我们 **echo %time:~0,-3%**，表示显示时间的时、分、秒，后面的毫秒就不显示出来了。**set** 说到底，还是赋值功能。

上面的**%a:~[m[,n]]%**，**m** 和 **n** 可以取正数也可以取负数。分别代表的意义如下：

m 是正数或 0，**n** 是正数	舍弃变量 **a** 的前 **m** 位，后再取其 **n** 位
m 是正数或 0，**n** 是负数	舍弃变量 **a** 的前 **m** 位和舍弃末尾的 **n** 位后剩下的值
m 是正数或 0，**n** 缺省	舍弃变量前 **m** 位后，取剩下的所有值
m 是负数，**n** 是正数	取变量 **a** 的末尾 **m** 位，后再取前 **n** 位
m 是负数，**n** 是负数	取变量 **a** 的末尾 **m** 位，后舍弃末尾的 **n** 位后剩下的值
m 是负数，**n** 缺省	取变量末尾 **m** 位

注意：**n** 值可以缺省，但不可以为 0。而 **m** 值可以是任意值。

假如，变量 **a** 代表字母表的一串 26 个字母，

%a:~7,3%	的值为 **HIJ**
%a:~0,1%	的值为 **A**
%a:~7,-3%	的值为 **HIJKLMNOPQRSTUVW**
%a:~-7,3%	的值为 **TUV**
%a:~-7,-3%	的值为 **TUVW**
%a:~3%	的值为 **DEFGHIJKLMNOPQRSTUVWXYZ**
%a:~-3%	的值为 **XYZ**
%a:~0,-3%	的值为 **ABCDEFGHIJKLMNOPQRSTUVW**

经典的实时显示时间的例子：

```
@echo off
::time 取出时间值，-3 表示不显示后三位字符。这个格式只显示时间的时、分、秒。
echo %time:~0,-3%
ping -n 2 127.1>nul&cls&%0
```

set 用法格式总结

```
set var=abc                                  直接给变量赋值
set /p var=请输入：                          将用户输入赋值到变量
set /a var=num1+num2                         数值计算 1
set /a var+=1                                数值计算 2
```

一种比较怪异的写法：

```
@echo off
set var=123456
set cmdstr=echo %var%
%cmdstr%
pause
```

程序输出的是 123456

注意：

set 不可以对常量进行赋值。

set 不可以对参数**%0-%9** 进行计算**!**

set 不可以对 **for** 循环变量**%%i** 进行计算**!**

set 对 **for** 的循环上下限变量进行计算不会改变循环次数**!**

```
set /a 9=4+5
set /a %1+=1
set /l %%i in (1,1,10) do set %%i*=3
```

上面三种形式的计算都是不可以的。

下面对 **for** 上下限进行计算，但不会改变循环次数。也就是说，循环次数在循环开始前就固定了，不会在循环中发生任何变化。

```
@echo off
setlocal enabledelayedexpansion
set /a min=1,max=10
for /l %%i in (!min!,1,!max!) do (
                        echo min=!min! max=!max!
                        set /a min+=1
                        set /a max*=2
)
pause
```

set 技巧高级篇：

1、利用 **set /a** 进行赋值。
在开启变量延迟情况下，我们要判断数组 **S!n!**的值的情况，代码不可以这么写：
if !S!n!!==0 **echo zero**
也不可以这么写：
if S!n!==0 **echo zero**
我们可以利用 **set /a** 对右边表达式变量不必添加变量符号的规则来实现赋值，赋值后就可以进行判断啦。
set /a var=S!n!
if !var!==0 **echo zero**
在这里，如果我们不是用 **set /a**，而仅仅 **set var=S!n!**，得到的 **var** 仅仅是数组名而已，同样也是行不通的。

当多个变量需要赋统一个值时，也可以用 **set /a** 来达到简化代码的目的：

```
@echo off
set /a A=B=C=D=1
echo %A% %B% %C% %D%
pause
```

有多个计算式，同样也可以用 **set /a** 来简化代码：

```
@echo off
set /a A=1,B=2,C=3,D=4
set /a A+=1,B+=2,C+=3,D+=4
echo %A% %B% %C% %D%
pause
```

2、利用 **set /p var=%num%<nul** 输出数据
如果我们输出大量的数据，用 **echo** 命令，每行只输出一个数据，这样会浪费大量的显示界面，不符合我们的习惯。如何将大量数据在同一行输出呢？这个就不能用 **echo** 命令了，必须用 **set** 命令。

set /p var=%num%<nul，可以实现对数字连续输出的效果。
例如：

```
@echo off
```

```
for /l %%i in (1,1,10) do set /p var=%%i<nul
pause
```

上面输出为：12345678910，全部数据都揉成一团，不符合我们的写字习惯。我们可以增加空格隔开数据：

```
@echo off
for /l %%i in (1,1,10) do set /p var=%%i    <nul
pause
```

输出为：1　2　3　4　5　6　7　8　9　10

数字与数字之间是隔开了，但是，如果我们输出的数字位数不一样，而间隔都相同，就会造成下面一种可能的结果：

1 2 3 4 5 6 7 8 9

10 11 12 13 14 15 16 17 18 19

也就是说，上下没有对齐，依然不符合我们的书写和阅读习惯。

如何实现上下对齐呢？这就要借助输出格式：**%var:~m,n%**

具体实现：我们先对每个数字前面增加若干个空格，然后用**%var:~m,n%**取末尾 **k** 位输出就可以了。另外，外加一个计数器，每输出 **i** 位数后换行。

格式：

```
set var=            %str%
set /p    print=%var:~-k%<nul
```

程序例子：

```
@echo off
setlocal enabledelayedexpansion
for /l %%i in (1,1,10000) do (
                        set var=                    %%i
                        set /p print=!var:~-7!<nul
                        set /a num=%%i%%10
                        if !num!==0 echo.
)
pause
```

这个程序不仅会每输出 10 个数后换行，而且上下数字会靠右对齐。另外，**set /p print=%num%<nul** 中的表达式左边变量 **print** 并不是必须的，可以省略为 **set /p =%num%<nul**，不过这样写不太规范，不建议这么写。

关于“**set e=   %b%**”，这个是为了后面的 “**%e:~-7%**”这种输出格式作准备，将数字“**b**”前面添加一些空格。“**%e:~-7%**”意

思是输出 **e** 的后面 7 个字符。
有关输出的格式：
%a:~[m[,n]]%，**m** 和 **n** 可以取正数也可以取负数。分别代表的意义如下：

m 是正数或 0，**n** 是正数	舍弃变量 **a** 的前 **m** 位，后再取其 **n** 位
m 是正数或 0，**n** 是负数	舍弃变量 **a** 的前 **m** 位和舍弃末尾的 **n** 位后剩下的值
m 是正数或 0，**n** 缺省	舍弃变量前 **m** 位后，取剩下的所有值
m 是负数，**n** 是正数	取变量 **a** 的末尾 **m** 位，后再取前 **n** 位
m 是负数，**n** 是负数	取变量 **a** 的末尾 **m** 位，后舍弃末尾的 **n** 位后剩下的值
m 是负数，**n** 缺省	取变量末尾 **m** 位

例如：变量 **a** 代表字母表的一串 26 个字母，

%a:~7,3%	的值为 **HIJ**
%a:~0,1%	的值为 **A**
%a:~7,-3%	的值为 **HIJKLMNOPQRSTUVW**
%a:~-7,3%	的值为 **TUV**
%a:~-7,-3%	的值为 **TUVW**
%a:~3%	的值为 **DEFGHIJKLMNOPQRSTUVWXYZ**
%a:~-3%	的值为 **XYZ**
%a:~0,-3%	的值为 **ABCDEFGHIJKLMNOPQRSTUVW**

程序例子：
读入两个数据，输入两个数据之间的所有数。要求：每输出 10 个数据后换行，并且上下数字要靠右对齐。

```
@echo off
setlocal enabledelayedexpansion
set /p a=请输入起始数字：
set /p b=请输入最终数字：
set    num=1
:begin
if !a! leq !b! (
                    set var=              !a!
                    set /p v=!var:~-5!<nul
                    set /a a+=1
                    set /a num+=1
                    if   !num!   ==   11   (
                        echo.
```

```
                    set num=1
                )
goto begin
)
echo.&pause
```

上面是用 **goto** 实现的，用 **for** 实现如下：

```
@echo off
setlocal enabledelayedexpansion
set /p a=请输入起始数字：
set /p b=请输入最终数字：
set    num=0
for /l %%i in (%a%,1,%b%) do (
                    set var=              !a!
                    set /p v=!var:~-5!<nul
                    set /a a+=1
                    set /a num+=1
                    set /a tem=!num!%%10
                    if !tem!==0 echo.
)
echo.& pause
```

小结：

<nul 是 **0<nul** 的简写，0 句柄是默认的输入句柄，**nul** 是空设备，所以 **<nul** 表示从空设备获取输入。

set /p 执行时会等待用户的输入，**<nul** 使得 **set /p** 输入设备为空（默认情况下为 **0<con**，也就是键盘输入），此时 **set /p** 就会自动终止输入，此处的 **<nul** 和手动按回车是一样的效果，只是它不需要人工操作。也就是：

set /p=提示语句**<nul**

等同于：

set /p=提示语句【按回车】

利用这个特性，再加上 **set /p** 内置的不换行显示提示语句的功能，就能够实现不换行显示，不过它有一个缺陷，就是无法显示以等号开头的行（而且 **win7** 下会把开头连续的空格与制表符忽略）。

批处理简易计算器：

```
@echo off
title DOS 简易计算器
::设置窗口标题
mode con cols=118 lines=30
::设置窗口大小
color 9f
::设置窗口颜色
set xianshi=N
::设置 xianshi 的值为 N，用于判断是否显示运算优先顺序表
echo. & echo.
set /p xianshi=是否显示 DOS 运算符的运算优先顺序表？(Y/N):
:begin
cls
::清屏
if /i "%xianshi%" == "Y" (
                        echo. & echo.
                        echo
DOS 的各种运算符运算优先顺序表
                        echo. & echo.
                        echo   "   1       ( )                                  分
组                                                                          "
                        echo   "   2       !   ~   -
一元运算符                                                                    "
                        echo   "   3       *   /   %%
算数运算符                                                                    "
                        echo   "   4       +   -
算数运算符                                                                    "
                        echo   "   5       <<   >>
二进制逻辑移位          输入表达式时，该符号需添加双引号或转义字符    "
                        echo   "   6       &
二进制按位“与”          输入表达式时，该符号需添加双引号或转义字符    "
                        echo   "   7       ^
二进制按位“异”          输入表达式时，该符号需添加双引号或转义字符    "
                        echo   "   8       |
二进制按位“或”          输入表达式时，该符号需添加双引号或转义字符    "
                        echo   "   9       =    *=    /=    %%=    +=    -=
算数赋值                输入表达式时，该符号需添加双引号或转义字符    "
                        echo   " 10       &=    ^=    |=    <<=    >>=
二进制运算赋值          输入表达式时，该符号需添加双引号或转义字符    "
                        echo   " 11       ,                                    逗
号运算符                                                                    "
::上面的各个回显，回显的内容都添加了双引号，假如不添加双引号，程序就无法正常显示，这个课题有待进一步研究。另外，第三行的百分号最难处理，它即使在双引号内还可以作怪，即使添加了转义字符^也不管用，非得用两个百分号
```

才能将其显示出来。第九行也是如此，%%=在屏幕显示为%=

```
                    echo.
)
echo.
echo ============================================================
==================
echo                    DOS 计算器只能计算整数(小数四舍五入)，并且计
算结果范围为-2147483647 到 2147483647
echo ============================================================
==================
echo. & echo.
set /p input=请输入计算表达式:
set /a var=%input%
::这里进行运算，结果为 var。
echo.
echo  计算结果：        %input%=%var%
echo.
set /p qingqiu=要不要继续计算？(按回车继续，按任意键退出):
if /i "%qingqiu%" == "" (goto begin) else (exit)
::判断，如果输入的是回车，将进行下一次运算，否则退出程序。
```

一个困惑的地方：

```
@echo off
echo  请分别输入三个数字
set /p a=
set /p b=
set /p c=
echo  合并结果为
set e=          %a%
set /p  var=%e:~-7%<nul
set e=          %b%
set /p  var=%e:~-7%<nul
set e=          %c%
set /p  var=%e:~-7%<nul
echo.
pause
```

这个程序的执行和显示结果如下：

```
请分别输入三个数字
12
34
```

56
合并结果为
　　12　　34　　56
请按任意键继续. . .
多测试几遍，就会发现这个程序存在 **bug**：我们分别输入“**a**、**b**、**c**”，或者“**,**、**+**、**=**”，或者其它任何字符，结果都没什么问题，但分别输入个位数字就不行了，例如“1、2、3”，程序无法显示！
这是为什么呢？不明白！对程序修改一下：

```
@echo off
echo  请分别输入三个数字
set /p a=
set /p b=
set /p c=
echo  合并结果为
set e=          %a%
set /p   var=%e:~-7% <nul
set e=          %b%
set /p   var=%e:~-7% <nul
set e=          %c%
set /p   var=%e:~-7% <nul
echo.
pause
```

也就是说，在每个**<nul** 前面增加一个空格，则程序又没问题了，非常奇怪！这个问题暂时还想不明白。

---

for

for 语句很强大，配合 **if**、**call**、**goto** 等流程控制语句，更是可以实现脚本复杂的自动化、智能化操作，离开了 **for** 语句，很多事情都办不了。但是 **for** 命令是如此之复杂，学习 **for** 语句很艰难，假如想通过帮助命令里的信息来学习 **for** 语句，会让人感到绝望！能否熟练充分使用 **for** 命令，是高手与低手的分水岭，下面将 **for** 语句彻底拆解！

FOR
对一组文件中的每一个文件执行某个特定命令。
备注：晕菜，帮助信息的这个解释等于没解释。

FOR 语句
标准格式：　　**FOR %%variable IN (set) DO command [command-parameters]**
这是帮助信息里的语法，我认为这样写不详细，应该这么写：

语法：
FOR [-d | -l | -r | -f] %%variable IN (set) DO command [command-parameters]
中文翻译：
FOR [-d | -l | -r | -f] %%变量名　　**IN (**相关文件或命令**) DO [**执行命令**]**
这里用**%%**，两个百分号是批处理中书写的格式，如果是用户在命令行输入的形式，输入只需一个**%**号。

注解：
从上面可以看到，**for** 有四个参数：**/d　/l　/r　/f**
%%变量名：　　这个变量名可以是单个的大小写字母，它们区分大小写！**for** 会把从**(set)**按顺序读取到的每个值赋值给它。
in　　命令格式，照写就是。
(相关文件或命令**)**：　　**for** 把这里的值读取后赋值给变量 **%%variable**。
do　　命令格式，照写就是。
执行命令：　　对每个变量的值要执行的操作，当然也可以没有任何操作，这个看实际需要。例如利用 **for** 循环来延迟程序，就不需要 **for** 执行任何操作，只需要不断循环即可。

参数详解：

for 没有任何开关的情况：
for %%i in (*) do echo %%i
这条命令的变量**%%i** 取值*，而不是取具体值，那么这里就有特殊的含义：表示显示当前目录下，所有非文件夹的文件名字。（包括**.rar　.jpg　.exe　.bat　.sys** 等等所有非文件夹）

若**%%i** 取具体的值，那么就表示分别取这列出来的具体的值：
for %%i in (a, b c,d;e) do echo %%i
那么执行结果显示：
a
b
c
d
e
这说明空格、逗号和分号都可以作为分隔符。一般情况下，我们都用逗号作为分隔符，这个在数据是数字的情况下使人们更加容易阅读。

利用这个特点，我们可以将变量**%%i** 取一些不连续的变量。
例如我们要将变量分别取值为 1、1945、1949、2008、2012 五个值，由于变量之间不是等差数列关系，所以用 **for /l %%i in () do ()**循环的形式无法达到目的。代码应该这么写：
for %%i in (1,1945,1949,2008,2012) do echo %%i
变量**%%i** 就可以分别取这五个值了。

输出字母表，因为批处理不可以直接转换为 **ASCII** 值，所以比 **C** 语言麻烦多了。在批处理里，用 **for** 输出字母表应该是最简洁的：
@echo off
for %%i in (a b c d e f g h i j k l m n o p q r s t u v w x y z) do echo %%i
pause

或者
@echo off
set var=a b c d e f g h i j k l m n o p q r s t u v w x y z
for %%i in (%var%) do echo %%i
pause

/d
只搜索当前目录下的目录（也就是文件夹，不包括子目录）。注意：**/d** 无法搜索到隐藏的文件夹。

语法：
for /d %%variable in (set) do command [command-parameters]
搜索 **set**（可以包含通配符*和?）里的文件夹后执行指定的 **Command**。用于目录搜索，不会搜索文件。
通配符*和？的区别：
*表示任意 **n** 个字符，用一个*符号就够了。而？表示任意一个字符，需要表达“不多于 **k** 个字符的文件夹”，就写上 **k** 个？，任意多个中文都算 0 个字符。
例如：**???**三个问号，不仅仅代表有三个字母的文件夹，还包括一个字母的文件夹和两个字母的文件夹，所有这些都会被搜索出来。

程序例子：
@echo off
for /d %%i in (C:*) do echo %%i
pause
运行结果显示（本机 **C** 盘非常简洁，只有 3 个文件夹）：
C:\Documents and Settings
C:\Program Files
C:\WINDOWS
请按任意键继续...

/d 不能显示文件名字，只能显示目录。说白了就是显示该目录下的所有文件夹名字。

再看一个例子：
@echo off
for /d %%i in (C:\"program files"*) do echo %%i
pause
运行结果将 **C:\program files** 里面的所有文件夹都显示出来了。注意，因为“**program files**”含有空格，所以必须用双引号，否则出错。

程序例子：
@echo off
for /d %%i in (???) do echo %%i
pause
把这个批处理程序放在哪里运行，程序会把当前路径下文件夹的名字

只有 1~3 个字母显示出来，没有就不显示了。
也可以指定路径搜索：
@echo off
for /d %%i in (C:\windows\????) do echo %%i
pause
表示把 **C:\windows** 里面所有文件夹名字不多于 4 个字符的所有文件都搜索出来。

程序例子：
@echo off
for /d %%i in (win??????????) do @echo %%i
pause
这个如果保存到 **C** 盘的 **Program Files** 下，有可能会显示：
Windows NT
Windows Media Player
WinRAR
也就是说，显示以 **win** 开头的文件。**echo** 前添加**@**，是为了运行时不要每次显示 **echo** 命令行本身。这个如果批处理头有**@echo off**，那么这里添不添加**@**就没什么所谓，但如果我们是在 **CMD** 窗口输入这条语句，没有**@**的执行结果如下：
C:\Program Files>for /d %i in (win??????????) do echo %i
C:\Program Files>echo Windows NT
Windows NT
C:\Program Files>echo Windows Media Player
Windows Media Player
C:\Program Files>echo WinRAR
WinRAR
C:\Program Files>
回显太多，显得很杂乱。

for /d 不能匹配带隐藏属性的文件夹，在灵活性上不及 **for /f** 和 **dir** 的组合；当“元素集合”中不包含任何统配符的时候，它完全是 **"for %%i in (**元素集合**) do ……"** 语句的翻版，但是又稍显复杂。感觉 **for /d** 就是鸡肋一块。

/r
只搜索目录中（包括子目录和子目录的子目录，也就是目录树）的所有文件。这个和**/d** 作用互补。

语法格式：
FOR /R [[drive:]path] %%variable IN (set) DO command [command-parameters]
如果在 **/r** 后没有指定目录，则使用当前目录。如果 **set** 仅为一个单点（.）字符，则枚举该目录树。例如：**for /r C:\windows %%i in (.) do echo %%i**　　枚举 **C:\windows** 目录树的所有文件。

/r 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。
注意两点：
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的与 **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**(.)**枚举目录是一个道理，单点代表当前目录，也可视为一个文件。

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

注意上面的书写格式，**%%i,**两个百分号必不可少。另外，**c:**和**%%i** 之间有空格。
这个程序随便保存到哪里都可以，它把 **C** 盘根目录和每个目录的子目录下面的所有 **exe** 文件都列出来了。
不过这个程序一闪而过，我们来不及一条条看它列举了什么内容，可以修改成这样
```
@echo off
for /r c:\ %%i in (*.exe) do (echo %%i & pause > nul)
pause
```
这个程序每列举一条就会等待按任意键继续。

```
@echo off
for /r D:\TTPlayer %%i in (*.exe) do echo %%i
pause
```
这个程序表示搜索 **D:\TTPlayer** 目录下的所有 **exe** 文件。（本机千千静听装在 **D** 盘根目录）运行结果如下：

D:\TTPlayer\bdupdate.exe
D:\TTPlayer\TTPlayer.exe
D:\TTPlayer\ttpsvr.exe
D:\TTPlayer\uninst.exe
D:\TTPlayer\APKMgr\aapt.exe
D:\TTPlayer\APKMgr\adb.exe
D:\TTPlayer\APKMgr\ttpdev32**.exe**
D:\TTPlayer\APKMgr\ttpdev64**.exe**
请按任意键继续. . .

程序例子：

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

这个程序命令前没有加 **C:**，也就是搜索路径，这样它就会以当前目录为搜索路径，比如这个 **bat** 放在 **D:\test** 目录下执行，那么它就会把 **D:\test** 目录和它下面的子目录的全部 **exe** 文件全部都列出来。

程序例子：

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

程序列举了 **C** 盘所有目录，虽然每个目录最后都显示 **boot.ini**，其实那不是真的存在 **boot.ini**。之所以会这样，是因为计算机在搜索所有的目录，搜索过程伴随回显。为了只列举 **boot.ini** 存在的目录，程序应该这样写：

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

于是，计算机就只显示 **boot.ini** 存在的目录了。

若要对获取到的路径进行进一步处理，则需要把 **dir** 语句放入 **for /f** 语句中进行分析，写成 **for /f %%i in ('dir /ad /b /s') do ……** 的形式；由于 **for /r** 语句是边列举路径边进行处理，所以，在处理大量路径的时候，前期不会感到有停顿，而 **for /f** 语句则需要等到 **dir /ad /b /s** 语句把所有路径都列举完之后，再读入内存进行处理，所以，在处理大量路径的时候，前期会感到有明显的停顿。

当列举目录时，**for /r** 和 **dir /ad /b /s** 的效果是非常类似的，那么当我们要获取目录路径并进行进一步处理的时候，两者之间，我们该如何选择？

1、**for /r**：
优点：
① 只通过 1 条语句就可以同时实现获取目录路径和处理目录路径的操作；
② 遍历文件夹的时候，是边列举边处理的，获取到一条路径就处理一条路径，内存占用小，处理大量路径的时候不会产生停顿感；
缺点：
① 不能获取到带隐藏属性的目录，会产生遗漏；
② 不能获取带指定属性的目录

2、**dir /ad /s**：
优点：
① 能一次性获取带任意属性的目录，不会产生遗漏；
② 能通过指定不同的参数获取带任意属性的目录，更具灵活性。
缺点：
① **dir /ad /s** 语句仅能获取到目录路径，若要实现进一步的处理，还需要嵌入 **for /f** 语句中才能实现，写法不够简洁；
② 嵌入 **for /f** 语句之后，需要写成 **for /f "delims=" %%i in ('dir /ad /b /s') do ……** 的格式，受 **for /f** 语句运行机制的制约，需要先列举完所有的路径放入内存之后，才能对每一条路径进行进一步的处理，处理大量路径时，内存占用量偏大，并且在前期会产生明显的停顿感，用户体验度不够好；

综合上述分析，可以做出如下选择：
1、若仅仅是为了获取某文件夹及其所有子文件夹的路径的话，请选择 **dir /ad /b /s** 语句；
2、若需要过滤带隐藏属性的文件夹的话，**for /r** 和 **dir** 语句都可以实现，但 **for /r** 内存占用小，处理速度快，是上上之选；
3、若需要获取所有文件夹，则除了 **dir /ad /b /s** 外，别无选择，因为 **for /r** 语句会遗漏带隐藏属性的文件夹；

/l

语法格式：
FOR /L %%variable IN (start,step,end) DO command [command-parameters]
从 **start** 开始，以 **step** 为步长，直至最接近 **end** 那个整数为止，这中间有多少个整数，**do** 后面就执行多少次。
(1,1,5)将产生序列 12345，5 次重复，并依次将序列赋值到变量**%%i**。**(5,-1,1)**将产生 54321 序列，也是 5 次重复，并依次将序列赋值到变量**%%i**。这个常用于循环结构。对于 **in** 后面括号里的三个变量，逗号隔开改为空格隔开也是可以的，不过不建议用空格隔开。

```
@echo off
for /l %%i in (1,1,5) do echo %%i
pause
```
运行显示：
1
2
3
4
5

程序例子：
```
@echo off
for /l %%i in (1,1,5) do start cmd
pause
```
执行后计算机打开了 5 个 **cmd** 窗口。假如将**(1,1,5)**改成**(1,1,65535)**，不死机算你强！
我们将 **start cmd** 改成 **md %%i** ，这样就会建立指定个数目录了！

```
for /l %%i in (1,1,100) do echo %%i
for /l %%i in (1 1 100) do echo %%i
```
上面这两种写法都是可以的，但建议用上面的写法好，数字有逗号隔开，我们更加容易阅读。

注意：在循环里，我们不可以改变循环变量的值！对循环上下限变量允许计算，但不会影响循环的次数！
例如：
```
@echo off
setlocal enabledelayedexpansion
for /l %%i in (1,1,10) do (
                        set %%i+=10
```

```
                echo %%i
)
pause
```

上面程序执行出错。

```
@echo off
setlocal enabledelayedexpansion
set /a min=1,max=10
for /l %%i in (!min!,1,!max!) do (
                echo min=!min! max=!max!
                set /a min+=1
                set /a max*=2
)
pause
```

上面虽然上下限 **min** 和 **max** 的值改变了，但循环次数不变。

for 里的循环变量**%%i** 和参数**%0-%9**，就代表它是变量了，在引用它时，不可以再对齐增加变量符号，即**%%%i%** 、**!%%i!**、**%%1%**、**!%1!**等形式都是不可以的。

循环结构的选择 **for** 还是 **goto**？

当循环次数确定的时候，首选 **for /l** 语句，代码会比较简单明了，也可使用 **goto** 语句但不推荐；当循环次数不确定的时候，用 **goto** 语句将是唯一的选择，因为，这个时候需要用 **if** 之类的条件语句来判断何时结束 **goto** 跳转。

高级篇：----无限循环

```
for /l %%i in () do echo Hello
```

/f 这是四个参数中最复杂的一个，灰常强大，不过其复杂性令人望而生畏。

/f 用途：

它能够对字符串进行操作，也能够对命令的返回值进行操作，还可以访问硬盘上的 **ASCII** 码文件，比如 **txt** 文档等。常用于文本信息编辑，如查询、提取、替换等等。

格式：
FOR /F ["options"] %%variable IN (file-set) DO command [command-parameters]　　文件名
file-set 不用引号（如果文件名含有空格时，就需要借助 **usebackq** 了）
FOR /F ["options"] %%variable IN ("string") DO command [command-parameters]　　字符串
string 要用双引号
FOR /F ["options"] %%variable IN ('command') DO command [command-parameters]　　命令
command 要用单引号
注意：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：**eol=c**、**skip=n**、**delims=xxx**、**tokens=x,y,m-n**、**usebackq**。

当前目录下有一个《**test.txt**》文件
@echo off
for /f %%i in (test.txt) do echo %%i
pause
会一下子显示出文件的所有内容。
@echo off
for /f %%i in (test.txt) do echo %%i & pause>nul
pause
会逐行显示文件内容。

下面对每个详细讲解：
（一） **eol=c**　　忽略（隐藏、屏蔽）所有以字符是 **C** 开头的行。
格式：
FOR /F "eol=c" %%I　IN (Command1)　DO Command2
“指定行尾字符（仅一个字符）。”这是帮助文件里的话，目前不理解它的用法。
eol=
默认情况下屏蔽冒号(无论是中文冒号还是英文冒号，都会被屏蔽掉)开头的行，但冒号本身不会被屏蔽，还是会被显示出来。
注意：这里被处理的文本里的冒号和后面的文字字符必须有空格隔开，否则无法对文本进行屏蔽！

“**eol=c**”程序举例
例如：文本《静夜思**.txt**》的内容是：

床前明月光，
疑是地上霜；
举头望明月，
低头思故乡。

要略掉文中的第三行：“举头望明月，”程序如下：
@echo off
for /f "eol=举" %%i in (静夜思.txt) do echo %%i
pause>nul
运行结果将隐藏第三行。运行结果显示：
床前明月光，
疑是地上霜；
低头思故乡。
上面代码没有指出《静夜思**.txt**》路径，那么脚本就只搜索当前路径有没有《静夜思**.txt**》文本，有就执行程序，没有则显示“找不到文件”。假如《静夜思**.txt**》和脚本不在同一目录，《静夜思**.txt**》是保存在 **D** 盘根目录，

那么脚本就需要指明路径了。指明路径的程序为：
@echo off
for /f "eol=举" %%i in (D:\静夜思.txt) do echo %%i
pause >nul
运行结果与上面一样。

问：**eol=c** 格式，能否屏蔽以空格开头的行呢？答案是：不能！
例子：文本《春夜喜雨**.txt**》内容如下：
　好雨知时节，
当春乃发生。
　随风潜入夜，
润物细无声。
　野径云俱黑，
江船火独明。
　晓看红湿处，
花重锦官城。
批处理代码如下：

@echo off
for /f　"eol= "　%%i in (春夜喜雨.txt) do echo %%i
pause>nul

上面代码“=”号后面有一个空格。程序执行结果如下：
好雨知时节，
当春乃发生。
随风潜入夜，
润物细无声。
野径云俱黑，
江船火独明。
晓看红湿处，
花重锦官城。
结果将原来文本里的前导空格都屏蔽掉了，但所有文本原封不动输出。

问：如果空格不在行首，**eol=c** 格式，又能否屏蔽掉呢？答案是：能，会屏蔽掉该行空格以后的内容。
例子：文本《念奴娇**.txt**》内容如下：
大江东去　浪淘尽　千古风流人物
故垒西边　人道是　三国周郎赤壁
乱石穿空　惊涛拍岸　卷起千堆雪　江山如画　一时多少豪杰
遥想公瑾当年　小乔初嫁了　雄姿英发 羽扇纶巾
谈笑间　樯橹灰飞烟灭
故国神游　多情应笑我　早生华发
人生如梦　一樽还酹江月
这里，没有用现代的标点符号断句，而是用空格。执行代码如下：

```
@echo off
for /f "eol= " %%i in (念奴娇.txt) do echo %%i
pause>nul
```

执行结果如下：
大江东去
故垒西边
乱石穿空
遥想公瑾当年
谈笑间
故国神游
人生如梦
所有空格以后的内容都被屏蔽掉了。但空格前面的内容还是会显示出来。

问：**eol=c** 格式，如果 **c** 字符不是某一行的开头字符，那么能否屏蔽掉该行呢？答案是：不能！
例如：文本还是用上面的带前导空格的《春夜喜雨**.txt**》文本，批处理代码如下：

```
@echo off
for /f   "eol=风"   %%i in (春夜喜雨.txt) do echo %%i
pause>nul
```

运行结果和上面结果一样，也将前导空格屏蔽掉了，其余原封不动。

“eol=”程序举例
文本《春夜喜雨**.txt**》内容如下：

```
：  好雨知时节，
当春乃发生。
：  随风潜入夜，
润物细无声。
：  野径云俱黑，
江船火独明。
：  晓看红湿处，
花重锦官城。
```

文中的冒号采用中文或英文冒号的效果都是一样的，本例采用中文冒号。文本里，冒号与后面的文字有空格隔开，否则文中的所有内容都还会被显示出来，达不到屏蔽的效果。
批处理代码如下：

```
@echo off
for /f   "eol="   %%i in (春夜喜雨.txt) do echo %%i
pause>nul
```

执行结果显示：

```
：
当春乃发生。
：
润物细无声。
：
江船火独明。
：
花重锦官城。
```

屏蔽并没有屏蔽冒号本身。假如我们的批处理代码修改为：

```
@echo off
for /f　"eol=："　%%i in (D:\春夜喜雨.txt) do echo %%i
pause>nul
```

注意这里的"eol=："中的冒号，必须和上面文本采用的格式一样，都采用中文，否则屏蔽无效。
程序执行结果如下：
当春乃发生。
润物细无声。
江船火独明。
花重锦官城。
所有以冒号开头的行都被屏蔽掉了，连冒号本身都被屏蔽掉了。

（二）**skip=n**　忽略（屏蔽、隐藏）文本前 **N** 行的内容。（**N** 不能等于 0，**N** 必须大于 0）
格式：

```
FOR /F "skip=n" %%I　IN (Command1)　DO Command2
```

例：忽略文本的前四行。
看例子：文本《古诗十九首.txt》内容如下：

```
迢迢牵牛星，
　皎皎河汉女。
纤纤擢素手，
　札札弄机杼。
终日不成章，
　泣涕零如雨。
河汉清且浅，
　相去复几许。
盈盈一水间，
　脉脉不得语。
```

上面有些行前多添加了空格，是为了下面输出时对比。从下面输出可以看出，无论有没有前导空格，计算机都会忽略掉前导空格，输出效果与没有前导空格是一样的。
批处理代码如下：

```
@echo off
for /f "skip=4" %%i in (古诗十九首.txt) do echo %%i
pause>nul
```

运行结果如下：
终日不成章，
泣涕零如雨。
河汉清且浅，
相去复几许。
盈盈一水间，
脉脉不得语。
前四行被屏蔽掉了，并且前导空格也被屏蔽掉了。

如果文本有前导回车，那么结果又如何呢？
程序例子：文本《古诗十九首**.txt**》内容如下：

迢迢牵牛星，
皎皎河汉女。
纤纤擢素手，
札札弄机杼。
终日不成章，
泣涕零如雨。
河汉清且浅，
相去复几许。
盈盈一水间，
脉脉不得语。
上面文本的开头有十个回车，执行代码如下：

```
@echo off
for /f "skip=4" %%i in (古诗十九首.txt) do echo %%i
pause>nul
```

执行结果显示：
迢迢牵牛星，

皎皎河汉女。
纤纤擢素手，
札札弄机杼。
终日不成章，
泣涕零如雨。
河汉清且浅，
相去复几许。
盈盈一水间，
脉脉不得语。

说明：屏蔽时，前导回车会算作一行而屏蔽掉，但输出时，所有的回车都不输出。
要彻底弄清上面情况，再看例子：
文本《古诗十九首**.txt**》内容如下：

迢迢牵牛星，
皎皎河汉女。
纤纤擢素手，

札札弄机杼。
终日不成章，
泣涕零如雨。
河汉清且浅，
相去复几许。
盈盈一水间，
脉脉不得语。

注意到，“迢迢牵牛星”上面有两行回车空行，“札扎弄机杼”这一行前面有一行回车空行。
执行代码如下：

```
@echo off
for /f "skip=4" %%i in (古诗十九首.txt) do echo %%i
pause>nul
```

根据上面的结论可以知道，输出时，屏蔽掉开头两个回车行以及“迢迢牵牛星，皎皎河汉女。”这两行，屏蔽共计四行。输出剩下的诗句，但“札扎弄机杼”前面的行是回车行，所以不会被输出。结果如下：

纤纤擢素手，
札札弄机杼。
终日不成章，
泣涕零如雨。
河汉清且浅，
相去复几许。
盈盈一水间，
脉脉不得语。

（三）**delims=xxx(xxx** 是被定义的符号，该符号在文本中存在，将作为分隔符**)**　　定义分隔符（用于切分文本）。
格式：
FOR /F "Delims=符号**" %%I　IN (Command**1**)　DO Command**2
用法：
1、以文本中的标点符号来定义分隔符。
2、被定义为分隔符的标点符号将被忽略（隐藏）。没有使用 **tokens** 定义显示，默认情况下只显示第一列内容。
3、被定义的分隔符前后内容将变成独立的小节（或列）。
4、**for** 默认以空格作分割符，当我们没有写**"delims="**，就默认以空格分隔。
5、分隔符可以是一个，也可以是多个。
6、用 **delims** 定义多个分隔符时，分隔符之间不能有空格。假如要包含空格，空格不能居于所有符号之前或中间，必须放在最后。
7、　我们写的代码为**"delims="**，也就是有写 **delims** 参数，但没指明什么符号作为分隔符，那么程序将认为文本没有分隔符，所有内容都会被显示。
有人说**"delims="**这个句子，在某些情况下，程序会将等号后面的双引号定义成了分隔符，从而使我们没有达到预期的效果。于是，有些人将**"delims="**代码改写为**"delims=="**，将等号作为分隔符。我测试了那么多程序，还没发现**"delims="**会将双引号定义成分隔符，所以，不要采用**"delims=="**这种格式。假设真的有程序发生错乱，我们要定义的分隔符的最佳选择也不是等号，而是后引号“`”，这个符号在 **Esc** 键下面，这个符号非常罕见，将它定义为分隔符就等于定义没有任何符号为分隔符。

分隔符和标点符号的联系和区别
分隔符是用“**delims=**”定义了的文本中的符号。文本中的符号，只

要没有被定义过，就不是分隔符。分隔符需要定义，用“**delims=**”来定义，定义中的符号都来自文本。

要弄清楚分节、分列。我们把每行两个分隔符之间的文字内容称为小节、列。注意：是分隔符之间，而非文中标点符号之间。

例：《易经·乾》的开头，保存为文本《易经**.txt**》，所有标点符号的格式都采用中文格式，内容如下：

乾：元，亨，利，贞。
初九：潜龙，勿用。
九二：见龙再田，利见大人。
九三：君子终日乾乾，夕惕若，厉无咎。
九四：或跃在渊，无咎。
九五：飞龙在天，利见大人。
上九：亢龙有悔。
用九：见群龙无首，吉。

下面我们做实验。
如果使用 **Delims** 来定义分隔符，你可以使用原文中的冒号（：），写法是“**Delims=：**”（使用英文双引号）。冒号作为分隔符后，冒号前后的内容被分成两个小节**(**下面可以列一个表格**)**

	第一小节	分隔符	第二小节
第一行	乾	：	元，亨，利，贞。
第二行	初九	：	潜龙，勿用。
第三行	九二	：	见龙再田，利见大人。
第四行	九三	：	君子终日乾乾，夕惕若，厉无咎。
第五行	九四	：	或跃在渊，无咎。
第六行	九五	：	飞龙在天，利见大人。
第七行	上九	：	亢龙有悔。
第八行	用九	：	见群龙无首，吉。

备注：默认只显示第一列，默认忽略分隔符，默认忽略第一个分隔符后面的内容。
代码：

```
@echo off
for /f "delims=：" %%a in (易经.txt) do echo %%a
pause>nul
```

这里的**"delims=："** 中的符号（冒号），要与文本中的符号（冒号）格式一致。上面文本的冒号采用中文格式，那么这里也必须采用中文格式。
运行结果显示如下：

乾
初九
九二
九三
九四
九五
上九
用九

运行结果将显示第一小节（列）的内容。原因是没有定义显示的列，默认情况下忽略第一个分隔符（冒号）前面的内容。
冒号改用逗号，效果又如何呢？代码如下：

```
@echo off
for /f "delims=，" %%a in (易经.txt) do echo %%a
pause>nul
```

结果显示：

乾：元
初九：潜龙
九二：见龙再田
九三：君子终日乾乾
九四：或跃在渊
九五：飞龙在天
上九：亢龙有悔。
用九：见群龙无首

可见，第一个逗号和第一个逗号后面的所有内容都没有显示出来。

程序例子：

```
@echo off
for /f "delims=" %%a in ("Hello "AnsiPeter" World") do
echo.%%a
pause>nul
```

这里，**delims** 写出来了，但没指明什么符号是分隔符，于是，计算机认为字符串**“Hello "AnsiPeter" World”**没有分隔符，显示第一列也就是显示所有内容就是：

Hello "AnsiPeter" World

（四）**tokens=x,y,m-n**　　显示指定的列

tokens=x	只显示第 **x** 列
tokens=x,y,z	只显示第 **x**、**y** 和 **z** 列。**x**、**y**、**z** 等字母之间必须用英文逗号。
tokens=m-n	只显示从 **m** 列到 **n** 列（**m<n**）
tokens=*	显示文本中的所有内容（此时，不管有没有用 **delims** 定义分隔符或者定义了多少分隔符，文本的所有内容包括所有符号都会被显示出来）

注意，如：

```
@echo off
for /f "tokens=1,2 delims=：" %%a in (易经.txt) do echo %%a %%b
pause>nul
```

表示的意思是：以冒号为分隔符，显示第一列和第二列内容。

运行结果显示：

乾 元，亨，利，贞。
初九 潜龙，勿用。
九二 见龙再田，利见大人。
九三 君子终日乾乾，夕惕若，厉无咎。
九四 或跃在渊，无咎。
九五 飞龙在天，利见大人。
上九 亢龙有悔。
用九 见群龙无首，吉。

注意，冒号同样没有显示出来。

注意格式细节：

1、“**tokens=1,2**”中 1 和 2 数字之间，用的逗号必须是英文格式逗号。

2、这里需要显示两列的内容，那么，**echo** 后面就需要用多个变量符号**%%a** 和**%%b**。同样，如果需要显示三列内容，就必须这么写 **echo %%a%%b%%c**。

3、**for /f "tokens=1,2 delims=：" %%a in (易经.txt) do echo %%a %%b**，对于 **in** 前面用“**%%a**”，那么 **echo** 后面就必须以“**%%a**”变量开始！而且，后面的变量必须以字母表顺序排列下去，否则无效！

例如：

for /f "tokens=1-4 delims=：" %%i in (易经.txt) do echo %%i %%j%%k%%l

正确，这里 **i**、**j**、**k** 和 **l** 按照字母表顺序。

for /f "tokens=1-4 delims=：" %%b in (易经.txt) do echo %%a %%b%%c%%d

错误，**echo** 后面没有以**%%b** 开始。

for /f "tokens=1-4 delims=：" %%e in (易经.txt) do echo %%e %%g%%h%%i

错误，**echo** 后面的变量 **e**、**g**、**h**、**i** 不是字母表顺序。

如果执行下面代码：

```
@echo off
for /f "tokens=1,3 delims=，" %%a in (易经.txt) do echo %%a %%b
pause>nul
```

表示的意思是：以逗号为分隔符，显示第一列和第三列的内容（其余内容忽略），执行结果如下：

乾：元 利
初九：潜龙
九二：见龙再田
九三：君子终日乾乾 厉无咎。
九四：或跃在渊
九五：飞龙在天
上九：亢龙有悔。
用九：见群龙无首

有些行没有第三列，就没有显示什么东西。“乾：元，亨，利，贞。”，这一行被逗号分隔为“乾：元”、“亨”、“利”、“贞”四列，显示第一列和第三列就是“ 乾：元 利”，作为分隔符的逗号以及后面的所有内容都被忽略。

也可以使用多个不同符号都作为分隔符。例如将上面的代码改为：

```
@echo off
for /f "tokens=1,3 delims=:， " %%a in (易经.txt) do echo %%a
%%b
pause>nul
```

表示的意思是：以冒号和引号作为分隔符，只显示第一列和第三列的内容。执行结果如下：

乾 亨
初九 勿用。
九二 利见大人。
九三 夕惕若
九四 无咎。
九五 利见大人。
上九
用九 吉。

这里只分析第一句：“乾：元，亨，利，贞。”这一行被冒号和逗号分隔为“乾”、“元”、“亨”、“利”、“贞”五列，只显示第一列和第三列就是“乾 亨”，作为分隔符的冒号和逗号以及其它内容都不予显示。

代码：

```
@echo off
for /f "delims=:，" %%a in (易经.txt) do echo %%a
pause>nul
```

这里，虽然用多个符号将文本分成更多的列，但显示的时候没有指出显示哪些列，所以运行结果将显示第一小节（列）的内容。运行结果显示如下：

乾
初九
九二
九三
九四
九五
上九
用九

对于**"delims=:，"**这个句子，这里有冒号和双引号两个符号了，当然我们还可以添加任意多的符号都作为分隔符，例如**"delims=，。？；"**，这里有逗号、句号、问号和分号四个符号都作为分隔符。注意，第一个符号前面以及所有符号之间不能有空格！**"delims=，。？；"**这个句子，四个符号间都不存在空格。像这样写就是错的：**"delims= ，。？；"**，错在第一个符号逗号前面有空格。**"delims=，　。？；"**也是错的，错在逗号和句号之间有空格。如果要空格也作为分隔符，那么空格必须写在所有符号的最后面，例如**"delims=，。？；　"**，这里分号后面有一个空格。当然，也可以单独将空格作为分隔符**"delims= "**。空格作为分隔符后，其余效果与之前一样，举例从略。

例如：文本“静夜思**.txt**”的内容是：

床前明月光，疑是地上霜，举头望明月，低头思故乡。

文中有四句诗，诗句之间全都用逗号分隔。**Delims=**，的意思就是定义逗号为分隔符，以此将诗句分成四个小列（或小节）。

	第一列	分隔符	第二列	分隔符	第三列	分隔符	第四列
第一行	床前明月光	，	疑是地上霜	，	举头望明月	，	低头思故乡

如果我要用 **for** 将四句诗都显示到屏幕上，不需要任何参数即能实现：

```
@echo off
for /f %%i in (静夜思.txt) do echo %%i
pause>nul
```

运行结果显示：
床前明月光，疑是地上霜，举头望明月，低头思故乡。

如果运行的代码改为：
@echo off
for /f %%i in (易经.txt) do echo %%i
pause>nul
那么，显示结果是：
乾：元，亨，利，贞。
初九：潜龙，勿用。
九二：见龙再田，利见大人。
九三：君子终日乾乾，夕惕若，厉无咎。
九四：或跃在渊，无咎。
九五：飞龙在天，利见大人。
上九：亢龙有悔。
用九：见群龙无首，吉。
所有行的前导空格都被忽略掉了。

也就是说：如果没有任何参数的语句：
for /f %%i in (filename) do echo %%i
会忽略掉所有行的前导空格后将全部文本内容都显示出来。

再看一个例子：

乾：元　亨　利　贞。
初九：潜龙　勿用。
九二：见龙再田　利见大人。
九三：君子终日乾乾　夕惕若　厉无咎。
九四：或跃在渊　无咎。
九五：飞龙在天　利见大人。
上九：亢龙有悔。
用九：见群龙无首　吉。

执行代码为：

@echo off
for /f　　%%e　　in (易经.txt) do echo %%e
pause>nul

执行结果为：
乾：元
初九：潜龙
九二：见龙再田
九三：君子终日乾乾
九四：或跃在渊
九五：飞龙在天
上九：亢龙有悔。
用九：见群龙无首

如果执行代码改为：
@echo off
for /f　"tokens=1,3" %%e　in (易经.txt) do echo %%e%%f
pause>nul

执行结果为：
乾：元利
初九：潜龙
九二：见龙再田
九三：君子终日乾乾厉无咎。
九四：或跃在渊
九五：飞龙在天
上九：亢龙有悔。
用九：见群龙无首

这里没有使用 **delim** 进行分隔，而带有空格的行空格后面的内容被忽略了，原因就是默认情况下，也就是没有即使没有用 **delims** 选项，**for** 也会以空格作为分隔符，将空格后的内容屏蔽掉。

使用 **delims** 的目的主要是通过切分字符串获得可编辑的字符串以便于进行下一步编辑。
Delims 好比一把刀，文本信息就好比猪扒，使用 **delims** 这把刀将文本这块猪扒切分成许多小块，这样吃起来就方便了。
如果你想吃其中的某一块，那么直接用 **tokens** 这把刀叉提取就行了。

（五）**usebackq**　　主要用于路径或文件名有空格时的情况（说白了就是对 **in** 后面的括号内集合进行转义）
如果没有任何人指点，单靠看帮助文件就能确切明白此符号的含义与

用法，可以称之为神人！
这个符号很多人一头雾水，包括编写教程的很多批处理高手，对其也是一知半解。于是，很多人写教材是，干脆就不写这个符号的用法。这个符号不是可有可无的，在文件名或路径名存在空格这个情况下，没有它的存在是不行的。

讲解开始：

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]
　　或者，如果有 **usebackq** 选项:
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]

上面六条格式，是直接从帮助文件复制过来的，猛一看，上面三条格式和下面三条格式完全一样，有点云里雾里。再接着看帮助里关于 **usebackq** 参数的说明：
　　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。
看了这句注解，还能保持清醒的都不是正常人。这句话艰深的程度堪比《易经》。
帮助里有举一例：
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。
所举的例子能懂否？还是不懂的话，现在开始学习：

1、当集合中的文件名 **file-set** 含有空格时，使用双引号。
语法：
FOR /F "usebackq" %%I　IN ("FileNameSet")　DO Command2
集合中的文件名或路径含有空格时，此时需要用双引号，保持文件名或路径的完整性，并在参数后打开“**usebackq**”。
我们在文件名没有含空格的情况下，是根本不需要 **usebackq** 的，例如（假如《**note.txt**》位于 **C** 盘根目录）：

```
@echo off
for /f %%a in (C:\note.txt) do echo %%a
pause>nul
```

程序没什么问题，可以正常读取 **note.txt** 文本内容并显示出来。
这里说的正常，不是说《**note.txt**》的所有内容都可以显示出来。由于这里没有使用参数 **delims**，那么默认空格是分隔符，假如《**note.txt**》存在空格，那么存在空格的那一行，空格后面的内容都不予显示。对这个还不是很懂的，重温上面 **delims** 参数命令教程。
假如我们的文件名或路径存在空格，仍旧按照上面习惯写代码：

```
@echo off
for /f   %%a in (C:\Documents and Settings\Administrator\桌面
\note.txt) do echo %%a
pause > nul
```

运行结果显示：
系统找不到文件 **C:\Documents**。

原因是“**Documents and Settings**”存在空格。那么我们对其添加双引号又如何呢？代码如下：

```
@echo off
for /f   %%a in ("C:\Documents and Settings\Administrator\桌面
\note.txt") do echo %%a
pause > nul
```

运行结果显示：

```
C:\Documents
```

这个程序可以运行，但它是认为“**C:\Documents and Settings\Administrator\桌面\note.txt**”这是一条字符串，**for** 对其作为字符串处理。由于这里没有用 **delims** 参数，默认分隔符就是空格，字符串空格后面的字符都不予显示，所以就显示前面的“**C:\Documents**”这一小节。显然，这也不是我们的本意。
正确的做法是使用 **usebackq** 参数。代码如下：

```
@echo off
for /f   "usebackq" %%a in ("C:\Documents and
Settings\Administrator\桌面\note.txt") do echo %%a
pause > nul
```

这时，计算机就会认为“**C:\Documents and Settings\Administrator\桌面\note.txt**”这时一条路径，程序可以正常显示《**note.txt**》的所有内容。

2、当使用 **usebackq** 处理文字字符串时，使用单引号。
格式：**FOR /F "usebackq" %%variable IN ('command') DO command [command-parameters]**
注意：**'command'**使用的是单引号，不是后引号！

```
@echo off
for /f "usebackq delims=" %%a in ('Hello "AnsiPeter" World') do echo.%%a
pause>nul
```

程序运行显示：
Hello "AnsiPeter" World

讲解：
这里只是说明 **usebackq** 也可以显示字符串，此时使用单引号，不同于用 **usebackq** 使用双引号的情况。
其实，要想显示字符串而已，是不需要烦劳 **usebackq** 的：

```
@echo off
for /f   "delims=" %%a in ("Hello "AnsiPeter" World") do echo.%%a
pause>nul
```

运行显示：
Hello "AnsiPeter" World

3、当集合内是可执行的命令式，使用“`”后引号（**Esc** 键下面那个键）。
语法：
FOR /F "usebackq" %%variable IN (`command`) DO command [command-parameters]

例子：

```
@echo off
for /f "usebackq delims==" %%i in (`set`) do echo %%i
pause>nul
```

显示结果：
枚举当前环境中的环境变量名称。
讲解：
`set`由于使用了 **usebackq** 和后引号，因此它表示命令执行，即括号内的**`set`**是 **for** 以外的一个命令，它在 **do** 后面的 **echo** 之前已经先被

执行了。
命令区别于文件名和字符串的地方是：命令式批处理中设定可以作用于其它命令、文件名和字符串执行相应任务的字母和标点符号。而文件名和字符串通常是被动的接受命令的安排的。因此，命令是早先设计好的，我们只是自由的组合它们而已。

再来一个例子：

```
@echo off
for /f "usebackq delims=" %%i in (`dir`) do echo %%i
pause>nul
```

这个搜索并显示当前目录。

未使用 **usebackq** 和使用 **usebackq** 时，集合内的符号比较：

双引号

未使用参数 **usebackq**，双引号表示字符串，即**"string"**。
使用参数 **usebackq**，当文件路径或名称中有空格时，用双引号括起来。

单引号

未使用参数 **usebackq**，表示执行命令，即**'command'**。
使用参数 **usebackq**，表示字符串，即**'string'**

后引号

未使用参数 **usebackq**，运行后会提示找不到某某文件，程序却又不认为其是文件，程序发生错乱。所以未用 **usebackq**，不可以使用后引号。
使用参数 **usebackq**，表示命令执行，即**'command'**。

程序例子：

```
@echo off
for /f "delims=" %%i in ('net user') do echo %%i
pause>nul
```

本程序会将你计算机所有的账号都显示出来，例如显示：

```
\\XXX 的用户帐户
-------------------------------------------------------------------------------
Administrator            ASPNET                   Guest
HelpAssistant            SUPPORT_388945a0
命令成功完成。
```

对 **usebackq** 总结：
usebackq 可以认为仅仅用于当路径或文件名有空格时的情况。
格式：**for /f　"usebackq" %%i ("C:\program files\temp.txt") do echo %%i**　　计算机会认为“**C:\program files\temp.txt**”是一个文件的完整路径，对该路径的文件进行操作。

其余的情况不建议使用，多此一举。
字符串情况：　　**for /f %%i ("string") do echo %%i**
计算机会认为 **string** 是字符串，对该字符串进行操作。
执行命令情况：　**for /f %%i ('command') do echo %%i**
计算机会认为 **command** 是命令，对该命令执行。

批处理编程怎么那么麻烦，就仅仅为了增加一个空格转义，就弄多了一个 **usebackq** 参数。而且，为了使用这个 **usebackq** 符号，在我们执行命令时，还需要使用后引号！吐血！

for 实际运用样例（**/f** 的使用不列出来）：
for %%i in (*) do echo %%i
显示当前目录下，所有非文件夹的文件名字。（包括**.rar .jpg .exe .bat .sys** 等等所有非文件夹）
for %%i in (*.*) do echo %%i
搜索当前目录下的所有文件。
for /d %%i in (*) do echo %%i
搜索当前路径下所有文件夹的名字。
for /d %%i in (C:\abc\"ef gh"*) do echo %%i
搜索 **C** 盘 **abc** 文件夹里 **ef gh** 文件夹里面的所有文件的名字。
for /d %%i in (C:\"program files"\????) do echo %%i
搜索 **C** 盘 **program files** 文件夹里，文件夹名字为 1~4 个字符的所有文件夹。
for /d %%i in (win??????????) do @echo %%i
搜索当前路径下以 **win** 开头，后面名字不多于 10 个字符的所有文件夹。
for /d %%i in (C:\"program files"\win*) do echo %%i
搜索 **C:\program files** 文件夹里，所有以 **win** 开头的文件夹。
for /d　%%i in (C:\windows\w???) do echo %%i
搜索 **C:\windows** 所有以 **w** 开头，名字不超过 4 个字符的文件夹。
for /d　%%i in (C:\windows\wbem) do echo %%i
这种搜索就没多大意义了，计算机在 **C:\windows** 搜索到 **wbem** 就显示：**C:\windows\wbem**，否则不显示。

for /r c:\ %%i in (*.exe) do echo %%i
搜索 **C** 盘下包括子目录的所有 **exe** 文件。
for /r %%i in (*.txt) do @echo %%i
搜索当前路径目录树（当前脚本所处的路径）的所有 **txt** 文件。
for /r C:\"program files" %%i in (????.txt) do echo %%i
搜索 **C:\program files** 目录树的所有名字不多于 4 个字符的 **txt** 文件。
for /r C:\windows %%i in (*) do echo %%i
搜索 **C:\windows** 目录树的所有文件。
for /r C:\"program files" %%i in (*.???) do echo %%i
搜索 **C:\program files** 目录树下所有扩展名不超过 3 个字符的所有文件。
for /r C:\windows %%i in (???.??) do echo %%i
搜索 **C:\windows** 目录树下所有文件名不超过 3 个字符并且扩展名不超过 2 个字符的文件。
for /r C:\"program files" %%i in (???.*) do echo %%i
搜索 **C:\program files** 目录树下文件名不超过 3 个字符的所有格式文件。
for /r C:\ %%i in (win*) do echo %%i
搜索 **C** 盘所有文件名以 **win** 开头的文件。
for /r C:\ %%i in (win*.e??) do echo %%i
搜索 **C** 盘所有文件名以 **win** 开头，并且扩展名为 **e** 开头且不超过 3 个字符的所有文件。
for /r C:\ %%i in (boot.ini) do if exist %%i echo %%i
搜索整个 **C** 盘，找出 **boot.ini** 文件。

这是搜索具体文件名的例子，我们一定要记得用 **exist** 判断文件是否存在，存在才输出，否则整个 **C** 盘的文件都会被输出。
for /r C:\windows %%i in (.) do echo %%i
枚举 **C:\windows** 目录树的所有文件。
for /l %%i in (1,1,100**) echo %%i**
循环 100 次，并且输出从 1 到 100 的连续数字。
for /l %%i in (100,-2,0**) echo %%i**
循环 50 次，并且输出从 100 到 0 的连续偶数。

```
@echo off
set str=c d e f g h i j k l m n o p q r s t u v w x y z
echo  当前硬盘的分区有：
for %%i in (%str%) do if exist %%i: echo %%i:
pause
```

这段代码能检测硬盘都有哪些分区，包括 **U** 盘和移动硬盘的分区，但是，当光驱中有盘的时候，也会被列出来，这是一个缺憾。一个解

决办法：利用光驱写保护功能，我们对分区写入数据，如果不能写入，那么就不要显示出来，将上面一段代码修改为：

```
for %%i in (%str%) do (
    if exist %%i: (
        (echo.>>%%i:\1.txt)&&echo %%i:
    )
)
```

用 **wmic**，可以将隐藏分区也显示出来（**wmic diskquota get quotavolume /value**）：

```
@echo off
echo 您的当前分区是：
for /f "tokens=3 delims==" %%a in ('wmic diskquota get quotavolume /value') do call :fuck %%a
pause
:fuck
set shit=%1
echo %shit:~1,2%
goto :eof
```

下面一句代码搞定：

```
wmic logicaldisk where drivetype=3 get caption|findstr :
```

接下来要学的的是 **for** 中的变量，关于 **for** 中的变量。那么多个变量，看着就头疼，背诵就蛋疼！

下面这一段直接复制帮助里的内容：

FOR 变量参照的替换已被增强。您现在可以使用下列选项语法：

~I - 删除任何引号(")，扩充 **%I**

%~fI - 将 **%I** 扩充到一个完全合格的路径名

%~dI - 仅将 **%I** 扩充到一个驱动器号

%~pI - 仅将 **%I** 扩充到一个路径

%~nI - 仅将 **%I** 扩充到一个文件名

%~xI - 仅将 **%I** 扩充到一个文件扩展名

%~sI - 扩充的路径只含有短名

%~aI - 将 **%I** 扩充到文件的文件属性

%~tI - 将 **%I** 扩充到文件的日期/时间

%~zI - 将 **%I** 扩充到文件的大小

%~$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饰符来得到多重结果：

%~dpI　　- 仅将 **%I** 扩充到一个驱动器号和路径
%~nxI　　- 仅将 **%I** 扩充到一个文件名和扩展名
%~fsI　　- 仅将 **%I** 扩充到一个带有短名的完整路径名
%~dp$PATH:i - 查找列在路径环境变量的目录，并将 **%I** 扩充到找到的第一个驱动器号和路径。
%~ftzaI　　- 将 **%I** 扩充到类似输出线路的 **DIR**

在以上例子中，**%I** 和 **PATH** 可用其他有效数值代替。**%~** 语法用一个有效的 **FOR** 变量名终止。选取类似 **%I** 的大写变量名比较易读，而且避免与不分大小写的组合键混淆。

说明：
上面就是 **for** 帮助里关于变量的中的内容。上面每行后面都有一个大写字母 **I**，这个 **I** 其实就是我们在 **for** 代入的变量。我们 **for** 语句代入的变量名是什么，这里就写什么，不能仅仅局限于一个字母 **I** 而已。不过，前后字母的大小

写一定要一致。例如：
for /f　%%t in ('set') do @echo %%t
这里我们要代入的变量名是 **t**，就是帮助里的那个 **I** 改成了 **t**。再比如：
for /f　%%d in ('set') do @echo %%~dd
类似 **dd** 这样重复写两个变量字符的例子是可以的，但不建议这么写。
但不能这么写：
for /f %%t in ('set') do @echo %%T
前后两个 **T** 变量大小写不一致，计算机认为是不同的两个变量。
至于 **I** 前面的例如“**%~f**”这样的内容，就是有固定含义的语法了，字母不能随便改，但大小写倒没什么所谓，例如：**%~fi** 和**%~Fi** 是一样的。
注意：**%**符号，在批处理里要使用两个，但在 **CMD** 窗口直接输入执行，只要一个就行了。
例如：
在批处理执行的代码：
for /f "delims="　%%i in ('dir /b') do @echo %%~ti
在 **CMD** 窗口执行的代码要改为：
for /f "delims="　%i in ('dir /b') do @echo %~ti
注意，**echo** 与后面的**%**之间要有空格。

下面对每个变量逐个探究：

~I
对操作内容删除双引号后扩充到**%I**。注意，单引号不会被删除。

程序例子：
```
@echo off
set var=">>"
for %%i in (%var%) do echo %%~i
pause
```
输出为>>，没有双引号。

```
@echo off
for %%i in (a,"b c",d) do echo %%i
pause
```
执行结果显示：
```
a
"b c"
d
```
上面会显示双引号，假如我不想要显示双引号，但仍旧要显示 **b** 和 **c** 为同一组，**b** 和 **c** 之间有一个空格，代码应该改为：
```
@echo off
for %%i in (a,"b c",d) do echo %%~i
pause
```

再看一例子：
我们首先建立临时文件 **temp.txt**，里面内容为：
```
"1111
"2222"
3333"
"4444"44
"55"55"55
```

也可以通过批处理建立临时文件 **temp.txt** 文件：
```
@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
```

::上面建立临时文件，注意不成对的引号要加转义字符^，重向符号前不要留空格。（这个，我测试过了，有无空格都没问题，书上这么说，也许是因为程序代码不够复杂，复杂了就容易出错）

建立好文件后，在同目录下运行批处理：

```
@echo off
for /f "delims=" %%i in (temp.txt) do echo %%~i
pause
del temp.txt
```

执行结果显示：

```
1111
2222
3333"
4444"44
55"55"55
请按任意键继续. . .
```

从上面规则可以看出删除规则如下：

1、若字符串首尾同时存在引号，则删除首尾引号。

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。

总结：无头不删，有头连尾删。

如果我们不用**%%~i** 变量，程序改为如下：

```
@echo off
for /f "delims=" %%i in (temp.txt) do echo %%i
pause
```

那么，代码执行的结果是将 **temp.txt** 里面所有的内容都显示出来：

```
"1111
"2222"
3333"
"4444"44
"55"55"55
请按任意键继续. . .
```

上面代码里，有一个**"delims="**，它的作用是即使文本包含空格，也可以显示所有内容。详细用法参见上面 **delims** 参数讲解。

```
@echo off
for /f   "delims="   %%i in ('dir /b') do echo %%~i
pause
```

运行程序将当前目录下的所有文件以及文件夹的名字显示出来。**Windows** 规定，文件名称不能含有双引号，所以这个代码看不出 **%%~i** 有什么效果。

%~fI
将**%I** 扩充到完整路径。

程序例子（前提条件：我桌面有且只有两个文件 **test.bat,test.vbs**）：
```
@echo off
for /f    %%i in ('dir /b') do echo %%~fi
pause
```
把代码随便放在一个地方，例如桌面，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

当我们将代码中的**%%~fi** 直接改成**%%i**
```
@echo off
for /f    %%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，后面那个没有路径了，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，它就会把这个文件所在的绝对路径显示出来，而不只显示一个文件名而已。这里，注

意**'dir /b'**的作用，如果我们将程序改为：
```
@echo off
for /f "delims=" %%i in ('dir') do echo %%~fi
pause
```
那么，程序不仅会显示文件的完整路径，而且还会显示日期和大小等信息，这不是我们想要的结果。关于 **dir** 的详细用法，参见 **dir** 专题研究。

%~dI
仅将 **%I** 扩充到一个驱动器号。
程序例子（前提条件：我桌面就两个文件 **test.bat,test.vbs**）：
```
for /f    %%i in ('dir /b') do @echo %%~di
pause
```
执行后在 **CMD** 里显示如下：
```
C:
C:
```
上面例子说明**%%~di** 的作用是：如果变量**%%i** 的内容是一个文件或者目录名，它就会把这个文件或者目录所在盘显示出来。

%~pI
仅将 **%I** 扩充到一个路径
程序例子（前提条件：我桌面就两个文件 **test.bat,test.vbs**）：

```
@echo off
for /f    %%i in ('dir /b') do @echo %%~pi
pause
```

运行结果：
```
\Documents and Settings\Administrator\桌面\
\Documents and Settings\Administrator\桌面\
```
上面运行结果说明**%~pI** 只显示路径，不显示盘符和文件名。

%~nI
仅将 **%I** 扩充到一个文件名
程序例子（前提条件：我桌面就两个文件 **test.bat,test.vbs**）：

```
@echo off
for /f "delims==" %%i in ('dir /b') do @echo %%~ni
pause
```

运行结果：
```
test
test
```
上面说明：**%~nI** 仅显示文件的主文件名，连扩展名都不显示。

%~xI
仅将 **%I** 扩充到一个文件扩展名
程序例子（前提条件：我桌面就两个文件 **test.bat,test.vbs**）：

```
@echo off
for /f "delims==" %%i in ('dir /b') do @echo %%~xi
pause
```

运行结果：

```
.bat
.vbs
```

显示扩展名时，会将一点也显示出来。

%~sI
扩充的路径只含有短名
程序例子（前提条件：我桌面就两个文件 **test.bat,test.vbs**）：

```
@echo off
for /f "delims==" %%i in ('dir /b') do @echo %%~si
pause
```

运行结果：

```
C:\DOCUME~1\ADMINI~1\桌面\test.bat
C:\DOCUME~1\ADMINI~1\桌面\test.vbs
```

这里，运用了 **DOS** 的显示规则，当文件名超过 8 个字符时，采用“**ABCDEF~1**”短名形式显示出来，也就是显示出原来名字的前六个字符，然后是字符~，再加上一个数字，共计还是 8 个字符。数字是为了区分前六个字符完全相同

的文件名，例如有两个文件的名字为 **Abcdefghijk.txt** 和 **Abcdefgh.txt**，缩减后显示为 **ABCDEF~1.TXT**，**ABCDEF~2.TXT**，缩减后的名字，连同扩展名都会采用全部大写的形式。一个中文文字占用两个字符大小，所以如果是《新建

文本文档**.txt**》，那么这个文件缩减后的名字为“新建文~1.TXT”，上面的 **test.bat** 和 **test.vbs** 两个文件的名字比较短，所以还显示出全名来，文件名包括扩展名的大小写没有改变。

%~aI
将 **%I** 扩充到文件的文件属性
程序例子（前提条件：我桌面就两个文件 **test.bat,test.vbs**）：

```
@echo off
for /f "delims==" %%i in ('dir /b') do @echo %%~ai
pause
```

运行结果显示：
--a------
--a------

下面列举一些可能出现的属性例子：
----
d--------
dr-------
--a------
-ra------
d-a------
这些是什么意思呢？网上只查出来这几个：
R 只读文件属性。
A 存档文件属性。
S 系统文件属性。
H 隐藏文件属性。
I 无内容索引文件属性。
解释：
假如一个文件的属性是隐藏的，用上面的代码不会显示出其 **H** 属性，而是根本没搜索到这个文件。假如一个文件的属性是存档，那么就会显示**--a------**，假如是一个文件夹，那么就会显示 **d--------**，假如一个文件的属性是存

档并且只读，那么显示**-ra------**，假如一个文件夹的属性是存档并且只读，那么属性是 **dr-------**，假如不是文件夹的文件，所有属性都不是，那么显示**---------**

%~tI
将 **%I** 扩充到文件的日期/时间
显示文件建立的日期和时间。
程序例子（前提条件：我桌面就两个文件 **test.bat,test.vbs**）：

```
@echo off
for /f "delims==" %%i in ('dir /b') do echo %%~ti
pause
```

运行结果显示：
2013-05-29 09:03
2013-06-24 11:34
也就是说**%~tI** 显示文件建立的日期和时间。

%~zI
将 **%I** 扩充到文件的大小
大小的默认单位是字节（**B**）。
程序例子：

```
@echo off
for /f "delims=" %%i in ('dir /b') do @echo %%~zi
pause
```

显示
456
156467
这里只显示出数字，默认的单位是 **B**，也就是字节。

%~$PATH:I
查找列在路径环境变量的目录，并将 **%I** 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。
这个和上面的都不太一样，看程序例子：

```
@echo off
for /f "delims=" %%i in ("notepad.exe") do echo %%~$path:i
pause
```

运行结果：
C:\WINDOWS\system32\notepad.exe
上面程序的意思是在**%path%**变量里指定的路径里搜索 **notepad.exe** 文件，并显示第一个合格的路径，没有就显示出一个错误。
%path%的路径有哪些？我用代码在本机运行：
@echo.%path%
pause
显示如下：
C:\WINDOWS\system32;
C:\WINDOWS;C:\WINDOWS\System32\Wbem;
C:\Program Files\Microsoft SQL Server\100\Tools\Binn\;
C:\Program Files\Microsoft SQL Server\100\DTS\Binn\;
D:\QuickTime\QTSystem
请按任意键继续. . .

再比如程序例子：
@echo off
for %%a in (notepad.exe calc.exe explorer.exe) do echo 查找**%%a** 的第一个结果是：**%%~$PATH:apause**
pause
运行结果为：
查找 **notepad.exe** 的第一个结果是：
C:\WINDOWS\system32\notepad.exepause
查找 **calc.exe** 的第一个结果是：
C:\WINDOWS\system32\calc.exepause
查找 **explorer.exe** 的第一个结果是：**C:\WINDOWS\explorer.exepause**
请按任意键继续. . .
程序会将目标文件逐一搜索并显示出来。

进一步分析：
其实，**%~$path:i** 可以将其分解为**%i** 和**%path%**，**%I** 和 **PATH** 可用其他有效数值代替。
程序例子：
@echo off
set a=D:\Tencent\QQ\Bin
for /f "delims=" %%i in ("qq.exe") do echo %%~$a:i
pause
运行结果显示：
D:\Tencent\QQ\Bin\QQ.exe
请按任意键继续. . .

这个，似乎没什么用，因为要指定的路径时精确的路径，不能搜索目录树，好像失去了搜索的意义了。

一个问题分析：
我们在 **CMD** 窗口输入：
dir
dir /b
dir /b /ah
dir C:\Windows
dir C:\Windows /b
dir C:\windows /b /ah
dir C:\boot.ini /b /ah
所有上面的指令都可以执行。将上面保存为批处理文件，也可以正确执行。
下面程序：
```
for /f "delims=" %%i in ('dir /b') do @echo %%~ti
pause
```
上面代码也可以正确执行，执行结果显示文件时间。这段代码，**dir** 是没有指名搜索路径的，那么搜索就默认为搜索当前路径。
下面的代码指明了搜索路径：
```
for /f "delims=" %%i in ('dir C:\windows /b') do @echo %%~ti
pause
```
程序不能执行！我们对搜索的格式无论怎么改，例如：
```
for /f "delims=" %%i in ('dir C:\ /b /ah') do @echo %%~ti
```
程序还是无法正确执行。执行结果显示：**echo** 处于打开状态。

将代码改为直接在 **CMD** 窗口执行又如何呢？代码要稍微改动一下：
```
for /f "delims="    %i in ('dir C:\ /b /ah') do @echo %~ti
```
也是不能执行！

%~aI
%~tI
%~zI
%~$path:I
这四个变量都有这个现象，就是不能指定路径！但其余的变量都不会出现这个问题。是不是百思不得其解？这个是在挑战我们的智商吗？！

出现上面问题的原因是
%~aI
%~tI
%~zI
需要指明完整路径，而 **dir** 搜索出的只是文件名，这样**%%~ti** 就把当前工作目录设置为文件的路径，所以找不到文件。可以用 **pushd** 来转到相应目录就行了：

```
@echo off
pushd C:\windows
for /f "delims=" %%i in ('dir C:\windows /b') do echo %%~ti
pause
```

也可以用 **cd /d** 来指明路径：

```
@echo off
cd /d C:\windows
for /f "delims=" %%i in ('dir C:\windows /b') do echo %%~ai
pause>nul
```

最后一个，**%~$path:I** 非常特殊，

```
@echo off
pushd C:\
for /f "delims=" %%i in ('dir c:\boot.ini /b /ah') do echo
%%~$path:i
pause
```

还是不能正确执行。不过，程序改为：

```
@echo off
for %%i in ('dir C:\boot.ini ') do echo. %%~path:i
pause
```

则显示

```
C:\boot.ini

请按任意键继续. . .
```

显示两个空字符串和一个正确的结果。

程序例子：
for %a in ("%path:;=" "%") do　echo %~a

运行结果如下：

C:\WINDOWS\system32
C:\WINDOWS
C:\WINDOWS\System32**\Wbem**

目的就是把 **PATH** 环境变量的每个路径分离出来进行逐行显示

拾遗：
一：
for 语句里，**do** 后面一般会有括号，有了括号就是复合语句，假如我们需要用到括号里的变量，就需要延迟变量。于是，基本上有 **for** 的地方，就需要用到延迟变量。

二：**for** 语句里，不能改变循环变量**%%i** 的值！
例如：

```
@echo off
setlocal enabledelayedexpansion
for /l %%i in (1,1,10) do (
                        set /a %%i-=2
                        set /a i-=2
                        set /a !i!-=2
                        set /a %%i=%%i+2
                        set /a %%i=!%%i!+2
                        set /a %%i=-%%i
                        set /a %%i=-!%%i!
                        set %%i=-%%i
                        set %%i=-!%%i!
)
Pause
```

上面一切形式对循环变量**%%i** 进行运算以及添加负号都是无效的！循环变量**%%i** 值不会有任何变化，并且会显示“找不到操作数”。

三：**for** 语句里，可以改变上下限的值，但不会影响循环次数。

```
@echo off
setlocal enabledelayedexpansion
set min=1
set max=10
for /l %%i in (!min!,1,!max!) do (
                        set /a min+=2
                        set /a max*=3
                        echo min=!min! max=!max!
)
pause
```

四：**for** 语句里，我们的**%var:~m,n%**形式截取字符，会变得掣肘！主要原因是因为 **for** 语句里不能对循环变量**%%i** 进行运算，以及在延迟变量环境下，很难对 **m**、**n** 是变量的情况下的表达。

情况 1：假如 **m** 恒等于**%%i**，而 **n** 总比 **m** 相差一个定值 **P**，如何表达**%var:~m,n%**这个形式？
情况 2：假如 **m** 或 **n** 需要用到**%%i** 的负值，也就是**-%%i** 形式，如何表达**%var:~m,n%**这个形式？

对于情况 1，我们的第一反应就是引入一个新变量 **j**，我们对 **j** 值可以进行运算，然后将 **j** 代入**%var:~m,n%**就行了。代码如下：

```
@echo off
setlocal enabledelayedexpansion
set var=abcdefghijklmnopqrstuvwxyz
for /l %%i in (1,1,10) do (
                        set /a j=%%i-1
                        set tem=!var:~%%i,j!
                        echo !tem!
)
pause
```

上面的代码肯定不行，因为 **set tem=!var:~%%i,j!**这里的 **j** 是变量，没有用到变量符号将其表达出来。我们的代码用了延迟变量，那么表达式改为 **set tem=!var:~%%i,%j%!**肯定不对。
那么用感叹号如何呢？**set tem=!var:~%%i,!j!!**这个也不行，因为从计算机角度看来，感叹号括到的范围为**!var:~%%i,!**和 **j!!**，这个结果不是我们想要的。那么，如果先对 **j** 进行处理，**set j=!!j!!**，然后

set tem=!var:~%%i,j!，这样也不行。也就是说，假如 **m** 值和 **n** 值，是有一定关系差值的动态变化情况，我们就无法简单用**%var:~m,n%**这个形式表达出来，得另想办法，具体实现，要看实际要达到什么效果，这个在我写的一些数值计算例子里有介绍。

对于情况 2，由于不能对**%%i** 进行取负值，于是我们可以从一开始循环时对循环参数取负值。

```
@echo off
setlocal enabledelayedexpansion
set var=abcdefghijklmnopqrstuvwxyz
for /l %%i in (-1,-1,-10) do (
                        set tem=!var:~%%i!
                        echo !tem!
)
Pause
```

上面，我们只需要用到 **m** 值，假如我们的 **n** 值随 **m** 值变化而变化，那么我们想用**%var:~m,n%**这个形式又束手无策了，得另想办法，具体实现，要看实际要达到什么效果。

一个变通的例子：输入一串字符，要求从字符串末尾一个一个字符输出到字符串开头。

```
@echo off
setlocal enabledelayedexpansion
set var=abcdefghijklmnopqrstuvwxyz
for /l %%i in ( -1,-1,-26 ) do (
                        set tem=!var:~%%i!
                        set tem= !tem!
                        set tem=!tem:~1,1!
                        echo !tem!
)
pause
```

五：读入一个有两个数的算式，将其中的两个数赋值到字符串并回显出来。

我们在输入数学表达式的时候，都是输入像 12+34,25-6 等这样的表达式，所以实现上面效果对编辑计算器非常有用，符合我们的输入习惯。

```
@echo off
setlocal enabledelayedexpansion
set num=123+456
for /f "delims=+ tokens=1,2" %%i in ("!num!") do (
set num1=%%i
set num2=%%j
echo num1=!num1! num2=!num2!
)
pause
```

这里要注意："%num%"一定要用双引号引起来，否则计算机会显示找不到文件。

常用特殊符号

@　关闭命令行回显符号，放在命令行前面，可以屏蔽命令行本身。

%　严格来说，这个符号算不上命令，它只是批处理中的参数而已(多个%一起使用 情况除外)。这个符号非常特殊，用法复杂。

|　命令管道符

把前一命令的输出结果当后一命令的输入参数来使用。

格式：第一条命令 | 第二条命令 [| 第三条命令.....]

例如：

dir C:\ | find "e"

这个命令的执行结果为：在 C 盘查找，列出其中包含 e 字符的所有文件和文件夹。

再例如：

在不使用 **format** 的自动格式化参数时，可以这样来自动格式化 A 盘

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

用过 **format** 的都知道，再格式盘时要输入 y 来确定是否格盘，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。(此命令有危险性，测试请慎重！)

>、>>　这两个命令都是输出重定向符，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“>>”追加新的内容到原文件末尾，不会改动其中的原有内容。

程序举例：

if /? > if.txt

执行上面代码后，将在本文件所在目录产生一个名为 **if** 的 **txt** 文件，

里面记录 **if** 的所有帮助内容。

如果输出的文件名含有空格，那么文件名就需要用双引号括起来：
reg add /? >"reg add".txt
reg add /? >"reg add.txt"
上面这两种格式都是可以的。

如果要将某个变量写入文件，就要借助 **echo** 了。
程序例子：
@echo off
set num=123
echo %num%>>D:\num.txt
上面是输出到指定路径的例子。

程序例子：
@echo off
echo Hello World!>file.doc
格式细节注意：上面的的“**Hello World!**”与“**>**”之间没有空格，那么输出的“**Hello World!**”末端也没有空格。如果上面的的“**Hello World!**”与“**>**”之间有空格，那么输出为“**Hello World! **”，也有空格。
而**>**或者**>>**与 **file.doc** 之间有无空格都无所谓。

程序例子：
for /l %%i in (1,1,100**) do set /p var=%%i<nul>>num.txt**
将 1 到 100 连续不间断写入文本文档 **num.txt** 中。

<　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。
例如，**D** 盘里有一个名为日期的 **txt** 文件，并且文件里的内容为：
2013-06-09
那么执行下面的批处理：
@echo off
date < D:\日期**.txt**
这样就可以不用等待而直接将当前日期修改为 2013 年 6 月 9 日。

>& 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。

---

^　　转义字符

是<、>、&这三个符号以及回车的前导符，作用是将<、>、&这三个符号的特殊功能和回车去掉，仅仅把它当成符号而不使用它们的特殊意义。

例如：

```
echo abc^>tmp.txt
```

结果是在窗口中显示

```
abc>tmp.txt
```

没有将“abc”内容写入 tmp.txt，而是将“abc”连同一个空格一起写入到 tmp.txt 文件里。

注意：^与>之间不可以有空格，否则无法转义。如果“abc”与“^”之间有空格，这个空格会当做字符一并显示出来。“>”与“tmp”如果有空格，同样这个空格也会被当做字符一并显示出来。

如果有括号包含 set /a 计算表达式，表达式也有括号，那么表达式的括号必须用转义字符，否则会显示括号不成对：

```
@echo off
setlocal enabledelayedexpansion
for /l %%i in (1,1,10) do (
                        set /a var=(%%i+1^)*2
                        echo !var!
)
pause
```

注意到上面 set /a var=(%%i+1^)*2 括号有一个转义字符。

输出“^ > >> & && | ||”要借助转义字符。另外，在延迟变量的情况下输出感叹号，则需要连续使用两个转义字符：

```
@echo off
echo ^^
echo ^>
echo ^>^>
echo ^&
```

echo ^&^&
echo ^|
echo ^|^|
pause

@echo off
setlocal enabledelayedexpansion
echo ^^!
pause

转义字符高级篇：转义字符用于续行

@echo off
echo 中^
华人^
民^
共和国
pause

窗口显示：
中华人民共和国
按任意键继续...

注意，最后的“共和国”后面不要加转义字符，否则 **pause** 也被续上去了。另外，^后面不要有空格，否则达不到续行的作用。为什么转义字符放在行尾可以起到续行符的作用呢？原因很简单，因为每个行尾都有一个看不见的符号，即回车符。转义字符位于行尾时，就让回车符失效了，从而起到续行的作用。但是，由于 **echo** 可以显示多达 8184 个字符，所以嘛，用转义字符来续行徒增麻烦，用处不大。

&　　组合命令

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。

语法：
第一条命令 **&** 第二条命令 **[&** 第三条命令**...]**

程序例子：
dir Z:\&dir Y:\&dir C:
以上命令会连续显示 **Z**、**Y**、**C** 盘内容，不理会该盘是否存在。
格式细节：**dir** 是关键字，后面一定要有空格。而其余地方有无空格都无所谓。当然，我们这样写也是可以的：
dir Z:\ & dir Y:\ & dir C:
这样明朗一点。
copy a.txt b.txt /y & del a.txt
这里**&**两边的命令是有执行顺序的，从前往后执行。

&&　　组合命令

这个命令也是把它前后两个命令连接起来，并按这些命令的顺序执行。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。

语法：
第一条命令 **&&** 第二条命令 **[&&** 第三条命令**...]**

程序例子：
dir Z:\ && dir Y:\ && dir C:
程序执行到 **dir Z:**，显示“系统找不到路径”后就不执行后面的命令了。

|| 组合命令符

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，将不执行后面所有的命令。如果没有出现正确的命令，则一直执行完所有的命令。
提示：组合命令和重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别高于组合命令。
程序例子：
问题：把 C 盘和 D 盘的文件和文件夹列出到 a.txt 文件中。
dir C:\ && dir D:\ > a.txt
这样执行后，**a.txt** 里只有 **D** 盘的信息！为什么？因为组合命令的优先级别没有重定向命令的优先级别高！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，必须这么写：
dir C:\> a.txt && dir D:\ >>a.txt
这样，根据优先级别的高低，**DOS** 将把这句话分成下面两部分：**dir C:\ >a.txt** 和 **dir D:\>>a.txt**。
当然，这里还可以用**&**命令：
dir C:\ >a.txt & dir D:\ >>a.txt

"" 双引号是字符串界定符

表示双引号内的内容界定为字符。
常用于文件、文件夹名字带空格的情况，以及表示字符串。
另外，我们在 **if** 语句里，比较两个变量时，有时用双引号可以对程序的健壮性有很大好处。
例如：
@echo off
set /p choose=请选择(Y/N)：
if /i "choose"=="Y" (echo Yes) else (echo No)
pause
这里，如果我们不用双引号，那么假如我们读入的数据是一个回车，那么程序就会出错而自动退出。有双引号，则输入回车，输出为 **No**。

比较两个数字时，如果用双引号，那么计算机就会认为它们不再是数字，而作为字符来比较。
例如 987 比 1234 小，但"987"会比"1234"大。测试代码如下：

```
@echo off
set A=987
set B=1234
if %A% gtr %B% (echo A 大于 B) else (echo A 小于 B)
if "%A%" gtr "%B%" (echo "A"大于"B") else (echo "A"小于"B")
pause
```

例子：

```
C:\>dir "program files"
```

要用双引号才有效，因为文件名“**program files**”存在空格。

字符界定符高级篇：
假如两个字符串没有包含任何内容，就表示空值，这个可以用来判断某个变量的值是否存在！
测试代码：

```
@echo off
if "%A%" equ "" (echo A 是空值。) else (echo A 值不是空值，A 值等于%A%)
set /p A=请输入 A 值：
if "%A%" equ "" (echo A 是空值。) else (echo A 值不是空值，A 值等于%A%)
pause
```

当然，用 **if** 的 **defined** 也可以判断字符是否存在，详见 **if** 命令。

,　　逗号相当于空格，在某些情况下可以当做空格使用。

程序例子：

```
@echo off
echo,Hello world!
start,iexplore.exe
dir,E:\
pause
```

程序可以执行。但这种写法很邪恶，应该摒弃。

对于 **set /a** 语句，可以连续赋值多个变量：

```
@echo off
set /a X=Y=Z=0
echo x=%X% y=%Y% z=%Z%
set /a x+=1,y*=2,z-=0
echo x=%X% y=%Y% z=%Z%
pause
```

;　　分号

当命令相同时，可以将不同目标用分号隔离，执行效果不变。

此符号属于旁门左道，不宜采用！

程序例子：

```
dir c:\;d:\;e:\;z:\
pause
```

上面命令相当于

```
dir C:\
dir D:\
dir E:\
dir Z:\
pause
```

而如果没有 **Z** 盘存在，则所有 **dir C**、**D**、**E** 盘都不会执行，后面如果还有命令，也不会执行。

语法细节：“**dir c:\;d:\;e:\;z:**”指令，分号与别的字符间可以有空格，也可以没有空格。

程序例子：

```
dir C:\;D:\;E:\adobe\;F:\
```

假若 **E** 盘不存在名字为 **adobe** 的文件夹，程序仍然会执行 **dir C**、**D**、**F** 盘，只是执行到 **E** 盘后找不到 **adboe** 文件夹，会提示“找不到文件”。

结论：对于 **dir** 命令，如果目标驱动器不存在，则所有命令都不会执行。如果驱动器存在，仅文件或文件夹不存在，所有有效的命令都会执行。

再看一个例子：

假如我们的 **qq.exe** 和 **qqgame.exe** 的路径分别为 **D:\Tencent\QQ\Bin\qq.exe** 和 **D:\Tencent\QQGame\qqgame.exe**，那么下面程序：

```
@echo off
start D:\Tencent\QQ\Bin\qq.exe
start D:\Tencent\QQGame\qqgame.exe
pause
```

程序可以正常执行。但如果用分号修改为：

```
start D:\Tencent\QQ\Bin\qq.exe;D:\Tencent\QQGame\qqgame.exe
```

则程序无法执行。程序修改为：

```
start D:\Tencent\QQ\Bin\qq.exe ; D:\Tencent\QQGame\qqgame.exe
```

这个程序和上面程序的区别仅仅是分号前后各添加了空格，程序可以执行，但只打开了 **qq.exe**，后面的没有执行。

再举一例：

```
@echo off
set a=123;b=456
echo %a%
echo %b%
pause
```

这个程序根本不会将 456 赋值到 **b**，而 **a** 的值就是“123;b=456”。

结论：对于 **set**，分号无效。**set** 语句要用逗号：**set /a a=123,b=456**，在这里，逗号是必须的，而且还不能用空格。

结论：分号存在的意义不大，尽可能不要用分号。

( )　　左右括号必须成对使用，括号中可以包含多行命令，这些命令将被看成一个整体，视为一条命令行，即复合语句。此时，如果要用到括号里的代码动态实时值，就需要用到延迟变量了。

例如：**echo** 1 **&** **echo** 2 **&** **echo** 3 可以写成

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

例子二：

```
@echo off
setlocal enabledelayedexpansion
set var=0
(
set /a var+=1
echo var1=%var% var2=!var!
)
pause
```

输出结果为：第一个 **var** 等于 0，第二个 **var** 等于 1。
从上面可以看出，有了括号后，要实时得到变量的动态值，就必须用延迟变量。

!　　感叹号在变量延迟问题**(setlocal enabledelayedexpansion)**中，用来表示变量。即**%var%**变成**!var!**。详见延迟变量专题 **setlocal enabledelayedexpansion**。

易出错举例

批处理的语法格式比较复杂，有好多地方都很容易出错，令程序员苦恼不已。下面列举一些容易出错的语法供大家参考参考：

变量的书写格式：

所有情况下，对于 **for** 循环变量，**%%i** 变量的书写保持不变。

例如：

```
@echo off
setlocal EnableDelayedExpansion
set num=100
set var=a
set t=80
for /l %%i in (1,1,%num%) do (set S%%i=1)
for /l %%i in (1,1,%num%) do (echo !S%%i!)
pause
if 1 lss 2 (
    echo Hello
    echo World!
    pause
)
for /l %%i in (1,1,%num%) do (
    for /l %%j in (1,1,%%i) do (
        echo i=%%i j=%%j
                    set /a var=%%i%%%%j
        echo %var%
        echo !var!
        set /a n=!var!%%10
        set /a u=%t%%%2
        set /a m+=1
        if !m!==%t% echo Hello! %t% & pause
    )
)
pause
```

语法格式分析：
1、**if** 语句，1 **lss** 2 是条件，条件后面必须带空格！不能因为是括号就忘记带空格！写成 **if** 1 **lss** 2**(echo Hello World!)**错误！
再举例子：

```
if 2 neq 3(
    echo Hello!
    echo world!
)

if 2 neq 3 (
    echo Hello!
    echo world!
)
```

这是两种写法，初学者一粗心就写成上面写法，在条件 2 **neq** 3 后面没有空格，导致程序语法错误，而且因为是括号的缘故，还愣查不出原因，郁闷啊！第二种写法才正确，切记切记！

2、**(echo !S%%i!)**这里因为**!S%%i!**在括号里，所以即使这里只有一句代码，它也是复合语句，所以不能用**%S%%i%**形式。

3、**(1,1,%num%)**，这个 **num** 可以用**!num!**也可以用**%num%**

4、**(1,1,%%i)**，这个**%%i** 本身就代表变量，无需添加**%%**或**!!**

5、**set /a var=%%i%%%%j** 表示变量 **i** 对变量 **j** 取模（余数）。这种写法会让人很别扭，可以这么写明朗点 **set /a var=%%i %% %%j**

6、**echo !var!**，这里 **var** 因为在复合语句里，那么**%var%**恒等于 **a**，**!var!** 则是变量 **i** 和 **j** 的余数。

7、**if !m!==%t%**，这里 **m** 为了使用复合语句里的动态值，所以必须用**!m!**格式，**%t%**因为一开始赋值是再复合语句之外 **set t=**80，此时用**%t%**可以，值为 80，用**!t!**也可以，值也等于 80。假如 **t** 是再复合语句之内，要调用 **t** 的动态值时，就必须用**!t!**格式。

---

set /a 后面表达式用双引号括起来，可以防止一些意外的错误。

if 语句不会进行运算，所以下面程序：
if %**num**%%%5== 0 **echo** %**num**%可以被 5 整除
这种写法不可以，运行错误。必须分开写。
set /a panduan=%**num**%%%5
if %**panduan**% == 0 **echo** %**num**%可以被 5 整除
分成这两段写就行了。

当引用文件名带有空格时，要记得添加双引号。

for 语句里，**do** 后面一般会有括号，有了括号就是复合语句，假如我们需要用到括号里的变量，就需要延迟变量。于是，基本上有 **for** 的地方，就需要用到延迟变量。

for 语句里，不能对循环变量%%**i** 进行运算，添加负号都不行！
例如：

```
@echo off
setlocal enabledelayedexpansion
for /l %%i in (1,1,10) do (
    set /a %%i-=2
    set /a i-=2
    set /a !i!-=2
    set /a %%i=%%i+2
    set /a %%i=!%%i!+2
    set /a %%i=-%%i
    set /a %%i=-!%%i!
    set %%i=-%%i
    set %%i=-!%%i!
)
pause
```

上面一切形式对循环变量%%**i** 进行运算以及添加负号都是无效的！循环变量%%**i** 值不会有任何变化。

for 语句里，我们的%**var:~m,n**%形式截取字符，会变得掣肘！主要原因是因为 **for** 语句里不能对循环变量%%**i** 进行运算，以及在延迟变量环境下，很难对 **m**、**n** 是变量的情况下的表达。

问题 1：假如 **m** 恒等于%%**i**，而 **n** 总比 **m** 相差一个定值 **P**，如何表达%**var:~m,n**%这个形式？
问题 2：假如 **m** 或 **n** 需要用到%%**i** 的负值，也就是-%%**i** 形式，如何表达%**var:~m,n**%这个形式？

对于问题 1，我们的第一反应就是引入一个新变量 **j**，我们对 **j** 值可以进行运算，然后将 **j** 代入%**var:~m,n**%就行了。代码如下：

```
@echo off
setlocal enabledelayedexpansion
set var=abcdefghijklmnopqrstuvwxyz
for /l %%i in (1,1,10) do (
    set /a j=%%i-1
    set tem=!var:~%%i,j!
    echo !tem!
)
pause
```

上面的代码肯定不行，因为 **set tem=!var:~%%i,j!**这里的 **j** 是变量，没有用到变量符号将其表达出来。我们的代码用了延迟变量，那么表达式改为 **set tem=!var:~%%i,%j%!**肯定不对。
那么用感叹号如何呢？**set tem=!var:~%%i,!j!!**这个也不行，因为从计算机角度看来，感叹号括到的范围为**!var:~%%i,!**和 **j!!**，这个结果不是我们想要的。那么，如果先对 **j** 进行处理，**set j=!!j!!**，然后 **set tem=!var:~%%i,j!**，这样也不行。也就是说，假如 **m** 值和 **n** 值，是有一定关系差值的动态变化情况，我们就无法简单用%**var:~m,n**%这个形式表达出来，得另想办法。

对于问题 2，由于不能对%%**i** 进行取负值，于是我们可以从一开始循环时对循环参数取负值。

```
@echo off
setlocal enabledelayedexpansion
set var=abcdefghijklmnopqrstuvwxyz
for /l %%i in (-1,-1,-10) do (
    set tem=!var:~%%i!
    echo !tem!
)
pause
```

上面，我们只需要用到 **m** 值，假如我们的 **n** 值随 **m** 值变化而变化，那么我们想用%**var:~m,n**%这个形式又束手无策了，得另想办法。

一个变通的例子：输入一串字符，要求从字符串末尾一个一个字符输出到字符串开头。

```
@echo off
setlocal enabledelayedexpansion
set var=abcdefghijklmnopqrstuvwxyz
for /l %%i in ( -1,-1,-26 ) do (
    set tem=!var:~%%i!
    set tem= !tem!
    set tem=!tem:~1,1!
    echo !tem!
)
pause
```

一些常用的命令

学习完上面知识后，对批处理的运行机制基本清楚了，可以写出一些初步的代码。但还远远不够，要想实现更多的功能，必须学习更多的 **DOS** 命令。

time 和 date

time 显示或设置系统时间
date 显示或设置系统日期

语法：
TIME [/T | time]
DATE　[/T | date]

显示当前时间/日期设置和输入新时间/日期的提示，请键入不带参数的 **TIME/DATE**。
输入代码：
time
程序运行后显示如下

当前时间：　9:19:30.92
输入新时间:

如果我们想修改系统时间，则按提示输入新时间；不想修改系统时间，按 **ENTER** 结束输入。

输入代码：
date
程序运行如下：

当前日期: 2013-09-25 星期三
输入新日期: (年月日)

如果我们想修改系统日期，则按提示输入新日期；不想修改系统日期，按 **ENTER** 结束输入。

/t 参数，只显示时间/日期，不提示输入新的时间/日期。对于 **time**，会省略显示秒，只显示小时和分；对于 **date**，有没有加参数**/t** 效果一样。
例如输入代码：
time /t
系统显示：
09:23

例如输入代码：
date /t
系统显示：
2013-09-25 星期三

直接修改时间/日期：
输入：

time 09	修改当前时间为 09:00 整
time 09:13	修改当前时间为 09:13:00 整
time 09:13:30	修改当前时间为 09:13:13.00 整
time 09:13:30.25	修改当前时间为 09:13:30.25 精确修改

date 2013-10-01
修改日期只有这一种修改方法，下面的方法与这个一样。
date 2013-10-1
date 2013/10/3
date 2013/09-25
而“**date** 2013”或“**date** 2013-10”都是不可以的。

代码：
date 2013/09/19
time 19:13:00
直接将系统日期和时间修改为上面指定的时间。

VOL

显示磁盘卷标和序列号(如果存在)。

语法：**VOL [drive:]**

输入：
vol D:
显示：
驱动器 **D** 中的卷是 软件
卷的序列号是 000**B**-6514

VER

显示 **WindowsXP** 版本
语法：
ver

输入：
ver
显示：
Microsoft Windows XP [版本 5.1.2600]

TREE

以图形模式显示驱动器或路径的目录结构。

语法：
TREE [drive:][path] [/F] [/A]
/F 显示每个文件夹中文件的名称。
/A 使用 **ASCII** 字符，而不使用扩展字符。

没有参数**/f**，则列出目录下的所有目录**(**文件夹名**)**以及子目录**(**子文件夹名**)**。带参数**/f** 则列出目录下的所有目录、子目录、每个目录下的所有文件。

例子：

tree D:	显示 **D** 盘根目录结构
tree D:\tencent	显示 **D** 盘 **tencent** 目录结构
tree D:\tencent /f	显示 **D** 盘 **tencent** 目录结构及其所有文件
tree D:\tencent /a	以 **ASCII** 码显示 **D** 盘 **tencent** 目录结构
tree D:\tencent /f /a	以 **ASCII** 码显示 **D** 盘 **tencent** 目录结构及其所有文件

TYPE

显示文本文件的内容。

语法：

TYPE [drive:][path]filename

例子：

type D:\宇宙.txt

执行后就会在命令行窗口显示：

《文子 •自然》:“往古来今谓之宙，四方上下谓之宇。”宇宙（**Universe**）是由空间、时间、物质和能量，所构成的统一体。

DIR

显示目录列表

语法：

DIR [drive:][path][filename] [/A[[:]attributes]] [/B] [/C] [/D] [/L] [/N] [/O[[:]sortorder]] [/P] [/Q] [/S] [/T[[:]timefield]] [/W] [/X] [/4]

参数说明：

[drive:][path][filename]　　指定要列出的驱动器、目录和/或文件。

需要注意的是：**dir** 返回的只是个文件名，不可以返回路径，这一点在 **for** 应用中尤其明显。详见 **for** 命令

/A　　显示具有指定属性的文件。例如 **dir /ah** 只显示隐藏文件。 **dir/aa** 只显示存档文件。

attributes　　**D**　目录　　**R**　只读文件

H　隐藏文件　　**A**　准备存档的文件

S　系统文件　　**-**　表示“否”的前缀，例如**-r** 为非只读文件，**-h** 为非隐藏文件如此等等。

/B　　只显示文件名和扩展名。

/C　　在文件大小中显示千位数分隔符。这是默认值。用 **/-C** 来停用分隔符显示。

/D　　跟宽式相同，但文件是按栏分类列出的。

/L　　用小写。

/N　新的长列表格式，其中文件名在最右边。
/O　用分类顺序列出文件。
sortorder　**N**　按名称(字母顺序)　**S**　按大小(从小到大)
E　按扩展名(字母顺序)　**D**　按日期/时间(从先到后)
G　组目录优先　**-**　颠倒顺序的前缀
/P　在每个信息屏幕后暂停。
/Q　显示文件所有者。
/S　显示指定目录和所有子目录中的文件。
/T　控制显示或用来分类的时间字符域。
timefield　**C**　创建时间
A　上次访问时间
W　上次写入的时间
/W　用宽列表格式。
/X　显示为非 8**dot**3 文件名产生的短名称。格式是 **/N** 的格式，短名称插在长名称前面。如果没有短名称，在其位置则显示空白。
/4　用四位数字显示年

可以在 **DIRCMD** 环境变量中预先设定开关。通过添加前缀 **-**(破折号)来替代预先设定的开关。例如，**/-W**。

程序例子：
例 1：　**C:\>dir**　显示 **C** 盘根目录文件列表
例 2：**C:\>dir "program files"**　显示 **C:\program files** 里面的内容。注意：如果文件名包含空格，就必须用双引号，否则失败。文件名不包含空格，则双引号可有可无。
例 3：　**C:\>dir /a**　显示所有文件，包括系统文件和隐藏文件。**/a** 后面可以带参数，表示搜索具有指定属性的文件。这里缺省情况下，就表示搜索所有文件。在 **Windows** 下，用“显示所有文件”都无法看到的文件，用这个都能显示出来，灰常强大！切勿乱删用这个命令显示出来的文件！例如 **AUTOEXEC.BAT**、**CONFIG.SYS**、**MSDOS.SYS** 等等都是系统文件。
例 4：**C:\>dir /as**　显示 **C** 盘里的系统文件及隐藏的文件及目录，其它不显示。
例 5：　**C:\>dir /ah**　显示隐藏文件及文件夹。
例 6：　**C:\>dir /ad**　只显示 **C** 盘的目录而不显示文件。
例 7：　**C:\>dir /a-d**　只显示 **C** 盘的文件而不显示目录。以上都是**/a** 后面带参数，表明搜索具备或者不具备**(-d -r -h -a -s)**指定属性的文件。

例 8：　**C:\>dir /o**　　显示 C 盘目录和文件顺序。
例 9：　**C:\>dir /o-g**　　目录在下面，文件在上面。
例 10：**C:\>dir /on**　　按名称的字母顺序排列 C 盘的目录和文件。
例 11：**C:\>dir /o-n**　　按名称的字母逆序排列 C 盘的目录和文件。
例 12：**C:\>dir /oe**　　按扩展名的字母顺序排列 C 盘的目录和文件。
例 13：**C:\>dir /o-e**　　按扩展名的字母逆序排列 C 盘的目录和文件。
例 14：**C:\>dir /od**　　按日期和时间顺序排列 C 盘的目录和文件（早的排前）
例 15：**C:\>dir /o-d**　　按日期和时间逆序排列 C 盘的目录和文件（晚的排前）
例 16：**C:\>dir /os**　　按文件的大小排列（大的排前）
例 17：**C:\>dir /o-s**　　按文件的大小排列（小的排前）
例 18：**C:\>dir /p /a /og windows**　　分页显示 C 盘 Windows 目录和文件，也包括隐藏的目录和文件，并按照在文件之前分组显示。windows 没有包含空格，那么有无双引号无所谓。
例 19：**C:\>dir /p /w /a /og windows**　　分页并宽屏显示 C 盘 Windows 目录和文件，也包括隐藏的目录和文件，并按照在文件之前分组显示。
例 20：**C:\>dir /s /b regedit.exe**　　在 C 盘搜索 regedit.exe 路径。如果要求显示详细信息，可以这么写 **C:\>dir /s regedit.exe**
例 21：**D:\music>dir /s /b *.mp3>E:\mp3.txt&start E:\mp3.txt**　　假如我们在 D 盘 music 文件夹里收藏后很多 mp3 文件，那么在命令行里输入这命令，会将 music 文件夹里的所有 mp3 文件目录都输出到 E:\mp3.txt，并且打开 E:\mp3.txt。
例 22：**F:\>dir /s /b *.mp3>E:\mp3.txt**　　搜索 F 盘每个角落的 mp3 文件，生成列表后保存在 E 盘。
例 23：**C:\>dir read???.txt**　　搜索 C 盘根目录下所有以 read 开头，后面最多跟三个字符的 txt 文件。

CD 等于 CHDIR

显示当前目录名或改变当前目录。

语法：

CD [/D] [drive:][path]
CD [..]
CD [\]

1、**CD [/D] [drive:][path]**
改变当前目录为输入目录。
假如我们想要进入的目录的根目录与当前目录根目录相同，则不用参数**/d**。
例如：
C:\Documents and Settings>cd administrator
执行后，当前目录就变为：
C:\Documents and Settings\Administrator>
假如我们想要进入的目录的根目录与当前目录根目录不同，那么就必须加参数**/d**。
例如：
C:\Documents and Settings>cd /d D:\Tencent
执行后，当前目录就变为：
D:\Tencent>

2、

cd..	返回上一级目录
cd..\..	返回上两级目录
cd..\..\..	返回上三级目录
cd..\..\..\..	返回上四级目录

例如：
C:\Documents and Settings\Administrator\My Documents\My Music\iTunes\Album Artwork\Cache>cd..\..\..\..\..
变为：
C:\Documents and Settings\Administrator>

3、**cd**
进入根目录。

注意一个细节：
假如我们当前的目录为
C:\Documents and Settings\Administrator>
然后我们键入
cd /d D:\Tencent
当前目录就变为
D:\Tencent>
此时我们键入
cd /d C:
目录会变回为我们刚才跳转前的目录：
C:\Documents and Settings\Administrator>

而不是 **C:\>**这个根目录。这个对于其他盘也一样。

EXIT

退出 **CMD.EXE** 程序(命令翻译程序)或当前批处理脚本。

语法:

EXIT [/B] [exitCode]

/B 指定要退出当前批处理脚本而不是 **CMD.EXE**。如果从一个
批处理脚本外执行，则会退出 **CMD.EXE**

exitCode 指定一个数字号码。如果指定了 **/B**，将 **ERRORLEVEL**
设成那个数字。如果退出 **CMD.EXE**，则用那个数字设置
过程退出代码。

exit 命令： **exit /B** 返回值

例： **aa.bat**

@echo aa.bat 调用中

@exit /b 2

---

bb.bat

@echo 调用 **aa.bat**

@call aa.bat

@echo aa.bat 的返回值**:%errorlevel%**

@pause

MD 等于 MKDIR

创建目录(也就是创建文件夹)

语法：
MD [drive:]path

md xxx
在当前目录下新建文件夹《**xxx**》

如果当前目录为：
D:\Tencent\>
那么执行：
md abc
后，在 **Tencent** 目录下可以看到新建了一个《**abc**》文件夹。

md D:\xxx\yyy
在指定的路径“**D:\xxx**”新建一个文件夹《**yyy**》

如果在指定路径上新建的文件夹与原来某一个文件夹同名，那么新建不会进行。
例如：**D** 盘根目录已有一个名为《**file**》的文件夹
md D:\file
此时程序会提示“子目录或文件 **D:\file** 已经存在。”新建没有进行。

假如当前目录不存在《**a**》文件夹，那么我们执行
md \a\b\c\d
等同于连续执行

md a
cd a
md b
cd b
md c
cd c
md d
cd d
建立一个目录串。

建立的文件名有空格，则要用双引号引起来：
md D:\"Adobe Flash CS5"

高级篇：
用 **MD** 创建一个无法打开和删除的文件。
md D:\abc..
在 **D** 盘根目录创建一个名为《**abc.**》的文件夹，该文件夹无法被访问和删除。**(**当然，用一些软件暴力删除还是可以删除的**)**

删除方法：
rd D:\abc..
这就将该奇怪的文件夹《**abc.**》删除了。

REN 等于 RENAME

重命名文件。

语法：
REN [drive:][path]filename1 filename2.
请注意，您不能为目标文件指定新的驱动器或路径。

例如：
ren D:\Tencent tengxun
将 **D** 盘的 **Tencent** 重命名为 **tengxun**
又如：
ren D:\program files\efg.doc M.txt
将 **D:\program files** 下的 **efg.doc** 文件改为 **M.txt** 文件，文件类型虽然可以更改，但改类型后不一定能打开。

LABEL

创建、更改或删除磁盘的卷标。
卷标：在磁盘名**(C、D、E、F、G . . .)**的左侧，例如“软件**(D:)**”，**D:**是磁盘名，软件是卷标。

语法：
LABEL [drive:][label]
LABEL [/MP] [volume] [label]

drive:	指定驱动器名。
label	指定卷标签。
/MP	指定卷应该被当作安装点或卷名。
volume	指定驱动器**(**后面跟一个冒号**)**、装入点或卷名。如果指定了卷名，**/MP** 标志则不必要。

上面是直接从帮助信息复制过来的。

例如：
label D: 安装软件
执行上面命令后，我们可以看到磁盘的卷标变为“安装软件”。

例如：
label D:
执行后显示如下：

驱动器 **D:** 中的卷是 软件
卷序列号为 000**B**-6514
卷标**(**如果没有，请按 **ENTER)**?

我们按回车后显示：

是否删除当前的卷标**(Y/N)**?

选择 **Y** 可以删除卷标，选择 **N** 则退出。没有卷标的磁盘会显示诸如“本地磁盘”字样。

按照我实际操作的经验，可以直接在根目录里直接修改卷标：
C:\>label 系统
这样直接就将 **C** 盘的卷标改为“系统”。

D:\>label 安装
这样直接就将 **D** 盘的卷标改为“安装”。

E:\>label
这个会提示是否删除 **E** 盘卷标。

也就是说，没有指明磁盘的情况下，默然为当前磁盘。
C:\program files>label G: 下载
这个指明了磁盘，将 **G** 盘卷标改为“下载”。

对于 **LABEL [/MP] [volume] [label]**这个语法，不必理会！**/mp** 纯粹多余的参数！有和没有一个样。

ATTRIB

显示或更改文件属性。

语法：
ATTRIB [+R | -R] [+A | -A] [+S | -S] [+H | -H] [[drive:] [path] filename] [/S [/D]]

+　设置属性。
-　清除属性。
R　只读文件属性。
A　存档文件属性。
S　系统文件属性。
H　隐藏文件属性。
[drive:][path][filename]指定要处理的文件路径。
/S　处理当前文件夹及其子文件夹中的匹配文件。
/D　也处理文件夹。

注意：如果将文件属性修改为系统属性后，将无法对属性再进行修改，所以**-s** 没用！

缺省参数情况下，显示属性。
例如：
attrib
显示当前目录下的所有文件的文件属性（不包含文件夹）。

attrib D:\picture.bmp
显示 **D:\picture.bmp** 的文件属性

attrib C:\Documents and Settings\Administrator\My Documents
显示 **My Documents** 文件夹的属性。

程序例子：
md autorun
attrib +a +s +h autorun
上面的命令先建立 **autorun** 文件夹，然后将其设置为存档、系统、隐藏属性。
例如：
attrib +R -A D:\files\flash.txt
设置 **flash.txt** 为只读属性，取消其存档属性。

attrib +a E:*.*　　将 **E** 盘根目录的所有文件都设置为存档属性。
attrib -a F:*.bak　将 **F** 盘所有**.bak** 文件都取消存档属性。
xcopy a: b: /a　将 **A** 盘上的所有标志为"归档"属性的文件拷到 **B** 盘
xcopy a: b: /m　将 **A** 盘上的所有文件拷到 **B** 盘后并移去归档属性

删除命令：RD 和 DEL

用 **rd** 和 **del** 命令删除资料后，都不会到回收站，也就是彻底删除，所以删除前需谨慎。

DEL
ERASE
这两个命令的功能完全一样，都是删除一个或数个文件。
del 支持通配符。**del** 只能删除文件，不能删除文件夹。

语法：
DEL　　[/P] [/F] [/S] [/Q] [/A[[:]attributes]] names
ERASE　　[/P] [/F] [/S] [/Q] [/A[[:]attributes]] names

names	指定一个或数个文件或目录列表。通配符可被用来删除多个文件。如果指定了一个目录，目录中的所有文件都会被删除。
/P	删除每一个文件之前提示确认。
/F	强制删除只读文件。
/S	从所有子目录删除指定文件。
/Q	安静模式。删除全局通配符时，不要求确认
/A	根据属性选择要删除的文件。
attributes	**R**　只读文件　　**S**　系统文件 **H**　隐藏文件　　**A**　存档文件 **-**　表示“否”的前缀

如果命令扩展名被启用，**DEL** 和 **ERASE** 会如下改变**:**

/S 开关的显示句法会颠倒，即只显示已经
删除的文件，而不显示找不到的文件。

程序例子：
del d:\test.txt
执行后，程序不经询问就直接将 **D** 盘下的 **text.txt** 文件删除。

del 没有带参数的情况下，无法删除具有隐藏属性、只读属性或者系统属性的文件。
如果是只读文件，执行删除命令时会显示“拒绝访问”。
如果是隐藏文件，执行删除命令时会显示“找不到 **XXX**”。
如果是系统文件，执行删除命令时会显示“找不到 **XXX**”。

注意，无论 **del** 带不带参数，所有被删除的文件都不会到回收站里，都是直接彻底删除。

删除具有隐藏属性的文件 **del /a:h D:\test.txt**
删除具有只读属性的文件 **del /a:r D:\test.txt**
删除具有系统属性的文件 **del /a:s D:\test.txt**

当我们不知道文件是什么属性时，可以直接用 **del /a D:\test.txt**
不过这个无法删除具有只读属性的文件。
所有文件都可以删除的指令：**del /a /f D:\test.txt**

del D:\test*.txt
将 **D:\test** 文件夹下的所有名字任意，扩展名为 **txt** 文件全部不加确认就删除。

del D:\test\abc.*
将 **D:\test** 文件夹下的所有名字为 **abc**，扩展名任意的文件不加确认就删除。

del D:\test
del D:\test*
将 **D:\test** 文件夹下的所有文件都删除，删除前会有删除确认对话。对于这种情况，不想要对话就需要加参数/**q** 了。

参数/**p**
删除之前增加确认对话。
程序例子：
del /a /f /p D:\test.txt

参数/**q**
安静模式，不显示任何对话就直接删除。
程序例子：
del /a /f /q D:\test.txt

参数/**s**
从所有子目录删除指定文件。
程序例子：
假如我们现在的目录是 **D** 盘 **program files** 目录
D:\program files>
我们输入
del /a /f /p /s test.txt
那么，程序就会搜索 **program files** 目录下的所有子目录，将所有的 **test.txt** 文件全部找出并删除，由于有/**p** 参数，在删除每个文件时，都会出现删除确认对话框。

del /f /q *.*
会将当前目录下的所有文件都删除，包括该批处理文件。

RD
RMDIR

这两个命令完全一样，作用都是删除一个目录。
注意：**rd** 不支持通配符！所以类似 **rd *.***之类的命令无效。**rd** 可以删除文件和文件夹。

语法：
RD　　[/S] [/Q] [drive:]path
RMDIR　[/S] [/Q] [drive:]path

/S 除目录本身外，还将删除指定目录下的所有子目录和文件。用于删除目录树。
/Q 安静模式，带 **/S** 删除目录树时不要求确认

rd 在没有带**/q** 参数的情况下，假如被删除的文件夹不为空，那么都会有删除确认对话。

程序例子：
rd D:\test
假如 **D:\test** 文件夹为空，那么删除直接进行，没有删除确认对话。
假如 **D:\test** 文件夹不空，那么删除无法进行，删除被拒绝。

当 **D:\test** 文件夹不为空时，用下面命令删除：
rd /s D:\test　　删除前会有删除确认对话
rd /s /q D:\test　　删除直接进行，没有对话

如果想要删除所有文件和文件夹，因为 **rd** 不支持通配符，所以不能用类似 **rd *.***之类的命令，应该用 **rd** 删除指定目录以及目录下的所有子目录及文件就行了。
如果当前删除程序批处理在删除的目录下，那么批处理本身也会被删除，不过当前目录不会被删除。

At

列出在指定的时间和日期在计算机上运行的已计划命令或计划命令和程序。必须正在运行“计划”服务才能使用 **at** 命令。
at [\\computername] [[id] [/delete] | /delete [/yes]]
at [\\computername] time [/interactive] [/every:date[,...]|
/next:date[,...]] "command"

参数

无
如果在没有参数的情况下使用，则 **at** 列出已计划的命令。
这个，我们先在开始命令输入
at 23:30 **shutdown /s**
执行后，我们在 **cmd** 命令行窗口输入
at
就可以看到这个关机命令。
但是，我们在控制面板的任务计划里添加软件运行计划，在 **cmd** 窗口输入 **at** 是无法看到的。

\\computername
指定远程计算机。如果省略该参数，命令将在本地计算机执行。

id
指定指派给已计划命令的识别码。

/delete
取消已计划的命令。如果省略了 **id**，计算机中已计划的命令将被全部取消。

/yes
当删除已计划的事件时，对系统的查询强制进行肯定的回答。

time
指定运行命令的时间。将时间以 24 小时标记（00:00 [午夜] 到 23:59）的方式表示为小时：分钟。

/interactive
允许作业与在作业运行时登录用户的桌面进行交互。

/every:date[,...]
在每个星期或月的指定日期（例如，每个星期四，或每月的第三天）运行命令。将 **date** 指定为星期的一天或多天 **(M,T,W,Th,F,S,Su)**，或月的一天或多天（使用 1 到 31 的数字）。用逗号分隔多个日期项。如果省略了 **date**，将假定为该月的当前日期。

/next:date[,...]

在重复出现下一天（例如，下个星期四）时，运行指定命令将 **date** 指定为星期的一天或多天 **(M,T,W,Th,F,S,Su)**，或月的一天或多天（使用 1 到 31 的数字）。用逗号分隔多个日期项。如果省略了 **date**，将假定为该月的当前日期。

command
指定要运行的 **Windows** 2000 命令、程序（**.exe** 或 **.com** 文件）或批处理程序（**.bat** 或 **.cmd** 文件）。当命令需要路径作为参数时，请使用绝对路径，也就是从驱动器号开始的整个路径。如果命令在远程计算机上，请指定服务器和共享名的 **UNC** 符号，而不是远程驱动器号。如果命令不是可执行 **(.exe)** 文件，必须在命令前加上 **cmd /c**，例如：
cmd /c dir > c:\test.out

MOVE

移动文件并重命名文件和目录。相当于剪切。

语法：
要移动至少一个文件：
MOVE [/Y | /-Y] [drive:][path]filename1[,...] destination
要重命名一个目录：
MOVE [/Y | /-Y] [drive:][path]dirname1 dirname2

[drive:][path]filename1　指定您想移动的文件位置和名称。
destination　指定文件的新位置。目标可包含一个驱动器号和冒号、一个目录名或组合。如果只移动一个文件并在移动时将其重命名，您还可以包括文件名。
[drive:][path]dirname1　指定要重命名的目录。
dirname2　指定目录的新名称。

/Y　取消确认改写一个现有目标文件的提示。
/-Y　对确认改写一个现有目标文件发出提示。

程序例子：
move E:\test.txt D:\file
将 **E** 盘根目录里的《**test.txt**》文件移到 **D:\file** 去。
注意：**move** 不会建立新文件夹。也就是说，上面假如 **D** 盘根目录没有 **file** 文件夹，那么程序不会自动新建一个 **file** 文件夹，移动无法进

行。所以，如果不确定文件夹是否存在，可以这么写代码：

md D:\file
move E:\test.txt D:\file
假如 **D** 盘根目录原本就有 **file** 文件夹，那么 **md** 不会再新建 **file** 文件夹或者覆盖掉原来的 **file** 文件夹。

move E:\test.txt D:\file\temp.txt
移动 **E** 盘根目录下的 **test.txt** 文件到 **D:\file** 文件夹里，并重命名为《**temp.txt**》（当然，文件里面的内容不变）。

我们在移动过程中，最好加上参数/**-Y**，否则如果移动的目的目录如果有与移动文件的文件名相同的文件，那么程序会自动覆盖掉原来的文件。
假如我们的 **E** 盘和 **D** 盘根目录原来都就有一个名为《**test.txt**》的文件
move /-y D:\test.tmp E:
计算机会提示
改写 **E:\temp.txt** 吗? **(Yes/No/All):**
我们可以根据需要选择 **Y**、**N** 或 **A**

当然，如果我们确定，当目标文件存在时，文件直接覆盖掉目标文件，那么程序可以这么写：
move /y D:\test.tmp E:
计算机会不加提示的覆盖掉目标文件。

移动文件夹：
假如我们的 **D** 盘根目录有一个名为《**music**》的文件夹，里面有若干首歌曲
move /-y D:\music E:
这样就直接将 **music** 整个文件夹移动到了 **E** 盘根目录。
这里注意一个细节：假如 **D:\music** 文件夹里包含有子文件夹，那么子文件夹也会被移动到目标路径去。

如果我们只想将 **D:\music** 文件夹里的所有**.mp**3 格式歌曲移到 **E:\mp**3 文件夹里
move /-y D:\music*.mp3 **E:\mp**3

如果我们想将 **D:\music** 文件夹里的所有文件移动到 **E:\media** 文件里
move /-y D:\music*.* E:\midia

这里注意一个细节：假如 **D:\music** 文件夹里还包含有子文件夹，那么子文件夹不会被移动！

COPY

将一份或多份文件复制或合并到另一个位置。
copy 是内部命令，**xcopy** 是外部命令。
copy 不可以复制文件夹，复制文件夹应该用 **xcopy** 命令。
copy 不可以复制具有隐藏、系统属性的文件，要复制这些文件，要先用 **attrib** 去除文件属性或者改用 **xcopy** 命令。

语法：
COPY [/D] [/V] [/N] [/Y | /-Y] [/Z] [/A | /B] source [/A | /B] [+ source [/A | /B] [+ ...]] [destination [/A | /B]]

source　指定要复制的文件，由驱动器号和冒号、文件夹名、文件名组成（也就是路径）。
destination　为新文件指定目录和/或文件名，由驱动器号和冒号、文件夹名、文件名组成（也就是路径）。
/A　表示一个 **ASCII** 文本文件。
/B　表示一个二进位文件。
/D　允许将复制的加密文件在目标处作为解密文件保存。
/V　拷贝后验证新文件写入是否正确。
/N　复制带有非 8.3 格式名称的文件(文件名超过 8 个字符，扩展名超过 3 个字符)时，尽可能使用短文件名。
/Y　不显示确认是否要改写现有目标文件的提示。
/-Y　显示确认是否要改写现有目标文件的提示，与**/Y** 是互反参数。
/Z　在重启模式中复制网络文件。

上面是帮助信息，解释如下：
[/D] [/V] [/N] [/Y | /-Y] [/Z] [/A | /B] 前导参数
source [/A | /B]被复制的源文件及其参数。
[+ source [/A | /B] [+ ...]]每个源文件及其参数，当有多个源文件需要合并时，就用“+”来连接。
[destination [/A | /B]]目标文件及其参数。
前导参数的 **[/A | /B]**和 **source**、**destination** 这两个的参数**[/A | /B]**是

一样的，作用都是指明当前文件是二进制文件还是 **ASCII** 文本文件。未指明参数的情况下，系统默认是 **ASCII** 文本文件，也就等于参数**/a**。**/a** 和 **/b** 两个标记符，其作用范围都是直到遇到对方为止。

一个细节：
我们合并几个 **txt** 格式文件为一个 **txt** 文件，在合并的文件末尾总会出现一个结束字符。原因是复制完成的 **ASCII** 文件，程序都会自动添加结束字符。在没有指明文件参数的情况下，系统默认文件的参数为**/a**，也就是 **ASCII** 文本文件。解决的办法：将合并的文件的参数改为**/b**，二进制形式就没有这个结束字符了。代码如下：

```
copy  1.txt+2.txt 3.txt /b
```

因为参数**/a** 和**/b** 的作用范围都是知道遇到对方为止，假如一直没有遇到对方，那么作用范围就一直持续到末尾。于是下面的代码的效果和前面代码的效果是一样的：

```
copy /b 1.txt+2.txt 3.txt
copy 1.txt /b +2.txt 3.txt
copy 1.txt +2.txt /b 3.txt
```

上面这四种形式都是可以的，不过，考虑程序的健壮性，建议采用

```
copy /b 1.txt+2.txt 3.txt
```

也就是所有的文件都统一一种格式。对**/a** 参数也一样，统一在前导参数设置，作用范围为所有文件。

再看一个程序例子：

```
copy 1.txt+2.txt /b +3.txt+4.txt /a+5.txt /b +6.txt /a +7.txt+8.txt+9.txt
/b  0.txt
```

分析：
1.txt 没有参数，它的前面也没有参数，那么系统默认为 **ASCII** 文本文件，等于 **1.txt** 的参数为**/a**。
2.txt 文件有**/b** 参数，那么 **2.txt** 就是二进制文件。接下来**/b** 作用到 **3.txt** 文件，所以 **3.txt** 文件也是**/b** 参数。
4.txt 文件有**/a** 参数，前面 **2.txt** 文件的**/b** 参数的作用范围到此为止。所以 **4.txt** 文件的参数是**/a**。
5.txt 文件有**/b** 参数，前面 **4.txt** 文件的**/a** 参数的作用范围到此为止。所以 **5.txt** 文件的参数是**/b**。
6.txt 文件有**/a** 参数，前面 **5.txt** 文件的**/b** 参数的作用范围到此为止。所以 **6.txt** 文件的参数时**/a**。从 **6.txt** 参数顺下来，**7.txt** 和 **8.txt** 文件的参数也是**/a**。
9.txt 的参数为**/b**，所以前面 **6.txt** 文件的**/b** 参数的作用范围到此为止。所以 **9.txt** 文件的参数是**/b**。接下来，**0.txt** 文件的参数继承了前面的

参数，也是**/b**。

/V
如果写入操作不能校验，则会出现错误消息。尽管使用 **copy** 很少发生记录错误，但是仍可以使用 **/v** 来确保重要的数据已正确记录。因为要检查驱动器上记录的每一个扇区，所以该 **/v** 命令行选项也会使 **copy** 命令的运行速度减慢。

/Y
当复制的目标文件存在时，程序会暂停并提示是否改写目标文件。如果我们想要程序自动执行，不要出现对话，就要用到**/y** 参数了。
例如 **D** 盘有两个文件《**test.txt**》和《**abc.txt**》，我们要将《**test.txt**》不加确认就全部复制到《**abc.txt**》里去，覆盖掉《**abc.txt**》原来的内容，程序如下：
copy /y D:test.txt D:\abc.txt

/-Y
显示确认是否要改写现有目标文件的提示，与**/Y** 是互反参数。

/Z
如果在复制阶段连接丢失（例如，如果服务器脱机切断了连接），**copy /z** 将在重新建立连接后继续复制。**/z** 同时会显示每个文件已完成的复制操作的百分比。

程序例子：
copy D:\test.txt D:\abc.txt
这里分为两种情况：
1、当前 **D** 盘根目录没有 **abc.txt** 文件
执行命令后，会不加提示地在 **D** 盘根目录新建一个名为《**abc.txt**》的文件，并将 **D:\test.txt** 文本内容复制到 **abc.txt** 里去。
2、当前 **D** 盘根目录有 **abc.txt** 文件
执行命令后，程序会提示是否是否改写《**abc.txt**》，选择 **n** 则复制没有进行，程序运行结束。选择 **Y**，则将《**abc.txt**》里面原来的所有内容都删除，然后将 **D:\test.txt** 文本内容复制到 **abc.txt** 里去。

如果复制的路径有空格，就需要用双引号。
copy D:\"program files"\test.txt F:\"abc def"\123.txt

如果不指定目标文件，复制的文件和源文件将会有相同的名称、创建日期和时间、修改日期和时间。其中，修改日期和时间与源文件相同，但创建时间与源文件会有上百微秒的出入。如果源文件位于当前驱动器的当前目录中，而您没有为目标文件指定其他驱动器或目录，则 **copy** 命令将终止并显示下列错误消息：
File cannot be copied onto itself
0 **File(s) copied**
中文意思是：
文件无法自身复制。
已复制　　0 个文件。

例如：
copy D:\test.txt E:
这里，只写出了复制的目标路径，那么程序会自动在目标路径 **E:**新建一个与源文件名字时间和日期相同的文件《**test.txt**》，并将其所有内容复制到目标文件上。
假如我们当前驱动器为 **F:**
F:\>copy D:\test.txt
那么程序会在 **F** 盘根目录新建一个《**test.txt**》文件，并将 **D:\test.txt** 复制到里面去。

下面两个程序：
copy D:\test.txt D:\test.txt
copy D:\test.txt D:
假如当前路径为 **D** 盘根目录，源文件《**test.txt**》位于当前根目录下：
D:\>copy D:\test.txt
D:\>copy test.txt
这四个程序均执行错误，复制无法进行，不可以自身复制。

用 **copy** 合并文件
语法：
copy 文件 1+文件 2+文件 3+…文件 **n** 新文件
使用通配符（*或?）例如：
copy *.* 新文件
copy *.doc 新文件

合并文件时，**copy** 会使用当前日期和时间来标记目标文件。如果省略 **Destination**，则文件被合并并且会使用首个指定的文件的名称存储。

程序例子：
copy D:\test.txt+D:\abc.txt D:\123.txt
这个将两个文件合并为一个文件。注意，即使是在同一个文件夹下的文件要合并，也必须将每个文件的完整路径写出。
copy　D:\test.txt+E:\"program files"\help.txt+F:\soft\tmp.txt D:
这里合并的目标文件名字没有写出来，那么程序就会将所有文件的内容不加询问就合并在 **D:\test.txt** 文件里，覆盖掉 **D:\test.txt** 原来的内容。
如果程序是这样：
copy　D:\test.txt+E:\"program files"\help.txt+F:\soft\tmp.txt G:
那么会将所有内容合并到 **G** 盘新建的《**test.txt**》文件里。
从上面两个例子可以看出，如果没有指定目标文件名，那么程序会将第一个源文件的名字作为目标文件名。

这里需要注意的是，合并的文件类型最好相同，不同类型的文件合并可能会出错。

copy *.txt combin.doc
在当前驱动器的当前目录中，将所有具有 **.txt** 扩展名的文件合并进名为 **Combin.doc** 的文件中。
使用通配符存在一个排序问题。合并的顺序是按名称排序。按名称排序时要注意，计算机智能程度不高，假如我们有 100 个文件，文件名是从 **1.txt** 到 **100.txt**，合并时计算机不会按照数字大小来排序，12 会排在 3 前面，因为数字 1 比 3 小。56 会排在 9 前面，因为数字 5 比 9 小，如此等等。
使用“+”就不存在排序问题，计算机会按照“+”所列出的顺序排列文件。
例如：《**12.txt**》的内容为“共和国”，　《**3.txt**》的内容为“人民”，《**56.txt**》的内容为“中华”
copy /b 56.txt+3.txt+12.txt 0.txt
合并后，**0.txt** 的内容为：中华人民共和国

批量复制：
md D:\music
copy *.mp3 D:\music
::实现 **D** 盘根目录新建一个《**music**》文件夹，并把当前文件夹下的所有 **mp3** 文件都复制一份到《**music**》文件夹下。

一一对应合并文件
copy *.txt+*.ref　*.doc
copy 命令将每个带有 **.txt** 扩展名的文件和它对应的 **.ref** 文件合并起来。结果是文件名相同但扩展名为 **.doc** 的文件。**copy** 将 **File1.txt** 与 **File1.ref** 合并形成 **File1.doc,**然后 **copy** 将 **File2.txt** 与 **File2.ref** 合并形成 **File2.doc**，如此等等。

合并同类文件
copy *.txt+*.ref　combin.doc
要合并所有具有 **.txt** 扩展名的文件，然后合并所有具有 **.ref** 扩展名的文件，并将合并结果置于 **Combin.doc** 文件中。

注意：
如果要使用通配符将几个二进制文件合并进一个文件，请添加 **/b** 命令行选项。这可防止 **Windows XP** 将 **CTRL+Z** 当作文件结束符处理。
copy /b *.exe combin.exe
警告：
如果合并二进制文件，结果文件可能因为内部格式而无法使用。

---

copy 高级技巧----用 **copy** 命令隐藏文件

copy 命令能将一个文本文件合并到一个非文本文件中，实现隐藏秘密的效果。
比如你的 **D** 盘根目录有一个重要的文本文件：001**.txt**，想对其进行隐藏。找一个 **mp**3 歌曲，体积最好不要太大**(**500**KB** 左右为宜，太大会引起打开不那么方便**)**，比如：**wanfei.mp**3（假设也放在 **D** 盘根目录），输入：
copy wanfei.mp3 **/b**+001**.txt /a wanfei**-1**.mp**3
或
copy wanfei.mp3 **/b**+001**.txt wanfei**-1**.mp**3
这样会生成一个新文件 **wanfei**-1**.mp**3。播放它，歌曲很正常。右击 **wanfei**-1**.mp**3，选择“打开方式”并选择 **EmEditor** 之类的文本编辑器或者 **Word**、记事本打开该文件，看到什么了？一堆乱码？但如果按下 **Ctrl+End** 组合键将光标移到文件的尾部，是不是 001**.txt** 文件的内容？

小提示：其中参数**/b** 指定以二进制格式复制、合并文件；参数**/a** 指定以 **ASCII** 格式复制、合并文件。这里要注意文件的顺序，二进制格式的文件应该放在加号前，文本格式的文件放在加号后。有一点要提醒大家：就是这个文本文件的前面最好至少空 3 行，这样它的头部的内容就不易丢失。我实验了一下，这个**/a** 参数可有可无。

同样道理，你可以把文件隐藏进图片、**WAV** 声音文件，甚至还能在 **Word**、**Excel** 等文档中藏进秘密，比如：

copy wanfei.jpg/b+001.txt/a wanfei-1.jpg（把 001**.txt** 藏进 **wanfei-1.jpg** 图片）
copy photo.jpg /b +secret.txt photo-1.jpg
可以用 **Word** 或记事本打开图片，在一堆乱码后面看到隐藏的 **txt** 内容。
copy wanfei.doc/b+001.txt/a wanfei-1.doc（把 001**.txt** 藏进 **wanfei-1.doc** 文档）
copy file.doc /b +secret.txt file-1.doc
将 **txt** 文件隐藏进 **doc** 文件后，用 **Word** 打开 **doc** 文件很正常。但用记事本打开 **doc** 文件，会看到开头的一堆乱码和后面原来的 **txt** 内容！
copy wanfei.wav/b+001.txt/a wanfei-1.wav（把 001**.txt** 藏进 **wanfei-1.wav** 文件）
copy music.wav /b +secret.txt music-1.wav
copy wanfei.exe/b+001.txt/a wanfei-1.exe（把 001**.txt** 藏进 **wanfei-1.exe** 文件）
copy execute.exe /b +secret.txt execute-1.exe

还可以这么隐藏文件：
假如我们有一张图片 **photo.jpg**，和一份秘密资料 **secret.txt**，要将秘密资料隐藏在图片里。
1、用 **winrar** 将《**secret.txt**》压缩，文件为 **secret.rar**
2、在当前目录下新建一个文本文件，里面的代码：
copy /b photo.jpg + secret.rar photo-1.jpg
copy photo.jpg /b + secret.rar photo-1.jpg
copy photo.jpg /b + secret.rar=photo-1.jpg
上面三种写法都是可以的，建议采用第一种写法。感觉有无等号“=”效果都一样。
3、将文本文件的后缀**.txt** 改为**.bat**，双击运行它后就会生成一个 **photo-1.jpg** 图片。
3、打开 **photo-1.jpg** 是一张正常的图片，但是右击 **photo-1.jpg** 选择打开方式，选择 **rar** 解压缩，就可以得到 **secret.txt** 文本。
这种方法要注意：**photo.jpg** 素材最好不要太大，一般 **KB** 级大小就行了。**MB** 级别大小的图片，合成的图片 **photo-1.jpg** 可能会损坏而无法还原 **secret.txt** 文件。
这个不单可以隐藏 **txt**，**office** 文件也都可以，例如 **doc**、**xls**、**ppt** 等等，也可以将一首歌曲隐藏进一张图片等等，非常强大。

XCOPY

复制文件和目录树。
copy 是内部命令，**xcopy** 是外部命令。

XCOPY source [destination] [/A | /M] [/D[:date]] [/P] [/S [/E]] [/V] [/W] [/C] [/I] [/Q] [/F] [/L] [/G] [/H] [/R] [/T] [/U] [/K] [/N] [/O] [/X] [/Y] [/-Y] [/Z] [/EXCLUDE:file1[+file2][+file3]...]

source　指定要复制的文件。
destination　指定新文件的位置和/或名称。
/A　只复制有存档属性集的文件，但不改变属性。
/M　只复制有存档属性集的文件，并关闭存档属性。
/D:m-d-y　复制在指定日期或指定日期以后更改的文件。如果没有提供日期，只复制那些源时间比目标时间新的文件。
/EXCLUDE:file1[+file2][+file3]...　指定含有字符串的文件列表。每一个字符串必须在文件的单独行中。如果有任何字符串与要被复制的文件的绝对路径相符，那个文件将不会得到复制。例如，指定如 **\obj** 或 **.obj** 的字符串会排除目录 **obj** 下面的所有文件或带有**.obj** 扩展名的文件。
/P　创建每个目标文件前提示。
/S　复制目录和子目录，除了空的。
/E　复制目录和子目录，包括空的。与 **/S /E** 相同。可以用来修改 **/T**。
/V　验证每个新文件。
/W　提示您在复制前按键。
/C　即使有错误，也继续复制。
/I　如果目标不存在，又在复制一个以上的文件，则假定目标一定是一个目录。
/Q　复制时不显示文件名。
/F　复制时显示完整的源和目标文件名。
/L　显示要复制的文件。
/G　允许将没有经过加密的文件复制到不支持加密的目标。
/H　也复制隐藏和系统文件。
/R　覆盖只读文件。
/T　创建目录结构，但不复制文件。不包括空目录或子目录。**/T /E** 包括空目录和子目录。
/U　只复制已经存在于目标中的文件。
/K　复制属性。一般的 **Xcopy** 会重置只读属性。

/N	用生成的短名复制。
/O	复制文件所有权和 **ACL** 信息。
/X	复制文件审核设置**(**隐含 **/O)**。
/Y	复制文件审核设置**(**隐含 **/O)**。现存目标文件。
/-Y	导致提示以确认改写一个 现存目标文件。
/Z	用重新启动模式复制网络文件。

命令行开关 **/Y** 可以预先在 **COPYCMD** 环境变量中设置。这可能被命令行上的 **/-Y** 改写。

copy C:*.* D:
xcopy C:*.* D:
上面两个命令的区别是：
copy C:*.* D:　　会将 **C** 盘除了文件之外的所有文件全部复制到 **D** 盘。
xcopy C:*.* D: 会将 **C** 盘的所有文件、文件夹以及子文件夹和子文件夹中的所有文件全部复制到 **D** 盘，等于将整个 **C** 盘全部复制到 **D** 盘。

xcopy F:\ G:\abc /e /s /h
执行这个命令后，计算机会提示问题：**abc** 是文件还是目录？假如我们一开始知道是目录（文件夹），那么代码可以这么写：
xcopy F:\ G:\abc\ /e /s /h
这样计算机就知道 **abc** 是个文件夹，不会有提示了。

程序例子：
xcopy F:\ G:\ /e
将驱动器 **F** 的所有文件和子目录（包括所有空的子目录）复制到驱动器 **G**。

xcopy F:\ G:\ /e /s /h
将驱动器 **F** 的所有文件和子目录（包括所有空的子目录）以及系统文件和隐藏文件全都复制到驱动器 **G**。

xcopy D:\rawdata E:\reports /d:12-29-1993
复制 **Rawdata** 目录中 1993 年 12 月 29 日以及以后更改的文件更新到 **Reports** 目录中的文件。
(这个无法指定日期复制？？ **)**
xcopy D:\rawdata E:\reports /u
从 **rawdata** 中复制在 **reports** 存在的文件到 **reports**，也就是更新所有

reports 中的文件。

xcopy E:*.* D:\ /s /h
复制 **E:/** 盘上所有文件到 **D:/** 盘上。
如果在复制过程中，按键盘热键“**Ctrl " +"Pause /break"**热键可以中断复制。

xcopy E:*.* D:\abc /d
有**/d** 参数的存在，复制会将 **E** 盘中在 **D:\abc** 没有的文件复制到 **D:\abc** 中去。这个可以用于断点复制，就是今天复制一点，明天接着再复制一点。

我们会碰到这种情况，一些文件我们已经修改，网上面又有最新版本的啦。原来备份的那份文件我们要及时更新啦。那怎么才能同步更新呢？当然我们可以把最新的好份文件复制过去。 如果修改的文件有很多的，自己到时候也记不清楚那份文件是更新过的。这么多的原来备份的那份文件都要更新，都要复制过去，可不是件省力气的活，而且容易出错和一些文件被忘记啦！ 输入**" xcopy E:*.* d:\abc /s /h /d /y "**就行啦！

还有一种情况我们经常也碰到的，因为复制某个文件出错或者这个文件在使用中，而停止了复制工作，（比如，复制 **C** 盘上的 **windows xp** ）我们这时候想跳过某个出错的文件和某个正在使用中文件而继续复制其他文件。 再在多加入一个参数**" /c "**就是 **" xcopy E:*.* d:\ /s /h /d /c /y "**就行啦！

有时候我们想 **XCOPY** 变为自动复制，并且复制完成后关闭电脑，代码如下：
xcopy e:*.* d:\ /s /h /c /y
shutdown -s
运行这个批处理，你这时候可以出去玩啦，电脑会自己复制备份完成文件然后自动关闭电脑。

xcopy "//192.XXX.1.20/GameAdmin/VSS_Local/DebugEnv" "E:/DevEnv" /i /s /c /y
这个是从目标机器上复制东西。

xcopy a: b:\ /d:08/18/98 **/s /b**
复制 98.08.18 年后的文件。

copy 和 **xcopy** 比较
1、**copy** 是内部命令，**xcopy** 是外部命令。
2、**copy** 只能复制文件，不能复制文件夹。**xcopy** 可以复制文件和文件夹。
3、**copy** 可以合并文件，**xcopy** 不能合并文件。
4、**copy** 不支持断点复制，**xcopy** 支持断点复制。
5、复制隐藏文件和系统文件时，**copy** 需要先用 **attrib** 去掉文件的属性后才能复制，而 **xcopy** 只需要添加几个参数就行了。
6、高级技巧里，**copy** 可以用来隐藏文件，**xcopy** 不可以隐藏文件。
点评：**copy** 和 **xcopy** 各有千秋，看用哪个更方便就用哪个。

FIND

在文件中搜索字符串。

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

参数说明：

/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。
程序例子：

```
@echo off
mode con cols=100
find /v /i "abc" D:\*.txt
pause
```

忽略大小写，将 **D** 盘根目录所有 **txt** 文件中所有不含 **abc** 的所有行都显示出来。

假如我们要查找指定目录及其子目录的所有文件，那么就要配合 **for** 语句来实现了。
例如我们要实现以下功能：
当前目录（***.bat** 文件位置）下有多个文件夹，每个文件夹中都包含两个后缀为**.key** 的文件，其中一个**.key** 文件的名字中总是包含 **Ped** 字符串（我需要打开的）。现在我需要用指定的应用程序（**F:\LSDYNA\manager.exe**）来依次打开名字中包含 **Ped** 字符串的文件。

```
@echo off
for /f "delims=" %%f in ('dir /b /s /a-d *.key ^| find /i "Ped"') do
(F:\LSDYNA\manager.exe "%%f")
```

这个比较复杂，看不懂可以略过。

find 常和 **type** 一起使用的例子：
假如我们在 **D** 盘中有一个 **C.txt** 文件，我们在另外一个地方新建一个 **tmp.bat**，里面代码为：

```
@echo off
mode con cols=100
type D:\c.txt | find /n "return"
pause
```

将找出所有包含 **return** 字符串的行，并将所有行都分别显示出来，包括显示行号。
这样的用法比较复杂。

FINDSTR

在文件中寻找字符串。

语法：

```
FINDSTR [/B] [/E] [/L] [/R] [/S] [/I] [/X] [/V] [/N] [/M] [/O] [/F:file]
[/C:string] [/G:file] [/D:dir list] [/A:color attributes] [/OFF[LINE]]
strings [[drive:][path]filename[ ...]]
```

/B	在一行的开始配对模式。
/E	在一行的结尾配对模式。
/L	按字使用搜索字符串。
/R	将搜索字符串作为一般表达式使用。

/S　在当前目录和所有子目录中搜索匹配文件。有了这个参数，就不可以指定搜索路径了，否则出错。
/I　指定搜索不分大小写。
/X　打印完全匹配的行。
/V　只打印不包含匹配的行。
/N　在匹配的每行前打印行数。
/M　如果文件含有匹配项，只打印其文件名。
/O　在每个匹配行前打印字符偏移量。
/P　忽略有不可打印字符的文件。
/OFF[LINE]　不跳过带有脱机属性集的文件。
/A:attr　指定有十六进位数字的颜色属性。请见 **"color /?"**
/F:file　从指定文件读文件列表 **(/** 代表控制台**)**。
/C:string　使用指定字符串作为文字搜索字符串。
/G:file　从指定的文件获得搜索字符串。 **(/** 代表控制台**)**。
/D:dir　查找以分号为分隔符的目录列表
strings　要查找的文字。
[drive:][path]filename　指定要查找的文件。

单个字符或字符串搜索，可以带双引号，也可以不带双引号。多个字符串搜索，字符串要用双引号引起来，每个字符串之间用空格隔开。例如：
find windows test.txt
find "windows" test.txt
find "computer windows" test.txt
下面的不合法：
find computer windows test.txt
find "computer" "windows" test.txt

不带参数**/C:**的情况下，没有空格隔开的字符串，每个字符串都会被独立搜索。带参数**/C:**，则所有字符会被当做一整个字符串来搜索。例如**:**
FINDSTR "hello there" x.y　在文件 **x.y** 中寻找 **"hello"** 或 **"there"** 。
FINDSTR /C:"hello there" x.y　在文件 **x.y** 中寻找 **"hello there"**。

一般表达式的快速参考**:**
.　通配符**:** 任何字符
*　重复**:** 以前字符或类别出现零或零以上次数

　^　　行位置：行首
　$　　行位置：行尾
　[class]　包含的字符：任何在字符集中的字符　　例如：[aeiou]包含 aeiou 五个字母中的任意一个字符。
　[^class]　不包含字符：任何不在字符集中的字符　例如：[12345]不包含 12345 五个字符中任意一个字符。
　[x-y]　　范围：在指定范围内的任何字符。　　例如：[0-9]包含 0 到 9 中任意一个字符。[^a-z]不包含所有小写字母。
　\x　　Escape：元字符 x 的文字用法
　\<xyz　　字位置：字的开始
　xyz\>　　字位置：字的结束

上面是帮助里的信息。

findstr 也是个很复杂很让人头疼的命令，例如用 **findstr /a** 参数实现窗口彩色效果，没多少人可以掌握。对 **findstr** 各种潜在技巧的挖掘更是困难异常！**findstr** 的功能非常强大，下面只讲解其常规用法：

参数说明：
/b
在一行的开始配对模式。
/e
在一行的结尾配对模式。

说明：这两个符号合起来将。

请看例子：
```
@echo off
echo a1984>test.txt
findstr /b "[0-9]" test.txt
del test.txt
pause>nul
```
从一行的开始匹配，a1984，一开始不是数字，所以匹配不上，没有任何显示。如果是 1984a 则可以匹配上。

```
@echo off
echo 1984d>test.txt
findstr /e "[0-9]" test.txt
del test.txt
pause>nul
```

从一开结束开始匹配，1984**d**，末尾不是数字，所以匹配不上，没有任何显示。如果是 **d**1984 则可以匹配上。

```
@echo off
echo 19xyz84>test.txt
findstr /e "[0-9]" test.txt
del test.txt
pause>nul
```

由于 19**xyz**84 非常狡猾，无论我们单独**/b** 还是**/e**，都匹配得上。但是，**/b** 和**/e** 同时用，这个数字就匹配不上了，非得纯数字才可以匹配显示出来，代码改为：

```
findstr /be "[0-9]" test.txt
findstr /b /e "[0-9]" test.txt
```

这两个是等价的，都只搜索纯数字。

在某些情况下，单独使用**/b** 或**/e** 会失效！

例如：

```
@echo off
set /p a=请输入：
echo %a%|findstr /be "[0-9.]*" >nul &&echo 是数字||echo 不是数字
& %0
pause > nul
```

这里如果参数只是**/b** 或**/e**，那么全部都可以正确执行，无论我们输入的是纯字母还是纯特殊符号，都显示是数字。但**/b /e** 一起用，就可以正确执行了。

/i

默认情况下，**findstr** 搜索字符串会严格区分大小写。

```
findstr Windows proposal.txt
```

在当前目录下的 **Proposal.txt** 文件中搜索字符串 **Windows (**字首的字母 **W** 为大写**)**。

如果不想搜索时区分大小写，就要用参数**/i**

```
findstr /i Windows proposal.txt
```

在当前目录下的 **Proposal.txt** 文件中搜索字符串 **Windows (**不区分大小写，**windows** 也会被搜索到**)**

/c:

搜索字符串，有多个字符串时，用空格隔开，并且所有字符串要用双引号引起来。

findstr "Hello There" D:\test.txt
在 **D:\test.txt** 中搜索 **"Hello"** 或 **"there"**字符串，如果有出现该字符串，那么程序会将该字符串所在的行全部打印出来；如果全都没索索到，则没什么显示。假如 **D** 盘根目录的《**test.txt**》文本的内容为：
Hello world!
computer program
I'm here.
here and there
There are ten beautiful girls!
那么执行 **findstr "Hello There" D:\text.txt** 后显示为：
Hello world!
There are ten beautiful girls!
说明：之所以没出现 **here and there**，是因为搜索中的 **There** 首字母大写。没有说明的情况下，**findstr** 会严格区分大小写。
如果执行 **findstr "hello here" D:\test.txt**，执行结果为：
Hello world!
I'm here.
here and there
There are ten beautiful girls!
因为 **there** 中包含 **here**，所以这些全都被搜索出来了。

假如我们要将 **hello there** 当做一整个字符串来搜索，就需要添加参数 **/c:**了
findstr /c:"hello there" D:\test.txt
在 **D:\test.txt** 中搜索字符串**"hello there"**

/s
搜索当前目录及其子目录。
@echo off
findstr /s /i /n Windows *.*
pause
在目前目录及其所有子目录下，不区分大小写，在每一个档案中搜索字符串 **Windows**。这个搜索过程，会将批处理本身也搜索。注意这里*.*的应用，没有这个就搜索不出来了。

假如我们在当前目录及其子目录下，有个 **MHL.txt** 的文件，里面有 **windows** 字符，那么执行上面结果后为：
a.bat:2**:findstr /s /i /n　windows *.***
MHL.txt:10**:windows**
MHL.txt:13**:windows**
MHL.txt:14**:WINDOWS**

有了**/n**，则将文件里字符串 **windows** 所在的行号都显示出来了。

/n
对搜索出匹配的行添加行号。
findstr /b /n /c:"*FOR" *.bas
在 **bas** 文件中搜索含有文字 **"FOR" (**前面可以有任意个空格**)** 的行，并在找到的行前添加行号。

一些例子：
findstr /g:finddata.txt /f:filelist.txt > results.out
在 **Finddata.txt** 中搜索文件 **Filelist.txt** 中所列的文件，然后将结果保存到 **Results.out** 中。

findstr /s /i /m "\<computer\>" *.*
在当前目录及子目录下，不区分大小写搜索含有 **computer** 文字的每一个文件。

findstr /s /i /m "\<comp.*" *.*
在当前目录以及子目录下，搜索以 **comp** 开头的所有字符串。

FINDSTR 正则表达式

findstr 的正则表达式是用来定义字符串样式的，它的元字符有 **.**、*、[-]、\、\<、\>、^、$等，参与运算的对象主要是字母、数字、符号、还有汉字。而且该运算具有特殊的规则。

findstr 中正则表达式用法规则小结
一：行首行尾规则
如**"^rem"**、**"bat$"** 表示从行首匹配 **rem**、从行尾匹配 **bat**。
1、行首行尾可以是英文、数字，还可以是汉字。

二：字符集规则
如 [}>]、**[abc]**、[123]、**[a-zA-Z]**、[0-9] 表示行中含有集里的任意字符即匹配。
注意，这是集合，不是串。如：**"[news]"**不能理解为查找含有 **news** 单词的行，只能是定位含有 **n e w s** 4 个字母之一的行。
1、需要说明的是，该字符集里的集元素可以是字母和数字和一般的半角字符。

如可以是 **:** } { **, .**] [等，但双引号"不被识别，不能是汉字（汉字不是 **ASCII** 码）。
2、**"[.*]"** 集合中出现的 **.***，作为普通字符，没有特殊含意。
3、可以组合使用,如 **[aef1-3x-z]**表示该字符集是 **aef** 和数字 1-3 和字母 **x-z** 等元素的并集**.**
4、**"[ah][1-3]"** 表示 2 个字符。

三：减法规则
[^abc] 参照帮助信息，本该理解为，匹配不含 **abc** 三个字母的行。但在 **xp** 系统下，却不被正确解释。
1、**"[^echo.]"** 实际表示在查找结果中去除为**"echo."**字符串的行。

四：通配符和重复符规则
即 **.***
1、通配符 **.** 代表任何一个字符，包括字母、数字、半角符号还有空格，但不包括空行。
2、重复符 * 代表前面字母的重复（重复次数从 0 到多次）。如 **.*** **[a-z]*** [0-9]* **[abc]* a***

五：单词前缀后缀定位规则
"\<cal" 和 **"ed\>"**
该单词可以是英文单词和数字，该单词规则不适用于汉字。符号 \ 为转义符。
1、**"\<cal"**表示查找文本中，英文单词以 **cal** 为前缀的单词 如 **call called calling calculation**
2、**"ed\>"** 查找文本中，英文单词以 **ed** 为后缀的单词 如 **called added changed**
3、**"\<call\>"** 用来精确查找单词。这里用来精确查找单词 **call** ，那么 **calling called** 就不被匹配。
4、"\<3389\>" 表示精确查找 3389 那么 33898、233895 就不被匹配。

六：关键字规则 **"string"**
该 **string** 可以是英文单词、汉字、数字、符号以及上面规则的组合**.** 能够正确解读汉字的只有行首行尾规则，和该规则。

七：转义符 \
把表达式中的特殊字符**(**元字符**)**转化为普通字符。但不能对双引号" 和大于号 > 转义。

常见写法
\.、*、\\、\[、\]、\-、\\<
如 **"\.bat"** 把通配符转化为普通的句点，这里表示匹配**".bat"**结尾的批处理文件名。也可以写成 **"[.]bat"**

示例 1：
dir /a /b /s d:\ | findstr /i "\.bat$"
查找行尾是**".bat"**字符串的行，并显示出来。
\是转义符号，使句点.不再是通配符，而是文件拓展名中的点。用于表达式 的 关键字查询，在这里是列举后缀是 **bat** 的批处理文件。

dir /a /b /s d:\ | findstr /i ".bat$"
同效于 **dir /a /b /s d:\ | findstr /i "bat$"**

findstr /r /i /n Google d:\bat\wenben.txt | more
使用 **Strings** 和 **[Drive:][Path] FileName [...]** 在命令字符串中，所有 **findstr** 命令行选项必须在 **Strings** 和 **[Drive:][Path] FileName [...]** 之前。

findstr /r /i /n /x /c:"Google news" d:\bat\htm\meinv.txt | more
/x 打印完全匹配的行.查找只有“**Google news**”单词的行。
如果是带空格的字符串 ，要用双引号引起来，而且要用参数 **/C:"string1 string2"**

findstr . 2.txt 或 **Findstr "." 2.txt**
从文件 **2.txt** 中查找任意字符，不包括空字符或空行。
空字符不是空格。.可以匹配空格
// F:>echo hi pz|findstr "...pz"
// hi pz
// F:>echo hi pz|findstr "....pz"

findstr /i /n [g-k] d:\bat\htm\meinv.txt | more
表达式中**[g-k]**，可以是字母 **a-z** 或者数字 0-9,不能是汉字。而且要顺序排列，由小到大。
示例 2
下面的命令实现提取网页的特定行
Findstr /r /i /n %string% %htmfile%
%htmfile% 表示网页文件（文本文件）。

%string% 表示 关键字或字符串表达式。
"<title>" 提取**<title>**所在行的信息，即标题信息。
"" 提取在****内加粗的文本
"<h[1-6]**"** 提取所有的正文标题信息包括 **h**1-**h**6
[59]　查找含有数字 5 或 9 的行
1[58] 查询含有 15 或 18 的行
[@,.] 查找含有符号"@"或逗号","或句点"."的行，定位有该符号任意之一的行。但双引号"不被正确解释。//？
"^<div"　查询定位行首字符是**<div** 的行**.**
"}$"　定位行尾是右花括号的行
"[}>]$"　定位行尾是右花括号或者大于号的行**.** 使用行首或行尾进行综合定位时必须加双引号才能被正确解释。
"[^echo.]" 减法规则，在查找结果中去除含有**"echo."**字符串的行，而且有严格规定，该行仅仅只有字符串**"echo."**不能再含有其它字符，甚至不能有空格。否则视为无效。但是可以使用多个减法规则，用来去除多行。
"[^echo.][^add][^call]" 减法规则，查找去除含有 **echo.** 或者 **add** 或者 **call** 单词的行。
"[.*]" 如果在字符集内插入通配符和重复符号将视为普通字符，没有通配和重复的含义。
"[abcl][^echo.]" 减法规则也可以与字符集规则联合使用。
"[abcl][^echo.][^add][^call]"

```
@echo off
title 循环测试正则表达式用法
: begin
cls
set /p string=输入关键字或字符串表达式
set htmfile=d:\bat\htm\meinv.htm
echo.
Findstr /r /i /n %string% %htmfile%|more
echo.
pause&&goto begin
rem Q 键退出 more
rem 参数/R 强调以正则表达式规则来解读字符串。
```

程序例子：
1.

```
findstr . test.txt
Findstr "." test.txt
findstr . D:\test.txt
```

从文件 **test.txt** 中查找任意字符**(**不包括空行，也就是没有任何字符的行**)**。这个功能除了空行不显示出来之外，其余内容都可以显示出来。

2.
findstr .* test.txt
findstr ".*" test.txt
从文件 **test.txt** 中查找任意字符**(**包括空行和空字符**)**，这个功能等于显示出整个文件的内容。

3.
findstr "[0-9]" test.txt
从文件 **test.txt** 中搜索包含数字 0－9 的字符串。只要该行包含任意一个字符，那么都会被搜索出来。例如 2.71828，**abc**123 等等都是符合要求的。

4.
findstr "[a-zA-Z]" test.txt
从文件 **test.txt** 中查找包括英文字母**(**大写或小写**)**任意字符。不管大小写，只要包含任意一个英文字母，都会被搜索出来。

5.
findstr "[abcezy]" test.txt
从文件 **test.txt** 中查找只包含 **a**、**b**、**c**、**e**、**z**、 **y** 字母的字符串。

6.
findstr "[a-fl-z]" test.txt
从文件 **test.txt** 中查找小写字符 **a-f l-z** 的字符串，但不包含 **g h I j k** 这几个字母。

7.
findstr "M[abc][hig]Y" test.txt
从文件 **test.txt** 中搜索符合第一个字母是 **M**，第二个字母是 **abc** 中任意一个字母，第三个字母是 **hig** 中任意一个字母，第四个字母是 **Y** 的字符串。
注：中括号里的字母没有次序之分。

8.
^和$，匹配行首行尾。 ^ 表示行首；$ 表示行尾。
"^step"仅匹配 **"step hello world"**中的第一个单词
"step$"仅匹配 **"hello world step"**中最后一个单词

9.
finstr "[^0-9]" test.txt
过滤掉纯数字的行后将所有内容显示出来。例如 2323423423 这样的字符串就不显示，如果是 345**hh**888 或者 2.71828 这样的形式就不会被过滤掉，可以显示出来。

10.
findstr "[^a-z]" test.txt
过滤掉纯字母的行，例如 **sdlfjlkjlksjdklfjlskdf** 这样的字符，如果是 **sdfksjdkf**99999 这样的形式，掺杂着数字就不会被过滤掉了。

11.
. 通配符：任何字符
* 重复：以前字符或类别出现零或零以上次数
*号的作用
".*"表示搜索的条件是任意字符
*号在这里的作用不是任何字符，而是表示左侧字符或者表达式的重复次数，*号表示重复的次数为零次或者多次。

假如在当前目录或子目录下，有一份文件里面包含有“**windows**”字符串，那么 **findstr /s /i windows *.*** 可以正确执行，搜索出字符串。而 **findstr /s /i windows** 就不能执行，为什么？*.*是什么意思？

12.
findstr "^[0-9]*$" 2.txt
这个是匹配找到的纯数字，例如 234234234234，如果是 2133234**kkjl**234 就被过滤掉了。
Findstr "^[a-z]*$" 2.txt
这个是匹配找到的纯字母，例如 **sdfsdfsdfsdf**，如果是 213**sldjfkljsdlk** 就被过滤掉了
如果在搜索条件里没有*号，也就是说不重复左侧的搜索条件，也就是[0-9] [**a-z**]那只能匹配字符串的第一个字符也只有这一个字符，因为有行首和行尾的限制，"^[0-9]$"第一个字符如果是数字就匹配，如果不是就过滤掉，如果字符串是 9 就匹配，如果是 98 或者 9**j** 之类的就不可以了。

13. "\<…\>"这个表达式的作用
这个表示精确查找一个字符串，\<**sss** 表示字的开始位置，**sss**\>表示

字的结束位置
echo hello world computer|findstr "\<computer\>"这样的形式
echo hello worldcomputer|findstr "\<computer\>" 这样的形式就不成了，他要找的是 **"computer"**这个字符串，所以不可以。
echo hello worldcomputer|findstr ".*computer\>"这样就可以匹配了

当然，上面的例子仅仅只是说明"\<…\>"的作用而已。上面程序其实直接就 **echo hello world computer|findstr "computer"**就可以了。
---
findstr 应用实例：
1、判断字符串是否是数字：
用 **findstr** 来判断字符串是否数字非常方便。

```
@echo off
set /p a=请输入：
echo %a%|findstr /be "[0-9.]*" >nul &&echo 是数字||echo 不是数字
& %0
pause > nul
```

本程序可以判断一串字符串是否是数字。但是，仍有改进之处，就是我们输入小数 1.2，可以判断是数字，但输入 1..2，这种不合法的字符也判定为数字。

读入一串字符串，判断如果是数字，那么求出其平方值。如果不是数字，则要求用户重新输入。

```
@echo off
setlocal enabledelayedexpansion
:loop
set /p var=请输入：
echo %var%|findstr /be "[0-9.]*">nul && (
    set /a var*=var
    echo 您的输入的%var%是个数字，其平方值为!var!
    pause >nul
    ) || (
        echo 您的输入不是数字，请重新输入。
        pause >nul & goto :loop
    )
)
```

注：这上面用到的判断没有用 **if**，而是用组合命令&&和||。&&表示前面运行正确，那么执行&&后的语句；||表示前面运行错误，才执行||后的语句。于是可以达到判断的效果。

2、搜索指定文本文件里的内容

```
@echo off
findstr /i "windows" D:\MHL.txt
pause>nul
```

不区分大小写，搜索 **D:\MHL.txt** 里的 **windows** 字符串。

```
@echo off
findstr /be [0-9]*    D:\MHL.txt
pause>nul
```

搜索出 **D:\MHL.txt** 里的纯数字。

3、**findstr** 显示彩色效果文字

findstr /a 可以在同一窗口设置不同的颜色，具体用法如下：

现在假设当前环境是在 **D** 盘根目录，**D** 盘根目录下有一个名为《**High precision computation.txt**》文件，即 **D:\High precision computation.txt**，《**High precision computation.txt**》里面的内容是（没有分割线）：

```
------------
高精度加法
高精度减法
高精度乘法
高精度除法
高精度开方
高精度计算圆周率
-----------------
```

我们在任意地方，比如 **C** 盘根目录，新建一个文本文档《**test.txt**》，里面的代码为：

```
findstr /a:2 . "D:\High precision computation.txt*"
pause>nul
```

上面代码要注意，因为文件名 **High precision computation** 含有空格，所以要用双引号。将文件改为《**test.bat**》后双击运行，那么就可以显示出 **High precision computation.txt** 这一串字符颜色是彩色的。注意：不是里面的内容彩色，而是文件名彩色。

```
findstr /a:2 . "D:\High precision computation.txt*"
```

这条格式比较讨厌，**/a** 和后面的冒号不可以有空格，冒号和颜色代码 2 之间也不可以有空格。这里颜色代码只是 2，说明只设置了前景颜色，如果要设置背景颜色，代码可以改为：**/a:2f**，就可以一并设置背景颜色了。

.是分隔符，照写就是。后面是文件完整路径，注意，如果文件路径含有空格，那么老规矩，必须用双引号引起来。最后，路径后面还要含有通配符*，这一点切记！

读入一串字符，输出彩色效果：

```
@echo off
set /p str=请输入显示字符（不能有^\^/^:^*^?^"^<^>^|）：
set /p var= <nul>%str%
findstr /a:5 .* "%str%*"
echo. & pause
```

::这个程序不够完美，就是每次都会在当前路径建立一个不知类型文件。并且，输出的结果会多加个冒号，要去除冒号，就要用退格符号帮助，这里就不去弄了。而如果不想新建一个文件，则无法实现。

对于输出冒号的问题，我们可以用退格符号“◘”来将其删除。退格键“◘”如何获得？方法一：百度上直接复制下来。方法二：打开 **cmd**，输入 **edit** 打开代码编辑窗口，先 **ctrl+p**，然后再 **ctrl+H** 或 **alt+8**（小键盘的 8）就可以得到这个符号了。对于建立了文件的问题，我们可以随手将其删除。于是代码改动如下：

```
@echo off
set /p var=请输入字符：
set /p col=请选择颜色
set /p var=◘ <nul>%var%
findstr /a:%col% .* %var%*
del %var%
pause>nul
```

上面的代码就可以实现对输入的字符显示指定的颜色。

```
@echo off
set /p var=请输入字符：
set /p col=请选择颜色
echo ◘ ◘>%var%
findstr /a:%col% . %var%*
del %var%
pause
```

我们还可以通过参数的形式，程序会简便一些，代码改动如下：

```
@echo off
call :x c  第一行输出红色
call :x a  第二行输出绿色
call :x f  第三行输出白色
pause
:x
echo.□ □>%2
findstr /a:%1 . %2*
del %2
```

读入的形式为：

```
@echo off
set /p str=请输入字符：
set /p col=请输入颜色：
call :x %col% %str%
pause
:x
echo.□ □>%2
findstr /a:%1 . %2*
del %2
```

findstr 不存在不新建一个文件而实现彩色的效果。**color** 则不可以设置成彩色效果，只有单调的前景颜色和背景颜色。
综上所述，批处理实现彩色显示效果非常麻烦，而且效果不理想。所以批处理彩色显示意义不大，笔者研究到此为止，有兴趣的朋友可以继续研究。

---

pushd，popd

这对个命令很生疏，之前的教程也没说个所以然来，调出帮助文件，看得更是一头雾水。
说白了，就是用 **pushd** 标记路径，然后 **popd** 恢复路径。
使用例子：
例如在命令行窗口里，当前的路径为 **C:\Documents and Settings\Administrator**，这条路径有点长，我们不想等会用手工打这条路径，那么就可以用 **pushd** 标记该路径。具体方法是：

```
C:\Documents and Settings\Administrator>pushd C:\Documents and Settings\Administrator
```

这样，路径就被记住了。接下来，我们随便到别的目录下，例如我们先到 C 盘根目录，然后又溜达到 **D:\Program files** 目录后，我们想回到原先的 **C:\Documents and Settings\Administrator** 目录，这是我们在当前窗口输入 **popd** 并回车就行了。

D:\Program files>popd

那么命令行的路径就显示为：

C:\Documents and Settings\Administrator>

需要注意的是：我们用 **pushd** 标记路径的时候，必须在所在路径标记，否则无效。

例如：

C:\Documents and Settings>pushd C:\Documents and Settings\Administrator

执行后窗口显示：

C:\Documents and Settings\Administrator>

然后我们进入到 C 盘根目录 **C:\>**，此时输入 **popd**，屏幕显示：

\C:\Documents and Settings>

没有得到我们想要的路径 **C:\Documents and Settings\Administrator**

其次，我们用 **pushd** 标记路径时，虽然是在所在的目录标记，但仍然要输入完整的路径名。

例如：

C:\Documents and Settings\Administrator>pushd Documents and Settings\Administrator

这样的标记是无效的。

每次使用 **pushd** 命令时，都将储存一个目录以备用户使用。可以通过不断重复使用 **pushd** 命令存储多个目录，目录按顺序储存到一个虚拟堆栈中（压栈）。第一次使用 **pushd** 命令，该命令的目录就会被置于栈底，重复使用该命令，则每次目录就会被置于前一个目录之上（压栈）。

可以使用 **popd** 命令将当前目录切换（出栈）为最近由 **pushd** 命令存储的目录（栈顶），并且该位于栈顶的目录会从堆栈中被删除。重复使用 **popd** 命令，可以不断切换到堆栈中每次的栈顶目录，直到栈底。最后一次，栈底会被删除，以后再执行 **popd** 命令就无效了，因为堆栈已是空栈。

如果启用了命令扩展，**pushd** 命令将接收网络路径或本地驱动器盘符和路径。如果指定网络路径，**pushd** 命令临时将第一个未使用的驱动器盘符（从 **Z** 开始）分配到指定的网络资源。然后命令将当前驱动器和目录更改为新分配驱动器上的指定目录。如果与已启用的命令扩展一起使用 **popd** 命令，**popd** 命令将删除由 **pushd** 创建的驱动器

盘符分配。
实际操作例子：
C:\Documents and Settings\Administrator>pushd C:\Documents and Settings\Administrator
C:\Documents and Settings\Administrator>cd..
C:\Documents and Settings>pushd C:\Documents and Settings
C:\Documents and Settings>cd..
C:\>popd
C:\Documents and Settings>popd
C:\Documents and Settings\Administrator>cd
C:\>popd
C:\>
这是一次压栈和出栈的实际操作过程。每执行一次 **popd** 都会自动删除一次出栈路径。当到达栈底后，所有路径都被清空，再执行 **popd** 就没有什么效果了，于是最后执行 **popd** 后，根目录还是根目录，没变化。

批处理的命令实在太多，限于笔者的时间精力，没能一一深入研究。下面几个命令只是简单复制帮助里面的信息，有兴趣的朋友可以自行研究学习：

CMD

启动 **Windows XP** 命令解释程序一个新的实例
备注：**CMD** 的用法也是奇复杂！笔者稍微看了一下帮助内容，望而却步，不敢进一步研究学习。

语法：
CMD [/A | /U] [/Q] [/D] [/E:ON | /E:OFF] [/F:ON | /F:OFF] [/V:ON | /V:OFF] [[/S] [/C | /K] string]

/C　　执行字符串指定的命令然后终断
/K　　执行字符串指定的命令但保留
/S　　在 **/C** 或 **/K** 后修改字符串处理(见下)
/Q　　关闭回应
/D　　从注册表中停用执行 **AutoRun** 命令(见下)
/A　　使向内部管道或文件命令的输出成为 **ANSI**
/U　　使向内部管道或文件命令的输出成为 **Unicode**

/T:fg　设置前景/背景颜色(详细信息，请见 **COLOR /?)**
/E:ON　启用命令扩展(见下)
/E:OFF　停用命令扩展(见下)
/F:ON　启用文件和目录名称完成字符 (见下)
/F:OFF　停用文件和目录名称完成字符(见下)
/V:ON　将 **!** 作为定界符启动延缓环境变量扩展。如**: /V:ON** 会允许 **!var!** 在执行时允许 **!var!** 扩展变量 **var**。**var** 语法在输入时扩展变量，这与在一个 **FOR** 循环内不同。
/V:OFF　停用延缓的环境扩展。

请注意，如果字符串有引号，可以接受用命令分隔符 '&&' 隔开的多个命令。并且，由于兼容原因，**/X** 与 **/E:ON** 相同，**/Y** 与**/E:OFF** 相同，并且 **/R** 与 **/C** 相同。忽略任何其他命令行开关。

如果指定了 **/C** 或 **/K**，命令行开关后的命令行其余部分将作为命令行处理；在这种情况下，会使用下列逻辑处理引号字符**("):**

1.　如果符合下列所有条件，那么在命令行上的引号字符将被保留**:**

- 不带 **/S** 命令行开关
- 整整两个引号字符
- 在两个引号字符之间没有特殊字符，特殊字符为下列中的一个**: <>()@^|**
- 在两个引号字符之间有至少一个空白字符
- 在两个引号字符之间有至少一个可执行文件的名称。

2.　否则，老办法是，看第一个字符是否是一个引号字符，如果是，舍去开头的字符并删除命令行上 的最后一个引号字符，保留最后一个引号字符之后的文字。

如果 **/D** 未在命令行上被指定，当 **CMD.EXE** 开始时，它会寻找以下 **REG_SZ/REG_EXPAND_SZ** 注册表变量。如果其中一个或两个都存在，这两个变量会先被执行。

HKEY_LOCAL_MACHINE\Software\Microsoft\Command Processor\AutoRun

和/或

HKEY_CURRENT_USER\Software\Microsoft\Command Processor\AutoRun

命令扩展是按默认值启用的。您也可以使用 **/E:OFF**，为某一特定调用而停用扩展。您可以在机器上和/或用户登录会话上启用或停用 **CMD.EXE** 所有调用的扩展，这要通过设置使用 **REGEDT32.EXE** 的注册表中的一个或两个 **REG_DWORD** 值:

HKEY_LOCAL_MACHINE\Software\Microsoft\Command Processor\EnableExtensions

和/或

HKEY_CURRENT_USER\Software\Microsoft\Command Processor\EnableExtensions

到 **0x1** 或 **0x0**。用户特定设置比机器设置有优先权。命令行开关比注册表设置有优先权。

命令行扩展包括对下列命令所做的更改和/或添加:

DEL 或 **ERASE**
COLOR
CD 或 **CHDIR**
MD 或 **MKDIR**
PROMPT
PUSHD
POPD
SET
SETLOCAL
ENDLOCAL
IF
FOR
CALL
SHIFT
GOTO
START (同时包括对外部命令调用所做的更改)
ASSOC
FTYPE

有关详细信息，请键入 **HELP** 命令名。

延迟变量环境扩展不按默认值启用。您可以用**/V:ON** 或 **/V:OFF** 命令行开关，为 **CMD.EXE** 的某个调用而启用或停用延迟环境变量扩充。您可以在机器上和/或用户登录会话上启用或停用 **CMD.EXE** 所有调用的完成，这要通过设置使用 **REGEDT32.EXE** 的注册表中的一个或两个 **REG_DWORD** 值:

HKEY_LOCAL_MACHINE\Software\Microsoft\Command

Processor\DelayedExpansion
　　和/或
　　HKEY_CURRENT_USER\Software\Microsoft\Command Processor\DelayedExpansion

到 0x1 或 0x0。用户特定设置比机器设置有优先权。命令行开关比注册表设置有优先权。

如果延迟环境变量扩充被启用，惊叹号字符可在执行时间，被用来代替一个环境变量的数值。

文件和目录名完成不按默认值启用。您可以用 **/F:ON** 或 **/F:OFF** 命令行开关，为 **CMD.EXE** 的某个调用而启用或停用文件名完成。 您可以在机器上和/或用户登录会话上启用或停用 **CMD.EXE** 所有调用的完成，这要通过设置使用 **REGEDT32.EXE** 的注册表中的一个或两个 **REG_DWORD** 值:
　　HKEY_LOCAL_MACHINE\Software\Microsoft\Command Processor\CompletionChar
　　HKEY_LOCAL_MACHINE\Software\Microsoft\Command Processor\PathCompletionChar
　　和/或
　　HKEY_CURRENT_USER\Software\Microsoft\Command Processor\CompletionChar
　　HKEY_CURRENT_USER\Software\Microsoft\Command Processor\PathCompletionChar

由一个控制字符的十六进制值作为一个特定参数(例如，0x4 是 **Ctrl-D**，0x6 是 **Ctrl-F**)。 用户特定设置优先于机器设置。命令行开关优先于注册表设置。

如果完成是用 **/F:ON** 命令行开关启用的，两个要使用的控制符是:目录名字完成用 **Ctrl-D**， 文件名完成用 **Ctrl-F**。 要停用注册表中的某个字符，请用空格(0x20)的数值，因为此字符不是控制字符。

如果键入两个控制字符中的一个，完成会被调用。完成功能将路径字符串带到光标的左边，如果没有通配符，将通配符附加到左边，并建立相符的路径列表。然后，显示第一个相符的路径。如果没有相符的路径，则发出嘟嘟声，不影响显示。之后，重复按同一个控制字符会循环显示相符路径的列表。将 **Shift** 键跟控制字符同时按下，会倒着显示列表。如果对该行进行了任何编辑，并再次按下控制字符，保存

的相符路径的列表会被丢弃，新的会被生成。如果在文件和目录名完成之间切换，会发生同样现象。两个控制字符之间的唯一区别是文件完成字符符合文件和目录名，而目录完成字符只符合目录名。如果文件完成被用于内置式目录命令**(CD、MD 或 RD)**，就会使用目录完成。
将引号将相符路径括起来，完成代码可以正确处理含有空格或其他特殊字符的文件名。同时，如果备份，然后从行内调用文件完成，完成被调用是位于光标右方的文字会被丢弃。

需要引号的特殊字符是**:**

<space>
& ( ) [ ] { } ^ = ; ! ' + , ` ~

FC

比较两个文件或两个文件集并显示它们之间的不同

语法**:**

FC [/A] [/C] [/L] [/LBn] [/N] [/OFF[LINE]] [/T] [/U] [/W] [/nnnn] [drive1:][path1]filename1 [drive2:][path2]filename2
FC /B [drive1:][path1]filename1 [drive2:][path2]filename2

/A	只显示每个不同处的第一行和最后一行。
/B	执行二进制比较。
/C	不分大小写。
/L	将文件作为 **ASCII** 文字比较。
/LBn	将连续不匹配的最大值设为指定的行数。
/N	在 **ASCII** 比较上显示行数。
/OFF[LINE]	不要跳过带有脱机属性集的文件。
/T	不要将 **tab** 扩充到空格。
/U	将文件作为 **UNICODE** 文字文件比较。
/W	为了比较而压缩空白**(tab** 和空格**)**。
/nnnn	指定不匹配处后必须连续匹配的行数。
[drive1:][path1]filename1	指定要比较的第一个文件或第一个文件集。
[drive2:][path2]filename2	指定要比较的第二个文件或第二个文件集。

FORMAT

格式化磁盘以供 **Windows XP** 使用。

语法：

FORMAT volume [/FS:file-system] [/V:label] [/Q] [/A:size] [/C] [/X]
FORMAT volume [/V:label] [/Q] [/F:size]
FORMAT volume [/V:label] [/Q] [/T:tracks /N:sectors]
FORMAT volume [/V:label] [/Q]
FORMAT volume [/Q]

volume　　指定驱动器(后面跟一个冒号)、装入点或卷名。
/FS:filesystem　指定文件系统类型(**FAT**、**FAT32** 或 **NTFS**)。
/V:label　　指定卷标。
/Q　　执行快速格式化。
/C　　仅适于 **NTFS:** 默认情况下，将压缩在该新建卷上 创建的文件。
/X　　如果必要，先强制卸下卷。那时，该卷所有 已打开的句柄不再有效。
/A:size　　替代默认配置单位大小。极力建议您在一般状况下使用 默认设置。

NTFS 支持 512、1024、2048、4096、8192、16**K**、32**K**、64**K**。

FAT 支持 512、1024、2048、4096、8192、16**K**、32**K**、 64**k**，(128**k**、256**k** 用于大于 512 字节的扇区) 。

FAT32 支持 512、1024、2048、4096、8192、16**k**、32**k**、64**k**，(128**k** 、256**k** 用于大于 512 字节的扇区)。

注意 **FAT** 及 **FAT32** 文件系统对卷上的群集数量有以下限制:

FAT: 群集数量 <= 65526

FAT32: 65526 < 群集数量 < 4177918

如果判定使用指定的群集大小无法满足以上需求，格式 化将立即停止。

NTFS 压缩不支持大于 4096 的分配单元。

/F:size　　指定要格式化的软盘大小(1.44)
/T:tracks　　为磁盘指定每面磁道数。
/N:sectors　　指定每条磁道的扇区数。

MORE

逐屏显示输出。

语法：

MORE [/E [/C] [/P] [/S] [/Tn] [+n]] < [drive:][path]filename
command-name | MORE [/E [/C] [/P] [/S] [/Tn] [+n]]
MORE /E [/C] [/P] [/S] [/Tn] [+n] [files]

[drive:][path]filename　指定要逐屏显示的文件。
command-name　　指定要显示其输出的命令 。

/E　　启用扩展功能
/C　　显示页面前先清除屏幕
/P　　扩展 **FormFeed** 字符
/S　　将多个空白行缩成一行
/Tn　　将跳格键扩展成 **n** 个空格(默认值为 8)
　　命令行开关可以出现在 **MORE** 环境变量中。
+n　　从第 **n** 行开始显示第一个文件
files　　要显示的文件列表。 用空格分开列表中的文件。

如果扩展的功能已经启用，在 **-- More --** 提示处会接受下列命令**:**

P n　　显示下 **n** 行
S n　　略过下 **n** 行
F　　显示下个文件
Q　　退出
=　　显示行号
?　　显示帮助行
<space>　显示下一页
<ret>　　显示下一行

RECOVER

从损坏的磁盘中恢复可读取的信息。

语法：

RECOVER [drive:][path]filename

REPLACE

替换文件。

语法：

REPLACE [drive1:][path1]filename [drive2:][path2] [/A] [/P] [/R] [/W]
REPLACE [drive1:][path1]filename [drive2:][path2] [/P] [/R] [/S] [/W] [/U]

[drive1:][path1]filename 指定源文件。
[drive2:][path2] 指定要替换文件的目录。
/A 把新文件加入目标目录。不能和 **/S** 或 **/U** 命令行开关搭配使用。
/P 替换文件或加入源文件之前会先提示您进行确认。
/R 替换只读文件以及未受保护的文件。
/S 替换目标目录中所有子目录的文件。不能与 **/A** 命令选项搭配使用。
/W 等您插入磁盘以后再运行。
/U 只会替换或更新比源文件日期早的文件。不能与 **/A** 命令行开关搭配使用。

---

assoc 和 ftype

这两个是文件关联
assoc 设置'文件扩展名'关联，到'文件类型'
ftype 设置'文件类型'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。

批处理编程格式

我们要养成良好的编程风格，下面是批处理编程的格式：

1、**@echo off**

作为批处理开头，关闭程序所有命令回显，否则程序显示会很杂乱。除非特殊需要的情况下才不关闭。

2、**setlocal enabledelayedexpansion**

延迟变量　　稍微复杂点的程序，都要用到延迟变量。

3、**color**

设置颜色　　为了美观，可以设置一下颜色。

4、**mode**

设置窗口大小。这个在需要固定窗口大小时才使用，可以省略。

5、**title**

设置窗口标题。 这个也有必要设置，在窗口显示程序名称。另外，对某些程序，用它来显示运行进度。

6、**echo**

程序名称　　这个很多人都会忽略，一个好的程序，需要介绍其功能，至少需要写明程序名称，防止忘记。

程序里一定要有注解，便于阅读和维护。注解要写在被注解程序上一行。

7、**“set /p var=请输入一个数据：”**，这种用户输入变量，一定要有提示语，这个例子的提示语就是“请输入一个数据”。

8、不使用^续行。

9、不使用分号。

10、可以不用组合命令的情况下，就不使用组合命令，程序改为分行写，比较明朗且不易出错。

11、要有缩进，规范的缩进用 **TAB** 键。括号回括的位置也是 **TAB** 键位置。

12、子程序要用空行隔开。

例如：

```
@echo off
title Program
color 9f

::子程序要空行
:loop
for /l %%i in (1,1,%all%) do (
    if !remaining!==%remain% (goto :loopend)
        if !S%%i!==1 (
            if !circulate!==%count% (
                set g=              %%i <nul
                                            set /p var=!g:~-7!<nul
                                            if !f!==10 (echo. & set f=0)
                                            set /a f+=1
                                            set S%%i=0
                                            set /a remaining-=1
                                            set    circulate=1
            ) else (set /a circulate+=1)
        )
    )
)
goto :loop

:end loop
echo The end!
pause
```

上面程序是不能运行的，这里只作为一个程序书写范例。**:loop** 和**:end loop** 这两个都是子程序段，要用空行与上面程序隔开，便于阅读。注意上面括号位置，回括的括号分别都是对应在 **TAB** 键位置。

由于 **set** 对变量的大小写不敏感，而 **for** 对变量的大小写敏感。所以我们在写程序时，应该做到用过大写名字的变量就不再用小写同名变量，用过小写名字的变量就不再用大写同名变量。并且，在书写变量时，对统一变量名始终严格统一大小写。例如一开始使用 **var** 变量，那么以后我们就不再使用 **Var vAr vaR VAr vAR VaR VAR** 这些形式的变量，并且书写时始终保持 **var** 这唯一一种写法。

批处理编程举例

程序一：**界面设计**

```
@echo off
::设置标题和时间(时间一般不显示在这里，这里只是说明时间可以显示在标题栏)
title 界面设计样板 1.0 版　　当前日期:%date%
::设置颜色
color 9f
::设置窗口大小
mode con cols=60 lines=20

:begin
echo.&echo.&echo　　批处理软件界面设计样板
echo.&echo.&echo　　==============================
echo　　请选择要进行的操作，然后按回车
echo　　==============================
echo.
echo　　1.隐藏文件
echo　　2.关机
echo　　Q.退出
echo.
set /p choose=请选择：
::我们选择的字母一般不区分大小写，所以 if 语句一般都用/i 关闭大小写敏感
if /i "%choose%"=="1" goto hide
if /i "%choose%"=="2" goto shut
if /i "%choose%"=="Q" ( exit ) else (
    echo 输入有误，请重新输入。
    pause>nul
    ::要注意清屏
    cls & goto begin
)

:hide
echo. & echo 文件隐藏完毕
pause>nul & exit

:shut
echo. & set /p s=请选择　　( F 定时关机　　D 倒计时关机 ) :
if /i "%S%"=="F" (echo 系统将在 23:00 关机，请保存好文件) else (
    if /i "%s%"=="D" (echo 系统将在 1 小时后关机，请保存好文件) else (
        echo 输入有误，请重新输入
```

```
        pause>nul & cls & goto begin
    )
)
pause >nul & exit
```

上面就是一个简单的界面设计例子，仅作示例，所以程序中没有给出具体的隐藏文件程序和关机程序。

程序二
问题：**echo 最多可以显示多少个字符？**
测试程序：

```
@echo off
set word=a
set var=%b%
set num=1
:kaishi
echo %var%
set var=%var%%word%
set /a num+=1
echo %num% > num.txt
goto kaishi
```

这个程序一直运行到自动崩溃为止。
测试结果：
word=a，num=8184
word=道，**num**=8184
也就是说，无论中文、英文、标点符号等等，**echo** 一次最多只能显示 8184 个。这个足够我们一般的显示需要了，所以也可以说 **echo** 回显没有上限。

程序三：**调出 DOS 中 help 的所有帮助信息。**

```
@echo off
setlocal enabledelayedexpansion
set "ko=                                    "
for /f "delims= " %%a in ('help^|findstr /i "^[a-z]"') do (
    set /a n+=1&set _!n!=%%a!ko!
)
```

```
set _72=共 71 个命令
:loop
color 9f
title cmd  命令帮助
for /l %%a in (1 3 !n!) do (
    set /a t=%%a+1,s=t+1
    if !t! lss 10 (set l=0) else set "l="
    call  echo  !l!%%a.  !_%%a:~0,20!  !l!!t!.  %%_!t!:~0,20%%  !l!!s!.
%%_!s!:~0,10%%
)
set /p m=请输入编号查询相应命令帮助：
cls
set /a m=100!m!%%100
call set ok=%%_!m!%%
title %ok:~0,10%  命令帮助
color 9f
%ok% /? | more
color 9f
echo.
echo  按任意键回主菜单。。。
pause>nul
cls
goto loop
```

程序四：

问：**1 到 4 组成无重复数字的三位数，共有几个？**

```
@echo off
for /l %%i in (1,1,4) do (
    for /l %%j in (1,1,4) do (
        for /l %%k in (1 1 4) do (
            if not "%%i"=="%%j" if not "%%j"=="%%k" if not
"%%i"=="%%k" echo %%i%%j%%k & set/a num+=1
        )
    )
)
echo  共能组成%num%个互不相同且无重复数字的三位数
pause>nul
```

或者这样写也可以，会减少一部分不必要的循环：

```
@echo off
set a=0
for /l %%i in (1,1,4) do (
    for /l %%j in (1,1,4) do (
        if %%i neq %%j (
            for /l %%k in (1,1,4) do if %%i neq %%k (
                if %%k neq %%j (
                    set/a a=a+1
                    echo %%i    %%j    %%k
                )
            )
        )
    )
)
echo     共%a%个
pause>nul
```

程序五：输出乘法口诀

例子一

```
@echo off
for /l %%i in (1,1,9) do (
    for /l %%j in (1,1,9) do call :print %%i %%j
    echo.
)
pause>nul

:print
if %2 gtr %1 goto :eof
set /a num=%1*%2
set /p var=%1×%2=%num%    <nul
```

例子二

```
@echo off
setlocal enabledelayedexpansion
for /l %%i in (1,1,9) do (
    for /l %%j in (1,1,9) do (
        set /a result=%%i*%%j
        set var=%%j×%%i=!result!
        if %%j leq %%i set/p=!var! <nul
```

```
        )
    echo.
)
pause>nul
```

程序六：**分解质因数**

```
@echo off
setlocal enabledelayedexpansion
::boor 值用于判断要不要输出乘号。
set boor=1
set /p num=请输入一个数字：
::因为批处理不能计算开方，所以循环改为循环到数字的一半。
set /a max=num/2
set /p var=%num%=<nul
:start
for /l %%i in (2,1,%max%) do (
    set /a yu=num%%%%i
    if !yu!==0 (
        set /a num/=%%i
        if !boor!==0 (set /p var=×<nul)
        set /p var=%%i<nul
        set boor=0
        goto :start
    )
)
echo.&echo  计算结束，按任意键退出。&pause>nul
```

网友作品：

```
@echo off
setlocal enabledelayedexpansion
set /p n=请输入整数
set /p=%n%=<nul
set /a k=2
:main
if !k!==%n% set /p=!k!<nul & goto :ex
set /a c=!n! %% !k!
if !c!==0 (
set /p=!k!*<nul
set /a n=!n!/!k!
) else (
```

```
set /a k+=1
)
goto :main
:ex
pause>nul
```

程序七：**输出国际象棋棋盘**

```
@echo off
setlocal enabledelayedexpansion
set b=0
:loop
for /l %%i in (1,1,4) do (
        set a=□■
        set/p=!a!<nul
)
echo.
for /l %%i in (1,1,4) do (
        set a=■□
        set/p=!a!<nul
)
echo.
set/a b+=1
if %b% geq 4 pause>nul&goto :eof
goto :loop
```

程序八：**十进制数转换为任意进制数**

```
@echo off
set /p x=要转换的十进制数:
set /p y=要转换的目标进制:

call :change %x% %y%
pause>nul
goto :eof

:change
set/a t=%1/%2
```

```
if %t% gtr 0 call :change %t% %2
set /a b=%1%%%2
set /p=%b% <nul
```

注：上面转换，当进制超过 10 进制时，例如十六进制，那么输出结果以 10 代表 **A**，11 代表 **B**，12 代表 **C** 如此等等。

程序久：**输出杨辉三角，要求呈金字塔状**

```
@echo off
setlocal enabledelayedexpansion
mode con cols=130 lines=30
set con=60
title 杨辉三角
:start
echo.&echo.
echo 请输入杨辉三角层数(限于显示器的宽度，层数不可超过 20)：
set /p tier=
if !tier! gtr 20 echo 输入层数过大，导致最后数字无法整齐排列，请重新输入。& cls & goto :start
echo.&echo.
set space=
set /a X1=Y1=1
for /l %%j in (1,1,!con!) do set /p var=!space!<nul
set /p var=         !X1!<nul & echo.
set /a con-=3
set /a max=tier-1
for /l %%i in (1,1,%max%) do (
    set X%%i=1
    for /l %%j in (2,1,%%i) do (
        set /a num=%%j-1
        set /a Y%%j=X!num!+X%%j
    )
    for /l %%j in (1,1,!con!) do set /p var=!space!<nul
    set /a con-=3
    for /l %%j in (1,1,%%i) do (
        set X%%j=!Y%%j!
        set tem=            !Y%%j!
        set /p var=!tem:~-6!<nul
    )
    set tem=            !Y1!
    set /p var=!tem:~-6!<nul
```

```
    echo.
)
pause>nul
```

网友作品：

```
@echo off&setlocal enabledelayedexpansion
:top
::原创作者：wudixin96
::    in=行数：
set/a ab=1,var=30,in=10
set str=1
for /l %%i in (1,1,%in%) do (
    set "num="
    set /a num2=0
    for %%a in (!str!) do (
    set /a num2+=1
    if !num2!==1 set "str1="
    set /a num1=%%a+num
    set "str1=!str1! !num1!"
    set num=%%a
    )
    call :lis "!str1:~1!"
    set "str=!str1! 0"
)
pause>nul
exit
:lis
set max=%~1
for /l %%a in (0 1 300) do (
   if not "!max:~%%a,1!"=="" set /a ci+=1
)
set /a ki=var-ci/2
for /l %%a in (1 1 !ki!) do set "kg= !kg!"
echo !kg!!max!
set nam!ab!=!kg!!max!
set /a ab+=1
set ci=0&set "kg="
goto :eof
```

程序十：**搜索完全数**
题目：一个数如果恰好等于它的因子之和，这个数就称为“完全数”。
例如 6=1＋2＋3
分析：目前知道的完全数全都是偶数，所以本程序只对偶数搜索就行了。

```
@echo off
setlocal enabledelayedexpansion
set /p num=请输入完全数搜索上限
for /l %%i in (6,2,!num!) do (
    set /a k=%%i
    set /a n=!k!/2
    set /a s=1
    for /l %%j in (2,1,!n!) do (
            set /a kk=!k! %% %%j
            if !kk!==0    set /a s+=%%j
    )
    if !s!==!k! echo !k!
    title  当前计算进度%%i
)
echo  计算结束。& pause>nul
```

注：前五个完全数是 6、28、496、8128、33550336，批处理的计算速度是非常慢的，所以用批处理来搜索完全数没有实际意义。

程序十一：**判断 101-200 之间有多少个素数，并输出所有素数**
例子一：

```
@echo off
setlocal enabledelayedexpansion
for /l %%i in (101,2 200) do (
   set tag=
   set /a c=%%i/2
   for /l %%j in (2,1,!c!) do (
      set /a b=%%i %% %%j
      if !b!==0 set tag=1)
   if not defined tag echo %%i &set /a n+=1
)
echo  共!n!个素数
pause>nul
```

例子二：

```
@echo off&setlocal EnableDelayedExpansion
for /l %%a in (101 2 200) do (
   set "flag="
   for /l %%b in (3 2 15) do (
      set /a m=%%a %% %%b
       if !m! equ 0 set flag=A
   )
   if not defined flag echo %%a
)
echo.&pause
```

程序十二：**约瑟夫问题，100 个人围成一圈，每数 3 个人就去掉一人，问最后剩下谁？**

```
@echo off&setlocal enabledelayedexpansion
for /l %%a in (100,-1,1) do set "kk= %%a !kk!"
:main
set "ie=0"
for %%a in (%kk%) do set /a ie+=1
if %ie% neq 1 call :st
echo %kk%
pause&exit
:st
for %%a in (%kk%) do (
            set /a num+=1
            if !num! equ 3 (
                        set "num=0"
                        set "kk=!kk: %%a =!
                  )
      )
goto main
```

程序十三：**判断回文数**

说明：形如 12321、19499491 都是回文数。

```
@echo off&setlocal EnableDelayedExpansion
set /p nnn=请输入一个数  &cls
set mmm=%nnn%
:loop
```

```
set bbb=%mmm:~0,1%%bbb%
set mmm=%mmm:~1%
if not "%mmm%"=="" goto loop
if "%bbb%"=="%nnn%" (echo %nnn% 是回文数) else echo %nnn% 不是回文数
pause>nul
```

程序十四：批处理计时器

方法一：用 **for** 语句截取时间参数计时：

```
-------------------------------------------------
@echo off
for /f "tokens=1 delims=:." %%i in ('echo %time%') do ( set a=%%i)
::截取小时参数
for /f "tokens=2 delims=:." %%i in ('echo %time%') do ( set b=%%i)
::截取分参数
for /f "tokens=3 delims=:." %%i in ('echo %time%') do (set c=%%i)
::截取秒参数
for /f "tokens=4 delims=:." %%i in ('echo %time%') do (set d=%%i &
echo 开始时间是%time%)
::截取毫秒参数
ping /n 20 127.1>nul
for /f "tokens=1 delims=:." %%i in ('echo %time%') do ( set
aa=%%i)
for /f "tokens=2 delims=:." %%i in ('echo %time%') do ( set bb=%%i)
for /f "tokens=3 delims=:." %%i in ('echo %time%') do (set cc=%%i)
for /f "tokens=4 delims=:." %%i in ('echo %time%') do (set dd=%%i
& echo 结束时间是%time%)
set /a t=360000*(aa-a)+6000*(bb-b)+100*(cc-c)+(dd-d)
::将时间全部转换成毫秒
set /a ta=t%%100
set /a tb=(t-ta)/100
echo.
echo 总共耗时%tb%.%ta%秒
echo.
set /a t1=t%%100
set /a t=(t-t1)/100
set /a t2=t%%60
set /a t=(t-t2)/60
set /a t3=t%%60
```

```
set /a t=(t-t3)/60
echo  总共耗时%t%时%t3%分%t2%.%t1%秒
pause>nul
```

方法二：用 **set** 语句截取时间参数计时：

```
----------------------------------------------
@echo off
set h=%time:~0,2%
::截取小时参数
set m=%time:~3,2%
::截取分参数
set s=%time:~6,2%
::截取秒参数
set hs=%time:~9,2%
::截取毫秒参数
echo  开始时间%time%
echo. & echo  现在执行程序，请耐心等待 . . .

ping /n 10 127.1>nul

set hh=%time:~0,2%
set mm=%time:~3,2%
set ss=%time:~6,2%
set hhss=%time:~9,2%
echo  结束时间%time%
set /a t=(hh-h)*360000+(mm-m)*6000+(ss-s)*100+(hhss-hs)
::将时间全部转换成毫秒
set /a t1=t%%100
set /a t=(t-t1)/100
echo. & echo  总共耗时%t%.%t1%秒
set /a t2=t%%60
set /a t=(t-t2)/60
set /a t3=t%%60
set /a t=t-t3
echo  总共耗时%t%时%t3%分%t2%.%t1%秒。
pause>nul
```

两者比较：用 **for** 语句比 **set** 语句复杂一点，计算机执行效率也不高，所以首选应该是 **set**。

程序十五：模拟进度条

```
@echo off
mode con cols=113 lines=15 &color 9c
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
```

程序十六：模拟 Win2000 开机进度条

```
@echo off
setlocal enabledelayedexpansion
set /a b=81
set
a=">>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>=====================
====================================================
======="
:back
cls
for /l %%i in (1,1,20) do @echo.
echo Loading Windows 2000...
set /a b-=1
echo !a:~%b%,80!
if !b! == 1 goto :eof
call :delay 80
goto back
:delay
set /a num=num + 1
if %num% geq %1 (set num=) && goto
:eof
goto delay
```

程序十七：输出字母表
在 **C** 语言里，我们可以很方便的输入 65 就输出 **ASCII** 字符 **A**，输入 **ASCII** 字符 **A**，就输出 65。
但在批处理里，我们无法办到。批处理编程有时候非常麻烦，连输出 **ASCII** 字符都是个挑战！

如果想输出字母表或 **ASCII** 表，可以用这几种办法：
方法一：

```
@echo off
setlocal enabledelayedexpansion
set S1=a
set S2=b
set S3=c
set S4=d
set S5=e
set S6=f
set S7=g
set S8=h
set S9=i
set S10=j
set S11=k
set S12=l
set S13=m
set S14=n
set S15=o
set S16=p
set S17=q
set S18=r
set S19=s
set S20=t
set S21=u
set S22=v
set S23=w
set S24=x
set S25=y
set S26=z
for /l %%i in (1,1,26) do ( echo !S%%i! )
pause
```

方法二：
我们先在 **D** 盘根目录建立一个 **word.txt**，里面的内容如下：
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
然后我们执行批处理：

```
@echo off
type D:\word.txt
pause
```

这个需要借助外部力量。

方法三：

```
@echo off
setlocal enabledelayedexpansion
set var= abcdefghijklmnopqrstuvwxyz
for /l %%i in (1,1,26) do echo !var:~%%i,1!
pause
```

方法四：

```
@echo off
for %%i in (a b c d e f g h i j k l m n o p q r s t u v w x y z) do echo
%%i
pause
```

方法五：

```
@echo off
set var=a b c d e f g h i j k l m n o p q r s t u v w x y z
for %%i in (%var%) do echo %%i
pause
```

从上面可以看出，方法四和方法五最简洁。

程序十八：输出数字方阵

例子一：

```
@echo off
setlocal enabledelayedexpansion
set/a  行=9,列=9,b=列-1
for /l %%a in (1,1,%行%) do (
	set/a n+=1,m=%%a*%%a,k=100+m&set/p =!k:~-2! <nul
	for /l %%b in (1,1,%b%) do set/a
"c=^!(%%b/n)","m+=^!c*(2*%%b-1)-1*c",k=100+m&set/p =!k:~-2!
<nul
	echo;)
pause>nul
```

运行结果：

```
01 02 05 10 17 26 37 50 65
04 03 06 11 18 27 38 51 66
09 08 07 12 19 28 39 52 67
16 15 14 13 20 29 40 53 68
25 24 23 22 21 30 41 54 69
36 35 34 33 32 31 42 55 70
49 48 47 46 45 44 43 56 71
64 63 62 61 60 59 58 57 72
81 80 79 78 77 76 75 74 73
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

例子二：

```
@echo off
setlocal enabledelayedexpansion
set /a w=h=9
for /l %%a in (1 1 %h%) do (
        for /l %%b in (1 1 %w%) do (
                set /a
"st*=^!^!~-%%b","n=^!(~-%%b/%%a)*(%%a*%%a-~-%%b+100)+^!^
!(~-%%b/%%a)*(%%a+%%b+st+99)","st+=%%b*2-2"
                set "str=!str!!n:~-2! "
        )
        echo !str!&set str=
)
pause>nul
```

运行结果显示：

```
01 02 05 10 17 26 37 50 65
04 03 06 11 18 27 38 51 66
09 08 07 12 19 28 39 52 67
16 15 14 13 20 29 40 53 68
25 24 23 22 21 30 41 54 69
36 35 34 33 32 31 42 55 70
49 48 47 46 45 44 43 56 71
64 63 62 61 60 59 58 57 72
81 80 79 78 77 76 75 74 73
```

++

例子三：

```
@echo off
set/a c=9
for /l %%a in (1 1 %c%) do (
    for /l %%b in (1 1 %c%) do set/a
"d=%%a+%%b-1,e=(%%a+%%b)%%2,f=e*((d*d-d)/2+%%a)+!e*((d*
d+d)/2-%%a+1)-!!(d/(c+1))*(d-c)*(d-c)+100"&call set/p=%%f:~-2%%
<nul
    echo;
)
pause>nul
```

运行结果显示：

```
01 02 06 07 15 16 28 29 45
03 05 08 14 17 27 30 44 46
04 09 13 18 26 31 43 47 60
10 12 19 25 32 42 48 59 61
11 20 24 33 41 49 58 62 71
21 23 34 40 50 57 63 70 72
22 35 39 51 56 64 69 73 78
36 38 52 55 65 68 74 77 79
37 53 54 66 67 75 76 80 81
```

++

例子四：

```
@echo off
set/a c=19
for /l %%i in (1 1 %c%)do echo;&for /l %%j in (1 1 %c%)do (
	set/a i=%%i,j=%%j,d=i+j-1
	set/a "f=(d*d-d)/2+j+(i+j)%%2*(i-j)-!!((d-1)/c)*(d-c)*(d-c)+1000"
	call set/p=%%f:~-3%% <nul
)
pause>nul
```

运行结果如下：

```
001 002 006 007 015 016 028 029 045 046 066 067 091 092 120 121 153 154 190
003 005 008 014 017 027 030 044 047 065 068 090 093 119 122 152 155 189 191
004 009 013 018 026 031 043 048 064 069 089 094 118 123 151 156 188 192 225
010 012 019 025 032 042 049 063 070 088 095 117 124 150 157 187 193 224 226
011 020 024 033 041 050 062 071 087 096 116 125 149 158 186 194 223 227 256
021 023 034 040 051 061 072 086 097 115 126 148 159 185 195 222 228 255 257
022 035 039 052 060 073 085 098 114 127 147 160 184 196 221 229 254 258 283
036 038 053 059 074 084 099 113 128 146 161 183 197 220 230 253 259 282 284
037 054 058 075 083 100 112 129 145 162 182 198 219 231 252 260 281 285 306
055 057 076 082 101 111 130 144 163 181 199 218 232 251 261 280 286 305 307
056 077 081 102 110 131 143 164 180 200 217 233 250 262 279 287 304 308 325
078 080 103 109 132 142 165 179 201 216 234 249 263 278 288 303 309 324 326
079 104 108 133 141 166 178 202 215 235 248 264 277 289 302 310 323 327 340
105 107 134 140 167 177 203 214 236 247 265 276 290 301 311 322 328 339 341
106 135 139 168 176 204 213 237 246 266 275 291 300 312 321 329 338 342 351
136 138 169 175 205 212 238 245 267 274 292 299 313 320 330 337 343 350 352
137 170 174 206 211 239 244 268 273 293 298 314 319 331 336 344 349 353 358
171 173 207 210 240 243 269 272 294 297 315 318 332 335 345 348 354 357 359
172 208 209 241 242 270 271 295 296 316 317 333 334 346 347 355 356 360 361
```

++

例子五：

```
@echo off
setlocal enabledelayedexpansion
set /p n=输入整数阶数 n(n^>0)：
set /a line=%n%-1,line2=%n%*2-1,x=1
for %%o in (%n% %line%) do (
	if %%o equ %n% (set "run=1 1 %n%") else (set
"run=%line% -1 1")
	for /l %%i in (!run!) do (
		set str=
		set /a mod=%%i%%2
		for /l %%j in (1 1 %%i) do (
			set /a sum+=1
			if !mod! equ 1 (
				set "str=!sum! !str!"
			) else (
				set "str=!str! !sum!"
			)
		)

		set y=1
		if %%o equ %line% set /a y+=%n%-%%i

		for %%j in (!str!) do (
			set ary[!x!][!y!]=%%j
			set /a y+=1
		)
		set /a x+=1
	)
)
for /l %%i in (1 1 %n%) do (
	for /l %%j in (1 1 %line2%) do (
		if defined ary[%%j][%%i] (
			set /p=!ary[%%j][%%i]!		<nul
			set /a count+=1
			set /a mod=!count!%%%n%
			if !mod! equ 0 echo.
		)
	)
)

pause>nul
```

运行结果如下：
输入 3
输出
1　　2　　6
3　　5　　7
4　　8　　9

程序十九：输出动态时间

```
@echo off
echo %time:~0,-3%
ping -n 2 127.1>nul&cls&%0
```

程序二十：数字时钟

```
@echo off & setlocal enabledelayedexpansion & mode con cols=54
lines=6 & color 0a
title=
for %%a in (4 1 2 1 2 1 4 2 1 2 1 2 1 2 1 2 4 2 5 2 6 2 4 2 5 1 2
1 4 2 1 2 5 2 3 2 8 2 4 1 7 2 1 2 1 2 1 2 5 1 5 1 8 1 4 2 4) do (
   set /a cc=~cc
   for /l %%i in (1,1,%%a) do (if !cc!==0 (set dgts=!dgts!    ) else (set
dgts=!dgts!■))
)
for /l %%z in (0 0 0) do (
   if "!time:~7,1!" neq "!sec!" (
      set "sec=!time:~7,1!" & set "oc="
      for /l %%h in (0,1,4) do (
         for %%d in (0 sp 1 sp : sp 3 sp 4 sp : sp 6 sp 7) do (
            if "%%d"==":" (set /a tt=%%h*5&if "!tt:~-1!"=="0" (set
oc=!oc!    ) else set oc=!oc!●) else (
               if "%%d"=="sp" (set oc=!oc!    ) else (
                  set "timeP=!time: =0!"&set /a
s=!timeP:~%%d,1!*15+%%h*3
                  for %%o in (!s!) do set "oc=!oc!!dgts:~%%o,3!"
))))&cls & set /p=!oc!<nul))
```

批处理专题研究

shutdown

关机命令

语法：

shutdown [-i | -l | -s | -r | -a] [-f] [-m \\computername] [-t xx] [-c "comment"] [-d up:xx:yy]

说明：一开始看不懂上面的格式说明，现在总算看懂了。

1、**shutdown** 标准语法格式总共有 6 对中括号[ ]，中括号里的内容是可选项目，可选可省略。

2、中括号里面的内容用“|”分隔，表示可选择的几个项，不可以同时选定。例如：上面第一个中括号里有“**/s**”和“**/a**”，这两个是分隔符中的两个项，我们如果这样写代码：

shutdown /s /a，执行无效，计算机不知道我们究竟是要关机还是取消关机。又如“**ON | OFF**”表示可以输入 **ON**，也可以输入 **OFF**，根据需要来选定，但就不能既 **ON** 又 **OFF**。

3、尖括号<>用来表明可选参数下的子参数。

4、大括号{}表示其中的项必须选一项。

5、省略号**...**指可以输入重复的项，知道需要的数目。例如 **path** 的语法格式为：**PATH [[drive:]path[;...][;%PATH%]**，表示关键字“**path**”后面可以带上多个路径，具体多少完全由用户决定。

6、中括号[]、分隔符和省略号只在说明命令格式时使用，在实际使用命令中是不出现的。例如：**shutdown** [/s] [/t 3600]是无效代码。应该写成 **shutdown /s /t** 3600。除了这三种符号在实际代码中不出现，其余在语法格式中的符号在实际使用中都要输入。包括：逗号、分号、等号、问号、冒号、斜杆和反斜杠等，另外，连空格都不能忘记！

7、参数项，每个参数一般由一个斜杠加一个字母组成，对命令起辅助作用。有的命令可以选用多个参数项。

8、啰嗦一下，中括号与中括号的命令，没有次序之分。例如：

shutdown /s /t 3600 **/c** "因为你的人品有问题，所以计算机准备关机"

shutdown /c "因为你的人品有问题，所以计算机准备关机" **/t** 3600 **/s**

两条代码是等价的，执行后，计算机会在一个小时后关机，并且显示“因为你的人品有问题，所以计算机准备关机”字样。

shutdown.exe 的扩展名可以省略，**shutdown.exe** 可以简写成 **shutdown**，例如：**shutdown.exe -s -t** 1000 等同于 **shutdown -s -t** 1000

shutdown 显示帮助，问号可以省略，但别的 **DOS** 命令就不一定可以省略了。即：**shutdown** 等同于 **shutdown** -?，但 **cls** 却不等同于 **cls** /?

斜杠"/"等价于横杠"-"，例如：**shutdown -s** 等同于 **shutdwon /s**

用法**: shutdown** [**-i** | **-l** | **-s** | **-r** | **-a**] [**-f**] [**-m \\computername**] [**-t xx**] [**-c "comment"**] [**-d up:xx:yy**]

没有参数　显示此消息(与 ？ 相同)

-i　显示 **GUI** 界面（远程关机对话框），必须是第一个选项。（**/i** 选项必须是键入的第一个参数，之后的所有参数都将被忽略。）

-l　立即注销当前用户，没有超时期限(不能与选项 **-m** 一起使用)

-s　关闭计算机

-r　重启计算机

-a　取消关机（仅在超时期限内有效）。**a** 参数仅可以与 **/m \\ComputerName** 一起使用。

-m \\computername　远程计算机关机/重启动/放弃。不能与 **/l** 选项一同使用。

-t xx　设置关闭前的超时为 **xxx** 秒。有效范围是 0-315360000 (10 年)，默认值为 30 秒。

-c "comment"　重启动或关闭的原因的注释。最多允许 512 个字符。格式：**/c** 后面要加空格，注释内容需要用双引号引起来，双引号可以是中文双引号，也可以用英文双引号。

-f　强制关闭正在运行的应用程序而不提前警告用户。(使用 **/f** 选项可能导致未保存的数据丢失)

-d [u][p]:xx:yy　关闭原因代码

u 是用户代码

p 是一个计划的关闭代码

xx 是一个主要原因代码(小于 256 的正整数)

yy 是一个次要原因代码(小于 65536 的正整数)

shutdown 与 **at** 命令配合使用来定时关机，会更加的精确。
格式：**at** 关机时间 **shutdown** 选项
例如：
at 23:00 **Shutdown -s**　　到 23:00 时候倒计时 30 秒（没有特别指明时间就默认为 30 秒）关机。
at 12:45 **shutdown -s -t** 20 让机子在 12:45 关机，关机前倒计时 20 秒。

注意：最好每次关机都加上/**f**，以免出现对话框而无法关机的现象。

shutdown /s /t 800　　系统倒计时 800 秒关机。

shutdown -s -f -t 60 **-c "April Fools"**
系统倒计时 60 秒关机，关机时强制关闭所有正在运行程序（使用/**f** 就是防止人不在的时候，由于某些程序正在运行而系统关机出现关机对话框，无法正常关机），并显示关机信息“**April Fool's**”。
注意次序，参照标准格式，**-s** 必须在**-f** 之前，**-t** 必须在**-f** 之后，**-c** 在所有命令之后。

未验证示例
要强制让应用程序在一分钟延迟后关闭并重新打开本地计算机，并注明原因是“应用程序：维护（计划内）”，注释内容为“重新配置 **myapp.exe**”，请键入：
shutdown /r /t 60 **/c "Reconfiguring myapp.exe" /f /d p:4:**1

要使用相同的参数重新启动远程计算机 **\\ServerName**，请键入：
shutdown /r /m \\servername /t 60 **/c "Reconfiguring myapp.exe" /f /d p:4:**1

程序例子，将关机功能集成到一个程序里：
@echo off&setlocal enabledelayedexpansion
@mode con cols=64 **lines**=20**&color** 9**f**
title 关机程序　　作者 **MHL QQ**1208980380
echo.&echo.&echo　　关机程序
&echo.
echo 当前日期　　**%date%**
echo 当前时间　　**%time:~0,-3%**
echo.&echo 本程序在新建计划任务时，有些敏感的杀毒软件会拦截，放行即可。

```
for /l %%i in (1,1,64) do set /p var=-<nul
echo.&echo  请选择:
echo A（定时关机）B（倒计时关机）C（重启计算机）D（取消倒计时关机）
SET /P a=
if /I "!a!"=="A" (
    SET /P time1=请输入关机时间（24 小时制,例如 23:59）:
    at "!time1!" shutdown -s -f
) else if /I "!a!"=="B" (
    SET /P time2=请输入倒计时关机时间（单位秒）:
    shutdown -s -f -t "!time2!"
) else if /I "!a!"=="C" (
    echo.
    SET /P b=请选择    A（定时重启计算机）B（倒计时重启计算机）：
    if /I "!b!"=="A" (
        echo.
        SET /P time3=请输入定时重启时间（24 小时制,例如 23:59）:
        at "!time!%" shutdown -r -f
    ) else if /I "!b!"=="B" (
        echo.
        SET /P time4=请输入倒计时重启时间（单位秒）:
        shutdown -r -f -t "!time4!"
    )
) else if /I "!a!"=="D" shutdown /a
```

用批处理加密文件

```
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
```

```
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a
@echo off
==========================================================
```

上面就是加密代码

加密方法：将上面分割线以上的代码复制到需要加密的 txt 文本开头就可以了，打开就是乱码。

原理:

记事本程序在保存一篇新建的文档时，如果没有指定编码类型，会使用缺省的 ANSI 类型（对于中文版来说，对应的就是 GB 码）。

而在打开一篇已创建的文档时，它会分析文档的编码类型，它首先判断文档头部有无 BOM（Byte Order Mark，字节序标记，长度为(2 - 3 字节)，如有则根据其内容判断编码类型，FF、FE（Unicode）,FE、FF（Unicode big endian）,EF、BB、BF（UTF-8)

因为事实上有很多非 ANSI 编码的文档是没有任何 BOM 的“纯文本”，所以对这些文档不能简单的判断为 ANSI 编码。而需要使用一系列的统计学算法根据文档内容来猜测文档编码。记事本使用了 IsTextUnicode 函数来判断是否为 Unicode/Unicode big endian 编码，使用 IsTextUTF8 判断是否为 UTF8 编码。但既然是统计学算法，就难免存在误判，尤其在文档内容过短时，由于样本的容量太小，这种误判的概率会显著增大。比如那个有名的微软与联通有仇的笑话，就是记事本在打开只有"联通"二字的 ANSI 编码文档时，IsTextUTF8 函数将其误判为 UTF8 编码[2]；同样的误判也发生在 IsTextUnicode 函数上，比如具有 “this app can break”这种具有 4335 结构的文档，会被误判为 Unicode 编码[3][4]。

需要说明的是，这种误判的可能性是建立在文本较短且其字节位特征不被干扰的前提上的。如果将上述的文本做稍许修改（即使只是增加一个回车），则误判很难再发生。
而这种方法的特殊性在于，它的字节串不但具有 **Unicode** 特征，而且很长达到了 1288 字节，也就是说它的 **Unicode** 特征性很强，所以可以抵抗一 些较短的不具有 **Unicode** 特征串的干扰，这是由统计学的规律所决定的。但是在干扰串稍长时，**Unicode** 的特征将会受到显著干扰，直至被 **IsTextUnicode** 函数认定为非 **Unicode**。所以，有些朋友总是无法测试成功，应该是与附加的批处理代码长度和内容相关。因为其他的编辑器（比如 **Word / Wordpad / EditPlus / UltraEdit**）使用了更新的编码类型判断算法，所以在 **Unicode** 判断上改进了不少，而 **UTF8** 的判断仍然不尽如人意。但因为理论上来说完全准确地算法并不存在，所以我们只能依靠避免使用无 **BOM** 的非 **ANSI** 文档，或者打开文档时手动指定编码类型。

另外，如果使用记事本保存了这些误判了编码类型的文件，则将难以恢复。如果使用误判编码保存，则将给原文档加上 **BOM** 标记，则使用其他编辑器也再无法 观察到原文档。如果使用 **ANSI** 编码保存，则原文档将会被当作 **Unicode** 文档而被转换，还原的可能性接近于零。

问：这里究竟有多少个**%%a** 才可以进行加密？
答案是：没有规定多少个，但不应该少于 150 个，否则加密就可能不成功，至于上限，没测试过，不知道有没有上限。

注意三点：
1、第一个**%%a** 前面和最后一个**%%a** 后面不可以有空格
2、相邻两个**%%a** 之间用空格隔开
3、在最后一个**%%a** 后面换行，然后就是**@echo off**。请注意这最后一行的**@echo off**，一个字符都不可以少，而且**@echo off** 后面不可以加任何字符！
4、别以为到了**@echo off** 以后就什么字符都可以加密了，不是这样的！上面的代码**@echo off** 换行以后，第一行不可以带标点符号、数字，但可以是回车！
5、注意，如果某份 **txt** 文件是用这个方法加密过一次的，那么我们将原来里面的乱码都删除掉，再用这些代码加密一些内容是无法加密的，必须重新建立新的 **txt** 文档后才能加密。

解密方法：

1、用 **word** 打开文件，（选择 **MS-DOS** 编码）。

2、将加密的 **txt** 压缩成 **rar** 格式，然后右击此 **rar** 文件，打开方式选择用“**Winrar** 压缩文件管理器”打开，然后点击工具栏的“查看”按钮，就可以查看文件的原来内容。

例子一：

```
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a
@echo off
清平调
云想衣裳花想容，　春风拂槛露华浓。
若非群玉山头见，　会向瑶台月下逢。
```

例子二：

```
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a
@echo off
```

清平调

云想衣裳花想容， 春风拂槛露华浓。

若非群玉山头见， 会向瑶台月下逢。

例子三：

```
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
```

```
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a
@echo off
a
```

3

上面三个例子都是可以加密的。下面例子就不行了：

例子 1：

```
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
```

```
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a
@echo off
,
```

清平调

云想衣裳花想容， 春风拂槛露华浓。

若非群玉山头见， 会向瑶台月下逢。

不能加密的原因：**@echo off** 后面紧跟着是标点符号逗号。

例子 2：

```
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
```

%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a
@echo off
3.1415926
不能加密的原因是：**@echo off** 后面跟着数字。

例子 3：
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a
@echo off

2.71828
不能加密的原因是：**@echo off** 后面虽然跟着回车，但回车后跟着数字，仍旧不行。

例子 4：

```
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a %%a
%%a
@echo off
a
```

2.71828

批处理输出 ASCII 字符

批处理输出 **ASCII** 字符非常困难。

一般情况下，我们可以在 **cmd** 窗口，按住 **alt** 键，然后在小键盘输入相应的 **ASCII** 数字来输出相应的 **ASCII** 字符。

例如：

alt+65 结果输出字符“**A**”。

我们也可以这么输出 **ASCII** 字符：
先在命令行输入“**echo** ”，注意 **echo** 后面有一个空格。然后按住 **ctrl** 不放，再按住 **A**，屏幕上会显示“**echo** ^**A**”，但这个^可不是转义字符^，这一点千万要注意。按下回车就会显示一个笑脸的 **ASCII** 字符。

用上面的方法，对某些字符还是无法输出，例如退格符号的 **ASCII** 字符。这个字符很重要，我们在很多情况下会用到它。

输出这些用常规方法无法输出的字符，就需要下面这么来操作了：
首先进入 **CMD**，然后输入 **EDIT** 进入编码编辑窗口，然后按 **CTRL+P** 来允许非常特殊的字符输入，然后按住 **ctrl+H** 或者 **alt**+8（小键盘的 8），就能得到一个退格符了，将其保存到 **txt** 文本后就可以使用啦。

::纯批处理很难输出 **ASCII** 字符，下面是借助了汇编。

```
@echo off
(echo A100
echo MOV CX,0100
echo MOV DL,00
echo MOV AH,02
echo INT 21
echo INC DL
echo LOOP 0105
echo INT 20
echo.
echo G
echo q)>temp.txt
debug<temp.txt>tem.txt
for /f "delims=" %%a in ('findstr "#" tem.txt') do echo %%a
del /q tem*.txt&pause>nul
```

解除 IE 文件夹伪装成回收站

```
C:
cd\
cd    program files\internet explorer
attrib desktop.ini -s -h
del desktop.ini
```

判断一串字符串是否是数字

```
@echo off
set /p a=请输入：
echo %a%|findstr /be "[0-9]*" >nul &&echo  是数字||echo  非数字
&echo. & %0
pause > nul
```

注册表编辑

注册表根键（**Root Key**）：
是指在注册表编辑器左侧以“**HKEY**”作为名称前缀的位置，因其位于注册表最项层被称为根键，如 **Windows XP** 的注册表中包括五大根键，分别为
HKEY_CLASSES_ROOT
HKEY_CURRENT_USER
HKEY_LOCAL_MACHINE
HKEY_USERS
HKEY_CURRENT_CONFIG
这些根键都可缩写，分别是 **HKCR**、 **HKCU**、 **HKLM**、 **HKU**、**HKCC**。
“**HKEY_LOCAL_MACHINE\Software\Microsoft\Windows**”可以简写成“**HKLM\Software\Microsoft\Windows**”

注册表编辑关键字有以下十一个：
1、**REG ADD**　　将新的子项或项添加到注册表中
2、**REG COMPARE**　　比较指定的注册表子项或项
3、**REG COPY**　　复制注册表项到本地或远程计算机的指定位置
4、**REG DELETE**　　删除注册表项或子项
5、**REG EXPORT**　　导出本地计算机注册表项、子项和值
6、**REG IMPORT**
7、**REG LOAD**
8、**REG QUERY**
9、**REG RESTORE**
10、**REG SAVE**
11、**REG UNLOAD**

REG ADD

将新的子项或项添加到注册表中
如果添加的项的路径和项名全都相同，那么就会更改项的数据类型和数据为指定的数据类型和数据。利用这个就可以修改原来注册表项的数据类型和键值了。

语法：
REG ADD KeyName [/v ValueName | /ve] [/t Type] [/s Separator] [/d Data] [/f]

keyname
指定要添加的子项或项的完整路径。要指定远程计算机，请包括计算机名(以**ComputerName**\格式表示)，并将其作为 **keyname** 的一部分。省略**Computername**\则默认为对本地计算机的操作。**keyname** 必须包括一个有效的根键。有效根键包括：[**HKLM | HKCU | HKCR | HKU | HKCC**]。如果指定了远程计算机，则有效根键是 **HKLM** 和 **HKU**。
注意：如果路径中含有空格，整条路径都要用双引号括起来，否则计算机无法识别项。
有一个细微的细节，就是双引号和路径之间不允许有空格！

例如：
"HKEY_LOCAL_MACHINE\Software\Google\No Toolbar Offer Until"
因为“**No Toolbar Offer Until**”含有空格，所以整条路径都必须用双引号括起来，而且双引号和路径之间没有空格。

/v ValueName
　　指定要添加到指定子项下的注册表项名称。
注意：这个和 **keyname** 一样，假如项名含有空格，也必须用双引号括起来。同样，双引号和项名之间不允许有空格。
例如：
"PPStream Inc."

/ve
　　指定添加到注册表中的注册表项为空值，建立的名称为“**(**默认**)**”。**/ve** 后面就不可以添加名称了。例如：**/ve nat** 这样写是不允许的。
如果后面没有指定类型和数据值，那么默认为“**REG_SZ**”和“空值”。当然，我们用了**/ve** 后，仍然可以指定数据类型和数据值。
例如：
reg add "HKLM\Software\Google\No Toolbar Offer Until" /ve /t REG_DWORD /d 0 **/f**

/t Type
指定注册表项的数据类型。**Type** 必须是以下几种类型之一：
　　REG_SZ
　　REG_MULTI_SZ
　　REG_DWORD_BIG_ENDIAN

REG_DWORD
REG_BINARY
REG_DWORD_LITTLE_ENDIAN
REG_LINK
REG_FULL_RESOURCE_DESCRIPTOR
REG_EXPAND_SZ

没有用**/t** 指明数据类型，那么默认的数据类型为 **REG_SZ** 。

/s Separator

当指定了 **REG_MULTI_SZ** 数据类型并且需要列出多个项时，指定用来分隔数据的多个实例的字符。如果没有指定，将使用默认分隔符“\0”。

/d Data

指定新注册表项的键值数据。

默认情况下输入的数据是十进制，0**x**123 这种形式为十六进制。假如没有**/d** 项给定数据，那么该项默认数据为空。

例如：

/d 24　　没有前缀，数据位十进制中的 24，等于十六进制中的 18。

/d 0**x**16　有十六进制前缀，数据为十六进制中的 16，等于十进制中的 22。

/d 7b　　这个没有指定数据类型，不可以这样写，程序运行错误。尽管我们可以知道它是十六进制，等于十进制的 123，但没有前缀，计算机无法识别。

/f

表示不用提示就强行改写现有注册表项。 没有**/f**，程序会提示输入 **Y/N** 确认操作。

程序例子：

下面的代码操作，使得有隐藏属性的文件彻底被隐藏。隐藏后，我们无法通过“工具-->文件夹选项-->查看-->显示所有的文件和文件夹”来显示隐藏文件：

reg add HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced\Folder\Hidden\SHOWALL /v CheckedValue /t REG_DWORD /d 0 /f

因为路径和项名都没不含有空格，所以可以不用添加任何双引号。如果想添加，也是可以的：

reg add "HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced\Folder\Hidden\SHOWALL" /v CheckedValue /t REG_DWORD /d 0 /f

注释：

reg add　　添加注册表项的关键字，照写就是**"HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced\Folder\Hidden\SHOWALL"**这一条是路径

/v CheckedValue　　这个**/v** 后紧跟着要添加的注册表项名称

/t REG_DWORD　　这个**/t** 后紧跟着 **RegKey** 数据类型

/d 0　　这个**/d** 后就是键值

/f　　表示不用提示就强行改写现有注册表项

再看一个例子：

reg add "HKLM\Software\Google\No Toolbar Offer Until" /v "PPStream Inc." /t REG_DWORD /d 0x1335045

这里路径和项都含有空格，所以各个必须用双引号引起来。**/d** 后面的数据有 **0x** 前缀，表示该数据为十六进制数据。不含**/f** 开关，那么程序会提示我们输入 **Y/N** 确认操作。

下面是帮助信息里的例子：

REG ADD \\ABC\HKLM\Software\MyCo

添加远程机器 **ABC** 上的一个注册表项 **HKLM\Software\MyCo**

REG ADD HKLM\Software\MyCo /v Data /t REG_BINARY /d fe340**ead**

添加一个值(名称**: Data**，类型**: REG_BINARY**，数据**: fe**340**ead)**

REG ADD HKLM\Software\MyCo /v MRU /t REG_MULTI_SZ /d fax0**mail**

添加一个值(名称**: MRU**，类型**: REG_MUTLI_SZ**，数据**: fax**0**mail**0\0**)**

REG ADD HKLM\Software\MyCo /v Path /t REG_EXPAND_SZ /d %%systemroot%%

添加一个值(名称**: Path**，类型**: REG_EXPAND_SZ**，数据**: %systemroot%)**

注意：　在扩充字符串中使用双百分比符号**(%%)**

这里，**%systemroot%**不是我们批处理的变量**%var%**，不要以为计算机会用什么数据代替，注册表里的数据就是**“%systemroot%”**。

++

REG COMPARE

比较指定的注册表子项或项

语法

REG COMPARE KeyName1 KeyName2 [/v ValueName | /ve] [/oa | /od | /os | on] [/s]

参数

KeyName1

指定要比较的第一个子项的完整路径。要指定远程计算机，请包括计算机名（以 **\\ComputerName** 格式表示），并将其作为 **KeyName** 的一部分。省略 **\\ComputerName** 会导致默认对本地计算机的操作。**KeyName** 必须包括一个有效的根键。有效根键包括 **HKLM**、**HKCU**、**HKCR**、**HKU** 以及 **HKCC**。如果指定了远程计算机，则有效根键是 **HKLM** 和 **HKU**。

KeyName2

指定要比较的第二个子项的完整路径。要指定远程计算机，请包括计算机名（以 **\\ComputerName** 格式表示），并将其作为 **KeyName** 的一部分。省略 **\\ComputerName** 会导致默认对本地计算机的操作。只在 **KeyName2** 中指定计算机名会导致该操作使用到 **KeyName1** 中指定的子项的路径。**KeyName** 必须包括一个有效的根键。有效根键包括 **HKLM**、**HKCU**、**HKCR**、**HKU** 以及 **HKCC**。如果指定了远程计算机，则有效根键是 **HKLM** 和 **HKU**。

注意：**keyname** 中含有空格，整个 **keyname** 都要用双引号引起来，并且双引号与 **keyname** 之间不可以存在空格。

/v ValueName

指定要比较的子项下的值名称。如果名称含有空格，要用双引号引起来，并且双引号与名之间不可以存在空格。

/ve

比较空白值**<no name>**名称的值。

[/oa | /od | /os | on]

指定如何显示比较操作的结果。默认设置是 **/od**，每个选项注释如下：

/oa

显示所有不同和匹配结果。

/od

只显示不同的结果。这是默认操作。

/os
只显示匹配结果。
/on
不显示结果。
/s
比较所有子项和值。

返回代码：
0　比较成功且结果相同。
1　比较失败。
2　比较成功并找到不同点。
下表列出了结果中显示的符号。
=
KeyName1 数据等于 **KeyName**2 数据
<
KeyName1 数据小于 **KeyName**2 数据
>
KeyName1 数据大于 **KeyName**2 数据

程序例子：

```
@echo off
mode con cols=160 lines=20
reg compare "HKLM\SOFTWARE\Adobe\Acrobat Distiller\8.0\Installer" "HKLM\SOFTWARE\Adobe\Acrobat Elements\8.0\Installer"
pause
```

上面两个路径都含有空格，所以都要用双引号引起来。代码运行结果如下：

```
< Value:    HKLM\SOFTWARE\Adobe\Acrobat Distiller\8.0\Installer
InstallDate    REG_SZ    2013-7-27
> Value:    HKLM\SOFTWARE\Adobe\Acrobat Elements\8.0\Installer
InstallDate    REG_SZ
< Value:    HKLM\SOFTWARE\Adobe\Acrobat Distiller\8.0\Installer
InstallTime    REG_SZ    08:59:25
> Value:    HKLM\SOFTWARE\Adobe\Acrobat Elements\8.0\Installer
InstallTime    REG_SZ
< Value:    HKLM\SOFTWARE\Adobe\Acrobat Distiller\8.0\Installer
ReinstallMode    REG_SZ    omus
< Value:    HKLM\SOFTWARE\Adobe\Acrobat Distiller\8.0\Installer
CHS_GUID    REG_SZ    {AC76BA86-2052-0000-7760-000000000003}

Result Compared:    Different
```

操作成功结束
请按任意键继续. . .

程序例子二：

```
@echo off
mode con cols=160 lines=20
reg compare "HKLM\SOFTWARE\Adobe\Acrobat Distiller\8.0\Installer"  "HKLM\SOFTWARE\Adobe\Acrobat Elements\8.0\Installer" /v Installdate
pause
```

运行结果如下：

```
< Value:  HKLM\SOFTWARE\Adobe\Acrobat Distiller\8.0\Installer  Installdate  REG_SZ  2013-7-27
> Value:  HKLM\SOFTWARE\Adobe\Acrobat Elements\8.0\Installer  Installdate  REG_SZ

Result Compared:  Different
```

操作成功结束
请按任意键继续. . .

程序例子三：

```
@echo off
mode con cols=160 lines=20
reg compare "HKLM\SOFTWARE\Adobe\Acrobat Distiller\8.0\Installer"  "HKLM\SOFTWARE\Adobe\Acrobat Elements\8.0\Installer" /oa
pause
```

运行结果：

```
= Value:  HKEY_LOCAL_MACHINE\SOFTWARE\Adobe\Acrobat Distiller\8.0\Installer  ENU_GUID  REG_SZ  {AC76BA86-2052-0000-7760-000000000003}
< Value:  HKEY_LOCAL_MACHINE\SOFTWARE\Adobe\Acrobat Distiller\8.0\Installer  InstallDate  REG_SZ  2013-7-27
> Value:  HKEY_LOCAL_MACHINE\SOFTWARE\Adobe\Acrobat Elements\8.0\Installer  InstallDate  REG_SZ
< Value:  HKEY_LOCAL_MACHINE\SOFTWARE\Adobe\Acrobat Distiller\8.0\Installer  InstallTime  REG_SZ  08:59:25
> Value:  HKEY_LOCAL_MACHINE\SOFTWARE\Adobe\Acrobat Elements\8.0\Installer  InstallTime  REG_SZ
< Value:  HKEY_LOCAL_MACHINE\SOFTWARE\Adobe\Acrobat Distiller\8.0\Installer  ReinstallMode  REG_SZ  omus
< Value:  HKEY_LOCAL_MACHINE\SOFTWARE\Adobe\Acrobat Distiller\8.0\Installer  CHS_GUID  REG_SZ
```

{**AC**76**BA**86-2052-0000-7760-000000000003}

Result Compared:　Different

操作成功结束
请按任意键继续. . .

下面是帮助信息上的例子（不同的计算机，也许没有这些注册项）

要将 **MyApp** 项下的所有值与 **SaveMyApp** 项下的所有值进行比较，请键入：

REG COMPARE HKLM\Software\MyCo\MyApp HKLM\Software\MyCo\SaveMyApp

要比较 **MyCo** 项下的 **Version** 的值和 **MyCo**1 项下的 **Version** 的值，请键入：

REG COMPARE HKLM\Software\MyCo HKLM\Software\MyCo1 **/v Version**

要将计算机 **ZODIAC** 上 **HKLM\Software\MyCo** 下的所有子项和值与当前计算机上 **HKLM\Software\MyCo** 下的所有子项和值进行比较，请键入：**REG COMPARE \\ZODIAC\HKLM\Software\MyCo** \\。**/s**

++

REG COPY

将一个注册表项复制到本地或远程计算机的指定位置。

语法：

REG COPY KeyName1 **KeyName**2 **[/s] [/f]**

将 **KyeName**1 复制到 **KeyName**2

参数

KeyName1

指定要复制子项的完整路径。要指定远程计算机，请包括计算机名（以 **\\ComputerName** 格式表示），并将其作为 **KeyName** 的一部分。省略

\\ComputerName 会导致默认对本地计算机的操作。**KeyName** 必须包括一个有效的根键。有效根键包括 **HKLM**、**HKCU**、**HKCR**、**HKU** 以及 **HKCC**。

如果指定了远程计算机，则有效根键是 **HKLM** 和 **HKU**。

KeyName2

指定子项目的地的完整路径。要指定远程计算机，请包括计算机名（以 **\\ComputerName** 格式表示），并将其作为 **KeyName** 的一部分。省略**\\ComputerName** 会导致默认对本地计算机的操作。**KeyName** 必须包括一个有效的根键。有效根键包括 **HKLM**、**HKCU**、**HKCR**、**HKU** 以及 **HKCC**。如果指定了远程计算机，则有效根键是 **HKLM** 和 **HKU**。

/s　　复制所有子项和值
/f　　不用提示就强行复制**(**我实际操作的结果是无论有无**/f**，计算机都没有出现对话框，直接就复制过去了**)**

程序例子：

```
@echo off
reg copy "HKLM\SOFTWARE\Adobe\Acrobat Distiller\8.0"
"HKLM\SOFTWARE\abc" /s
pause
```

帮助信息的例子**:**

REG COPY HKLM\Software\MyCo\MyApp HKLM\Software\MyCo\SaveMyApp /s

将注册表项 **MyApp** 下的所有子项和值复制到注册表项 **SaveMyApp**

REG COPY \\ZODIAC\HKLM\Software\MyCo HKLM\Software\MyCo1

将 **ZODIAC** 上注册表项 **MyCo** 下的所有值复制到当前机器上的注册表项 **MyCo1**

++

REG DELETE

删除注册表项或子项。

语法：

REG DELETE KeyName [/v ValueName | /ve | /va] [/f]

参数

KeyName

指定要删除的子项或项的完整路径。要指定远程计算机，请包括计算机名（以 **\\ComputerName** 格式表示），并将其作为 **KeyName** 的一部分。省略 **\\ComputerName** 会导致默认对本地计算机的操作。**KeyName** 必须包括一个有效的根键。有效根键包括 **HKLM**、**HKCU**、**HKCR**、**HKU** 以及 **HKCC**。如果指定了远程计算机，则有效根键是 **HKLM** 和 **HKU**。

/v ValueName

删除子项下的特定项。如果未指定项，则将删除子项下的所有项和子项。

/ve

指定只可以删除为空值的项。

/va

删除指定子项下的所有项。使用本参数不能删除指定子项下的子项。

/f

无需请求确认而删除现有的注册表子项或项。

下表列出了 **reg delete** 操作的返回值。

0　成功

1　失败

程序例子：

reg delete　"HKLM\SOFTWARE\abc" /f

运行上面代码，会将 **abc** 项及其所有子项不经确认就直接删除。

reg delete "HKLM\SOFTWARE\abc\Language" /v UI /f

运行上面代码，会将 **Language** 项中键名为 **UI** 的键删除。

下面是帮助信息里的例子：

REG DELETE HKLM\Software\MyCo\MyApp\Timeout

删除注册表项 **Timeout** 以及其所有子项和值

REG DELETE \\ZODIAC\HKLM\Software\MyCo /v MTU

删除计算机 **ZODIAC** 上 **HKLM\Software\MyCo** 下的注册表值 **MTU**

++

REG EXPORT

导出本地计算机注册表项、子项和值。

语法

REG EXPORT KeyName FileName

参数

KeyName

指定项的完全路径。**Export** 操作仅可在本地计算机上工作。**KeyName** 必须包括一个有效的根键。有效根键包括 **HKLM**、**HKCU**、**HKCR**、**HKU** 以及 **HKCC**。

FileName

指定在操作期间创建的文件的名称和路径。该文件必须具有 **.reg** 扩展名。

程序例子：

reg export "HKLM\SOFTWARE\abc" "D:\123.reg"

运行上面代码，会将 **HKLM\SOFTWARE\abc** 项及其子项全部导出到 **D** 盘 **123.reg** 文件中。如果我们接下来不小心将“**HKLM\SOFTWARE\abc**”删除了，不要紧，双击 **123.reg**，就又恢复“**HKLM\SOFTWARE\abc**”及其子项的所有内容了。“123”是导出的 **reg** 文件名字，我们将其恢复到注册表里，项的名字肯定不会是“123”。

帮助信息的例子:

REG EXPORT HKLM\Software\MyCo\MyApp AppBkUp.reg

将注册表项 **MyApp** 的所有子项和值导出到文件 **AppBkUp.reg**

++

REG IMPORT

将包含已导出的注册表子项、项和值的文件的内容复制到本地计算机的注册表中。 和 **REG EXPORT** 是互逆操作。

语法

REG IMPORT FileName

参数

FileName （只适用于本地计算机）

指定其内容将复制到本地计算机注册表中的文件的名称和路径。此文件可以使用 **reg export** 创建 或者自己手动预先创建。

下表列出了 **reg import** 操作的返回值。

0　成功

1　失败

例如：

reg import "D:\123.reg"

运行上面代码和双击 123**.reg** 文件的效果一样，都是将 123**.reg** 导入到注册表里，区别只是一个是批处理操作，另一个是手工操作。

++

REG LOAD

将保存的子项和项写回到注册表的不同子项中。与用于进行疑难解答或编辑注册表项的临时文件一起使用。

语法

REG LOAD KeyName FileName

参数

KeyName

指定要加载的子项的完整路径。要指定远程计算机，请包括计算机名（以 **\\ComputerName** 格式表示），并将其作为 **KeyName** 的一部分。省略 **\\ComputerName** 会导致默认对本地计算机的操作。**KeyName** 必须包括一个有效的根键。有效根键包括 **HKLM**、**HKCU**、**HKCR**、**HKU** 以及 **HKCC**。如果指定了远程计算机，则有效根键是 **HKLM** 和 **HKU**。

FileName

指定要加载的文件的名称和路径。必须使用 **REG SAVE** 来预先创建这个文件，扩展名为 **.hiv** 。

/?

在命令提示符处显示 **reg load** 的帮助。

注释

? 下表列出了 **reg load** 操作的返回值。

值 描述
0　成功
1　失败
示例
要，请键入：
REG LOAD HKLM\TempHive TempHive.hiv
　　将名为 **TempHive.hiv** 的文件加载到 **HKLM\TempHive** 项

++

REG QUERY

返回位于注册表中指定的子项下的下一层子项和项的列表。

语法
REG QUERY KeyName [/v ValueName | /ve] [/s]
这个类似于 **DOS** 下对文件夹进行 **dir** 操作，也就是查找下一级目录。利用这个命令，我们可以获得注册表项、子项以及键值。

参数
　　KeyName
　　指定子项的完全路径。要指定远程计算机，请包括计算机名（以 **\\ComputerName** 格式表示），并将其作为 **KeyName** 的一部分。省略**\\ComputerName** 会导致默认对本地计算机的操作。**KeyName** 必须包括一个有效的根键。有效根键包括 **HKLM**、**HKCU**、**HKCR**、**HKU** 以及 **HKCC**。如果指定了远程计算机，则有效根键是 **HKLM** 和 **HKU**。
　　/v ValueName
　　指定要查询的注册表值名称。如果省略，则返回 **KeyName** 的所有值名称。
　　/ve
　　查询默认值或空白值名称**<no name>**
　　/s
　　查询所有子项和值。

　　reg query 操作的返回值。
　　0　成功
　　1　失败

程序例子

```
@echo off
reg Query
HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced\Folder\Hidden\SHOWALL /v CheckedValue
pause
```

运行结果显示：

```
! REG.EXE VERSION 3.0

HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced\Folder\Hidden\SHOWALL
    CheckedValue    REG_DWORD    0x1

请按任意键继续. . .
```

下面是帮助信息里的例子：

要显示 **HKLM\Software\Microsoft\ResKit** 项中的名称值 **Version** 的值，请键入：

REG QUERY HKLM\Software\Microsoft\ResKit /v Version

要显示远程计算机 **ABC** 上的 **HKLM\Software\Microsoft\ResKit\Nt\Setup** 项下的所有子项和值，请键入：

REG QUERY \\ABC\HKLM\Software\Microsoft\ResKit\Nt\Setup /s

++

REG RESTORE

将保存的子项和项写回到注册表。

语法

REG RESTORE KeyName FileName

用途：编辑任何注册表项之前，请使用 **reg save** 操作保存父子项。如果编辑失败，则可以使用 **reg restore** 操作还原原来的子项。

参数

KeyName

指定要还原的子项的完整路径。**Restore** 操作仅在本地计算机上工作。**KeyName** 必须包括一个有效的根键。有效根键包括 **HKLM**、

HKCU、HKCR、HKU 以及 **HKCC**。

FileName

指定其内容将写回到注册表中的文件的名称和路径。必须使用 **REG SAVE** 来预先创建这个文件，扩展名为 **.hiv** 。

reg restore 操作的返回值。

0　成功

1　失败

程序例子：

要将名为 **NTRKBkUp.hiv** 的文件还原到 **HKLM\Software\Microsoft\ResKit** 项，并覆盖该项的现有内容，请键入：

REG RESTORE HKLM\Software\Microsoft\ResKit NTRKBkUp.hiv

++

REG SAVE

将指定的子项、项和注册表值的副本保存到指定文件中。

语法

REG SAVE KeyName FileName

用途：编辑任何注册表项之前，请使用 **reg save** 操作保存父子项。如果编辑失败，则可以使用 **reg restore** 操作还原原来的子项。

使用 **REG LOAD** 之前也必须用 **REG SAVE** 预先建立文件。

参数

KeyName

指定子项的完全路径。要指定远程计算机，请包括计算机名（以 **\\ComputerName** 格式表示），并将其作为 **KeyName** 的一部分。省略**\\ComputerName** 会导致默认对本地计算机的操作。**KeyName** 必须包括一个有效的根键。有效根键包括 **HKLM、HKCU、HKCR、HKU** 以及 **HKCC**。如果指定了远程计算机，则有效根键是 **HKLM** 和 **HKU**。

FileName

指定所创建的文件的名称和路径。如果未指定路径，则默认当前路径。

reg save 操作的返回值。

0　成功

1　失败

示例

要将配置单元 **MyApp** 作为名为 **AppBkUp.hiv** 的文件保存到当前文件夹中，请键入：

REG SAVE HKLM\Software\MyCo\MyApp AppBkUp.hiv

++

REG UNLOAD

删除已加载的部分注册表。

语法

REG UNLOAD KeyName

参数

KeyName

指定要卸载的子项的完整路径。要指定远程计算机，请包括计算机名（以 **\\ComputerName** 格式表示），并将其作为 **KeyName** 的一部分。省略 **\\ComputerName** 会导致默认对本地计算机的操作。**KeyName** 必须包括一个有效的根键。有效根键包括 **HKLM**、**HKCU**、**HKCR**、**HKU** 以及 **HKCC**。如果指定了远程计算机，则有效根键是 **HKLM** 和 **HKU**。

reg unload 操作的返回值。

0　成功

1　失败

示例

要卸载 **HKLM** 中的配置单元 **TempHive**，请键入：**REG UNLOAD HKLM\TempHive**

实际应用例子：

写一个判断程序，运行程序，当目前系统处于“显示隐藏文件”状态时，运行它就自动不显示，当系统处于“不显示隐藏文件”状态时，运行它就自动显示。
程序难就难在怎么获取键值。
我们运行代码：

```
@echo off
mode con cols=102 lines=10
for /f "delims=" %%i in ('reg query
HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced\Folder\Hidden\SHOWALL　/v CheckedValue') do echo　%%i
pause>nul
```

运行上面代码显示内容如下：

```
----------------------------------------------------------------------------------------

! REG.EXE VERSION 3.0

HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced\Folder\Hidden\SHOWALL

        CheckedValue        REG_DWORD        0x1

----------------------------------------------------------------------------------------
```

我们需要的数据只有“0x1”一个，从第 4 行为

```
        CheckedValue        REG_DWORD        0x1
```

它有前导空格，很讨厌，我们观察到 1 的前面是字母 **x**，于是我们将 **x** 作为分隔符，显示第二行就行了。格式为 **"skip=4 tokens=2 delims=x"**，就可以截取数字“1”，嘿嘿。

写一个批处理程序，判断如果当前计算机处于显示状态，则隐藏文件，如果处于隐藏状态，则显示文件：

```
@echo off&setlocal enabledelayedexpansion
mode con cols=60 lines=20&color 9f
title 隐藏显示文件　　作者 MHL QQ1208980380
echo.&echo　　彻底隐藏和显示文件或文件夹
echo.&echo　　程序通过修改注册表来隐藏文件和文件夹。隐藏好后，即使选择“显示所有文件和文件夹”，具有隐藏属性的文件也无法显示。要想恢复显示文件，只有修改注册表回原样才行。操作难度不大，用这种办法保护信息安全，只能对付一般电脑小白。
echo　　注：有些杀毒软件对修改注册表很敏感，允许程序运行即可，程序无毒可以放心。
echo.&echo.&echo.
set var=HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced\Folder\Hidden\SHOWALL
for /f "skip=4 tokens=2 delims=x" %%i in ('reg query "%var%" /v "CheckedValue"') do if /i "%%i"=="1" (
	echo　　当前计算机文件或文件夹处于显示状态。
	set /p tem=是否隐藏文件?^(Y/N^)
	if /i "!tem!"=="Y" reg add "%var%" /v CheckedValue /t REG_DWORD /d 0 /f
) else (
	echo　　当前计算机文件或文件夹处于隐藏状态。
	set /p tem=是否显示文件?^(Y/N^)
	if /i "!tem!"=="Y" reg add "%var%" /v CheckedValue /t REG_DWORD /d 1 /f
)
echo.&echo.&echo 按任意键退出。&pause>nul
```

Run 项会使程序在用户每次登录时自动运行。

读取注册表下的 **RUN** 项代码：

```
@echo off
for /f "tokens=2 delims=:" %%i in ('reg query HKLM\Software\Microsoft\Windows\CurrentVersion\Run') do echo %systemDrive%%%i
pause>nul
```

---

ping

什么是 **Ping**

Ping 是典型的网络工具。**Ping** 是 **Windows** 系列自带的一个可执行命令，从 98 到最新的 2003 **Server** 版的计算机都带有这个命令行工具。**Ping** 能够辨别网络功能的某些状态。这些网络功能的状态是日常网络故障诊断的基础。特别是 **Ping** 能够识别连接的二进制状态(也就是是否连通)。**Ping** 命令通过向计算机发送 **ICMP** 回应报文并且监听回应报文的返回，以校验与远程计算机或本地计算机的连接。对于每个发送报文， **Ping** 最多等待 1 秒，并打印发送和接收把报文的数量。比较每个接收报文和发送报文，以校验其有效性。默认情况下，发送四个回应报文，每个报文包含 64 字节的数据。**Ping** 向目标主机(地址)发送一个回送请求数据包，要求目标主机收到请求后给予答复，从而判断网络的响应时间和本机是否与目标主机(地址)联通。

Ping

校验与远程计算机或本地计算机的连接。只有在安装 **TCP/IP** 协议之后才能使用该命令。

语法：

ping ip [-t] [-a] [-n count] [-l length] [-f] [-i ttl] [-v tos] [-r count] [-s count] [[-j computer-list] | [-k computer-list]] [-w timeout] destination-list

下面是帮助信息获得的语法格式：

ping [-t] [-a] [-n count] [-l size] [-f] [-i TTL] [-v TOS] [-r count] [-s count] [[-j host-list] | [-k host-list]] [-w timeout] target_name

两种基本格式：

1、**ping+IP**　　例如：**ping** 119.167.145.19

2、**ping+**网址　　例如：**ping www.youku.com**　这个可以方便查看网址的 **IP**

例如：

Ping 192.168.10.163

显示结果如下：

Pinging 192.168.10.163 **with** 32 **bytes of data:**

Reply from 192.168.10.163**: bytes**=32 **time**<1**ms TTL**=64
Reply from 192.168.10.163**: bytes**=32 **time**<1**ms TTL**=64
Reply from 192.168.10.163**: bytes**=32 **time**<1**ms TTL**=64
Reply from 192.168.10.163**: bytes**=32 **time**<1**ms TTL**=64

Ping statistics for 192.168.10.163**:**
　　Packets: Sent = 4**, Received** = 4**, Lost** = 0 **(**0% **loss),**
Approximate round trip times in milli-seconds:
　　Minimum = 0**ms, Maximum** = 0**ms, Average** = 0**ms**

上面的状态说明 **ping** 通了。
解释如下：
正在 **ping** 192.168.10.163 具有 32 字节的数据：

来自 192.168.10.163 的回复：字节=32 时间<1**ms TTL**=64
来自 192.168.10.163 的回复：字节=32 时间<1**ms TTL**=64
来自 192.168.10.163 的回复：字节=32 时间<1**ms TTL**=64
来自 192.168.10.163 的回复：字节=32 时间<1**ms TTL**=64

ping192.168.10.163 的统计信息：
　　数据包：已发送=4，已接收=4，丢失=0（0%丢失），
往返程的估计时间以毫秒为单位：
　　最短=0**ms,**最长=0**ms,**平均=0**ms**

备注：上面因为是 **ping** 本机，所以时间非常短暂，计算机认为没花时间。

再比如：**ping www.baidu.com**
运行结果如下：

Pinging www.a.shifen.com [115.239.210.27] **with** 32 **bytes of data:**

Reply from 115.239.210.27**: bytes**=32 **time**=37**ms TTL**=55
Reply from 115.239.210.27**: bytes**=32 **time**=33**ms TTL**=55
Reply from 115.239.210.27**: bytes**=32 **time**=32**ms TTL**=55
Reply from 115.239.210.27**: bytes**=32 **time**=37**ms TTL**=55

Ping statistics for 115.239.210.27**:**

Packets: Sent = 4**, Received** = 4**, Lost** = 0 **(**0% **loss),**
Approximate round trip times in milli-seconds:
Minimum = 32**ms, Maximum** = 37**ms, Average** = 34**ms**

这也是 **ping** 通的，同时我们知道了 **www.baidu.com** 的 **IP** 为 115.239.210.27。解释参照上面解释。
顺便说明一下：像百度是个大公司，服务器有多个，**IP** 地址有多个，这里只 **ping** 出一个。

下面的状态说明 **ping** 不通：

Pinging 192.168.10.2 **with** 32 **bytes of data:**

Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 192.168.10.2**:**
Packets: Sent = 4**, Received** = 0**, Lost** = 4 **(**100% **loss),**

解释如下：
正在 **ping**192.168.10.2 具有 32 字节的数据：

请求超时。
请求超时。
请求超时。
请求超时。

ping192.168.10.2 的统计信息：
数据包：已发送=4，已接收=0，丢失=4（100%丢失），

下面开始讲解：

Options:
-t **Ping the specified host until stopped. To see statistics and continue - type Control-Break; To stop - type Control-C.**

不断 **Ping** 指定的主机，查看是连接还是中断，直到按 **Ctrl+C** 快捷键停止。

例如：**ping www.baidu.com -t**
ping -t www.sina.com.cn
ping -t 206.188.193.62
ping 183.60.15.153 **-t**
上面几个例子都不断 **ping** 对方主机，直到我们按 **Ctrl+C** 终止操作为止。
这个参数常被用来攻击别人计算机。

-a　　Resolve addresses to hostnames.
将地址解析为计算机名。这里的主机名指的是 **netbios** 名，是通过微软的 **wins** 服务反向解析的。说一下：现在的防火墙都防 **ping**，所以对本机百分百可以 **ping** 出名字，但对别的计算机就不一定了。（具体进一步研究）

程序例子：
ping -a 192.168.1.21
结果显示如下：

Pinging iceblood.yofor com [192.168.1.21] **with** 32 **bytes of data:**

Reply from 192.168.1.21**: bytes**=32 **time**<10**ms TTL**=254
Reply from 192.168.1.21**: bytes**=32 **time**<10**ms TTL**=254
Reply from 192.168.1.21**: bytes**=32 **time**<10**ms TTL**=254
Reply from 192.168.1.21**: bytes**=32 **time**<10**ms TTL**=254

Ping statistics for 192.168.1.21**:**
Packets: Sent = 4**,Received** = 4**,Lost** = 0 **(**0**% loss),**
Approximate round trip times in milli-seconds:
Minimum = 0**ms,Maximum** = 0**ms,Average** = 0**ms**

从上面就可以知道 **IP** 为 192.168.1.21 的计算机 **NetBios** 名为 **iceblood.yofor com**。

-n count　　Number of echo requests to send.
发送由 **count** 指定数量的 **ECHO** 数据包数，默认值为 4。

在默认情况下，一般都只发送四个数据包，通过这个命令可以自己定义发送的个数，对衡量网络速度很有帮助，比如我想测试发送 10 个数据包的返回的平均时间为多少，最快时间为多少，最慢时间为多少就可以通过以下获知：

ping -n 10 202.108.22.5

这个是 **ping** 百度的首页，运行结果如下：

```
----------------------------------------------------------------------------
Pinging 202.108.22.5 with 32 bytes of data:

Reply from 202.108.22.5: bytes=32 time=141ms TTL=52
Reply from 202.108.22.5: bytes=32 time=140ms TTL=52
Reply from 202.108.22.5: bytes=32 time=139ms TTL=52
Reply from 202.108.22.5: bytes=32 time=140ms TTL=52
Reply from 202.108.22.5: bytes=32 time=141ms TTL=52
Reply from 202.108.22.5: bytes=32 time=140ms TTL=52
Reply from 202.108.22.5: bytes=32 time=139ms TTL=52
Reply from 202.108.22.5: bytes=32 time=139ms TTL=52
Reply from 202.108.22.5: bytes=32 time=139ms TTL=52
Reply from 202.108.22.5: bytes=32 time=139ms TTL=52

Ping statistics for 202.108.22.5:
    Packets: Sent = 10, Received = 10, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 139ms, Maximum = 141ms, Average = 139ms
----------------------------------------------------------------------------
```

上面的意思是：

发送了 10 个数据包，返回 10 个数据包，丢失 0 个数据包，丢失率为 0%，

这 10 个数据包当中，返回速度最快为 139**ms**，最慢为 141**ms**，平均速度为 139**ms**。

-l size　　Send buffer size.

定义 **echo** 数据包大小。

在默认的情况下 **windows** 的 **ping** 发送的数据包大小为 32**byte**，我们也可以自己定义它的大小，但有一个大小的限制，就是最大只能发送 65500**byte**，也许有人会问为什么要限制到 65500**byte**，因为 **Windows** 系列的系统都有一个安全漏洞（也许还包括其他系统）就是当向对方一次发送的数据包大于或等于 65532 时，对方就很有可能挡机，所以微软公司为了解决这一安全漏洞于是限制了 **ping** 的数据包大小。虽

然微软公司已经做了此限制，但这个参数配合其他参数以后危害依然非常强大，比如我们就可以通过配合**-t** 参数来实现一个带有攻击性的命令：（以下介绍带有危险性，仅用于试验，请勿轻易施于别人机器上，否则后果自负）

ping -l 65500 **-t** 192.168.1.21
运行结果如下：
Pinging 192.168.1.21 **with** 65500 **bytes of data:**
Reply from 192.168.1.21**: bytes**=65500 **time**<10**ms TTL**=254
Reply from 192.168.1.21**: bytes**=65500 **time**<10**ms TTL**=254
……
因为加上参数**-t**，它就会不停的向 192.168.1.21 计算机发送大小为 65500**byte** 的数据包，如果你只有一台计算机也许没有什么效果，但如果有很多计算机那么就可以使对方完全瘫痪，我曾经就做过这样的试验，当我同时使用 10 台以上计算机 **ping** 一台 **Win**2000**Pro** 系统的计算机时，不到 5 分钟对方的网络就已经完全瘫痪，网络严重堵塞，**HTTP** 和 **FTP** 服务完全停止，由此可见威力非同小可！

而百度公司比较刁钻，数据大小最大限定为 1464**b**，超过就 **ping** 不通了。
ping -l 1464 **-t** 202.108.22.5

-f　　Set Don't Fragment flag in packet.
在包中发送“不分段”标志。该包将不被路由上的网关分段。
在一般你所发送的数据包都会通过路由分段再发送给对方，加上此参数以后路由就不会再分段处理。
（问：不分段有何好处？）

-i TTL　　Time To Live.
指定 **TTL** 值在对方的系统里停留的时间。**ttl**：表示从 1 到 255 之间的数
此参数同样是帮助你检查网络运转情况的。
（问：具体怎么用？）

-v TOS　　Type Of Service.
将“服务类型”字段设置为 **tos** 指定的值。

（不明使用）

-r count　　Record route for count hops.

在“记录路由”字段中记录传出和返回数据包的路由。

在一般情况下你发送的数据包是通过一个个路由才到达对方的，但到底是经过了哪些路由呢？通过此参数就可以设定你想探测经过的路由的个数，不过限制在了 9 个，也就是说你只能跟踪到 9 个路由，如果想探测更多，可以通过其他命令实现

我试过了，公司内网可以很容易 **ping** 通，但 **ping** 外网就难以 **ping** 通了，网友们的解释是：与中间某层路由器的设置有关。正常的 **ping** 是点对点方式，加参数**-r** 是要中间路由器返回 **ICMP** 包信息。当某路由器禁止或者限制返回 **ICMP** 数据包信息时，就会不通。

下面是直接复制网友们的 **ping** 结果：

ping -n 1 **-r** 9 202.96.105.101

发送一个数据包，最多记录 9 个路由。运行结果如下：

Pinging 202.96.105.101 **with** 32 **bytes of data:**
Reply from 202.96.105.101**: bytes**=32 **time**=10**ms TTL**=249
Route: 202.107.208.187 ->
202．107.210.214 ->
61．153.112.70 ->
61．153.112.89 ->
202．96.105.149 ->
202．96.105.97 ->
202．96.105.101 ->
202．96.105.150 ->
61．153.112.90
Ping statistics for 202.96.105.101**:**
Packets: Sent = 1**,Received** = 1**,Lost** = 0 **(**0% **loss),**
Approximate round trip times in milli-seconds:
Minimum = 10**ms,Maximum** = 10**ms,Average** = 10**ms**

从上面我就可以知道从本机到 202.96.105.101 一共通过了 202.107.208.187 ，202.107.210.214,61.153.112.70,61.153.112.89,202.96.105.149,202.96.105.97 这几个路由。

再看一个网友例子：
ping 169.254.190.92　**-r** 4
运行结果如下：

```
---------------------------------------------------------------------------
Pinging 169.254.190.92 with 32 bytes of data:
Reply from 169.254.190.92: bytes=32 time<1ms TTL=128
    Route: 169.254.190.92
Reply from 169.254.190.92: bytes=32 time<1ms TTL=128
    Route: 169.254.190.92
Reply from 169.254.190.92: bytes=32 time<1ms TTL=128
    Route: 169.254.190.92
Reply from 169.254.190.92: bytes=32 time<1ms TTL=128
    Route: 169.254.190.92
Ping statistics for 169.254.190.92:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
---------------------------------------------------------------------------
```

结果说明（下同）

Ping 命令有两种返回结果：

1、**“Request timed out.”**表示没有收到目标主机返回的响应数据包，也就是网络不通或网络状态恶劣

2、**“Reply from X.X.X.X: bytes**=32 **time**<1**ms TTL**=255”表示收到从目标主机 **X.X.X.X** 返回的响应数据包，数据包大小为 32**Bytes**，响应时间小于 1**ms TTL** 为 255，这个结果表示您的计算机到目标主机之间连接正常。

3、**“Destination host unreachable”**表示目标主机无法到达

4、**“PING: transmit failed,error code XXXXX”**表示传输失败，错误代码 **XXXXX**

下面是我实际操作 **ping** 通电信的例子：
ping -r 9 113.87.226.9
这里没有指定数据包发送个数，那么默认就发送 4 个。裕兴结果如下：

```
---------------------------------------------------------------------------------------
Pinging 113.87.226.9 with 32 bytes of data:

Reply from 192.168.10.1: Destination net unreachable.
Reply from 192.168.10.1: Destination net unreachable.
Reply from 192.168.10.1: Destination net unreachable.
Reply from 192.168.10.1: Destination net unreachable.
```

Ping statistics for 113.87.226.9**:**
　　Packets: Sent = 4**, Received** = 4**, Lost** = 0 **(**0**% loss),**
Approximate round trip times in milli-seconds:
　　Minimum = 0**ms, Maximum** = 0**ms, Average** = 0**ms**

Destination net unreachable.表示对方主机不存在或者没有跟对方建立连接。
上面可以看出，经过的路由是 192.168.10.1，是公司的路由。

-s count　　Timestamp for count hops.
　　指定 **count** 指定的跃点数的时间戳。
此参数和**-r** 差不多，只是这个参数不记录数据包返回所经过的路由，最多也只记录 4 个。
（具体怎么用？）

-j host-list　　Loose source route along host-list.
　　利用 **computer-list** 指定的计算机列表路由数据包。连续计算机可以被中间网关分隔（路由稀疏源）**IP** 允许的最大数量为 9。

-k host-list　　Strict source route along host-list.
　　利用 **computer-list** 指定的计算机列表路由数据包。连续计算机不能被中间网关分隔（路由严格源）**IP** 允许的最大数量为 9。

-w timeout　　Timeout in milliseconds to wait for each reply.
　　以毫秒为单位指定超时间隔。

Ping 是个使用频率极高的网络诊断程序，用于确定本地主机是否能与另一台主机交换（发送与接收）数据报。根据返回的信息，你就可以推断 **TCP/IP** 参数是否设置得正确以及运行是否正常。需要注意的是：成功地与另一台主机进行一次或两次数据报交换并不表示 **TCP/IP** 配置就是正确的，你必须执行大量的本地主机与远程主机的数据报交换，才能确信 **TCP/IP** 的正确性。

返回信息

Ping 的返回信息有**"Request Timed Out"**、**"Destination Net Unreachable"**和**"Bad IP address"**还有**"Source quench received"**。

"Request Timed Out"这个信息表示对方主机可以到达到 **TIME OUT**，这种情况通常是为对方拒绝接收你发给它的数据包造成数据包丢失。大多数的原因可能是对方装有防火墙或已下线。

"Destination Net Unreachable"这个信息表示对方主机不存在或者没有跟对方建立连接。这里要说明一下**"destination host unreachable"**和**"time out"**的区别，如果所经过的路由器的路由表中具有到达目标的路由，而目标因为其它原因不可到达，这时候会出现**"time out"**，如果路由表中连到达目标的路由都没有，那就会出现**"destination host unreachable"**。

"Bad IP address" 这个信息表示你可能没有连接到 **DNS** 服务器所以无法解析这个 **IP** 地址，也可能是 **IP** 地址不存在。

"Source quench received"信息比较特殊，它出现的机率很少。它表示对方或中途的服务器繁忙无法回应。

测试网络

怎样使用 **Ping** 这命令来测试网络连通呢？

连通问题是由许多原因引起的，如本地配置错误、远程主机协议失效等，当然还包括设备等造成的故障。

首先我们讲一下使用 **Ping** 命令的步骤。

使用 **Ping** 检查连通性有六个步骤：

1．　使用 **ipconfig /all** 观察本地网络设置是否正确；

2．**Ping**127.0.0.1，127.0.0.1 回送地址 **Ping** 回送地址是为了检查本地的 **TCP/IP** 协议有没有设置好；

3．　**Ping** 本机 **IP** 地址，这样是为了检查本机的 **IP** 地址是否设置有误；

4．　**Ping** 本网网关或本网 **IP** 地址，这样的是为了检查硬件设备是否有问题，也可以检查本机与本地网络连接是否正常；（在非局域网中这一步骤可以忽略）

5．**Ping** 本地 **DNS** 地址，这样做是为了检查 **DNS** 是否能够将 **IP**。

6．**Ping** 远程 **IP** 地址，这主要是检查本网或本机与外部的连接是否正常。

常见错误

在检查网络连通的过程中可能出现一些错误，这些错误总的来说分为两种最常见。

1．　**Request Timed Out**

"request time out"这提示除了在《**PING**（一）》提到的对方可能装有防火墙或已关机以外，还有就是本机的 **IP** 不正确和网关设置错误。

①、**IP** 不正确：

IP 不正确主要是 **IP** 地址设置错误或 **IP** 地址冲突，这可以利用 **ipconfig /all** 这命令来检查。在 **WIN**2000 下 **IP** 冲突的情况很少发生，因为系统会自动检测在网络中是否有相同的 **IP** 地址并提醒你是否设置正确。在 **NT** 中不但会出现**"request time out"**这提示而且会出现**"Hardware error"**这提示信息比较特殊不要给它的提示所迷惑。

②、网关设置错误：这个错误可能会在第四个步骤出现。网关设置错误主要是网关地址设置不正确或网关没有帮你转发数据，还有就是可能远程网关失效。这里主要是在你 **Ping** 外部网络地址时出错。错误表现为无法 **Ping** 外部主机返回信息**"Request timeout"**。

2．　**Destination Host Unreachable**

当你在开始 **PING** 网络计算机时如果网络设备出错它返回信息会提示**"destination host unreachable"**。如果局域网中使用 **DHCP** 分配 **IP** 时，而碰巧 **DHCP** 失效，这时使用 **PING** 命令就会产生此错误。因为在 **DHCP** 失效时客户机无法分配到 **IP** 系统只有自设 **IP**，它往往会设为不同子网的 **IP**。所以会出现**"Destination Host Unreachable"**。另外子网掩码设置错误也会出现这错误。

还有一个比较特殊就是路由返回错误信息，它一般都会在**"Destination Host Unreachable"**前加上 **IP** 地址说明哪个路由不能到达目标主机。这说明你的机器与外部网络连接没有问题，但与某台主机连接存在问题。

编辑本段反馈信息

Request timed out

a.对方已关机：比如在上图中主机 **A** 中 **PING** 192.168.0.7,或者主机 **B** 关机了，在主机 **A** 中 **PING** 192.168.0.5 都会得到超时的信息。

b.对方与自己不在同一网段内，通过路由也无法找到对方，但有时对方确实是存在的，当然不存在也是返回超时的信息。

c.对方确实存在，但设置了 **ICMP** 数据包过滤（比如防火墙设置）

怎样知道对方是存在，还是不存在呢，可以用带参数 **-a** 的 **Ping** 命令探测对方，如果能得到对方的 **NETBIOS** 名称，则说明对方是存在的，是有防火墙设置，如果得不到，多半是对方不存在或关机，或不

在同一网段内。

d.错误设置 **IP** 地址

正常情况下，一台主机应该有一个网卡，一个 **IP** 地址，或多个网卡，多个 **IP** 地址（这些地址一定要处于不同的 **IP** 子网）。但如果一台电脑的“拨号网络适配器”（相当于一块软网卡）的 **TCP/IP** 设置中，设置了一个与网卡 **IP** 地址处于同一子网的 **IP** 地址，这样，在 **IP** 层协议看来，这台主机就有两个不同的接口处于同一网段内。当从这台主机 **Ping** 其他的机器时，会存在这样的问题：

A.主机不知道将数据包发到哪个网络接口，因为有两个网络接口都连接在同一网段。

B.主机不知道用哪个地址作为数据包的源地址。因此，从这台主机去 **Ping** 其他机器，**IP** 层协议会无法处理，超时后，**Ping** 就会给出一个“超时无应答”的错误信息提示。但从其他主机 **Ping** 这台主机时，请求包从特定的网卡来，**ICMP** 只须简单地将目的、源地址互换，并更改一些标志即可，**ICMP** 应答包能顺利发出，其他主机也就能成功 **Ping** 通这台机器了。

Destination host Unreachable

对方与自己不在同一网段内，而自己又未设置默认的路由，或者网络上根本没有这个地址，比如上例中 **A** 机中不设定默认的路由，运行 **Ping** 192.168.1.4 就会出现“**Destination host Unreachable**”。

网线出了故障

这里要说明一下“**destination host unreachable**”和 “**time out**”的区别，如果所经过的路由器的路由表中具有到达目标的路由，而目标因为其他原因不可到达，这时候会出现“**time out**”，如果路由表中连到达目标的路由都没有，那就会出现“**destination host unreachable**”。

Bad IP address

这个信息表示您可能没有连接到 **DNS** 服务器，所以无法解析这个 **IP** 地址，也可能是 **IP** 地址不存在。

Source quench received

这个信息比较特殊，它出现的机率很少。它表示对方或中途的服务器繁忙无法回应。

Unknown host——不知名主机

这种出错信息的意思是，该远程主机的名字不能被域名服务器（**DNS**）转换成 **IP** 地址。故障原因可能是域名服务器有故障，或者其名字不正确，或者网络管理员的系统与远程主机之间的通信线路有故障。

No answer——无响应

这种故障说明本地系统有一条通向中心主机的路由，但却接收不到它发给该中心主机的任何信息。故障原因可能是下列之一：中心主机没

有工作；本地或中心主机网络配置不正确；本地或中心的路由器没有工作；通信线路有故障；中心主机存在路由选择问题。
Ping 127.0.0.1：127.0.0.1 是本地循环地址
如果本地址无法 **Ping** 通，则表明本地机 **TCP/IP** 协议不能正常工作。
no rout to host：网卡工作不正常
transmit failed,error code：10043 网卡驱动不正常
unknown host name：**DNS** 配置不正确

TTL

ping 命令中返回的 **ttl** 即可反映跃点数。它每经过一个路由及减一。通过它一般可猜测目标机的系统。**TTL** 字段值可以帮助我们识别操作系统类型。

UNIX 及类 **UNIX** 操作系统 **ICMP** 回显应答的 **TTL** 字段值为 255 **Compaq Tru**64 5.0 **ICMP** 回显应答的 **TTL** 字段值为 64

Windows NT/2K 操作系统 **ICMP** 回显应答的 **TTL** 字段值为 128

Windows 95 操作系统 **ICMP** 回显应答的 **TTL** 字段值为 32

在一般情况下还可以通过 **ping** 对方让对方返回给你的 **TTL** 值大小，粗略的判断目标主机的系统类型是 **Windows** 系列还是 **UNIX/Linux** 系列，一般情况下 **Windows** 系列的系统返回的 **TTL** 值在 100-130 之间，而 **UNIX/Linux** 系列的系统返回的 **TTL** 值在 240-255 之间，当然 **TTL** 的值在对方的主机里是可以修改的，**Windows** 系列的系统可以通过修改注册表以下键值实现：

[HKEY_LOCAL_MACHINE \ sys tem \ CurrentControlSet \ Services \ Tcpip \ Parameters]

"DefaultTTL"=dword:000000ff

255---**FF**

128---80

64----40

32----20

当然，系统的 **ttl** 是可以修改的。

不同的操作系统，它的 **TTL** 值是不相同的。默认情况下，**Linux** 系统的 **TTL** 值为 64 或 255，**Windows NT**/2000/**XP** 系统的 **TTL** 值为 128，**Windows** 98 系统的 **TTL** 值为 32，**UNIX** 主机的 **TTL** 值为 255。

简单的伪装操作系统的方法,就是修改 **TTL.**

Windows 下修改方法**:**

修改 **TTL** 值其实非常简单，通过注册表编辑器就可以实现，点击“开始→运行”，在“运行”对话框中输入“**regedit**”命令并回车，弹出“注册表编辑器”对话框，展开“**HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Tcpip\ Parameters**”，找到“**DefaultTTL**”，将该值修改为十进制的“255”，重新启动服务器系统后即可。

Linux 下修改方法**:**

方法 1**(**重启后有效**)**：

#sysctl -w net.ipv4.ip_default_ttl=N

(N=0~255**),**若 **N>**255,则 **ttl**=0

方法 2**(**重启后无效**)**：

#echo N(N 为 0～255**) > /proc/sys/net/ipv4/ip_default_ttl**

---

ping 127.1

ping 127.0.0.1

这两个都是 **ping** 本机，两个是一样的，只不过上面的那个是缩写。

ping127.0.0.1 这个数据包不会发送到网络上，用来检查本机 **tcp/ip** 配置。可以这样理解：这个数据包从 **cmd** 的 **shell** 发到了 **kernel** 的 **tcp/ip** 模块，然后从 **tcp/ip** 模块返回到 **cmd** 下，就看到 **ping** 成功了。也表示本机 **tcp/ip** 配置没问题，这就是为什么拔掉网线后仍然能够 **ping** 通。

ping 本机 **ip** 地址 这个数据包发送到网络上了 一般指局域网 所有机器都收到了这个数据包 也包括自己的机器 然后自己的机器回映这个数据包 回映了就看到 **cmd** 下 **ping** 成功了 这也表示这个 **ip** 地址是正确的 是被其他机器承认的 **ip**。例如 **ping** 192.168.10.163，如果拔掉网线，那么就无法 **ping** 通。

由于 **ping**127.0.0.1，是 **ping** 本机，时间间隔很短，也没受到外界的什么干扰，所以这个常被用来作为时间延迟。

例如我们想让批处理暂停 10 秒后再继续执行，可以这么写：

ping /n 10 127.1**>nul**

程序执行到这里，会发送 10 个数据包到本机，时间间隔大概是 10 秒。

注意：这种方法的延迟是非常精确的，我曾经测试了 90 分钟的时间，结果没有误差。

格式：
批处理停顿几秒后再执行下一条命令
ping /n x 127**.1>nul**
其中“**x**”表示秒数，把它改成你想要的秒数就可以了。
程序例子：
start /w 1**.exe**
ping /n 5 127**.1>nul**
start /w 2**.exe**

上例就是执行完 1**.exe** 后，停顿 5 秒，后执行 2**.exe**。

显示本机 **IP**

```
@echo off
mode con cols=64 lines=20
title  本机 IP 地址
ipconfig
echo. & echo.
echo ------------------------------------------------------------
echo                                  中英文对照
echo.
echo Connection-specific DNS Suffix      连接特定的 DNS 后缀
echo IP Address                          IP 地址
echo Subnet Mask                         子网掩码
echo Default Gateway                     默认网关
pause > nul
```

显示 **DNS** 客户解析器缓存的内容

```
@echo off
mode con cols=80 lines=160
ipconfig /displaydns
pause >nul
```

清除 **DNS** 缓存记录

```
@echo off
ipconfig /flushdns
pause >nul
```

中文显示 **ping** 结果

```
@echo off&color f2&echo.
set for=于
set of=的
set with=用
set in=(以
set data:=数据
set milli-seconds:=毫秒为单位)
set Approximate=大约
set times=时间:
set round=来回
set trip=行程
set Reply=应答
set from=来自
set bytes=字节
set time=时间:
set timed=时间
set out=超过
set statistics=统计
set Packets:=包:
set Sent=已发送=
set Received=已收到=
set Lost=已丢失=
set loss)=丢失)
set Minimum=最小值=
set Maximum=最大值=
set Average=平均值=
set TTL=TTL=
setlocal enabledelayedexpansion
set a=
set/p a=请输入要 ping 的网址或 IP
for /f "delims=" %%i in ('ping %a%') do (
    set ret=
    for %%a in (%%i) do if defined %%a (set ret=!ret!!%%a!) else set
ret=!ret! %%a
    if not "!ret!"=="" (set ret=!ret:time=时间! && echo !ret!) else
echo.
)
pause>nul
```

ipconfig

IP 信息查看工具

1 具体功能

Ipconfig 是内置于 **windows** 的 **TCP/IP** 应用程序，用于显示本地计算机网络适配器的物理地址和 **IP** 地址等配置信息，这些信息一般用来检验手动配置的 **TCP/IP** 设置是否正确。当在网络中使用 **DHCP** 服务时，**ipconfig** 可以检测到计算机中分配到了什么 **IP** 地址，是否配置正确，并且可以释放、重新获取 **IP** 地址。这些信息对于网络测试和故障排除都有重要的作用。

2**ipconfig** 命令参数简介：

使用 **ipconfig** 命令如果不带参数，只显示简单的 **IP** 地址配置信息，如果配合参数使用，还可以实现其他的一些管理功能。

（1） 语法

从帮助信息获得的语法格式：

ipconfig [/? | /all | /renew [adapter] | /release [adapter] | /flushdns | /displaydns | /registerdns | /showclassid adapter | /setclassid adapter [classid]]

从参考书上获得的语法格式：

ipconfig [/all] [/renew [adapter]] [/release [adapter]] [/flushdns] [/displaydns] [/registerdns] [/showclassid adapter] [/setclassid adapter [class ID]]

（2） 参数说明

概念：网络适配器又称网卡或网络接口卡，缩写：**NIC**。

在这里，用 **ipconfig** 查询到的适配器可以代表物理接口（例如安装的网络适配器）或逻辑接口（例如拨号连接）。

ipconfig　　在没有该参数的情况下 **ipconfig** 只显示 **IP** 地址、子网掩码和各个适配器的默认网关值。

ipconfig /all　　显示所有适配器的完整 **TCP/IP** 配置信息。

控制面板-->网络连接-->右击本地连接（或别的连接）-->状态-->支持-->详细信息。这个也可以获得 **TCP/IP** 信息。

下面所有命令，斜杠前面都需加 **ipconfig**，例如 **ipconfig /renew**。

/?在命令提示符显示帮助。

/all　　显示所有适配器的完整 **TCP/IP** 配置信息。

/renew [adapter] 更新所有适配器（如果未指定适配器），或特定适配器（如果包含了 **adapter** 参数）的 **DHCP** 配置。该参数仅在具有配置为自动获取 **IP** 地址的网卡的计算机上可用。要指定适配器名称，请键入使用不带参数的 **ipconfig** 命令显示的适配器名称。（简单理解：重新自动获得 **IP**，网络连接。）

/release [adapter] 发送 **DHCPRELEASE** 消息到 **DHCP** 服务器，以释放所有适配器（如果未指定适配器）或特定适配器（如果包含了 **adapter** 参数）的当前 **DHCP** 配置并丢弃 **IP** 地址配置。该参数可以禁用配置为自动获取 **IP** 的适配器的 **TCP/IP**。要指定适配器名称，请键入使用不带参数的 **ipconfig** 命令显示的适配器名称。（简单理解：释放当前 **IP**，网络断开。）

/flushdns 清理并重设 **DNS** 客户解析器缓存的内容。

/displaydns 显示 **DNS** 客户解析器缓存的内容，包括从本地计算机文件预装载的记录以及计算机解析的名称查询而最近获得的任何资源记录。**DNS** 客户服务在查询配置的 **DNS** 服务器之前使用这些信息快速解析被频繁查询的名称。

/showclassid adapter 显示指定适配器的 **DHCP** 类别 **ID**。要查看所有适配器的 **DHCP** 类别 **ID**，可以使用星号（*）通配符代替 **adapter**。该参数仅具有配置为自动获取 **IP** 地址的网卡的计算机上可用。

/setclassid adapter [classID] 配置特定适配器的 **DHCP** 类别 **ID**。要设置所有适配器的 **DHCP** 类别 **ID**，可以使用星号（*）通配符代替 **adapter**。该参数仅在具有配置为自动获取 **IP** 地址的网卡的计算机上可用/**registerdns** 初始化计算机上配置的 **DNS** 名称和 **IP** 地址的手工动态注册。可以使用该参数对失败的 **DNS** 名称注册进行疑难解答或解决客户和 **DNS** 服务器之间的动态更新问题，而不必重新启动客户计算机。**TCP/IP** 协议高级属性中的 **DNS** 设置可以确定 **DNS** 中注册了哪些名称。如果未指定 **DHCP** 类别的 **ID**，则会删除当前类别的 **ID**。

释放 **IP** 地址：**ipconfig /release**
获得 **IP** 地址：**ipconfig /renew**
这是一对相反的指令。

如果网络中使用了 **DHCP** 服务，客户端计算机就可以自动获得 **IP** 地址。但有时因 **DHCP** 服务器或网络故障等原因，使一些客户端计算机不能正常获得 **IP** 地址（此时系统就会自动为网卡分配一个 169.254.**x.x** 的 **IP** 地址）或者有些计算机租约到期（需要更新或重新获得 **IP** 地址）等等，使得客户端计算机没有正确获得 **IP** 地址。这时就需要先将原先获得的 **IP** 地址释放掉（**ipconfig /release**，可以用命令提示符或批处理执行），释放以后，可以看到 **IP** 地址和子网掩码均变成 0.0.0.0，此时网络连接断开，然后就可以重新获得一个新的 **IP** 地址了。要重新获得 **IP** 地址，可以点击 **Windows** 下的连接状态-->支持-->修复，也可以用 **ipconfig /renew** 修复，系统就会自动从 **DHCP** 服务器获得一个新的 **IP** 地址，以及子网掩码、默认网关等信息，网络自动重新连接。

清除 **DNS** 客户解析器缓存内容
ipconfig /flushdns
运行这个后，会显示：
Successfully flushed the DNS Resolver Cache.（成功清空 **DNS** 解析器缓存。）

Ipconfig /displaydns
显示 **DNS** 客户解析器缓存的内容，包括从本地计算机文件预装载的记录以及计算机解析的名称查询而最近获得的任何资源记录。**DNS** 客户服务在查询配置的 **DNS** 服务器之前使用这些信息快速解析被频繁查询的名称。当用 **ipconfig /flushdns** 清空后再用 **Ipconfig /displaydns** 显示 **DNS** 缓存内容，如下显示（如果没有用 **ipconfig /flushdns** 清理过，则内容会比较多，在此补介绍了）：

Windows IP Configuration
Windows IP 配置

1.0.0.127**.in-addr.arpa**

Record Name . . . . . : 1.0.0.127**.in-addr.arpa.**

记录名称
Record Type . . . . . : 12
记录类型
Time To Live　. . . . : 576170
存在时间
Data Length . . . . . : 4
数据长度
Section . . . . . : Answer
部分
PTR Record　. . . . . : localhost
PTR 记录　　本机

localhost
本地的

Record Name . . . . . : localhost
记录名称　　本机
Record Type . . . . . : 1
记录类型
Time To Live　. . . . : 576170
存在时间
Data Length . . . . . : 4
数据长度
Section . . . . . : Answer
部分
A (Host) Record . . . : 127.0.0.1
一个（主机记录）
==========

ipconfig registerdns
运行后显示：
Registration of the DNS resource records for all adapters of this computer　has been initiated. Any errors will be reported in the Event Viewer in 15　**minutes..**
翻译：
此计算机所有的适配器 **DNS** 资源注册记录已经重启。在 15 分钟之内，任何错误信息都将被记录在事件查看器中。

计算专题

一些编程技巧

一、进位问题

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

::那么数组就是 5、10、15、20、25 假如这是一个数，需要进行进位处理，变成数字 61725，怎么进位处理呢？

```
set jin=0
for /l %%i in (5,-1,1) do (
    set /a jin=jin/10+A%%i
    set /a A%%i=jin%%10
)
for /l %%i in (1,1,5) do set /p var=!A%%i!<nul
echo.&pause
```

二、读入一个小数，去掉小数点后输出来

```
@echo off
set /p num=
set num=%num:.=%
echo %num%
pause
```

三、数组专题

要实现高精度计算，数组是一个很好的途径。批处理如何实现数组计算呢？

一维数组

S 数组的每一个元素为：S1,S2,S3,S4,S5...

那么写成批处理就是：

```
@echo off
set S1=a
set S2=b
set S3=c
set S4=d
set S5=e
echo S1=%S1%
echo S2=%S2%
```

```
echo S3=%S3%
echo S4=%S4%
echo S5=%S5%
pause
```

上面就是一个最简单的一维有限数组，这个数组只有 5 个元素。

如果我们的数组有 1000 个元素，那么就不可能用 **set** 对每个元素一一设置，这时可以借助 **for** 语句实现：

```
@echo off
setlocal enabledelayedexpansion
for /l %%i in (1,1,1000) do (set S%%i=%%i)
for /l %%i in (1,1,1000) do (echo S%%i=!S%%i!)
pause
```

合并起来写就是：

```
@echo off
setlocal enabledelayedexpansion
for /l %%i in (1,1,1000) do (
    set S%%i=%%i
    echo S%%i=!S%%i!
    )
pause
```

上面的格式，**S%%i** 这种格式是借助变量**%%i** 作为下标。有时为了便于辨认，我们可以添加个中括号 **S[%%i]**，例如：

```
@echo off
setlocal enabledelayedexpansion
for /l %%i in (1,1,1000) do (
    set HL[%%i]=%%i
    echo HL[%%i]=!HL[%%i]!
    )
pause
```

注意，上面的书写，中括号只是一个符号便于辨认而已，对数组没任何影响。

二维数组：

批处理也可以实现二维数组，例如：

```
@echo off
setlocal enabledelayedexpansion
for /l %%i in (1,1,100) do (
    for /l %%j in (1,1,100) do (
        set S%%i%%j=%%i%%j
        echo S%%i%%j=!S%%i%%j!
    )
)
pause
```

和一维数组一样，如果我们愿意用中括号来括住下标也是可以的：

```
@echo off
setlocal enabledelayedexpansion
for /l %%i in (1,1,100) do (
    for /l %%j in (1,1,100) do (
        set S[%%i%%j]=%%i%%j
        echo S[%%i%%j]=!S[%%i%%j]!
    )
)
pause
```

多维数组
仿照一维二维数组的形式，我们可以将数组扩展到高维数组。

批处理对数组下标没有限制，我们可以用负数、0 和正数作为下标。例如：

```
@echo off
for /l %%i in (-100,1,100) do set S%%i=%%i
echo %S-10%
echo %S0%
echo %S10%
pause
```

特别注意一下，批处理虽然允许使用负数作为数组下标，但是，在一些情况下，使用负数作为下标会出现错误！并且很难被发现。

看下面的例子：

```
@echo off
for /l %%i in (-10,1,10) do set B%%i=12345
set /a i=8,j=9
set /a num=i-j
set /a var=B%num%
echo %var%     %B-1%
pause>nul
```

我们看上面代码，计算 **set /a num=i-j**，此时 **num** 的值等于-1，接下来执行 **set /a var=B%num%**，我们是想令 **var** 等于数组元素 **B-1** 的值，其值为 12345。可是事与愿违，原因就是用负数作为下标！**set /a var=B%num%**语句，**/a** 有计算功能，从计算机角度来看，也就是执行

set /a var=B-1

我们没有定义过变量 **B**，其值等于空值，空值减去 1，结果 **var** 就等于-1。但此时的 **B-1** 数组值仍旧是 12345，于是输出-1 12345。假如我们一开始也定义一下 **B** 值，例如定义 **B** 值为 100，那么输出为 99 12345，结果是很明显说明 **set /a var=B%num%**进行了一次计算。

假如我们将上面代码的 **j** 值改为 8，**set /a var=B%num%**语句就是

set /a var=B0

B0 数组我们已经有定义了，其值等于 12345，所以输出为 12345 12345，这是我们想要的。

为了确保万无一失，数组不要使用负数下标。

四、字符串转换为数组

问题：输入一个只有 9 个数据的字符串，然后转换为数组后输出

```
@echo off
setlocal enabledelayedexpansion
set /p S=
for /l %%i in (1,1,9) do (
    set T%%i=!S:~%%i,1!
    )
for /l %%i in (1,1,9) do (echo T%%i=!T%%i!)
pause
```

输入：
123456789
输出：
T1=2
T2=3
T3=4
T4=5
T5=6
T6=7
T7=8
T8=9
T9=
请按任意键继续. . .
上面问题出在 **set T%%i=!S:~%%i,1!**这里，当 **i** 取值 1 时，**!S:~%%i,1!** 就是**!S:~1,1!**，也就是舍弃 **S** 字符串的前 1 位后取 1 位，那么 **T**1 取值为 2，往后的以此类推。最后的 **T**9，因为对 **S** 字符串舍弃了 9 位后取 1 位，没字符可取，所以 **T**9 是空值。
怎么解决这个问题呢？我们的第一反应就是设置一个变量 **j**，其值比 **i** 小 1，然后在 **T%%i=!S:~%%i,1!**这里赋值时，将后面的 **i** 改为 **j**，问题就解决啦。但是，问题来了，这个 **j** 在**!S:~%%i,1!**这里怎么表达？由于用了延迟变量，那么我们对 **j** 就自然会想到用**!j!**表达，于是写成 **T%%i=!S:~!j!,1!**。这个代码无法执行，从计算机角度来看，它认为感叹号的范围是是 **T%%i=!S:~!**和 **j** 和**!,1!**三部分，自然无法正确执行。
代码可以改为这样：

```
@echo off
setlocal enabledelayedexpansion
set /p S=
for /l %%i in (0,1,8) do (
    set T%%i=!S:~%%i,1!
    )
for /l %%i in (0,1,8) do (echo T%%i=!T%%i!)
pause
```

但这样，数组下标就不是从 1 开始，而是从 0 开始到 8。
如果我们非得要数组下标从 1 开始到 9，可以这样变通一下：在读入一个字符串后给字符串添加前导空格就行了

```
@echo off
setlocal enabledelayedexpansion
set /p Tem=
set S= %Tem%
for /l %%i in (1,1,9) do (
```

```
    ::舍弃数组 S 前 i 位后取 1 位。
    set T%%i=!S:~%%i,1!
    )
for /l %%i in (1,1,9) do (echo T%%i=!T%%i!)
pause
```

上面代码先用一个临时变量 **Tem** 读入字符串，然后再用 **set S=%Tem%**给字符串添加一个空格，于是运行结果为：

```
123456789
T1=1
T2=2
T3=3
T4=4
T5=5
T6=6
T7=7
T8=8
T9=9
请按任意键继续. . .
```

其实，不想对数组进行预处理，数组下标要从 1 开始到 9 也是可以的，就是赋值时需要两步，代码效率会稍有下降：

```
@echo off
setlocal enabledelayedexpansion
set /p S=
for /l %%i in (1,1,9) do (
    ::对数组 S 去掉前 0 位后取 1 位
    set T%%i=!S:~0,1!
    ::将数组 S 删除前 1 位后取剩下所有值
    set S=!S:~1!
    )
for /l %%i in (1,1,9) do (echo T%%i=!T%%i!)
pause
```

上面的字符串转换为数组，我们一开始知道字符串长度就是 9，所以赋值时有所准备。如果读入的字符串是任意长度，那么转换之前就必须对字符串的长度进行测量，否则在转换超出字符串长度的时候，会出现错误，切记！

如何测量一串字符串的长度？

假如我们读入一串字符串为 **str**

```
set long=0
set S=%str%
:len
set /a long+=1
set S=%S:~0,-1%
if defined S goto :len
```

说明：最后 **long** 就是数组 **str** 的长度。对于 **set S=%str%**这一步，是因为下面的测量行为会破坏被测量字符串直到变为空值。
为了不破坏原来字符串，所以增加这一步赋值。如果我们仅仅需要知道字符串长度，那么这一步可以省去。
set S=%S:~0,-1%这一步意思是将被测量字符串每次都去掉末尾一位后剩余的值。
if defined S goto :len 如果字符串还存在，那么循环继续，直到空值为止。

我们在做高精度计算时，例如高精度加法，为了便于对齐数组，我们需要将字符串赋值到数组末端。这个时候怎么转换？

```
@echo off
setlocal enabledelayedexpansion
title 字符串转换为数组，在数组末端对齐
echo                                字符串转换为数组，在数组末端对齐
set /p num=请输入一个数字：

::这里采用的是固定数组长度方式，一开始预定的数组长度为 group 长。
set group=20

::计算出字符串长度 M。
set M=0
set str=!num!
:len
set /a M+=1
set str=!str:~0,-1!
if defined str goto len

::有必要对数组 X 进行清零预处理，否则后面输出是空值，无法输出，空值元素和别的数进行计算也会出错。
for /l %%i in (1,1,!group!) do (set X%%i=0)
```

::因为下面字符串转换为数组时，循环数不能大于字符串长度，否则赋值出错，所以上面要先计算出字符串长度。
::因为批处理无法从字符串末端开始一个个剥离字符串元素，所以要用几次**%var:~m,n%**形式进行剥离并赋值。

```
set /a p=-!M!
set N=!group!
for /l %%i in (-1,-1,!p!) do (
    set tem=!num:~%%i!
    set tem= !tem!
    set tem=!tem:~1,1!
    set X!N!=!tem!
    set /a N-=1
)
echo. & echo.
echo  本次预定的数组长度为!group!
::输出数组。这里是输出有前导 0 的情况。
echo.
echo  输出数组数，前导 0 也输出。
for /l %%i in (1,1,!group!) do (set /p var=!X%%i!<nul)
echo. & echo.
echo  输出数组数，前导 0 不输出。
set boor=0
for /l %%i in (1,1,!group!) do (
    if !X%%i! neq 0 set boor=1
    if !boor!==1 (set /p var=!X%%i!<nul)
)
echo.&echo.&pause>nul
```

说明：
上面的转换过程

```
for /l %%i in (-1,-1,!p!) do (
    set tem=!num:~%%i!
    set tem= !tem!
    set tem=!tem:~1,1!
    set X!N!=!tem!
    set /a N-=1
)
```

比较繁琐，代码可以改变如下：

```
@echo off
setlocal enabledelayedexpansion
title 字符串转换为数组，在数组末端对齐
echo                                    字符串转换为数组，在数组末端对齐
set /p num=请输入一个数字：
set group=20

set M=0
set str=!num!
:len
set /a M+=1
set str=!str:~0,-1!
if defined str goto len

for /l %%i in (1,1,!group!) do (set X%%i=0)

set /a N=!group!-M+1
for /l %%i in (!N!,1,!group!) do (
    set X%%i=!num:~0,1!
    set num=!num:~1!
)

echo. & echo.
echo 本次预定的数组长度为!group!
echo. & echo 输出数组数，前导 0 也输出。
for /l %%i in (1,1,!group!) do (set /p var=!X%%i!<nul)
echo. & echo. & echo 输出数组数，前导 0 不输出。
set boor=0
for /l %%i in (1,1,!group!) do (
    if !X%%i! neq 0 set boor=1
    if !boor!==1 (set /p var=!X%%i!<nul)
)
echo.&echo.&pause>nul
```

读入一串数字字符串，将其转换为数组。
要求：因为数组用来模拟高精度计算，所以小数点建议不要转换到数组里面去，否则需要判断情况会变得复杂，妨碍计算。不将小数点转换到数组里去，但需要知道小数点的位置。如何办到？
用 **for** 语句的**/f** 参数可以做到，它可以智能判别小数点前和小数点后

的数字。但是，**/f** 参数是对文件进行操作，我们这样写：

```
@echo off
set /p num=
for /f "tokens=1 delims=." %%i in (%num%) do echo %%i
pause
```

这样写是不可以的，计算机会显示：找不到文件。那我们可以临时创建文件：

```
@echo off
set /p num=
echo %num%>num.txt
for /f "tokens=1 delims=." %%i in (num.txt) do echo %%i
del num.txt
pause
```

上面代码临时建立文本文件《**num.txt**》，虽然用完后顺手将其删除，但毕竟是借助了外部力量。有没有办法不借助外部力量来实现呢？答案是肯定的，那就是对**%num%**添加双引号后，计算机就可以对齐像处理文本那样处理了。

```
@echo off
set    num=3.1415926
for /f "tokens=1,2 delims=." %%i in ("%num%") do (set num1=%%i
& set num2=%%j)
pause
```

只要有了上面这个转换，分离出小数部分和整数部分，接下来的转换为数组的处理就简单啦。

高精度正负浮点数加减法

说明：为了便于读者理解批处理的运算原理，加减乘除原理都采用模拟手工计算这种最原始的算法。

```
@echo off
setlocal enabledelayedexpansion
title 高精度加减法程序
作者 MHL     QQ1208980380
echo. & echo.
echo                                        高精度加减法程序
echo. & echo.
echo        本程序支持正负小数或整数加减法运算，输入时按照数学表达式输入并按回车键。
echo  例如: 1.23-2.1    又例如: -5.6+456
echo. & echo.
::输入表达式
set /p equation=
echo.
echo  计算结果是
echo.
::判断第一个数的正负号
set x=!equation:~0,1!
if "!x!"=="-" (
        set x=0
        set equation=!equation:~1!
) else (set x=1)

::将表达式拆解，并判断第二个数正负号
for /f "tokens=1,2 delims=+" %%i in ("!equation!") do (set
num1=%%i& set num2=%%j) & set y=1
if "!num2!"=="" for /f "tokens=1,2 delims=-" %%i in ("!equation!")
do (set num1=%%i& set num2=%%j) & set y=0
::将两个数拆解为整数部分和小数部分
for /f "tokens=1,2 delims=." %%i in ("!num1!") do (set strA=%%i&
set strB=%%j)
for /f "tokens=1,2 delims=." %%i in ("!num2!") do (set strC=%%i&
set strD=%%j)
```

```
set /a LA=LB=LC=LD=0
set str1=!strA!
set str3=!strC!
::测量出两个数的整数部分长度
:len1
set /a LA+=1
set str1=!str1:~0,-1!
if defined str1 goto len1

:len3
set /a LC+=1
set str3=!str3:~0,-1!
if defined str3 goto len3
::测量出两个数的小数部分长度
if defined strB (
	set str2=!strB!
	:len2
	set /a LB+=1
	set str2=!str2:~0,-1!
	if defined str2 goto len2
)

if defined strD (
	set str4=!strD!
	:len4
	set /a LD+=1
	set str4=!str4:~0,-1!
	if defined str4 goto len4
)
::两个数去掉小数点
set num1=%num1:.=%
set num2=%num2:.=%
set /a Lmaxnum1=LA+LB+LD
set /a Lmaxnum2=LB+LC+LD

set /a Lv=LA+LB+LC+LD
for /l %%i in (1,1,!Lv!) do set /a t%%i=u%%i=v%%i=0

::将两个数转换为数组
for /l %%i in (!Lmaxnum1!,-1,!LD!) do (
	set t%%i=!num1:~0,1!
	set num1=!num1:~1!
```

```
)

for /l %%i in (!Lmaxnum2!,-1,!LB!) do (
        set u%%i=!num2:~0,1!
        set num2=!num2:~1!
)

::判断要进行加法还是减法运算
if !x!==!y! (
        if !x!==0 set symbol=-
        goto :additive
) else goto :subtraction

::加法运算
:additive
set jin=0
for /l %%i in (1,1,!Lv!) do (
        set /a v%%i=t%%i+u%%i
        set /a jin=jin/10+v%%i
        set /a v%%i=jin%%10
)
goto :print

::减法运算
:subtraction
::判断两个去掉符号的数的大小，以此确定结果是正数还是负数
for /l %%i in (!Lv!,-1,1) do (
        if !t%%i! gtr !u%%i! (
          set BOS=1
          if !x!==0 set symbol=-
          goto :operation
        ) else if !t%%i! lss !u%%i! (
          set BOS=0
          if !y!==0 set symbol=-
          goto :operation
        )
)
echo 0 & goto :end

:operation
::进行减法运算，大数减去小数
for /l %%i in (1,1,!Lv!) do if !BOS!==1 (set /a v%%i=t%%i-u%%i) else
```

```
(set /a v%%i=u%%i-t%%i)
::进位处理
for /l %%i in (1,1,!Lv!) do (
        if !v%%i! lss 0 (
          set /a v%%i+=10
          set /a j=%%i+1
          set /a v!j!-=1
        )
)

::输出最后结果
:print
::输出结果符号，要注意判断是否所有数字都是 0，否则像-0-0 就可能输出为-0，多了一个负号。
set boo=0
for /l %%i in (1,1,!Lv!) do if !v%%i! neq 0 set boo=1
if "!symbol!"=="-" if !boo! neq 0 set /p var=-<nul
::去掉前导 0，输出整数部分
set boor=0
set /a Lint=LB+LD+1
for /l %%i in (!Lv!,-1,!Lint!) do (
        if !v%%i! gtr 0 set /a boor+=1
        if !boor! neq 0 set /p var=!v%%i!<nul
)
if !boor!==0 set /p var=!boor!<nul
::去掉后继 0，输出小数部分。
set /a Lint=LB+LD
if !Lint! gtr 0 (
        for /l %%i in (1,1,!Lint!) do (
          if !v%%i! gtr 0 (
               set /p var=.<nul
               for /l %%j in (!Lint!,-1,%%i) do set /p var=!v%%j!<nul
               goto :end
          )
        )
)

:end
echo. & echo. & echo  计算结束，按任意键退出。& pause>nul & exit
```

++

对上面“高精度正负浮点数加减法”压缩代码如下：

```
　　@echo off&title 高精度加减法程序
作者 MHL　QQ1208980380
　　setlocal enabledelayedexpansion&echo. & echo.&echo
高精度加减法程序
　　echo. & echo.&echo　　本程序支持正负小数或整数加减法运
算，输入时按照数学表达式输入并按回车键。
　　echo 例如: 1.23-2.1　又例如: -5.6+456&echo. & echo.&set /p
equation=请输入算式　&echo.&echo 计算结果是&echo.
　　set x=!equation:~0,1!
　　if "!x!"=="-" (set x=0&set equation=!equation:~1!) else set x=1
　　for /f "tokens=1,2 delims=+" %%i in ("!equation!") do (set
num1=%%i& set num2=%%j) & set y=1
　　if "!num2!"=="" for /f "tokens=1,2 delims=-" %%i in
("!equation!") do (set num1=%%i& set num2=%%j) & set y=0
　　for /f "tokens=1,2 delims=." %%i in ("!num1!") do (set
strA=%%i& set strB=%%j)
　　for /f "tokens=1,2 delims=." %%i in ("!num2!") do (set
strC=%%i& set strD=%%j)
　　set /a LA=LB=LC=LD=0
　　set str1=!strA!&set str2=!strB!&set str3=!strC!&set str4=!strD!
　　set num1=%num1:.=%&set num2=%num2:.=%
　　:len1
　　set /a LA+=1&set str1=!str1:~0,-1!
　　if defined str1 goto len1
　　:len3
　　set /a LC+=1&set str3=!str3:~0,-1!
　　if defined str3 goto len3
　　if defined strB (
　　  :len2
　　  set /a LB+=1&set str2=!str2:~0,-1!
　　  if defined str2 goto len2
　　)
　　if defined strD (
　　  :len4
　　  set /a LD+=1&set str4=!str4:~0,-1!
　　  if defined str4 goto len4
　　)
　　set /a
Lmaxnum1=LA+LB+LD,Lmaxnum2=LB+LC+LD,Lv=LA+LB+LC+
LD,Lint1=LB+LD+1,Lint2=LB+LD,boor=boo=jin=0
```

```
    for /l %%i in (1,1,!Lv!) do set /a t%%i=u%%i=v%%i=0
    for /l %%i in (!Lmaxnum1!,-1,!LD!) do set
t%%i=!num1:~0,1!&set num1=!num1:~1!
    for /l %%i in (!Lmaxnum2!,-1,!LB!) do set
u%%i=!num2:~0,1!&set num2=!num2:~1!
    if !x!==!y! (
       if !x!==0 set symbol=-
       for /l %%i in (1,1,!Lv!) do set /a v%%i=t%%i+u%%i&set /a
jin=jin/10+v%%i&set /a v%%i=jin%%10
       goto :print
    )
    for /l %%i in (!Lv!,-1,1) do if !t%%i! gtr !u%%i! (
       if !x!==0 set symbol=-
       set BOS=1&goto :operation
    ) else if !t%%i! lss !u%%i! (
       if !y!==0 set symbol=-
       set BOS=0&goto :operation
    )
    echo 0&goto :end
    :operation
    for /l %%i in (1,1,!Lv!) do if !BOS!==1 (set /a v%%i=t%%i-u%%i)
else (set /a v%%i=u%%i-t%%i)
    for /l %%i in (1,1,!Lv!) do if !v%%i! lss 0 set /a v%%i+=10&set /a
j=%%i+1&set /a v!j!-=1
    :print
    for /l %%i in (1,1,!Lv!) do if !v%%i! neq 0 set boo=1
    if "!symbol!"=="-" if !boo! neq 0 set /p var=-<nul
    for /l %%i in (!Lv!,-1,!Lint1!) do (
       if !v%%i! gtr 0 set /a boor+=1
       if !boor! neq 0 set /p var=!v%%i!<nul
    )
    if !boor!==0 set /p var=!boor!<nul
    if !Lint2! gtr 0 for /l %%i in (1,1,!Lint2!) do if !v%%i! gtr 0 (
       set /p var=.<nul
       for /l %%j in (!Lint2!,-1,%%i) do set /p var=!v%%j!<nul
       goto :end
    )
    :end
    echo.&echo.&echo 计算结束，按任意键退出。&pause>nul&exit
```

高精度正负浮点数乘法

```
@echo off
setlocal enabledelayedexpansion
title 高精度乘法程序　　作者 MHL QQ1208980380
echo.&echo.
echo                                    高精度乘法程序
echo.
echo                                  本程序运算支持正负浮点数
echo.&echo.
set /p num1=请输入第一个数
set /p num2=请输入第二个数
echo.&echo.
echo 计算结果
echo.
::符号判断
set /a symbol1=symbol2=boo=0
set sym1=!num1:~0,1!
set sym2=!num2:~0,1!
if "!sym1!"=="-" set symbol1=1&set num1=!num1:~1!
if "!sym2!"=="-" set symbol2=1&set num2=!num2:~1!
::将两个浮点数分别分为整数部分和小数部分，共计四段。
for /f "tokens=1,2 delims=." %%i in ("!num1!") do (set strA=%%i&
set strB=%%j)
for /f "tokens=1,2 delims=." %%i in ("!num2!") do (set strC=%%i&
set strD=%%j)
set /a LA=LB=LC=LD=0
set str1=!strA!
set str2=!strB!
set str3=!strC!
set str4=!strD!
::测量第一个数整数部分长度
:len1
set /a LA+=1
set str1=!str1:~0,-1!
if defined str1 goto len1
::测量第二个数整数部分长度
:len3
set /a LC+=1
set str3=!str3:~0,-1!
if defined str3 goto len3
::测量第一个数小数部分长度
```

```
if defined strB (
        :len2
        set /a LB+=1
        set str2=!str2:~0,-1!
        if defined str2 goto len2
)
::测量第二个数小数部分长度
if defined strD (
        :len4
        set /a LD+=1
        set str4=!str4:~0,-1!
        if defined str4 goto len4
)
::将两个数去掉小数点
set num1=%num1:.=%
set num2=%num2:.=%
set /a
Lmaxnum1=LA+LB,Lmaxnum2=LC+LD,LZ=LA+LB+LC+LD,jin=
0,k=LB+LD+1,kk=LB+LD
::对积数组清零
for /l %%i in (1,1,!LZ!) do set /a Z%%i=0
::将两个乘数转换为数组
for /l %%i in (!Lmaxnum1!,-1,1) do (
        set X%%i=!num1:~0,1!
        set num1=!num1:~1!
)
for /l %%i in (!Lmaxnum2!,-1,1) do (
        set Y%%i=!num2:~0,1!
        set num2=!num2:~1!
)
::乘法运算
for /l %%i in (1,1,!Lmaxnum1!) do (
        for /l %%j in (1,1,!Lmaxnum2!) do (
            set /a f=%%i+%%j-1
            set /a Z!f!+=X%%i*Y%%j
        )
)
::进位计算
for /l %%i in (1,1,!LZ!) do (
        set /a jin=jin/10+Z%%i
        set /a Z%%i=jin%%10
)
```

```
for /l %%i in (!LZ!,-1,!k!) do (
        if !Z%%i! neq 0 (
            set /a kkk=%%i
            goto :pri
        )
)
set /a kkk=LB+LD+1

::输出
:pri
::判断数组是否全是 0，这个判断是为了避免出现“-0”乘以“0”等于“-0”的情况
for /l %%i in (1,1,!kkk!) do if !Z%%i! neq 0 set boo=1
::判断结果是否负数
if "!symbol1!" neq "!symbol2!" if !boo!==1 set /p var=-<nul
::输出积的整数部分
for /l %%i in (!kkk!,-1,!k!) do set /p var=!Z%%i!<nul
::输出积的小数部分。注意，如果小数部分全是 0，那么不输出后继 0 以及小数点，以符合我们的习惯。
if !kk! gtr 0 for /l %%i in (1,1,!kk!) do (
        if !Z%%i! gtr 0 (
            set /p var=.<nul
            for /l %%j in (!kk!,-1,%%i) do set /p var=!Z%%j!<nul
            goto :end
        )
)
:end
echo.&echo.
echo  计算结束，按任意键退出。
pause>nul
```

++

对上面“高精度正负浮点数乘法”压缩代码如下：

```
@echo off&setlocal enabledelayedexpansion&title 高精度乘法程序 作者 MHL QQ1208980380
echo. & echo.&echo                                                    高精度乘法程序&echo.
echo                                                    本程序运算支持浮点数和负数
echo.&echo.&set /p num1=请输入第一个数       &set /p num2=请输
```

```
入第二个数　　&echo. & echo.&echo  计算结果&echo.
set /a symbol1=symbol2=boo=0


set sym1=!num1:~0,1!&set sym2=!num2:~0,1!

if "!sym1!"=="-" set symbol1=1&set num1=!num1:~1!
if "!sym2!"=="-" set symbol2=1&set num2=!num2:~1!


for /f "tokens=1,2 delims=." %%i in ("!num1!") do (set strA=%%i&
set strB=%%j)
for /f "tokens=1,2 delims=." %%i in ("!num2!") do (set strC=%%i&
set strD=%%j)
set /a LA=LB=LC=LD=0&set str1=!strA!&set str2=!strB!&set
str3=!strC!&set str4=!strD!
:len1
set /a LA+=1&set str1=!str1:~0,-1!
if defined str1 goto len1
:len3
set /a LC+=1&set str3=!str3:~0,-1!
if defined str3 goto len3
if defined strB (
        :len2
        set /a LB+=1&set str2=!str2:~0,-1!
        if defined str2 goto len2
)
if defined strD (
        :len4
        set /a LD+=1&set str4=!str4:~0,-1!
        if defined str4 goto len4
)
set num1=%num1:.=%&set num2=%num2:.=%
set /a
Lmaxnum1=LA+LB,Lmaxnum2=LC+LD,LZ=LA+LB+LC+LD,jin=
0,k=LB+LD+1,kk=LB+LD
for /l %%i in (1,1,!LZ!) do set /a Z%%i=0
for /l %%i in (!Lmaxnum1!,-1,1) do set X%%i=!num1:~0,1!&set
num1=!num1:~1!
for /l %%i in (!Lmaxnum2!,-1,1) do set Y%%i=!num2:~0,1!&set
num2=!num2:~1!
for /l %%i in (1,1,!Lmaxnum1!) do for /l %%j in (1,1,!Lmaxnum2!)
do set /a f=%%i+%%j-1&set /a Z!f!+=X%%i*Y%%j
```

```
for /l %%i in (1,1,!LZ!) do set /a jin=jin/10+Z%%i&set /a
Z%%i=jin%%10
for /l %%i in (!LZ!,-1,!k!) do if !Z%%i! neq 0 set /a kkk=%%i &
goto :pri
set /a kkk=LB+LD+1
:pri
for /l %%i in (1,1,!kkk!) do if !Z%%i! neq 0 set boo=1
if "!symbol1!" neq "!symbol2!" if !boo!==1 set /p var=-<nul
for /l %%i in (!kkk!,-1,!k!) do set /p var=!Z%%i!<nul
if !kk! gtr 0 for /l %%i in (1,1,!kk!) do if !Z%%i! gtr 0 (
        set /p var=.<nul&for /l %%j in (!kk!,-1,%%i) do set /p
var=!Z%%j!<nul
        goto :end
)
:end
echo.&echo.&echo 计算结束，按任意键退出。&pause>nul
```

高精度正负浮点数除法

```
@echo off
setlocal enabledelayedexpansion
title 高精度除法程序                                    作者 MHL
QQ1208980380
echo. & echo.
echo                                          高精度除法程序
echo.
echo                                本程序支持小数或整数除法运算，不支持
负数。
echo.&echo.
set /p num1=请输入被除数
set /p num2=请输入除数
set /p degree=请输入精确到小数点后几位?
::degr 的用处是，当输入的 degree=0 时，为了使循环退出而设置。
set /a degr=degree
echo.
::测量出被除数的整数部分长度，除数的整数部分长度和小数部分长
度
for /f "delims=." %%i in ("!num1!") do (set strA=%%i)
```

```
for /f "tokens=1,2 delims=." %%i in ("!num2!") do (set strC=%%i&
set strD=%%j)

set /a LA=LB=LC=LD=0
set str1=!strA!
set str3=!strC!

:len1
set /a LA+=1
set str1=!str1:~0,-1!
if defined str1 goto len1

:len3
set /a LC+=1
set str3=!str3:~0,-1!
if defined str3 goto len3

if defined strD (
    set str4=!strD!
    :len4
    set /a LD+=1
    set str4=!str4:~0,-1!
    if defined str4 goto len4
)

::两个数去掉小数点，并将 num2 转换为数组
set num1=%num1:.=%
set num2=%num2:.=%

set /a Lmaxnum2=LC+LD
set /a zero=0
for /l %%i in (!Lmaxnum2!,-1,1) do (
    set /a X%%i=0
    set /a Y%%i=!num2:~0,1!
    if !Y%%i! neq 0 set /a zero=1
    set num2=!num2:~1!
)

::如果除数等于 0，除法无法进行，违反运算规则，计算自动退出。
if !zero!==0 echo  除数不可以等于 0，输入错误。  & goto :end

set /a k=LC+LD+1
```

```
set /a LA=LA+LD+1
set /a X!k!=Y!k!=point=boor=0

::除法开始
:division
for /l %%i in (!k!,-1,1) do (
    ::比较两个数组的大小，这里是大于的情况
    if !X%%i! gtr !Y%%i! goto :loopmax

    ::比较两个数组的大小，这里是小于的情况
    if !X%%i! lss !Y%%i! (
        for /l %%j in (!k!,-1,2) do (
            set /a tem=%%j-1
            set /a X%%j=X!tem!
        )
        if defined num1 (
            set X1=!num1:~0,1!
            set num1=!num1:~1!
        ) else   set X1=0
        set /a LA-=1
        if !LA!==0 (
            if !boor!==0 set /p var=!boor!<nul
            if !degr!==0 goto :end
            set /p var=.<nul
            set /a boor+=1
            set /a point=1
        )
        for /l %%e in (!k!,-1,1) do (
            if !X%%e! gtr !Y%%e! goto :next1
            if !X%%e! lss !Y%%e! (
                if !boor! gtr 0 (
                    set /p var=!LB!<nul
                    if !point!==1 set /a degree-=1
                    ::这里一定要判断被除数的整数部分是不是已经除完了，接着判断是不是达到所需精度了才可以退出。否则如果只判断精度是否达到要求，那么整数部分可能还未除完就退出了。
                    if !LA! leq 0 if !degree! leq 0 goto :end
                    goto :next1
                )
            )
        )
        :next1
```

```
        goto :division
    )
)
```

::如果能运行到这里，说明两个数组一样大，可以按照大于的情况处理（它的这一步的商会等于 1）。

```
:loopmax
set /a shang=0
:loopmin
for /l %%j in (1,1,!k!) do set /a X%%j-=Y%%j
for /l %%j in (1,1,!k!) do (
    if !X%%j! lss 0 (
        set /a X%%j+=10
        set /a tem=%%j+1
        set /a X!tem!-=1
    )
)
set /a shang+=1
for /l %%i in (!k!,-1,1) do (
    if !X%%i! gtr !Y%%i! goto :loopmin
    if !X%%i! lss !Y%%i! goto :next
)
goto :loopmin

:next
set /p var=!shang!<nul
set /a boor+=1
if !point!==1 set /a degree-=1

::如果小数精度未达到目的，则循环继续
if !degree! gtr 0 goto :division

::如果被除数的整数部分还有数字，循环继续。增加这一句，是为了防止请求的小数位数等于 0 的情况，会令即使被除数仍然有整数，除法也无法继续进行。
if !LA! gtr 0 goto :division
:end
echo.&echo.&echo 计算结束，按任意键退出。&pause>nul
```

++

对上面“高精度正负浮点数除法”压缩代码如下：

```
@echo off&setlocal enabledelayedexpansion
title 高精度除法程序                                    作者
MHL QQ1208980380
echo. & echo.&echo                                      高精度除
法程序&echo.
echo                          本程序支持小数或整数除法运算，不支持
负数。&echo.&echo.
set /p num1=请输入被除数                    &set /p num2=请输入除
数
set /p degree=请输入精确到小数点后几位?  &set /a
degr=degree&echo.
for /f "delims=." %%i in ("!num1!") do (set strA=%%i)
for /f "tokens=1,2 delims=." %%i in ("!num2!") do (set strC=%%i&
set strD=%%j)
set /a LA=LB=LC=LD=0&set str1=!strA!&set str3=!strC!
:len1
set /a LA+=1&set str1=!str1:~0,-1!
if defined str1 goto len1
:len3
set /a LC+=1&set str3=!str3:~0,-1!
if defined str3 goto len3
if defined strD (
    set str4=!strD!
    :len4
    set /a LD+=1&    set str4=!str4:~0,-1!
    if defined str4 goto len4
)
set num1=%num1:.=%&set num2=%num2:.=%
set /a Lmaxnum2=LC+LD,zero=0
for /l %%i in (!Lmaxnum2!,-1,1) do (
    set /a X%%i=0&set Y%%i=!num2:~0,1!&set num2=!num2:~1!
    if !Y%%i! neq 0 set zero=1
)
if !zero!==0 echo  除数不可以等于 0,输入错误。&goto :end
set /a k=LC+LD+1,LA=LA+LD+1&set /a X!k!=Y!k!=point=boor=0
:division
for /l %%i in (!k!,-1,1) do (
    if !X%%i! gtr !Y%%i! goto :loopmax
    if !X%%i! lss !Y%%i! (
```

```
        for /l %%j in (!k!,-1,2) do set /a tem=%%j-1&set /a
X%%j=X!tem!
        if defined num1 (set X1=!num1:~0,1!&set num1=!num1:~1!)
else set X1=0
        set /a LA-=1
        if !LA!==0 (
            if !boor!==0 set /p var=!boor!<nul
            if !degr!==0 goto :end
            set /p var=.<nul&set /a boor+=1,point=1
        )
        for /l %%e in (!k!,-1,1) do (
            if !X%%e! gtr !Y%%e! goto :next1
            if !X%%e! lss !Y%%e! if !boor! gtr 0 (
                set /p var=!LB!<nul
                if !point!==1 set /a degree-=1
                if !LA! leq 0 if !degree! leq 0 goto :end
                goto :next1
            )
        )
        :next1
        goto :division
    )
)
:loopmax
set /a shang=0
:loopmin
for /l %%j in (1,1,!k!) do set /a X%%j-=Y%%j&if !X%%j! lss 0 set /a
X%%j+=10,tem=%%j+1&set /a X!tem!-=1
set /a shang+=1
for /l %%i in (!k!,-1,1) do (
    if !X%%i! gtr !Y%%i! goto :loopmin
    if !X%%i! lss !Y%%i! goto :next
)
goto :loopmin
:next
set /p var=!shang!<nul&set /a boor+=1
if !point!==1 set /a degree-=1
if !degree! gtr 0 goto :division
if !LA! gtr 0 goto :division
:end
echo.&echo.&echo 计算结束，按任意键退出。&pause>nul
```

高精度开平方一（网友作品，计算速度非常快）

```
@echo off
title 正整数高精度开方程序
echo. & echo 本程序可以对正整数开方，精度不限。不能对负数、0和小数开方。
setlocal enabledelayedexpansion
set s=0
echo. & echo.
set /p s=请输入被开方数:
set /p w=精确到小数点后几位数?:
echo.
if defined W (
    for /l %%i in (1,1,%W%) do (
        set s=!s!00
    )
)else set W=0
set p=!s!
set len=0
set N=0
for %%i in (4096,2048,1024,512,256,128,64,32,16,8,4,2,1) do (
    IF "!p:~%%i!" NEQ "" (
        set/a len+=%%i
        set p=!p:~%%i!
    )
)
set /a N=N-~(len%%2)
set M=!s:~,%N%!
for /l %%i in (1,1,9) do (
    set/a Mx=%%i*%%i
    if !Mx! leq !M! (
        set i=%%i
        set /a j=100+M-Mx
    )
)
set /a len-=1
set /a Len_i=i/5+1
set /a _N=i/5+1
set /a p=i*20
set j=!j:~-%_N%!
set p=0!p!
```

```
set kl=0000000
set /a _N=8-_N
for /l %%i in (%N%,2,!len!) do (
    set j=!j!!s:~%%i,2!
    if "!j:0=!" neq "" (
        set /a Ln_i+=2
        set /a Len_i+=2
        if !p! lss !j! (
            set d=Z
            set in=!kl!!P!
            set /a Ln_i+=7
            for /l %%j in (9,-1,2) do (
                if "!d!" gtr "!j!" (
                    set x=%%j
                    set d=
                    set /a b=x*x
                    for /l %%k in (8,8,!Ln_i!) do (
                        set /a b=1!in:~-%%k,8!*%%j+b
                        set d=!b:~-8!!d!
                        set /a b=!b:~,-8!-%%j
                    )
                set d=!b!!d!
                for %%k in (!Len_i!) do set d=!d:~-%%k!
            )
        )
        if !d! gtr !j! (
            set d=!in!
            set x=1
            set b=1
        ) else (
            set d=!kl!!d!
            set b=0
        )
        set j=!kl!!j!
        set t=
        for /l %%j in (8,8,!Ln_i!) do (
            set /a b=3!j:~-%%j,8!-1!d:~-%%j,8!-!b:~,1!%%2
            set t=!b:~1!!t!
        )
        for %%j in (!Len_i!) do set j=!t:~-%%j!
        set "j=!j:~1!" ) else set x=0
        set /a Len_i-=1
        if !x! neq 0 (
```

```
            if !x! geq 5 (
                set p=
                set b=0
                set in=!kl!!i!!x!
                set /a Ln_i=Len_i+_N
                for /l %%j in (8,8,!Ln_i!) do (
                    set /a b=1!in:~-%%j,8!*2+!b:~,1!%%2
                    set p=!b:~1!!p!
                )
                set /a b=!b:~,1!%%2
                for %%j in (!Len_i!) do set p=!b:1=01!!p:~-%%j!0
            ) else (
                set /a t=x*2
                set p=!p:~,-1!!t!0
            )
        ) else (
            set p=!p!0
            set j=!j:~1!
        )
    ) else (
        set j=!j:~1!
        set p=!p!0
        set /a Len_i+=1
        set x=0
    )
    set i=!i!!x!
)
for /f "tokens=* delims=." %%i in ("!i:~,-%W%!.!i:~-%W%!") do set
sqrt=%%i
echo %sqrt%
echo. & echo  计算结束，按任意键退出。& pause>nul

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

高精度开方二（模拟手工计算，支持正负浮点数，计算速度慢）

```
@echo off
setlocal enabledelayedexpansion&echo.&echo.
title  高精度开平方程序                                    作者
MHL QQ1208980380
echo                                                高精度开平方程序
&echo.
echo                                              本程序支持正负浮点数计算
echo.&echo.
set /p num=请输入被开平方数
set /p degree=请输入平方根小数位数
echo.&echo.
set symbol=!num:~0,1!
if "!symbol!"=="-" set num=!num:~1!
::测量出被开方数整数部分长度 LN（一位一位数）
for /f "tokens=1,2 delims=." %%i in ("!num!") do (set strA=%%i&
set strB=%%j)
set str1=!strA!
set /a LN=LF=0
:len1
set /a LN+=1
set str1=!str1:~0,-1!
if defined str1 goto len1

if defined strB (
    set str2=!strB!
    :len2
    set /a LF+=1
    set str2=!str2:~0,-1!
    if defined str2 goto len2
)
set num=!num:.=!
::转换整数部分第一位数
set /a var1=LN%%2
set /a begin=1
if !var1!==1 (
    set Snum1=!num:~0,1!
    set num=!num:~1!
    set /a LN+=1
    set /a begin=2
```

```
)
set /a var2=LF%%2
if !var2!==1 (set num=!num!0& set /a LF+=1)
::Linteger 转换后两位数被开方数数组整数部分长度，等于 0 时要及时输出小数点
set /a Linteger=LN/2
::root 数组存储根（一位数一位数存储），长度 Lroot 等于整数开方后的位数与请求的小数精度之和
set /a Lroot=Linteger+degree
set /a Lr=Lroot-1
for /l %%i in (1,1,!Lroot!) do set root%%i=0
::转换被开放数的所有数字（转换成两位数两位数），转换到数组 Snum
set num=!num:.=!
::C 是转换后数组的总长度，超过 C 这个长度，余数补充数字的时候就应该补充 000 了。

set /a C=^(LN+LF^)/2
for /l %%i in (!begin!,1,!C!) do (
    for %%j in (10,1) do (
        set /a Snum%%i+=!num:~0,1!*%%j
        set num=!num:~1!
    )
)
::cou 计数，计量当前商算到几位了（一位一位数）
set /a cou=1,first=9
for %%i in (81,64,49,36,25,16,9,4,1,0) do (
    if %%i leq !Snum1! (
        set /a var=%%i
        goto :break
    )
    set /a first-=1
)

:break
set /a root1=first
set /p var=!root1!<nul
set /a yushu1=Snum1-var
if !C! gtr 1 (
    set /a yushu2=Snum2
    set /a C-=2
) else set /a yushu2=0 & set /a C-=1
```

```
::coun 是被开方数转换成两位两位数字后的位数
set /a coun=2

:sqrt
set /a Lcoutem=cou%%2
if !Lcoutem!==1 (
    set /a Loopmin=cou+1,roottem!Loopmin!=0
    for /l %%i in (1,1,!cou!) do set /a roottem%%i=root%%i
) else (
    set /a Loopmin=cou+2,roottem1=0
    for /l %%i in (!Loopmin!,-1,2) do (
        set /a j=%%i-1
        set /a roottem%%i=root!j!
    )
)
::转换 root 一位数一位数根到 sumtem 数组成两位数两位数根
set /a count=Loopmin/2
for /l %%i in (1,1,!count!) do (
    set /a var2=%%i*2,var1=%%i*2-1
    set /a Sumtem%%i=^(roottem!var1!*10+roottem!var2!^)*2
)
set /a x=9

:loop
for /l %%i in (1,1,!count!) do set /a
Sumtt%%i=Sumtem%%i,Sum%%i=0
set /a Sumtt!count!+=x
for /l %%i in (1,1,!count!) do set /a Sum%%i+=Sumtt%%i*x

::这里是对 Sum 数组进位处理，使之可以和余数数组 yushu 比较大小.
set /a zhong=count+1
set /a jin=Sum!zhong!=0
for /l %%i in (!zhong!,-1,2) do (
    set /a kk=%%i-1
    set /a Sum%%i=^(Sum!kk!+jin^)%%100
    set /a jin=^(Sum!kk!+jin^)/100
)
set /a Sum1=jin

if !jin! geq 100 set /a Sum0=jin/100,Sum1=jin%%100
::将数组 Sum 和余数数组对齐，便于下面比较大小和减法运算
```

```
set /a f=zhong+1
for /l %%i in (!coun!,-1,1) do (
    set /a f-=1
    set /a Sum%%i=Sum!f!
    if !f! lss 0 set /a Sum%%i=0
)
::比较数组大小
for /l %%i in (1,1,!coun!) do (
    if !yushu%%i! gtr !Sum%%i! goto :next
    if !yushu%%i! lss !Sum%%i! (
        set /a x-=1
        goto :loop
    )
)

:next
set /a cou+=1
set /a root!cou!=x
set /a Linteger-=1
if !Linteger!==0 (
    if !degree!==0 goto :end
    set /p var=.<nul
)
::输出当前根
set /p var=!x!<nul
::减法运算，求出下次的余数
for /l %%i in (!coun!,-1,1) do (
    set /a yushu%%i-=Sum%%i
    if !yushu%%i! lss 0 (
        set /a j=%%i-1
        set /a yushu%%i+=100
        set /a yushu!j!-=1
    )
)

set /a coun+=1
if !C! gtr 0 (set /a yushu!coun!=Snum!coun!) else set /a
yushu!coun!=0
set /a C-=1,Lr-=1
if !Lr! gtr 0 goto :sqrt

:end
```

```
if "!symbol!"=="-" set /p var=i<nul
echo.&echo.&echo.
echo  计算结束，按任意键退出。
pause>nul
```

++

对上面“高精度开方二”代码压缩如下：

```
@echo off&setlocal enabledelayedexpansion&echo.&echo.
title  高精度开平方程序                                    作者
MHL QQ1208980380
echo                                        高精度开平方程序
&echo.
echo                                        本程序支持正负浮点数计算
&echo.&echo.
set /p num=请输入被开平方数          &set /p degree=请输入平方根
小数位数    &echo.&echo.
set symbol=!num:~0,1!
if "!symbol!"=="-" set num=!num:~1!
for /f "tokens=1,2 delims=." %%i in ("!num!") do set strA=%%i& set
strB=%%j
set str1=!strA!&set num=!num:.=!&set /a LN=LF=0
:len1
set /a LN+=1&set str1=!str1:~0,-1!
if defined str1 goto len1
if defined strB (
    set str2=!strB!
    :len2
    set /a LF+=1&set str2=!str2:~0,-1!
    if defined str2 goto len2
)
set /a var1=LN%%2,var2=LF%%2,begin=1
if !var1!==1 set Snum1=!num:~0,1!&set num=!num:~1!&set /a
LN+=1,begin=2
if !var2!==1 set num=!num!0& set /a LF+=1
set /a
Linteger=LN/2,Lroot=LN/2+degree,Lr=LN/2+degree-1,C=^(LN+LF
^)/2,cou=1,first=9
for /l %%i in (1,1,!Lroot!) do set root%%i=0
for /l %%i in (!begin!,1,!C!) do for %%j in (10,1) do set /a
Snum%%i+=!num:~0,1!*%%j&set num=!num:~1!
```

```
for %%i in (81,64,49,36,25,16,9,4,1,0) do (
    if %%i leq !Snum1! set /a var=%%i&goto :break
    set /a first-=1
)
:break
set /a root1=first,yushu1=Snum1-var,coun=2
set /p var=!root1!<nul
if !C! gtr 1 (set /a yushu2=Snum2,C-=2) else set /a yushu2=0&set /a
C-=1
:sqrt
set /a Lcoutem=cou%%2
if !Lcoutem!==1 (
    set /a Loopmin=cou+1,roottem!Loopmin!=0
    for /l %%i in (1,1,!cou!) do set /a roottem%%i=root%%i
) else (
    set /a Loopmin=cou+2,roottem1=0
    for /l %%i in (!Loopmin!,-1,2) do set /a j=%%i-1&set /a
roottem%%i=root!j!
)
set /a count=Loopmin/2,x=9
for /l %%i in (1,1,!count!) do set /a var2=%%i*2,var1=%%i*2-1&set
/a Sumtem%%i=^(roottem!var1!*10+roottem!var2!^)*2
:loop
for /l %%i in (1,1,!count!) do set Sum%%i=0&set /a
Sum%%i+=Sumtem%%i*x
set /a Sum!count!+=x*x,zhong=count+1,f=count+2,jin=0
for /l %%i in (!zhong!,-1,1) do set /a kk=%%i-1&set /a
Sum%%i=^(Sum!kk!+jin^)%%100&set /a jin=^(Sum!kk!+jin^)/100
for /l %%i in (!coun!,-1,1) do (
    set /a f-=1&set /a Sum%%i=Sum!f!
    if !f! lss 0 set Sum%%i=0
)
for /l %%i in (1,1,!coun!) do if !yushu%%i! gtr !Sum%%i! (goto :next)
else if !yushu%%i! lss !Sum%%i! set /a x-=1&goto :loop
:next
set /a cou+=1,Linteger-=1&set /a root!cou!=x
if !Linteger!==0 (
    if !degree!==0 goto :end
    set /p var=.<nul
)
set /p var=!x!<nul
for /l %%i in (!coun!,-1,1) do (
    set /a yushu%%i-=Sum%%i
```

```
    if !yushu%%i! lss 0 set /a j=%%i-1&set /a yushu%%i+=100&set /a yushu!j!-=1
)
set /a coun+=1
if !C! gtr 0 (set /a yushu!coun!=Snum!coun!) else set yushu!coun!=0
set /a C-=1,Lr-=1
if !Lr! gtr 0 goto :sqrt
:end
if "!symbol!"=="-" set /p var=i<nul
echo.&echo.&echo.&echo 计算结束，按任意键退出。&pause>nul
```

高精度正负浮点数开立方

```
@echo off
setlocal enabledelayedexpansion
title 高精度开立方程序                                    作者 MHL  QQ1208980380
echo.&echo.&echo.
echo                                        高精度开立方程序
echo.
echo                                    本程序支持正负浮点数计算
echo.
set /p num=请输入被开立方数
set /p degree=请输入立方根小数位数
echo.&echo.
::判断符号，由于负数的立方根仍旧是负数，所以只需要输出负号，其余一切照例开立方即可。
set symbol=!num:~0,1!
if "!symbol!"=="-" set num=!num:~1!&set /p var=-<nul
::测量出被开方数整数部分长度 LN（一位一位数）
for /f "tokens=1,2 delims=." %%i in ("!num!") do (set strA=%%i&
set strB=%%j)
set str1=!strA!
set /a LN=LF=0
:len1
set /a LN+=1
set str1=!str1:~0,-1!
if defined str1 goto len1
::测量小数部分长度 LF
```

```
if defined strB (
    set str2=!strB!
    :len2
    set /a LF+=1
    set str2=!str2:~0,-1!
    if defined str2 goto len2
)
::去除数组字符串的小数点
set num=!num:.=!
::转换整数部分第一位数（三位数三位数转换），之所以要这么转换，而不是补充前导 0 后统一三位数三位数转换，是因为有前导 0 的数字，计算机会认为是八进制而出错。
set /a var1=LN%%3
set /a begin=1
if !var1!==1 (
    set Snum1=!num:~0,1!
    set num=!num:~1!
    set /a LN+=2
    set /a begin=2
) else if !var1!==2 (
    set Snum1=!num:~0,2!
    set num=!num:~2!
    set /a LN+=1
    set /a begin=2
)
::判断小数部分长度，看末尾需要补一个还是两个 0
set /a var2=LF%%3
if !var2!==1 (set num=!num!00& set /a LF+=2) else if !var2!==2 set num=!num!0& set /a LF+=1
::Linteger 转换后三位数被开方数数组整数部分长度，等于 0 时要及时输出小数点
set /a Linteger=LN/3
::root 数组存储根（一位数一位数存储），长度 Lroot 等于整数开方后的位数与请求的小数精度之和
set /a Lroot=Linteger+degree
set /a Lr=Lroot-1
for /l %%i in (1,1,!Lroot!) do set root%%i=0
::转换被开放数的所有数字（转换成三位数三位数），转换到数组 Snum
::C 是转换后数组的总长度，超过 C 这个长度，余数补充数字的时候就应该补充 000 了。
```

```
::这个转换要注意，在批处理里，我们不可以直接截取三位字符串。例如：1026145，截取的结果为 1 026 145 ，对于 026，因为有前导 0，于是计算机认为是八进制，等于十进制中的 22，错误。
set /a C=^(LN+LF^)/3
for /l %%i in (!begin!,1,!C!) do (
    for %%j in (100,10,1) do (
        set /a Snum%%i+=!num:~0,1!*%%j
        set num=!num:~1!
    )
)
::cou 计数器，计量当前根计算到几位了（一位一位数）
set /a cou=1,first=9
for %%i in (729,512,343,216,125,64,27,8,1,0) do (
    if %%i leq !Snum1! (
        set /a var=%%i
        goto :break
    )
    set /a first-=1
)

:break
::第一个根等于 first
set /a root1=first
set /p var=!root1!<nul
::余数数组
set /a yushu1=Snum1-var
::补充第二位余数数组
if !C! gtr 1 (
    set /a yushu2=Snum2
    set /a C-=2
) else set /a yushu2=0 & set /a C-=1

::coun 是被开方数转换成三位三位数字后的位数
set /a coun=2

:sqrt
set /a Lcoutem=cou%%3
::这里，将根添加到临时数组 roottem 时，移位的同时顺便乘以 10 了，节省下面的乘法运算。
if !Lcoutem!==2 (
    set /a Loopmin=cou+1
    for /l %%i in (1,1,!cou!) do set /a roottem%%i=root%%i
```

```
    set /a roottem!Loopmin!=0
) else if !Lcoutem!==1 (
    set /a Loopmin=cou+2
    for /l %%i in (!Loopmin!,-1,2) do (
        set /a j=%%i-1
        set /a roottem%%i=root!j!
    )
    set /a roottem1=0
) else (
    set /a Loopmin=cou+3
    for /l %%i in (!Loopmin!,-1,3) do (
        set /a j=%%i-2
        set /a roottem%%i=root!j!
    )
    set /a roottem1=roottem2=0
)
::转换 root 一位数一位数根到 sumtem 数组成三位数三位数根
set /a count=Loopmin/3
for /l %%i in (1,1,!count!) do (
    set /a var3=%%i*3
    set /a var2=var3-1
    set /a var1=var2-1
    set /a
Sumtem1%%i=roottem!var1!*100+roottem!var2!*10+roottem!var3!
    set /a Sumtem2%%i=Sumtem1%%i
)
::从 9 开始试开方
set /a x=9

:loop
for /l %%i in (1,1,!count!) do (
    set /a Sumtt1%%i=Sumtem1%%i
    set /a Sumtt2%%i=Sumtem2%%i
)
set /a Sumtt2!count!+=x
for /l %%i in (1,1,!count!) do set /a Sumtt2%%i=Sumtt2%%i*3*x
set /a abc=count*2+1
for /l %%i in (1,1,!abc!) do set /a Sum%%i=0
for /l %%i in (1,1,!count!) do (
    for /l %%j in (1,1,!count!) do (
        set /a k=%%i+%%j-1
        set /a Sum!k!+=Sumtt1%%i*Sumtt2%%j
    )
```

```
)
set /a Sum!k!+=x*x*x
::zhong 长等于 k+1，其实也等于 count*2
::这里是对 Sum 数组进位处理，使之可以和余数数组 yushu 比较大小.
set /a zhong=k+1
set /a jin=Sum!zhong!=0
for /l %%i in (!zhong!,-1,2) do (
    set /a kk=%%i-1
    set /a Sum%%i=^(Sum!kk!+jin^)%%1000
    set /a jin=^(Sum!kk!+jin^)/1000
)
set /a Sum1=jin
::如果 Snum1 大于 1000，那么还需要继续进位，并且整体数组往后移动一位，数组长度增加 1 位。
if !jin! geq 1000 (
    set /a Sum0=jin/1000
    set /a Sum1=jin%%1000
    set /a zhong+=1
    for /l %%i in (!zhong!,-1,1) do (
        set /a j=%%i-1
        set /a Sum%%i=Sum!j!
    )
)
::数组和余数数组对齐，便于后面比较和减法运算
set /a f=zhong+1
for /l %%i in (!coun!,-1,1) do (
    set /a f-=1
    set /a Sum%%i=Sum!f!
    if !f! leq 0 set /a Sum%%i=0
)

::比较数组与余数数组的大小，决定该试根是否就是当前根
for /l %%i in (1,1,!coun!) do (
    if !yushu%%i! gtr !Sum%%i! goto :next
    if !yushu%%i! lss !Sum%%i! (
        set /a x-=1
        goto :loop
    )
)

:next
```

```
set /a cou+=1
set /a root!cou!=x
set /a Linteger-=1
if !Linteger!==0 (
    if !degree!==0 goto :end
    set /p var=.<nul
)
set /p var=!x!<nul
::减法运算，得出新的余数数组
for /l %%i in (!coun!,-1,1) do (
    set /a yushu%%i-=Sum%%i
    if !yushu%%i! lss 0 (
        set /a j=%%i-1
        set /a yushu%%i+=1000
        set /a yushu!j!-=1
    )
)
set /a coun+=1
::对余数数组末尾添加被开方数或者 0
if !C! gtr 0 (
    set /a yushu!coun!=Snum!coun!
) else set /a yushu!coun!=0
set /a C-=1
set /a Lr-=1
if !Lr! gtr 0 goto :sqrt
:end
echo.&echo.&echo.
echo  计算结束，按任意键退出。
pause>nul
```

++

对上面“高精度正负浮点数开立方”压缩代码如下：

```
@echo off&setlocal enabledelayedexpansion&title 高精度开立方程序　　作者 MHL QQ1208980380
echo.&echo.&echo　　高精度开立方程序
echo.&echo　　本程序支持正负浮点数计算&echo.
set /p num=请输入被开立方数
```

```
set /p degree=请输入结果的小数位数
echo.&set symbol=!num:~0,1!
if "!symbol!"=="-" set num=!num:~1!&set /p var=-<nul
for /f "tokens=1,2 delims=." %%i in ("!num!") do set strA=%%i& set
strB=%%j
set str1=!strA!&set num=!num:.=!&set /a LN=LF=0
:len1
set /a LN+=1&set str1=!str1:~0,-1!
if defined str1 goto len1
if defined strB (
    set str2=!strB!
    :len2
    set /a LF+=1&    set str2=!str2:~0,-1!
    if defined str2 goto len2
)
set /a var1=LN%%3,var2=LF%%3,begin=1
if !var1!==1 (set Snum1=!num:~0,1!&set num=!num:~1!&set /a
LN+=begin=2) else if !var1!==2 set Snum1=!num:~0,2!&set
num=!num:~2!&set /a LN+=1,begin=2
if !var2!==1 (set num=!num!00& set /a LF+=2) else if !var2!==2 set
num=!num!0& set /a LF+=1
set /a Linteger=LN/3&set /a Lroot=Linteger+degree&set /a
Lr=Lroot-1
for /l %%i in (1,1,!Lroot!) do set root%%i=0
set /a C=^(LN+LF^)/3,cou=1,first=9
for /l %%i in (!begin!,1,!C!) do for %%j in (100,10,1) do set /a
Snum%%i+=!num:~0,1!*%%j&set num=!num:~1!
for %%i in (729,512,343,216,125,64,27,8,1,0) do (
    if %%i leq !Snum1! set /a var=%%i&goto :break
    set /a first-=1
)
:break
set /a root1=first,yushu1=Snum1-var,coun=2
set /p var=!root1!<nul
if !C! gtr 1 (set /a yushu2=Snum2,C-=2) else set /a yushu2=0&set /a
C-=1
:sqrt
set /a Lcoutem=cou%%3
if !Lcoutem!==2 (
    set /a Loopmin=cou+1,roottem!Loopmin!=0
    for /l %%i in (1,1,!cou!) do set /a roottem%%i=root%%i
) else if !Lcoutem!==1 (
    set /a Loopmin=cou+2,roottem1=0
```

```
    for /l %%i in (!Loopmin!,-1,2) do set /a j=%%i-1&set /a
roottem%%i=root!j!
) else (
    set /a Loopmin=cou+3,roottem1=roottem2=0
    for /l %%i in (!Loopmin!,-1,3) do set /a j=%%i-2&set /a
roottem%%i=root!j!
)
set /a count=Loopmin/3,x=9
for /l %%i in (1,1,!count!) do set /a
var3=%%i*3,var2=%%i*3-1,var1=%%i*3-2&set /a
Sumtem2%%i=Sumtem1%%i=roottem!var1!*100+roottem!var2!*10
+roottem!var3!
:loop
for /l %%i in (1,1,!count!) do set /a
Sumtt1%%i=Sumtem1%%i,Sumtt2%%i=Sumtem2%%i*3*x
set /a Sumtt1!count!+=x,ab=count*2+1
for /l %%i in (1,1,!ab!) do set Sum%%i=0
for /l %%i in (1,1,!count!) do for /l %%j in (1,1,!count!) do set /a
k=%%i+%%j-1&set /a Sum!k!+=Sumtt1%%i*Sumtt2%%j
set /a Sum!k!+=x*x*x,zhong=k+1&set /a jin=Sum!zhong!=0
for /l %%i in (!zhong!,-1,2) do set /a kk=%%i-1&set /a
Sum%%i=^(Sum!kk!+jin^)%%1000&set /a
jin=^(Sum!kk!+jin^)/1000
set Sum1=!jin!
if !jin! geq 1000 set /a
Sum0=jin/1000,Sum1=jin%%1000,zhong+=1&for /l %%i in
(!zhong!,-1,1) do set /a j=%%i-1&set /a Sum%%i=Sum!j!
set /a f=zhong+1
for /l %%i in (!coun!,-1,1) do (
    set /a f-=1&set /a Sum%%i=Sum!f!
    if !f! leq 0 set /a Sum%%i=0
)
for /l %%i in (1,1,!coun!) do (
    if !yushu%%i! gtr !Sum%%i! goto :next
    if !yushu%%i! lss !Sum%%i! set /a x-=1&goto :loop
)
:next
set /a cou+=1&set /a root!cou!=x,Linteger-=1
if !Linteger!==0 (
    if !degree!==0 goto :end
    set /p var=.<nul
)
set /p var=!x!<nul
```

```
for /l %%i in (!coun!,-1,1) do (
    set /a yushu%%i-=Sum%%i
    if !yushu%%i! lss 0 set /a j=%%i-1&set /a
yushu%%i+=1000,yushu!j!-=1
)
set /a coun+=1
if !C! gtr 0 (set /a yushu!coun!=Snum!coun!) else set /a
yushu!coun!=0
set /a C-=1,Lr-=1
if !Lr! gtr 0 goto :sqrt
:end
echo.&echo.&echo 计算结束，请按任意键退出。&pause>nul
```

我们运行一个程序，要想边运行边实时显示时间，怎么办呢？
下面就是一个双线程的例子：

```
@echo off
start /b "" cmd /v:on /c "@echo off&for /l %%a in () do
title !time:~0,8!&ping /n 2 localhost>nul"
setlocal enabledelayedexpansion&title 高精度开立方程序
作者 MHL QQ1208980380
echo.&echo.&echo                                    高精度开
立方程序
echo.&echo                                          本程序支持小数和
负数&echo.
set /p num=请输入被开立方数        :
set /p degree=请输入结果的小数位数 :
echo.&set symbol=!num:~0,1!
if "!symbol!"=="-" set num=!num:~1!&set /p var=-<nul
for /f "tokens=1,2 delims=." %%i in ("!num!") do set strA=%%i& set
strB=%%j
set str1=!strA!&set num=!num:.=!&set /a LN=LF=0
:len1
set /a LN+=1&set str1=!str1:~0,-1!
if defined str1 goto len1
if defined strB (
    set str2=!strB!
    :len2
    set /a LF+=1&     set str2=!str2:~0,-1!
    if defined str2 goto len2
```

```
)
set /a var1=LN%%3,var2=LF%%3,begin=1
if !var1!==1 (set Snum1=!num:~0,1!&set num=!num:~1!&set /a
LN+=begin=2) else if !var1!==2 set Snum1=!num:~0,2!&set
num=!num:~2!&set /a LN+=1,begin=2
if !var2!==1 (set num=!num!00& set /a LF+=2) else if !var2!==2 set
num=!num!0& set /a LF+=1
set /a Linteger=LN/3&set /a Lroot=Linteger+degree&set /a
Lr=Lroot-1
for /l %%i in (1,1,!Lroot!) do set root%%i=0
set /a C=^(LN+LF^)/3,cou=1,first=9
for /l %%i in (!begin!,1,!C!) do for %%j in (100,10,1) do set /a
Snum%%i+=!num:~0,1!*%%j&set num=!num:~1!
for %%i in (729,512,343,216,125,64,27,8,1,0) do (
    if %%i leq !Snum1! set /a var=%%i&goto :break
    set /a first-=1
)
:break
set /a root1=first,yushu1=Snum1-var,coun=2
set /p var=!root1!<nul
if !C! gtr 1 (set /a yushu2=Snum2,C-=2) else set /a yushu2=0&set /a
C-=1
:sqrt
set /a Lcoutem=cou%%3
if !Lcoutem!==2 (
    set /a Loopmin=cou+1
    for /l %%i in (1,1,!cou!) do set /a roottem%%i=root%%i
    set /a roottem!Loopmin!=0
) else if !Lcoutem!==1 (
    set /a Loopmin=cou+2
    for /l %%i in (!Loopmin!,-1,2) do set /a j=%%i-1&set /a
roottem%%i=root!j!
    set /a roottem1=0
) else (
    set /a Loopmin=cou+3
    for /l %%i in (!Loopmin!,-1,3) do set /a j=%%i-2&set /a
roottem%%i=root!j!
    set /a roottem1=roottem2=0
)
set /a count=Loopmin/3,x=9
for /l %%i in (1,1,!count!) do (
    set /a var3=%%i*3,var2=%%i*3-1,var1=%%i*3-2
    set /a Sumtem1%%i=roottem!var1!*100
```

```
    set /a Sumtem1%%i+=roottem!var2!*10
    set /a Sumtem1%%i+=roottem!var3!
    set /a Sumtem2%%i=Sumtem1%%i
)
:loop
for /l %%i in (1,1,!count!) do set /a
Sumtt1%%i=Sumtem1%%i,Sumtt2%%i=Sumtem2%%i
set /a Sumtt2!count!+=x,abc=count*2+1
for /l %%i in (1,1,!count!) do set /a Sumtt2%%i=Sumtt2%%i*3*x
for /l %%i in (1,1,!abc!) do set /a Sum%%i=0
for /l %%i in (1,1,!count!) do for /l %%j in (1,1,!count!) do set /a
k=%%i+%%j-1&set /a Sum!k!+=Sumtt1%%i*Sumtt2%%j
set /a Sum!k!+=x*x*x,zhong=k+1&set /a jin=Sum!zhong!=0
for /l %%i in (!zhong!,-1,2) do set /a kk=%%i-1&set /a
Sum%%i=^(Sum!kk!+jin^)%%1000&set /a jin=^(Sum!kk!+jin^)/1000
set /a Sum1=jin
if !jin! geq 1000 set /a
Sum0=jin/1000,Sum1=jin%%1000,zhong+=1&for /l %%i in
(!zhong!,-1,1) do set /a j=%%i-1&set /a Sum%%i=Sum!j!
set /a f=zhong+1
for /l %%i in (!coun!,-1,1) do (
    set /a f-=1&set /a Sum%%i=Sum!f!
    if !f! leq 0 set /a Sum%%i=0
)
for /l %%i in (1,1,!coun!) do (
    if !yushu%%i! gtr !Sum%%i! goto :next
    if !yushu%%i! lss !Sum%%i! set /a x-=1&goto :loop
)
:next
set /a cou+=1&set /a root!cou!=x,Linteger-=1
if !Linteger!==0 if !degree!==0 goto :end
if !Linteger!==0 set /p var=.<nul
set /p var=!x!<nul
for /l %%i in (!coun!,-1,1) do (
    set /a yushu%%i-=Sum%%i
    if !yushu%%i! lss 0 set /a j=%%i-1&set /a
yushu%%i+=1000,yushu!j!-=1
)
set /a coun+=1
if !C! gtr 0 (set /a yushu!coun!=Snum!coun!) else set /a
yushu!coun!=0
set /a C-=1,Lr-=1
if !Lr! gtr 0 goto :sqrt
```

```
:end
echo.&echo.&echo  计算结束，请按任意键重新计算&pause>nul
```

素数搜索（非高精度计算）

```
@echo off
setlocal EnableDelayedExpansion
title  素数搜索程序
echo. & echo                                                    素数搜索程序
echo.
set /p num=请输入搜索范围：
echo.
if !num!==1 set jishu=0 & goto :end
set var=               2
set /p    print=!var:~-7!<nul
set jishu=1
for /l %%i in (3,2,%num%) do (
     set prime=1
     set /a tem=%%i/2
     for /l %%j in (3,2,!tem!)do (
               set /a var= %%i%%%%j
               if !var!==0 set prime=0
     )
          if !prime!==1 (
          set var=               %%i
          set /p    print=!var:~-7!<nul
          set /a jishu+=1
          set /a k=!jishu!%%10
          if !k!==0 echo.
          )
)
echo. & echo.
:end
echo ----------------------------
echo  从 1 到%num%共有%jishu%个素数。
echo ----------------------------
echo  计算结束，按任意键退出。
pause>nul
```

素数搜索，这里采用试除法。对于数字 **n**，一般试除法是用从 1 到根号 **n** 之间的奇数逐一试除，所有数都无法整除时，可以判断 **n** 就是素数。但批处理无法直接计算开方，用批处理通过“旁门左道”可以实现开方，但那会消耗大量时间，效率反而更差。所以批处理采用的是从 1 到 **n**/2 之间的所有奇数逐一试除 **n** 就行了。

素数搜索（可指定搜索范围）

```
@echo off
setlocal EnableDelayedExpansion
title  搜索指定范围内的素数
echo. & echo                                        指定范围搜索素数程序
echo.
set /p numbegin=请输入搜索起始值(大于终止值):
set /p numend=请输入搜索终止值(小于起始值):
set abc=%numbegin%
echo.
set jishu=0
if %numend% lss %numbegin% ( echo  终止值小于起始值，输入错误。& goto :end )
if %numend%==1 goto :end
if %numbegin% leq 2 (
    if %numend% geq 2 (
        set var=              2
        set /p    print=!var:~-7!<nul
        set jishu=1
    )
)
if %numbegin% leq 1 set numbegin=3
for /l %%i in (%numbegin%,1,%numend%) do (
    set /a pan=%%i%%2
    if not !pan!==0 (
            set prime=1
            set /a tem=%%i/2
```

```
            for /l %%j in (3,2,!tem!)do (
                set /a var= %%i %% %%j
                if !var!==0 set prime=0
            )
            if !prime!==1 (
                set var=            %%i
                set /p    print=!var:~-7!<nul
                set /a jishu+=1
                set /a k=!jishu!%%10
                if !k!==0 echo.
            )
    )
)
echo. & echo.
:end
echo ---------------------------------
echo  从%abc%到%numend%共有%jishu%个素数。
echo ---------------------------------
echo  计算结束，按任意键退出。
pause>nul
```

3X+1 猜想（高精度计算）

```
@echo off
setlocal enabledelayedexpansion
title 3x+1 猜想游戏
echo. & echo.
echo                                        3x+1 猜想游戏
echo.
echo  游戏简介：
echo        对任意一个大于 0 的整数，如果它是偶数，将它除以 2；
如果它是奇数，将它乘以 3 再加上 1。
echo        不断重复以上步骤，最后必定等于 1。
echo        目前数学家们无法证明这个结论，所以叫做猜想。
echo.
set /p num=请输入一个数：
echo -------------------------------
echo %num%
```

```
::counter 是计数器，统计总共计算了几次。
set counter=0

::对输入分别是 0 和 1 的情况作出判断，增加程序的健壮性
if %num%==1 goto :end
if %num% leq 0 (echo  输入数字不能小于或等于 0，输入错误，按任意键退出。 & pause>nul & goto :eof)

::测出读入数字的位数
set M=0
set str=%num%
:len
set /a M+=1
set str=!str:~0,-1!
if defined str goto len

::峰值位数不会大于数字位数的两倍，所以数组长度最大为 max 就够了。
set /a max=!M!*2

::对数组清零
for /l %%i in (1,1,!max!) do (set X%%i=0)

::将数字字符串转换为数组
set /a t=-!M!
set n=!max!
for /l %%i in (-1,-1,!t!) do (
    set tem=!num:~%%i!
    set tem=!tem:~0,1!
    set X!n!=!tem!
    set /a n-=1
)

::游戏进行
:loop
set /a result=X!max!%%2
if !result!==0 (
    ::当偶数时的情况
    set u=0
    for /l %%i in (1,1,!max!) do (
        set /a u=u*10+!X%%i!
```

```
        set /a X%%i=!u!/2
        set /a u=!u!%%2
    )
    ::偶数情况处理完毕，输出当次结果
    set boor=0
    for /l %%i in (1,1,!max!) do (
        if !X%%i! neq 0 set boor=1
        if !boor!==1 set /p var=!X%%i!<nul
    )
    ::计数器加 1
    set /a counter+=1
    echo.
)else (
    ::当奇数时的情况
    for /l %%i in (!max!,-1,1) do (set /a X%%i=!X%%i!*3)
    set u=0
    for /l %%i in (!max!,-1,1) do (
        set /a z=!u!+!X%%i!%%10
        set /a u=!X%%i!/10
        set X%%i=!z!
    )
    ::上面是乘以 3，这里是加 1。即使是加 1 而已，也要对所有位数进行进位处理，否则像 9999999 就会出错。
    set /a X!max!+=1
    for /l %%i in (!max!,-1,1) do (
        if !X%%i! geq 10 (
            set /a X%%i=!X%%i!-10
            set /a d=%%i-1
            set /a X!d!+=1
        )
    )
    ::奇数情况处理完毕，输出当次结果
    set boor=0
    for /l %%i in (1,1,!max!) do (
        if !X%%i! neq 0 set boor=1
        if !boor!==1 set /p var=!X%%i!<nul
    )
    ::计数器加 1
    set /a counter+=1
    echo.
    )
)
```

```
::判断此时数组是否等于 1，不是的话游戏继续进行。
set /a v=!max!-1
for /l %%i in (1,1,!v!) do (if !X%%i! neq 0 goto :loop)
)
set /a w=X!max!
if !w! neq 1 goto :loop

:end
::游戏结束
echo --------------------------------
echo  总共计算%counter%次后出现数字 1。
pause>nul
```

++

对上面“3x+1 猜想”代码压缩一下：

```
@echo off&setlocal enabledelayedexpansion&title 3x+1 猜想游戏
echo. & echo.&echo                                        3x+1 猜想游戏
echo.&echo  游戏简介：
echo        对任意一个大于 0 的整数，如果它是偶数，将它除以 2；如果它是奇数，将它乘以 3 再加上 1。
echo        不断重复以上步骤，最后必定等于 1。目前数学家们无法证明这个结论，所以叫做猜想。&echo.
set /p num=请输入一个数：&echo --------------------------------
echo %num% & set /a counter=M=0
if %num%==1 goto :end
if %num% leq 0 (echo  输入数字不能小于或等于 0，输入错误，按任意键退出。 & pause>nul & goto :eof)
set str=%num%
:len
set /a M+=1
set str=!str:~0,-1!
if defined str goto len
set /a max=!M!*2,t=-M,n=max
for /l %%i in (1,1,!max!) do set X%%i=0
for /l %%i in (-1,-1,!t!) do (
    set tem=!num:~%%i!
```

```
    set tem=!tem:~0,1!
    set X!n!=!tem!
    set /a n-=1
)
:loop
set /a result=X!max!%%2
if !result!==0 (
    set /a u=boor=0,counter+=1
    for /l %%i in (1,1,!max!) do set /a u=u%%2*10+!X%%i!&set /a
X%%i=!u!/2
    for /l %%i in (1,1,!max!) do (
        if !X%%i! neq 0 set boor=1
        if !boor!==1 set /p var=!X%%i!<nul
    )
    echo.
)else (
    set /a u=boor=d=0,counter+=1
    for /l %%i in (!max!,-1,1) do set /a X%%i=!X%%i!*3
    for /l %%i in (!max!,-1,1) do set /a u=u/10+X%%i&set /a
X%%i=u%%10
    set /a X!max!+=1
    for /l %%i in (!max!,-1,1) do set /a d=d/10+X%%i&set /a
X%%i=d%%10
    for /l %%i in (1,1,!max!) do (
        if !X%%i! neq 0 set boor=1
        if !boor!==1 set /p var=!X%%i!<nul
    )
    echo.
)
set /a v=!max!-1,w=X!max!
for /l %%i in (1,1,!v!) do if !X%%i! neq 0 goto :loop
if !w! neq 1 goto :loop
:end
echo --------------------------------
echo  总共计算%counter%次后出现数字 1。  & pause>nul
```

广义斐波那契数列（高精度计算）

```
@echo off
setlocal enabledelayedexpansion
```

```
title 高精度广义斐波那契数列程序
echo. & echo. & echo                                    高精度广义斐波那契数列程序
echo. & echo 请输入两个初始值。
set /p str1=
set /p str2=
set /p M=请输入总项数：

::先计算出两个初始值的位数
set M1=0
set M2=0
set A=%str1%
set B=%str2%
:len1
set /a M1+=1
set A=!A:~0,-1!
if defined A goto len1
:len2
set /a M2+=1
set B=!B:~0,-1!
if defined B goto len2

::K 值作为数组长度。一般来说，最后一个数的长度不会超过总项数的 1/2，将 K 值缩小使计算更加快速。
set /a K=!M!/2
::下面这个设计是防止初始值两个数过大，而需要计算的项数过少的情况。
set /a S=!M1!+!M2!
if !K! lss !S! (set K=!S!)

::对数组全部清零
for /l %%i in (1,1,!K!)do (
    set X%%i=0
    set Y%%i=0
    set Z%%i=0
)

::将两个字符串形式的初始值转换成数组。
::由于批处理无法对循环变量%%i 取负值，所以只能在循环参数用负数。
::批处理也无法将字符串末尾一个个“剥离”出来，于是用了几次
```

%var:~m,n%形式剥离出单个数位，然后赋值到对应数组里。

```
set /a t=0-!M1!
set n=!K!
for /l %%i in (-1,-1,!t!) do (
    set tem1=!str1:~%%i!
    set tem1= !tem1!
    set tem1=!tem1:~1,1!
    set X!n!=!tem1!
    set /a n-=1
)
set /a t=0-!M2!
set n=!K!
for /l %%i in (-1,-1,!t!) do (
    set tem2=!str2:~%%i!
    set tem2= !tem2!
    set tem2=!tem2:~1,1!
    set Y!n!=!tem2!
    set /a n-=1
)
echo ---------------------------------------------
echo !str1!
echo !str2!
```

::进行第一次计算。因为第一次计算，不必要将 **XYZ** 数组交叉赋值，所以要和后面的循环计算分开。

```
for /l %%i in (1,1,%K%) do (
    set /a Z%%i=!X%%i!+!Y%%i!
)
for /l %%i in (%K%,-1,1) do (
    if !Z%%i! geq 10 (
        set /a Z%%i=!Z%%i!-10
        set /a j=%%i-1
        set /a Z!j!+=1
    )
)
```

::输出结果，前导 0 不输出。

```
set boor=0
for /l %%i in (1,1,%K%) do (
    if !Z%%i! neq 0 set boor=1
    if !boor!==1 (set /p var=!Z%%i!<nul)
    )
echo.
```

::对循环次数判断，因为有**!M!**-3，那么如果预定的总项数小于 3，就

```
不应该再进行计算了。
if !M! gtr 3 (set /a r=!M!-3) else (goto end)
::循环计算，直到算到预定的项数为止。
:loop
for /l %%i in (1,1,%K%) do (
    set X%%i=!Y%%i!
    set Y%%i=!Z%%i!
    set /a Z%%i=!X%%i!+!Y%%i!
)
::对数组 Z 进行进位处理。
for /l %%i in (%K%,-1,1) do (
    if !Z%%i! geq 10 (
        set /a Z%%i=!Z%%i!-10
        set /a j=%%i-1
        set /a Z!j!+=1
    )
)
::输出结果，前导 0 不输出。
set boor=0
for /l %%i in (1,1,%K%) do (
    if !Z%%i! neq 0 set boor=1
    if !boor!==1 (set /p var=!Z%%i!<nul)
    )
echo.
set /a r-=1
if !r! neq 0 goto :loop
:end
echo ----------------------------------------
echo  计算结束，按任意键退出  & pause>nul
```

++

对上面“广义斐波那契数列”压缩代码如下：

```
@echo off & setlocal enabledelayedexpansion&title  高精度广义斐波那契数列程序
echo. & echo. & echo                                    高精度广义斐波那契数列程序
```

```
echo. & echo  请输入两个初始值。&set /p str1=&set /p str2=
set /p M=请输入总项数：
set /a M1=M2=0
set A=%str1%&set B=%str2%
:len1
set /a M1+=1
set A=!A:~0,-1!
if defined A goto len1
:len2
set /a M2+=1
set B=!B:~0,-1!
if defined B goto len2
set /a K=M/2,S=M1+M2
if !K! lss !S! set K=S
for /l %%i in (1,1,!K!)do set /a X%%i=Y%%i=Z%%i=0
set /a t=-M1,n=K,tt=-M2,nn=K
for /l %%i in (-1,-1,!t!) do (
    set tem1=!str1:~%%i!
    set tem1= !tem1!
    set tem1=!tem1:~1,1!
    set X!n!=!tem1!
    set /a n-=1
)
for /l %%i in (-1,-1,!tt!) do (
    set tem2=!str2:~%%i!
    set tem2= !tem2!
    set tem2=!tem2:~1,1!
    set Y!nn!=!tem2!
    set /a nn-=1
)
echo ----------------------------------------------&echo !str1!&echo !str2!
for /l %%i in (1,1,%K%) do set /a Z%%i=!X%%i!+!Y%%i!
for /l %%i in (%K%,-1,1) do set /a j=j/10+Z%%i&set /a Z%%i=j%%10
set boor=0
for /l %%i in (1,1,%K%) do (
    if !Z%%i! neq 0 set boor=1
    if !boor!==1 set /p var=!Z%%i!<nul
)
echo.&if !M! gtr 3 (set /a r=!M!-3) else goto end
:loop
for /l %%i in (1,1,%K%) do      set X%%i=!Y%%i!&set
Y%%i=!Z%%i!&set /a Z%%i=!X%%i!+!Y%%i!
for /l %%i in (%K%,-1,1) do set /a j=j/10+Z%%i&set /a Z%%i=j%%10
```

```
set /a boor=0,r-=1
for /l %%i in (1,1,%K%) do (
    if !Z%%i! neq 0 set boor=1
    if !boor!==1 set /p var=!Z%%i!<nul
)
echo.&if !r! neq 0 goto :loop
:end
echo ----------------------------------------------&echo 计算结束，按任意键退出 & pause>nul
```

高精度阶乘

```
@echo off
setlocal enabledelayedexpansion
title 高精度阶乘程序
echo. & echo. & echo                                    高精度阶乘程序
echo. & set /p num=请输入一个数字：
echo. & echo 计算进行中，请稍候，计算进度请看标题栏...
echo. & echo.
::M 是计算结果的总位数，初始值为 1
set M=1
::将数组的首位设置为 1，否则后面乘法过程全等于 0
set S1=1
set i=1
:loop1
    ::当输入的数字大于 100 时，计算时间比较长，所以要将计算进度显示出来。
    title 高精度阶乘程序 当前计算到!i!
    for /l %%j in (1,1,!M!) do set /a S%%j*=i
    ::进位处理，数组处理到 M-1 位即可。第 M 位 S!M!处理涉及要将位数上限 M 增大的问题，所以另行处理。
    set /a x=!M!-1
    for /l %%k in (1,1,!x!) do (
        set /a z=S%%k/10
        set /a S%%k%%=10
        set /a y=%%k+1
```

```
        set /a S!y!+=z
    )
    ::处理!SM!的进位，并处理上限 M 值增量。
:loop2
    set /a t=S!M!
    if !t! geq 10 (
        set /a u=S!M!/10
        set /a S!M!%%=10
        set /a M+=1
        set /a S!M!+=u
    )
    set /a t=S!M!
    if !t! geq 10 goto :loop2
    if !i! lss !num! set /a i+=1 & goto :loop1
echo !num!的阶乘等于:
echo.
for /l %%i in (!M!,-1,1) do set /p var=!S%%i!<nul
echo. & echo. & echo  结果总共有!M!位。
echo. & echo.& echo  计算结束，按任意键退出。& pause>nul
```

==========

上面程序看似没有对 0 的阶乘这种情况进行处理，实际上，这种情况已经在程序过程处理好了，输入 0 会输出 1。

用批处理来计算高精度阶乘，有几个问题要明确：

问题分析：

高精度计算，用数组实现。那么，这个数组一开始要设置多少位？也等于问：最后阶乘结果是几位？

我们一开始很难计算出阶乘结果的总位数，这个涉及到高等数学方法。我们采用动态数组位数 **M** 的办法可以规避这个问题。不断增加数组的位数 **M**，以适应每次计算结果的总位数。

采用动态数组位数的办法，那么我们应该采用 **goto** 循环为佳，采用 **for** 循环比较麻烦，因为 **for** 不允许改变循环变量值，而我们的循环上限 **M** 要随着计算结果的增加而增加。

采用动态数组，那么存储数字时，数字的末位应该存在数组的第一位，数组的末位应该存储数字的末位，这样便于进位运算和增加位数上限 **M**。

用批处理来进行计算真不是明智的选择，用上面的批处理代码计算到 500!，花了大约三分钟时间。

++

对上面“高精度阶乘”压缩代码如下：

```
@echo off & setlocal enabledelayedexpansion & title 高精度阶乘程序
echo. & echo. & echo                                    高精度阶乘程序
echo. & set /p num=请输入一个数字：
echo. & echo. & set /a M=S1=i=1
:loop1
    title 高精度阶乘程序 计算进度!i!
    for /l %%j in (1,1,!M!) do set /a S%%j*=i
    set /a x=M-1
    for /l %%k in (1,1,!x!) do set /a z=S%%k/10,S%%k%%=10,y=%%k+1 & set /a S!y!+=z
:loop2
set /a t=S!M!
if !t! geq 10 set /a u=S!M!/10,S!M!%%=10,M+=1 & set /a S!M!+=u
set /a t=S!M!
if !t! geq 10 goto :loop2
if !i! lss !num! set /a i+=1 & goto :loop1
echo !num!的阶乘等于: & echo.&for /l %%i in (!M!,-1,1) do (set /p =!S%%i!<nul)
echo. & echo. & echo 结果总共有!M!位。计算结束，按任意键退出。 & pause>nul
```

约瑟夫问题

```
@echo off
title 约瑟夫问题
setlocal enabledelayedexpansion
echo. & echo.
echo                                                    约瑟夫问题
echo.
echo 问题描述
echo.
echo 据说著名犹太历史学家 Josephus 有过以下的故事：在罗马人占领乔塔帕特后，39 个犹太人与 Josephus 及他的朋友躲到一个洞中，39 个犹太人决定宁愿死也不要被敌人抓到，于是决定了一个自杀方式：41 个人围成一个圆圈，由第 1 个人开始报数，报数到第 3 个人，该人就必须自杀，然后再由下一个重新报数，报数到的第 3 个人，该人又自杀...直到所有人都自杀身亡为止，然而 Josephus 和他的朋友并不想遵从。
echo 一开始有 i 个人，数 j 去 1，最后剩余 k 个人。首先从一个人开始，越过 j-1 个人，并杀掉第 j 个人；接着再越过 j-1 个人，并杀掉第 j 个人...这个过程沿着圆圈一直进行，直到最终只剩下 k 个人，这 k 个人就可以继续活着。问题是，给定了 i、j、k 值，一开始要站在什么地方才能避免被处决？聪明的Josephus要他的朋友先假装遵从，他将朋友与自己安排在第 16 个与第 31 个位置，于是逃过了这场死亡游戏。
echo.&echo.
echo 游戏开始
echo                          输入参数
echo --------------------------------
set /p all=请输入一开始总人数：
set /p count=请输入循环点数数目：
set /p remain=请输入最后剩余人数：
echo --------------------------------
echo.
::计数器 1，当计数器 1 等于预先设定的循环点数数目时，该人出列。
set circulate=1
::计数器 2，当计数器 2 等于预先设定的剩余人数时，游戏结束，输出结果。
set remaining=%all%
::计数器 f，用于输出时，每输出 10 个数后换行。
set f=1
```

::预处理，初始时，将 S1、S2、S3、S4、S5...Sall 全部标记为活人，也就是值等于 1，后面标记出列的人的值等于 0。

```
for /l %%i in (1,1,%all% ) do (set S%%i=1)
echo.
echo  出局编号(按出局先后顺序输出):
```

::游戏进行，假如循环到 Sall，那么又重新开始 loop 循环，又从 S1 开始游戏，此时计数器 circulate 不清零，计数继续。这样设计就达到了和链表构造圆圈的效果相同。

```
:loop
for /l %%i in (1,1,%all%) do (
    if !remaining!==%remain% (goto :loopend)
        if !S%%i!==1 (
            if !circulate!==%count% (
                set g=              %%i <nul
                                        set /p var=!g:~-7!<nul
                                        if !f!==10 (echo. & set f=0)
                                        set /a f+=1
                                        set S%%i=0
                                        set /a remaining-=1
                                        set    circulate=1
            ) else (set /a circulate+=1)
        )
    )
goto :loop

:loopend
echo.
echo  剩余编号(按编号从小到大输出)
set f=0
for /l %%i in (1,1,%all%) do (
    if !S%%i!==1 (
        set /p var=%%i    <nul
        set /a f+=1
        set /a var=!f!%%8
        if !var!==0 echo.
        )
    )
echo. & echo.
echo  计算结束，请按任意键退出。
pause>nul
```

==========

约瑟夫问题编程思路：
由于批处理没有指针，所以我们没法构造链表。采用一维数组的形式，从头循环到尾。
我们将全部 **all** 个人，编号从 1 到 **all** 号。由于 **all** 是读读入用户输入的数据后才知道，所以编号要采用动态编号的方法，从 **S1**、**S2**、**S3**、**S4...Sall**，可以在 **for** 语句里用 **S%%i** 这种写法编号。
我们将活人标记为 1，将四人标记为 0，一开始，所有人都是活的，所以要用 **for /l %%i in (1,1,%all%) do (set S%%i=1)**，将所有人标记为活人。
游戏进行，从编号为 1 开始数数，满足循环数**(if !circulate!==%count%)** 就将人标记为出局 0，循环数重新计数**(set　circulate=1)**，循环到 **Sall**，计数器 **circulate** 不清零，又开始 **loop** 循环，直到剩余人数 **remaining** 等于预先设定剩余人数 **remain** 为止。

报数游戏

@echo off&setlocal enabledelayedexpansion&title 报数游戏　作者 **MHL** 1208980380
:begin
echo. & **echo**. & **echo**　　报数游戏
echo.&echo　　预设一个目标数字 **M** 和每回合数字增量上限 **N**，人和计算机轮流报数。从 1 开始，报数要大于对方的数字，但又不能大于对方的数字加上 **N** 或大于目标数字 **M**，也不可以弃权。谁先报出目标数字 **M** 就输了。**&echo**. & **echo**.
set /a num=%random%%%200+1
echo 预设目标数字为　　**!num!**
set /a n=%random%%%20+1
echo 每回数字增量上限为　　**!n!**
set /p order=请选择谁先开始？(0:计算机。1:人类。)
if not %order%==0 if not %order%==1 echo 输入错误，请重新输入。& **pause>nul** & **cls** & **goto :begin**
echo.&echo 游戏开始----------**&echo.**
set /a var=0,m=n+1
:loop
if !order!==0 (

```
    set /a tem=num-var
    if !tem! lss !m! (
        set /a stoch=tem-1
        if !stoch!==0 (set /a var+=1) else set /a var=var+tem-1
    )else (
        set /a tem0=tem%%m
        if not !tem0!==0 (
            set /a tem1=tem%%m-1
            if !tem1!==0 (set /a stoch=%random%%%n+1,var+=stoch)
else set /a var+=tem%%m-1
        ) else set /a var+=n
    )
    echo  计算机--:   !var! & echo.&set order=1
    if !var!==!num! echo. & echo  恭喜您赢了计算机！
&pause>nul&goto :eof
) else (
    :loopmin
    set /p tem=人类----:   &echo.
    set /a judge=var+n
    if !tem! leq !var! goto :error
    if !num! lss !tem! goto :error
    if !judge! lss !tem! goto :error
    set /a var=!tem!,order=0
    if !var!==!num! echo. & echo  很抱歉，您输了。
&pause>nul&goto :eof
)
goto :loop
:error
echo  输入错误，请重新输入。& goto :loopmin
```

说明：
这是在 QQ 批处理群一位网友向我提出的一个游戏批处理编程，晚上回家后写出来的。
这个游戏很简单，但里面小有窍门，类似于捡火柴棍游戏。除非人类掌握了游戏规律，否则本程序必胜！经过测试，好多人都无法赢计算机。

这其中的取胜秘籍是：
前提：目标数字 m，每回合增量 n
当 m%(n+1)不等于 1，人类先，并且占据 m%(n+1)-1，以后累加 n+1
当 m%(n+1)=1 让计算机先，第一个输入 n+1,以后累加 n+1

当 m%(n+1)=0 人类先，并且占据 n，以后累加 n+1
按照上面方法，对计算机就能保证每次都赢计算机了。

多个分式边计算边输出算法

题目：对 **S**=1/3+1/7+1/13+1/17 算式，计算到小数点后 **N** 位。
(注；算式的 30 位小数结果为：0.611937082525317819435466494290)

分析：
这题编程很容易，我们用一个有 **N** 位数组元素的数组 **S** 记录小数，对每次除法得到的结果累加到 **S** 数组就可以了，编程如下：

```
@echo off
setlocal enabledelayedexpansion
set /p N=请输入位数：
set /a X1=3,X2=7,X3=13,X4=17
::S 数组用于记录总和，一开始进行清零。
for /l %%i in (1,1,%N%) do set S%%i=0
for /l %%i in (1,1,4) do (
    ::Y 数组用于计算各个除法算式的商。这里初始化 Y 数组，个位数等于 1，其余小数部分都等于 0。
    for /l %%j in (1,1,%N%) do set Y%%j=0
    set Y1=1
    set tem=0
    ::对每个算式进行除法运算，运算结果存到 Y 数组
    for /l %%j in (1,1,%N%) do (
        set /a tem=tem*10+Y%%j
        set /a Y%%j=tem/X%%i
        set /a tem=tem%%X%%i
    )
    ::将 Y 数组，也就是每个式子的商，累加到 S 数组里。
    for /l %%j in (1,1,%N%) do set /a S%%j+=Y%%j
)
::S 数组累加完四个算式，那么 S 数组就是最终结果。下面对 S 数组进行进位处理。
for /l %%i in (%N%,-1,1) do (
    set /a jin=jin+S%%i
```

```
    set /a S%%i=jin%%10
    set /a jin=jin/10
)
::输出结果
for /l %%i in (1,1,%N%) do set /p var=!S%%i!<nul
echo.&pause>nul
```

上面程序用 **Y** 数组记录每个式子的商，用 **S** 数组记录总和。将 **Y** 数组累加到 **S** 数组得到最终结果。

程序可以稍微简化。我们观察到数组 **Y**，一开始第一位是 1，后面全是 0，规律性极强，于是就不必浪费一个数组来记录它值。将它边做除法运算边累加到 **S** 数组就可以了，程序简化如下：

```
@echo off
setlocal enabledelayedexpansion
set /p N=请输入位数：
set /a X1=3,X2=7,X3=13,X4=17
for /l %%i in (1,1,%N%) do set S%%i=0
for /l %%i in (1,1,4) do (
    set tem=1
    for /l %%j in (1,1,%N%) do (
        set /a tem=tem*10
        set /a S%%j+=tem/X%%i
        set /a tem=tem%%X%%i
    )
)
set jin=0
for /l %%i in (%N%,-1,1) do (
    set /a jin=jin+S%%i
    set /a S%%i=jin%%10
    set /a jin=jin/10
)
for /l %%i in (1,1,%N%) do set /p var=!S%%i!<nul
echo.&pause>nul
```

这个程序是按照常规思路写出来的，其运算式子还可以再进行压缩：

```
@echo off
setlocal enabledelayedexpansion
set /p N=请输入位数：
set /a X1=3,X2=7,X3=13,X4=17,jin=0
for /l %%i in (1,1,4) do (
    set tem=1
    for /l %%j in (1,1,%N%) do (
        set /a S%%j+=tem*10/X%%i
        set /a tem=tem*10%%X%%i
    )
)
for /l %%i in (%N%,-1,1) do (
    set /a jin=jin/10+S%%i
    set /a S%%i=jin%%10
)
for /l %%i in (1,1,%N%) do set /p var=!S%%i!<nul
echo.&pause>nul
```

这个程序用来计算若干个分子全都是 1 的分数相加的任意精度。

上面程序的计算是完全模拟手工除法及加法过程，比较容易想到。程序做每个除法算式运算，都必须除到目标精度为止，对加法运算也是一样，必须将所有精度都相加完毕。所以这个程序必须将所有精度都算出来后，才可以一次性输出最终结果。

还有一种算法比较奇特，这种算法的对每个除法算式的运算不必每次都算到目标精度，而是算出一小部分就可以了。同样，每次加法运算也只计算一小部分。于是这种算法可以边计算边输出，直到所需精度为止。

这种算法不符合我们常规手工计算，不容易想到，下面详解讲解：

仍旧以算式 S=1/3+1/7+1/13+1/17 为例：

分析：

要实现边计算边输出，主要就是对除法运算的余数进行适当处理，总体思路是：将每次除法运算后余数记录起来，作为下次计算的依据。这样就不必每次都计算到目标精度，而是边计算边输出，直到所需精度为止。

具体过程：

第一次计算：我们可以对四个式子分别算出整数商，用四个变量分别

来记录四个余数，将商相加后输出结果。
第二次计算：用第一次运算的四个余数来继续本次计算----算出四个商之和并输出结果，并用四个变量记录本次的四个余数，作为再下次计算的数据参数...如此循环，边计算边输出，直到所需的精度为止。

下面看算式的实际运算过程：
用 **Y**1，**Y**2，**Y**3，**Y**4 分别存储四个分式 1/3，1/7，1/13，1/17 每次得到的余数。

第一次计算：四个商都等于 0，四个余数都等于 1，输出四个商之和等于 0，作为结果小数的个位数，存储四个余数 **Y**1=**Y**2=**Y**3=**Y**4=1。

第二次计算：第一个商为 **Y**1*10/3=1*10/3=3，余数 **Y**1=1；第二个商为 **Y**2*10/7=1*10/7=1，余数 **Y**2=3；第三个商为 **Y**3*10/13=1*10/13=0，余数 **Y**3=10；第四个商为 **Y**4*10/17=1*10/17=0，余数 **Y**4=10。
　　四个商之和为 3+1+0+0=4，于是输出 4，作为结果小数的十分位。这里要注意，和手工计算一样，对余数需要乘以 10 后再参与计算。至于为什么？回去问小学二年级数学老师！

第三次计算：第一个商等于 **Y**1*10/3=1*10/3=3，本次余数 **Y**1=**Y**1*10%3=1*10%3=1；第二个商等于 **Y**2*10/7=3*10/7=4，本次余数 **Y**2=**Y**2*10%7=3*10%7=2；第三个商等于 **Y**3*10/13=10*10/13=7，本次余数等于 **Y**3=**Y**3*10%13=10*10%13=9；第四个商等于 **Y**4*10/17=10*10/17=5，本次余数等于 **Y**4=**Y**4*10%17=10*10%17=15。四个商之和为 3+4+7+5=19，于是输出结果 19，作为结果小数的百分位。

第四次计算：第一个商等于 3，余数 1；第二个商等于 2，余数 6；第三个商等于 6，余数 12；第四个商等于 8，余数 14。输出四个商之和等于 3+2+6+8=19

第五次计算：第一个商等于 3，余数 1；第二个商等于 8，余数 4；第三个商等于 9，余数 3；第四个商等于 8，余数 4。输出四个商之和 3+8+9+8=28
第六次计算：第一个商等于 3，余数 1；第二个商等于 5，余数 5；第三个商等于 2，余数 4；第四个商等于 2，余数 6。输出四个商之和 3+5+2+2=12
第七次计算：第一个商等于 3，余数 1；第二个商等于 7，余数 1；第三个商等于 3，余数 1；第四个商等于 3，余数 9。输出四个商之和 3+7+3+3=16

第七次计算：输出 9
第八次计算：输出 16
第九次计算：输出 20
第十次计算：输出 24

编程如下：

```
@echo off
setlocal enabledelayedexpansion
set /p N=请输入精度：
::为了简化程序，我们用四个变量 M1、M2、M3、M4 存储四个分式的分母
set /a M1=3,M2=7,M3=13,M4=17
::用四个变量 Y1、Y2、Y3、Y4 存储每次除法运算后的余数。这里对余数初始化，一开始都等于分子 1。
for /l %%i in (1,1,4) do set Y%%i=1
::shang 是我们每次计算四个商之和，作为每次的结果输出，值一开始等于 0
set shang=0
for /l %%i in (1,1,%N%) do (
    for /l %%j in (1,1,4) do (
        ::对上次计算剩下的余数乘以 10 后再参与运算。
        set /a tem=Y%%j*10
        ::记录本次除法运算的余数
        set /a Y%%j=tem%%M%%j
        ::算出本次四个式子的商之和
        set /a shang=shang+tem/M%%j
    )
    ::输出本次结果
    echo !shang!
    ::对本次结果清零，防止干扰下次计算。
    set shang=0
)
pause>nul
```

输出的数字按顺序排列出来：

0　4　19　19　28　12　16　9　16　20　24

这个结果不对，我们将上面数字进位处理结果为：

0　6　1　1　9　3　7　0　8　2　4

也就是：0.6119370824，结果是正确的。

现在问题就变成如何在计算中实现进位后才输出？

我们看到上面程序的计算，是将四个分数各自独立计算出商，然后再对四个商求和后输出，这就有可能造成四个商的和的位数多于 1 位，那么就需要进位处理了。但这个进位处理，需要将下次计算的商之和算出，才能知道要进位多少到本次商之和，所以上面计算方法无法实

现边计算边输出的效果。
将上面计算方法改进一下：
将第一个分式算出的商累积到第二个分数上去，参与第二个分式的除法运算；得到的商累积到第三个分数上去，参与第三个分式的除法运算；得到的商累积到第四个分数上去，参与第四个分式的除法运算。这样一来，在对每个分式运算的同时也进行了商的求和，也就实现了进位处理，算到第四个分式后得到的商就是正确结果了。

那么，前面说到的将前面的商“累积”到本次计算的分式上去，这个“累积”是怎么回事？如何实现？
其实就是通分过程。例如：算式 1/3+1/7，
第一次计算：先对 1/3 计算，得到的结果为 0 余 1。余数存储起来作为下次计算使用，下面计算第二个分式：0+2/7=(0*7)/7+1/7=(0*7+2)/7 等于 0 余 1，输出这个 0，作为结果小数的个位数。
第二次计算：计算第一个算式 1*10/3，结果 3 余 1，余数存储起来，作为下次计算使用。接下来对第二个式子计算：(3*7+1*10)/7，结果 4 余 3，输出 4
第三次计算：计算第一个算式 1*10/3，结果 3 余 1，余数存储起来，作为下次计算使用。接下来对第二个式子计算：(3*7+3*10)/7，结果 7 余 2，输出 7
第四次计算：计算第一个算式 1*10/3，结果 3 余 1，余数存储起来，作为下次计算使用。接下来对第二个式子计算：(3*7+2*10)/7，结果 5 余 6，输出 5
...
上面输出结果顺序为：0475，正确。

看了上面的实际运算过程，现在明白了，要将前面算式的结果累积到当前算式进行计算，方法为：
假如本次计算，前面分式为 **a/b**，结果为 **c** 余数 **d**
当前分式为 **e/f**，上一次计算 **e/f** 的余数为 **g**
那么本次计算的公式为**(c*f+g*10)/f**，算的的结果有一个商和余数，作为后面分式计算的依据，如此循环到最后一个算式。
当计算到最后一个算式，算的的商，是否就可以直接作为结果输出呢？答案是：还不可以！因为我们的这个运算，对每个分式依次计算的同时，商也进行了累加，那么商就有可能出现多于两位数的情况。如果不进行处理而直接输出，得到的结果是不正确的。
处理的办法是：
第一次：将本次商除以 10，得到的商作为最后结果输出，余数则用一个变量记录下来，作为下次输出处理的依据。
第二次：将本次商除以 10 后再加上上次计算的余数，作为最后结果输出。将本次商除以 10 的余数记录下来，作为下次输出处理的依据。
第三次：将本次商除以 10 后再加上上次计算的余数，作为最后结果输出。将本次商除以 10 的余数记录下来，作为下次输出处理的依据。

...

至此，编程的所有原理都解决了，编程如下：

```
@echo off
setlocal enabledelayedexpansion
set /p N=请输入位数：
set /a X1=3,X2=7,X3=13,X4=17
for /l %%i in (1,1,%N%) do set S%%i=1
for /l %%i in (1,1,!N!) do (
        for /l %%k in (1,1,4) do (
            ::将前面算式计算结果累积到当前算式，参与当前算式计算。
            set /a shang=shang*X%%k+S%%k*10
            ::算出当前算式余数。
            set /a S%%k=shang%%X%%k
            ::算出当前算式商
            set /a shang=shang/X%%k
        )
    ::tem 是上一次计算的余数，加上本次的商，作为结果输出
    set /a var=tem+shang/10
    ::记录本次余数
    set /a tem=shang%%10
    set /p kk=!var!<nul
    ::对商清零
    set shang=0
)
pause>nul
```

输入精度 10
输出为：0510119361082
这个结果不对！但仔细看上面程序，看不出哪里出了毛病。我们增加一句输出语句，跟踪一下每次计算的结果：

```
@echo off
setlocal enabledelayedexpansion
set /p N=请输入位数：
set /a X1=3,X2=7,X3=13,X4=17
for /l %%i in (1,1,%N%) do set S%%i=1
for /l %%i in (1,1,!N!) do (
```

```
        for /l %%k in (1,1,4) do (
            set /a shang=shang*X%%k+S%%k*10
            set /a S%%k=shang%%X%%k
            set /a shang=shang/X%%k
        )
    echo shang=!shang!
    set /a var=tem+shang/10
    set /a tem=shang%%10
    set /p kk=!var!<nul
    echo.
    set shang=0
)
pause>nul
```

输入精度 10

输出：

```
shang=4
0
shang=19
5
shang=19
10
shang=28
11
shang=12
9
shang=16
3
shang=9
6
shang=16
10
shang=20
8
shang=24
2
```

将每次输出排列出来：

0　5　10　11　9　3　6　10　8　2

我们看到，在一个位值里，出现了两位数，我们将其进行进位处理后如下：

0　6　1　1　9　3　7　0　8　2

进位处理后的结果就正确了。那么如何解决这个问题呢？我们看到数

字会出现两位数，于是改为每两位数两位数进行计算（此时的进制是 100 进制），输出时也是两位数两位数一起输出，问题不就解决啦！
代码如下：

```
@echo off
setlocal enabledelayedexpansion
set /p N=请输入位数：
set /a X1=3,X2=7,X3=13,X4=17
for /l %%i in (1,1,%N%) do set S%%i=1
for /l %%i in (1,1,!N!) do (
        for /l %%k in (1,1,4) do (
            set /a shang=shang*X%%k+S%%k*100
            set /a S%%k=shang%%X%%k
            set /a shang=shang/X%%k
        )
    echo shang=!shang!
    set /a var=tem+shang/100
    set /a tem=shang%%100
    set /p kk=!var!<nul
    echo.
    set shang=0
)
pause>nul
```

输入精度 10
输出为：

```
shang=59
0
shang=218
61
shang=136
19
shang=106
37
shang=224
8
shang=124
25
shang=130
25
shang=177
31
```

shang=118
78
shang=142
19
我们看到，每次输出结果都不超过两位数，这个和 100 进制正好匹配。我们将输出结果排列出来：
0　61　19　37　8　25　25　31　78　19
看看这个结果，第一个数 0 作为小数的个位数，不予理会。第二个数是 61，第三个数是 19，第四个数是 37...，这里除了第五个数是个位数 8 以外，其余所有数都是两位数。我们将结果排列输出为：
061193782525317819
与正确结果不一样，就是 8 的前面少了个 0。
所以问题到这里还没有最终解决，我们每两位数两位数计算，输出时也是两位数两位数输出，如果某次输出的结果少于两位数，那么需要在数的前面添加 0，补齐两位数输出，这样就正确啦！
怎么对每个数据补齐前导 0 呢？例如这里是两位数两位数进行计算，我们对每个数加上 100，例如第二个数 61+100=161，第三个数 19+100=119，第四个数 37+100=137，第五个数 8+100=108...
然后我们将每个数只输出末尾两位，依次是 6119370825...问题彻底解决！

代码如下：

```
@echo off
setlocal enabledelayedexpansion
set /p N=请输入位数：
set /a X1=3,X2=7,X3=13,X4=17
for /l %%i in (1,1,%N%) do set S%%i=1
::大循环精度控制，每次输出两位数，为了输出 N 位精度，循环 N/2 次即可。
for /l %%i in (1,2,!N!) do (
        for /l %%k in (1,1,4) do (
            set /a shang=shang*X%%k+S%%k*100
            set /a S%%k=shang%%X%%k
            set /a shang=shang/X%%k
        )
    ::这里的加上 100，是为少于两位数的情况补齐前导 0 作预备处理
    set /a var=tem+shang/100+100
    set /a tem=shang%%100
    ::输出后两位结果
```

```
    set /p kk=!var:~-2!<nul
    set shang=0
)
pause>nul
```

上面代码效率还不够高，如果我们每四位数进行计算，效率会更高：

```
@echo off
setlocal enabledelayedexpansion
set /p N=请输入位数：
set /a X1=3,X2=7,X3=13,X4=17
for /l %%i in (1,1,%N%) do set S%%i=1
for /l %%i in (1,4,!N!) do (
        for /l %%k in (1,1,4) do (
            set /a shang=shang*X%%k+S%%k*10000
            set /a S%%k=shang%%X%%k
            set /a shang=shang/X%%k
        )
    set /a var=tem+shang/10000+10000
    set /a tem=shang%%10000
    set /p kk=!var:~-4!<nul
    set shang=0
)
pause>nul
```

注：限于批处理数据的最大值为 2147483647，于是我们计算时，每四位数进行计算是上限啦，每五位数进行计算有可能发生溢出。

问题研究到这里，是不是完美了呢？
答案是：还没有！算法还可以进一步优化！
对于本题 **S**=1/3+1/7+1/13+1/17 算式，是不是想不出还能怎么优化？
确实，局限于看本题，很难看出还能怎么优化算法，看另一个算式：**S**=1/10+1/100+1/1000+1/10000，这个式子的值为：**S**=0.1111，假如我们要求的精度是算到小数点后两位就行，那么最后两项继续参与计算就是多余的。对于无穷级数，这样的问题就更加明显了，级数有无穷多项，我们不可能无穷无尽算下去，我们只能根据级数的收敛速度，选取有限的前若干项，计算到我们需要的精度为止。
例子：
S=1/7+1/70+1/700+1/7000+1/70000+1/700000...=0.1587301571428571
42857142857...
这个级数的收敛速度是每计算 1 项可得 1 位有效精度。

我们选取前三个分式 1/7，1/70，1/700 进行分析，根据上面算法

第一次计算：
(1/7 算式)　　以(0*7+1*10)/7 计算　　结果 1 余 3
(1/70 算式)　　以(1*70+1*10)/70 计算　　结果 1 余 10
(1/700 算式)　　以(1*700+1*10)/700 计算　　结果 1 余 10
输出 1

第二次计算：
(1/7 算式)　　以(0*7+3*10)/7 计算　　结果 4 余 2
(1/70 算式)　　以(4*70+10*10)/70 计算　　结果 5 余 30
(1/700 算式)　　以(5*700+10*10)/700 计算　　结果 5 余 100
输出 5

第三次计算：
(1/7 算式)　　以(0*7+2*10)/7 计算　　结果 2 余 6
(1/70 算式)　　以(2*70+30*10)/70 计算　　结果 6 余 20
(1/700 算式)　　以(6*700+100*10)/700 计算　　结果 7 余 300
输出 7

我们选取前五个分式 1/7，1/70，1/700，1/7000，1/700000 进行分析，根据上面算法

第一次计算：
(1/7 算式)　　以(0*7+1*10)/7 计算　　结果 1 余 3
(1/70 算式)　　以(1*70+1*10)/70 计算　　结果 1 余 10
(1/700 算式)　　以(1*700+1*10)/700 计算　　结果 1 余 10
(1/7000 算式)　　以(1*7000+1*10)/7000 计算　　结果 1 余 10
(1/70000 算式)　　以(1*70000+1*10)/70000 计算　　结果 1 余 10
输出 1

第二次计算：
(1/7 算式)　　以(0*7+3*10)/7 计算　　结果 4 余 2
(1/70 算式)　　以(4*70+10*10)/70 计算　　结果 5 余 30
(1/700 算式)　　以(5*700+10*10)/700 计算　　结果 5 余 100

(1/7000 算式)　以(5*7000+10*10)/7000 计算　结果 5 余 100
(1/70000 算式)　以(5*70000+10*10)/70000 计算　结果 5 余 100
输出 5

第三次计算：
(1/7 算式)　以(0*7+2*10)/7 计算　结果 2 余 6
(1/70 算式)　以(2*70+30*10)/70 计算　结果 6 余 20
(1/700 算式)　以(6*700+100*10)/700 计算　结果 7 余 300
(1/7000 算式)　以(7*7000+100*10)/7000 计算　结果 7 余 1000
(1/70000 算式)　以(7*70000+100*10)/70000 计算　结果 7 余 1000
输出 7

第四次计算：
(1/7 算式)　以(0*7+6*10)/7 计算　结果 8 余 4
(1/70 算式)　以(8*70+20*10)/70 计算　结果 10 余 60
(1/700 算式)　以(10*700+300*10)/700 计算　结果 14 余 200
(1/7000 算式)　以(14*7000+1000*10)/7000 计算　结果 15 余 3000
(1/70000 算式)　以(15*70000+1000*10)/70000 计算　结果 15 余 10000
输出 15

第五次计算：
(1/7 算式)　以(0*7+4*10)/7 计算　结果 5 余 5
(1/70 算式)　以(5*70+60*10)/70 计算　结果 13 余 40
(1/700 算式)　以(13*700+200*10)/700 计算　结果 15 余 600
(1/7000 算式)　以(15*7000+3000*10)/7000 计算　结果 19 余 2000
(1/70000 算式)　以(19*70000+10000*10)/70000 计算　结果 20 余 30000
输出 20

对比一下选取三项进行计算和选取五项进行计算的两组数据，可以看出这么个规律：
对于选取前三项进行计算的情况，假如我们第一次循环三次，计算前三项， 输出 1；第二次循环两次，计算前两项，输出 5，第三次循环一次，计算前一项，输出 2。最后一位和精确结果不同
对于选取前五项进行计算的情况，假如我们第一次循环五次，计算前五项，输出 1；第二次循环两次，计算前四项，输出 5；第三次循环三次，计算前三项，输出 7；第四次循环两次，计算前两项，输出 10；第五次循环一次，计算前一项，输出 5。最后两位和精确结果不同，但第三次计算输出 7，比选取前三项进行计算结果精确了。这是因为

最后几位数据，舍入存在误差导致的。所以，一般的，我们的算法可以这么简化：

一般的，对于一个收敛速度为每计算 **n** 项可得 **m** 位精度的级数，为了获得 **k** 位精度的值，我们总共要对级数的前**(nk)/m** 项进行计算。第一次计算级数的前**(nk)/m** 项，输出 **m** 位有效数值；第二次计算级数的前**[(nk)/m]-n** 项，输出 **m** 位有效数值；第三次计算级数的前**[(nk)/m]-2n** 项，输出 **m** 位有效数值；第四次计算级数的前**[(nk)/m]-3n** 项，输出 **m** 位有效数值；...最后一次计算级数的第一项即可，输出 **m** 位有效数值。

于是代码改进如下，最终最高效算法：

```
@echo off
setlocal enabledelayedexpansion
set /p N=请输入位数：
set /a X1=7,X2=70,X3=700,X4=7000,X5=70000,X6=700000
for /l %%i in (1,1,%N%) do set S%%i=1
::num 作为小循环的次数控制器，次数不断减少
set /a num=N
::大循环，由于每次输出是两位两位输出，为了输出 N 位精度，于是大循环的总次数为 N/2 次。
for /l %%i in (!N!,-2,1) do (
        set /a num-=1
        for /l %%k in (!num!,-1,1) do (
            set /a shang=shang*X%%k+S%%k*100
            set /a S%%k=shang%%X%%k
            set /a shang=shang/X%%k
        )
    set /a var=tem+shang/100+100
    set /p kk=!var:~-2!<nul
    set shang=0
)
pause>nul
```

计算圆周率

在很多计算机语言里，圆周率计算都是一个很经典的课题，它检验程序员对该门计算机语言掌握的熟练程度。在这里，笔者也尝试用批处理进行圆周率计算。

批处理计算圆周率分析

首先选择计算圆周率的公式，用下面的公式是首选：

$$\pi = 2+\frac{1}{3}(2+\frac{2}{5}(2+\frac{3}{7}(2+\cdots(2+\frac{k}{2k+1}(2+\cdots))\cdots)))$$

这条公式比较简单，乘法、除法和加法三个运算的周期性和规律性很强，很容易实现编程。

表达式每计算 7/2=3.5 项，可得一位数精度，收敛速度也算满意。这样，要得到 **n** 位精度的 **pi** 值，那么迭代 **n***7/2 次就行了。

我们可以用一个有 **N** 位元素的数组，对乘除和加法进行迭代运算，达到我们满意的精度为止。

编程如下：

```
@echo off&echo.&echo.&setlocal enabledelayedexpansion
echo                                                圆周率程序
echo. & echo. & set /p N=请输入圆周率位数：
::根据无穷表达式的收敛速度，算出迭代次数 max
set /a max=N*7/2+1
::对圆周率数组赋初始值 3.333...。初始值只要不等于 0 就可以了，当然，越接近 pi 越好，所以这里选择 3.3333...。
for /l %%i in (1,1,%N%) do (set S%%i=3)
for /l %%i in (%max%,-1,1) do (
    ::乘法运算
    for /l %%j in (%N%,-1,1) do (set /a S%%j*=%%i)
    ::除法运算
    set remain=0
    set /a num=%%i*2+1
    for /l %%k in (1,1,%N%) do (
        set /a remain=remain*10+S%%k
        set /a S%%k=remain/num
        set /a remain=remain%%num
    )
```

```
    ::加法运算
    set /a S1+=2
    ::显示进度。当计算的位数比较多时，比如 200 位以上，就有必要显示进度给用户看。
    title 当前剩余计算量%%i
)
::对最终结果数组元素进行进位处理
set jin=0
for /l %%i in (%N%,-1,1) do (
    set /a tem=jin+S%%i
    set /a jin=tem/10
    set /a S%%i=tem%%10
)
::输出结果
set /p var=3.<nul
for /l %%i in (2,1,%N%) do set /p var=!S%%i!<nul
echo. & echo. & echo 计算结束，按任意键退出。&pause>nul
```

压缩一下，紧凑型代码如下：

```
@echo off & setlocal enabledelayedexpansion
echo.&echo.&echo.&set /p N=请输入圆周率位数：
set /a max=N*7/2,jin=0
for /l %%i in (1,1,%N%) do set S%%i=3
for /l %%i in (%max%,-1,1) do (
    set /a num=%%i*2+1,remain=0
    for /l %%k in (1,1,%N%) do (
        set /a remain=remain%%num*10+S%%k
        set /a S%%k=remain/num*%%i
    )
    set /a S1+=2 & title 当前计算进度%%i
)
for /l %%i in (%N%,-1,1) do (
    set /a jin=jin/10+S%%i
    set /a S%%i=jin%%10
)
set /p var=3.<nul
for /l %%i in (2,1,%N%) do set /p var=!S%%i!<nul
echo.&echo.&echo 计算结束，按任意键退出。&pause>nul
```

将上面圆周率计算代码压缩如下：

```
@echo off & setlocal enabledelayedexpansion
echo.&echo.&echo.&set /p N=请输入圆周率位数：
set /a max=N*7/2,jin=0
for /l %%i in (1,1,%N%) do set S%%i=3
for /l %%i in (%max%,-1,1) do (
    set /a num=%%i*2+1,remain=0
    for /l %%k in (1,1,%N%) do set /a
remain=remain%%num*10+S%%k,S%%k=remain/num*%%i
    set /a S1+=2 & title 当前计算进度%%i
)
for /l %%i in (%N%,-1,1) do set /a jin=jin/10+S%%i,S%%i=jin%%10
set /p var=3.<nul& for /l %%i in (2,1,%N%) do set /p var=!S%%i!<nul
echo.&echo.&echo 计算结束，按任意键退出。&pause>nul
```

上面的每一个周期进行的乘法、除法和加法运算后，没有急于马上对数组进行进位处理，这样太浪费运算资源。将最终计算好后的数组进行一次进位处理就行了。

上面的编程思路很简单，程序也很简短，运算速度也还算不错，不过，程序还可以继续改进，这个改进就需要非常高的技巧了。用到的技巧请先参阅上面的《多个分式边计算边输出算法》专题，理解多个分式边计算边输出的原理后，下面继续讲解圆周率计算：

$$\pi = 2+\frac{1}{3}(2+\frac{2}{5}(2+\frac{3}{7}(2+\cdots(2+\frac{k}{2k+1}(2+\cdots))\cdots)))$$

我们知道公式的收敛速度是每计算 3.5 位，可以获得 1 位精度有效数字，那么我们可以这么设计：

请求的精度为 **N**，我们每四位数一起计算和输出，那么计算项数就是 **N***14/4。控制精度的循环，循环 **N**/14 次。

因为对迭代的初始值没多少要求，而每次计算都需要加 2，所以为了代码简单，可以设初始值就是 2。

代码如下：

```
@echo off
setlocal enabledelayedexpansion
set /p N=请输入精度：
set /a N=N/2*7-1
set a=2000
for /l %%i in (1,1,%N%) do set f[%%i]=2000
```

```
for /l %%i in (%N%,-14,0) do (
            for /l %%j in (%%i,-1,1) do (
                        set /a quotient=quotient+f[%%j]*10000
          set /a f[%%j]=quotient%%(%%j*2+1^)
          set /a quotient=quotient/(%%j*2+1^)*%%j
            )
            set /a var=a+quotient/10000+10000
     set /a a=quotient%%10000
     set /a quotient=0
            set /p=!var:~-4!<nul
)
echo.&pause
```

一般算法，有小数点输出，代码 12 行：

```
@echo off & setlocal enabledelayedexpansion
echo.&echo.&echo.&set /p N=请输入圆周率位数：
set /a max=N*7/2,jin=0
for /l %%i in (1,1,%N%) do set S%%i=3
for /l %%i in (%max%,-1,1) do (
     set /a num=%%i*2+1,remain=0
     for /l %%k in (1,1,%N%) do set /a
remain=remain%%num*10+S%%k,S%%k=remain/num*%%i
     set /a S1+=2 & title 当前计算进度%%i
)
for /l %%i in (%N%,-1,1) do set /a jin=jin/10+S%%i,S%%i=jin%%10
set /p var=3.<nul& for /l %%i in (2,1,%N%) do set /p var=!S%%i!<nul
echo.&echo.&echo 计算结束，按任意键退出。&pause>nul
```

高效率算法，有小数点的输出，代码 11 行：

```
@echo off&setlocal enabledelayedexpansion
echo.&echo.&set /p c=请输入圆周率精度：
set /a c=c*7/2,a=boor=2000 & set /p var=3.141<nul
for /l %%i in (1,1,%c%) do set f%%i=%a%
for /l %%i in (%c%,-14,1) do (
            for /l %%j in (%%i,-1,1) do set /a
d+=f%%j*10000,f%%j=d%%(%%j*2+1^),d=d/(%%j*2+1^)*%%j
            set /a var=a+d/10000,a=d%%10000+10000
     if !boor!==0 set /p=!var:~-4!<nul
     set /a d=boor=0
)
echo.&echo.&echo 计算结束，按任意键退出。&pause>nul
```

高效算法，无小数点的输出，代码 8 行：

```
@echo off&setlocal enabledelayedexpansion&echo.&echo.&set /p c=
请输入圆周率精度:
set /a c=c*7/2
for /l %%i in (1,1,%c%) do set /a f%%i=a=2000
for /l %%i in (%c%,-14,1) do (
    for /l %%j in (%%i,-1,1) do set /a
d+=f%%j*10000,b=%%j*2+1&set /a f%%j=d%%b,d=d/b*%%j
    set /a c=a+d/10000+10000,a=d%%10000,d=0&set /p =!c:~-4!<nul
)
echo.&echo  计算完毕，按任意键退出。&pause>nul
```

---

e 计算

我们看看 e 的表达式：

$$e=1+\frac{1}{1!}+\frac{1}{2!}+\frac{1}{3!}+\frac{1}{4!}+\frac{1}{5!}\cdots$$

用上面 e 的级数公式，很容易编写代码：

```
@echo off
setlocal enabledelayedexpansion
title e 高精度计算程序
echo. & echo. & echo                                    e 高精度
计算程序
echo. & echo. & set /p precision=请输入精确到的小数位数：
echo. & echo  计算进行中，进度请看标题栏...
::这里需要对请求的精度 precision 值加 1，是因为我们计算是从 1/2!
开始，假如数组一开始等于 100000，除以 2!就等于 050000，也就是
会产生一个前导 0，这个我们在输出时需要去掉，所以要对精度
precision 值加 1。
set /a precision+=1
::一开始的临时数组 X，首项要等于 1，否则除法永远等于 0，累加到
E 数组，得到的 e 值恒等于 0。然后对数组其余位全部清零。
for /l %%i in (1,1,!precision!) do (set /a E%%i=X%%i=0)
set X1=1
::对于 1/n!，根据阶乘的定义，n 是从 1 开始的，但除以 1 对数值没
```

任何变化，所以除法从 2 开始。

```
set N=2
:loop
set t=0
for /l %%j in (1,1,!precision!) do (
    set /a t=t*10+!X%%j!
    set /a X%%j=t/N
    set /a t%%=N
)
set /a N+=1
set /a boor=k=0
```

::此处对计算的 1/n!值进行判断，算出其前导 0 个数 k，这个可以作为进度显示出来，并且下面累加到 E 数组时，累加到 k+1 项就行了。

```
for /l %%i in (2,1,!precision!) do (
    if !X%%i! neq 0 (
        set boor=1
        set /a k+=1
        ::将每次的商累加到 E 数组里去。这里只管累加，不必进位处理，等需要输出最终结果时，再作进位处理。
        for /l %%j in (!precision!,-1,!k!) do (    set /a E%%j+=X%%j)
        ::由于批处理计算效率低下，这里显示当前计算进度。
        title e 高精度计算程序                                    当前计算到小数位数为!k!
        goto :loop
    ) else (if !boor!==0 set /a k+=1)
)
```

::对数组 E 作进位处理。这里对 E 数组元素是否大于等于 10 进行判断，复合条件才进位处理。其实这个判断可以省略，下面的处理照样进行。

```
for /l %%i in (!precision!,-1,1) do (
    set /a v=v/10+E%%i
    set /a E%%i=v%%10
)
title e 高精度计算程序                                    当前计算到小数位数为!k!
echo. & echo.
echo ------------------------------------------------------------------------
```

::增加这个判断，是防止用户请求的精度是 0 位小数的情况，那么我们输出时就只输出“2”就行了，不应该输出“2.”。

```
if !precision!==1 (set /p var=e=2 <nul) else (set /p var=e=2.<nul)
```

::输出结果

for /l %%i in (2,1,!**precision**!) **do** (**set /p var=**!**E%%i**!<**nul**)
echo. & **echo**. & **echo**
---
echo. & **echo**. & **echo** 由于每次计算都舍弃小数!**k**!位后的数字，所以结果的末尾 2 位或 3 位数可能是不精确的。
echo. & **echo** 计算结束，请按任意键退出。& **pause>nul**

::算法分析：还是采用 **e**=1+1/1!+1/2!+1/3!+1/4!+1/5!...来进行计算。对于每个加数 1/**n**!，我们采用 1/2/3/4/5/6.../**n**，也就是用 1 对连续数字 1、2、3、4、5、6...**n** 连续相除的方式计算，然后将结果加入 **e** 数组，这样编程会变得很简单。如果我们先算阶乘，然后再算倒数，就会造成高精度的乘法以及除法计算，程序复杂不说，效率也很低下。注意，计算时，不要对每一项 1/**n**!从 1 开始除到 **n**，这样效率很差。我们看到 1/**n**!等于 1/(**n**-1)再除以 **n** 就行了，也就是上一项再除以 **n** 值就等于 1/**n**!
我们知道了 **e** 的数值大约等于 2.71828，而前两项 1+1/1!等于 2，后面所有项的和等于 0.71828...那么我们的计算就只需要计算小数部分就行了，计算从 1/2!开始计算。

问题一：我们需要算到多少项才能达到我们预定的精度要求？
假如我们要求的 **e** 数组的精度为 **precision** 位小数，那么我们对每一项 1/**n**!的最后结果作判断，它数组的长度也是 **precision**，如果数组所有元素都等于 0，那么累加到 **e** 数组，**e** 数值不会产生任何变化，计算就到此为止，因为再往后进行 1/(**n**+1)!的计算，它的数组所有元素也会等于 0，同样累加到 **e** 数组，对 **e** 数组不会产生任何变化。
我们每次进行除法运算，都会舍弃末尾 **precision** 位后的小数，这就造成最终结果 **e** 值最后两到三位小数不准确。这个也是可以克服的，就是我们将用户请求的精度 **precision** 增加三位数，最后输出时少输出末尾的三位数就行了，不过上面程序没这么处理。

问题二：我们如何实时看出计算的进度？
因为是用批处理来计算 **e**，批处理的计算效率非常低下，要它计算出 500 位 **e** 值，会计算个老半天，这就要求程序实时显示计算进度。我们可以用当前计算精确到的小数位数的办法来计算进度。也就等于当前计算到的第 **n** 项“1/**n**!”，其结果数组前导 0 的个数就是当前的精确小数位数，也就是进度。因为数组所有 **i** 个前导 0 累加到 **e** 数组，对 **e** 数组的前 **i** 位不会产生任何影响，**e** 数组此时的前 **i** 位是精确值。

将上面计算 e 的代码压缩一下：

```
@echo off & setlocal enabledelayedexpansion
echo. & echo. & set /p precision=请输入 e 的精度：
for /l %%i in (1,1,!precision!) do (set /a E%%i=X%%i=0)
set /a X1=1,N=2
:loop
for /l %%j in (1,1,%precision%) do set /a
t=t%%N*10+X%%j,X%%j=t/N
set /a N+=1,boor=k=t=0
for /l %%i in (2,1,!precision!) do (
    if !X%%i! neq 0 (
        set /a boor=k+=1
        for /l %%j in (!precision!,-1,!k!) do set /a E%%j+=X%%j
        title  当前计算进度!k! & goto :loop
    ) else if !boor!==0 set /a k+=1
)
for /l %%i in (!precision!,-1,1) do set /a v=v/10+E%%i,E%%i=v%%10
echo. & echo.& set /p =e=2.<nul
for /l %%i in (2,1,!precision!) do set /p =!E%%i!<nul
echo. &echo.& echo. & echo  计算结束，请按任意键退出。&
pause>nul
```

++

编程高手可以看出，上面的编程不够完美，程序的效率还可以进一步提高。
我们先将级数公式变形：

$$e=1+\frac{1}{2}(1+\frac{1}{3}(1+\frac{1}{4}(1+\frac{1}{5}(1+\cdots))))$$

这个公式很容易编程，而且效率比较高。我们看到，这个公式的形状和上面圆周率公式形状非常相似，所以，我们仿照写圆周率代码的方法，写出 **e** 的最高效率代码：

```
@echo off&setlocal enabledelayedexpansion
title e 程序　　作者 MHL QQ1208980380
echo.&echo.&set /p c=请输入 e 的精度：
set /a c+=4,a=boor=1000
set /p var=2.718<nul
for /l %%i in (1,1,%c%) do set f%%i=%a%
for /l %%i in (%c%,-4,4) do (
        for /l %%j in (%%i,-1,1) do set /a d+=f%%j*10000&set /a
f%%j=d%%%%j&set /a d=d/%%j
        set /a var=a+d/10000,a=d%%10000+10000
    if !boor!==0 set /p=!var:~-4!<nul
    set /a d=boor=0
)
echo.&echo.&echo 计算结束，按任意键退出。&pause>nul
```

八皇后问题

八皇后问题是回溯算法的经典例子。

下面用批处理实现，至于回溯算法原理，在此不再赘述，有兴趣的朋友可以参考算法教材：

```
@echo off
setlocal enabledelayedexpansion
title 八皇后问题　　作者 MHL QQ1208980380
echo. & echo.
echo                                                    八皇后问题
echo. & echo.
echo. & echo.
for /l %%i in (0,1,14) do (set /a A%%i=B%%i=C%%i=0)
set conter=0
echo 按任意键开始计算 & pause>nul
call :loop 1
echo 共计!conter!个解
echo 解题完毕，按任意键退出。& pause>nul & exit
:loop
for /l %%j in (1,1,8) do (
    set /a jian=%%j-%1+7
    set /a subtract=B!jian!
    set /a jia=%%j+%1
    set /a add=C!jia!
    if !A%%j!==0 (
        if !subtract!==0 (
            if !add!==0 (
                set A%%j=1
                set B!jian!=1
                set C!jia!=1
                set S%1%%j=Q
                if %1 lss 8 (
                    set /a increment=%1+1
                    call :loop !increment!
                ) else (
                set /a conter+=1
                echo 第(!conter!^)个解
                    for /l %%u in (8,-1,1) do (
                        for /l %%v in (1,1,8) do (
```

```
                        if !S%%u%%v!==Q (
                            set /p var=Q <nul
                        ) else (set /p var=+ <nul)
                    )
                    echo.
                )
                echo. & echo.
            )
            set /a endjian=%%j-%1+7
            set /a endjia=%%j+%1
            set   A%%j=0
            set   B!endjian!=0
            set   C!endjia!=0
            set   S%1%%j=0
        )
    )
)
)
```

八皇后问题推广到 n 皇后

```
@echo off
setlocal enabledelayedexpansion
title 广义八皇后问题　作者 MHL QQ1208980380
echo. & echo.
echo                                                    八皇后问题
echo. & echo.
echo        八皇后问题，是一个古老而著名的问题，是回溯算法的典型例题。该问题是国际象棋棋手马克斯·贝瑟尔于 1848 年提出：在 8X8 格的国际象棋上摆放八个皇后，要求其不能互相攻击，即任意两个皇后都不能处于同一行、同一列或同一斜线上，问有多少种摆法？数学王子高斯认为有 76 种方案。1854 年在柏林的象棋杂志上不同的作者发表了 40 种不同的解，后来有人用图论的方法解出全部 92 种解（如果将旋转和对称的解归为一种的话，则一共有 12 个独立解）。本程序可以计算一般化 n 皇后情况。
echo. & echo.


set /p N=请输入皇后数量：
set /a max=2*N

for /l %%i in (0,1,%max%) do (set /a A%%i=B%%i=C%%i=0)

set conter=0
set /p choose=是否将计算结果保存到《!N!皇后问题答案》文本文档？(Y/N):
if /i "%choose%"=="Y" (set boor=1)
```

```
echo  按任意键开始计算  & pause>nul


call :loop 1
echo  共计!conter!个解
if !boor!==1 echo  共计!conter!个解>>!N!皇后问题答案.txt
echo  解题完毕，按任意键退出。& pause>nul & exit

:loop
for /l %%j in (1,1,%N%) do (
      set /a jian=%%j-%1+N
      set /a subtract=B!jian!
      set /a jia=%%j+%1
      set /a add=C!jia!
      if !A%%j!==0 (
            if !subtract!==0 (
                  if !add!==0 (
                        set A%%j=1
                        set B!jian!=1
                        set C!jia!=1
                        set S%1%%j=1
                        if %1 lss %N% (
                              set /a increment=%1+1
                              call :loop !increment!
                        ) else (
                              set /a conter+=1
                              echo  第(!conter!^)个解
                              if !boor!==1 echo  第(!conter!^)个解>>!N!皇后问题答案.txt
                              for /l %%u in (1,1,%N%) do (
                                    for /l %%v in (1,1,%N%) do (
                                          if !S%%u%%v!==1 (
                                                set /p var=Q <nul
                                                if !boor!==1 set /p var=Q <nul>>!N!皇后问题答
案.txt
                                          ) else (
                                                set /p var=+ <nul
                                                if !boor!==1 set /p var=+ <nul>>!N!皇后问题答
案.txt
                                          )
                                    )
                              echo.
                              if !boor!==1 echo.>>!N!皇后问题答案.txt
                              )
                              echo. & echo.
                              if !boor!==1 echo. & echo.>>!N!皇后问题答案.txt
                        )
                        set /a endjian=%%j-%1+N
                        set /a endjia=%%j+%1
                        set    A%%j=0
```

```
                set    B!endjian!=0
                set    C!endjia!=0
                set    S%1%%j=0
            )
        )
    )
)
```

网友批处理程序

下面这些程序都不是笔者的作品，全都来自网上，作者大多都不详。

猜数字游戏一

```
:start
@echo off&color 9f&mode con cols=55 lines=30
setlocal enabledelayedexpansion
title  猜数字游戏

echo.&echo.
echo.&echo                                  猜数字游戏
echo.&echo                              ***** 游戏规则 *****
echo.&echo              程序会自动生成一组四个互不相同的个位数，
echo.&echo        请你根据猜测和判断，每次输入一组四个互不相同
echo.&echo        0-9 内的数，数与数之间用空格隔开，如：1 2 3 4，
echo.&echo        程序会将你每次输入的数组与其生成的数组进行比
echo.&echo        较，并将比较的结果输出屏幕上以帮助你进一步的
echo.&echo        判断，直到你所输入的数组与其生成的数组的数值
echo.&echo        和次序完全相同，或你已用完了所有的猜测机会。
echo.&echo        看懂了规则，就按任意键开始游戏。
set var=A&pause>nul&gt;nul
:begin
set /a a=%random%%%10
for %%a in (%var%) do if %%a equ %a% goto begin
set var=%var% %a%
if "%var:~8%" equ "" goto begin
set var=%var:~2%&set m=0
:again
set /a n=0,v=0,ws=0,s=0,t=10-m
if %t% equ 0 goto end
cls&echo.&echo  请注意你总共只有 10 次机会,现在还剩下%t%次机会
if "%m%" neq "0" for /l %%i in (1,1,%m%) do echo.&echo !_%%i!
set codes=A
echo.&set /p code=  请输入四个数字:
echo %code: =%|findstr "[^0-9]"&&goto wrong
```

```
for %%a in (%code%) do (
for %%i in (!codes!) do if %%i equ %%a goto wrong
set codes=!codes! %%a
set /a n+=1
)
if %n% neq 4 goto wrong
set codes=%codes:~2%
for %%i in (%codes%) do (
set str=%%i
if "!str:~1!" neq "" goto wrong
)
for %%a in (%var%) do (
set /a v+=1,x=0
for %%i in (%codes%) do (
set /a x+=1
if %%a equ %%i if !x! equ !v! (
set /a ws+=1,s+=1
) else (
set /a s+=1
)
)
)
set /a m+=1
if "%var%" equ "%codes%" goto suc
set _%m%= %codes%中数值正确的有%s%个，其中位置数值都正确的有%ws%个&goto again
:wrong
cls
echo.&echo 所输入的非合法十进制整数，或数值不合要求，
echo.&echo 或有重复数，或输入的数值个数不是 4 个。
echo.&echo 请重新开始
pause>nul
goto start
:suc
if %m% equ 1 cls&echo.&echo 不会吧，1 次就搞定了，你是不是神仙下凡啊？ &goto select
if %m% leq 3 cls&echo.&echo 你真是聪明绝顶啊，只用%m%次就猜中了。&goto select
if %m% leq 5 cls&echo.&echo 看来你还是很厉害，用了%m%次就猜中了。&goto select
if %m% leq 9 cls&echo.&echo 呵呵，猜了%m%次，勉强过关了。&goto
```

```
select
if %m% equ 10 cls&echo.&echo 你太精明了，硬是把机会给用完成功过关。
:select
pause>nul
cls&echo.&set select=&set /p select= 是否再来一次(是就输入 Y，其它任意键退出)：
if /i "%select%" equ "y" (goto begin) else (goto exit)
:end
cls&echo.&echo 对不起你已经用完了所有的机会了。正确答案是 %var%&goto select
:exit
pause>nul
```

猜数字游戏二

```
@echo off
rem 将代码保存为 allyesno.bat 在 cmd shell 下运行即可
echo 这是一个猜数字的游戏
echo 游戏开始 已经生成了一个随机的数字
set ok=%RANDOM%>nul
set count=0>nul
echo 请输入你想猜的数字:
goto gamex
:game
echo 再猜一次!一共有 20 次机会，你已经猜了%count%次
:gamex
set/p guess=
echo 请等待。。。。。
rem 简单防止提交非法参数。。。。优化了好几次的代码。。。
if /i "%guess%"=="allyesno" echo 你要猜测的数字是 %ok%，你这个作弊的家伙&&goto endxx
echo "%guess%"|find "!">nul&&goto error
echo "%guess%"|find "+">nul&&goto error
echo "%guess%"|find "-">nul&&goto error
echo "%guess%"|find "*">nul&&goto error
echo "%guess%"|find "/">nul&&goto error
echo "%guess%"|find "~">nul&&goto error
echo "%guess%"|find "%">nul&&goto error
echo "%guess%"|find "&">nul&&goto error
```

```
echo "%guess%"|find "("&gt;nul&&goto error
echo "%guess%"|find ")"&gt;nul&&goto error
echo "%guess%"|find "="&gt;nul&&goto error
echo "%guess%"|find "|"&gt;nul&&goto error
echo "%guess%"|find ","&gt;nul&&goto error
echo "%guess%"|find "&lt;"&gt;nul&&goto error
echo "%guess%"|find "&gt;"&gt;nul&&goto error
echo %guess%|find/i "0x"&gt;nul&&goto error
if "%guess%"=="0" goto upallyesno
set/a debug=%guess%-0&gt;nul
if "%debug%"=="0" goto error
:upallyesno
if "%count%"=="0" goto countx
goto gameallyesno
:countx
set %count%"=="1"
goto gamenow
:error
echo  你输入的不是数字，请你输入数字
goto gamex
:gameallyesno
set/a count=%count%+1
if "%count%"=="20" echo  猪头啊你，猜了 20 次都猜不对&&goto pig
:gamenow
if "%guess%"=="%ok%" echo  哇靠你真 NB，你猜对了喔&&goto end
set/a %guess%-%ok%|find "-"&gt;nul&&echo  你猜得数字小了喔&& goto
game
echo  你猜得数字大了喔
goto game
:pig
echo  把你抓去卖了  正确答案是  %ok%
goto endxx
:end
if "%count%"=="1" echo  你太厉害了，一次就猜对了
:endx
echo  恭喜你,你一共猜了  %count%次
:endxx
```

俄罗斯方块

本代码非常牛，模拟俄罗斯方块游戏很成功。

```
@goto :` 俄罗斯方块游戏，由 netbenton 编写，在 bathome 首发，完成时间：2009 年 9 月 25 日  ver 1.3
e100 EB 02 00 00 CD 1A 89 16 02 01 31 C0 CD 1A 3B 16 02 01 75 0E
e114 B4 01 CD 16 74 F0 B4 00 CD 16 88 E0 EB 02 B0 00 B4 4C CD 21
g
:`
@echo off
set              "d-v=for /l %%a in (0,1,#) do set/a
one=0x!str:~%%a,1!,x=one/4+n,y=one%%4+m&(for %%b in ("r!x!.!y!")
do if "!%%~b!" equ "    " (set %%~b=■) else (set err=1))"
::函数 d-v，把 str 中的图标数据，放置到总坐标空间中，如果有覆盖，则 err=1。调用方法：(%d-v:#=图标点数%)
::变量使用：one    x    y
set "d-e=set aec=!cr!!cr!                ┌──────────────────────
┐ !cr!&(for /l %%a in (2,1,#) do for %%b in (" │ !ebuf:@=%%a!
 │        !ebu%%a!") do set aec=!aec!
%%~b!cr!)&cls&echo;!aec!                 └──────────────────────┘ "
::函数 d-e，把总坐标空间显示出来，调用方法：(%d-e:#=行数%)
set ebu4=第!guan!关
set ebu6=总分：!fen!
set cr=^


::各种图标定义
set ga1=0156
set ga2=1458
set gb1=1245
set gb2=0459
set za1=4567
set za2=159d
set qa1=1598
set qa2=0456
set qa3=0148
set qa4=0126
```

```
set qb1=0489
set qb2=0124
set qb3=0159
set qb4=2456
set ta1=0145
set sa1=0125
set sa2=1459
set sa3=1456
set sa4=0458
chcp 437 >nul
graftabl 936>nul
setlocal enabledelayedexpansion
::按键定义
set  转=k%%=mx,k+
set  速=down
set  左=m-
set  右=m+
set  停=paus%%=2,paus+
set  退=quit

cls
echo  批处理俄罗斯方块，300 分为一关
echo  一次消一行得 10 分
echo  一次消二行得 30 分
echo  一次消三行得 50 分
echo  一次消四行得 70 分
echo;
echo  请自定义操作按键
echo;
call :setkey  左  "左按键"
call :setkey  右  "右按键"
call :setkey  速  "加速键"
call :setkey  转  "转动键"
call :setkey  停  "暂停"
call :setkey  退  "退出"

set paus=1
mode con: rate=50 delay=0
for /l %%a in (0,1,10) do (set "ebuf=!ebuf!^!r@.%%a^!")
for %%a in (sa_4 ta_1 qb_4 qa_4 za_2 ga_2 gb_2) do (
```

```
		for /f "tokens=1,2 delims=_" %%b in ("%%a") do (
			set _%%b=%%c
			for /l %%d in (1,1,%%c) do set/a nx+=1&set
ran!nx!=%%b%%d
		)
)
::定义各种图型的可变型数，及单个图的随机号
:restart
for /l %%a in (0,1,20) do (for /l %%b in (0,1,10) do set r%%a.%%b=    )
set/a fen=0,guan=1
::初始化坐标空间 20 行，10 列
:loop
set/a "m=4,t=2,n=1,down=450/(guan<<2),bti=0"
set/a r=!random!%%nx+1
set err=
set ttr=!_ttr!
set _ttr=!ran%r%!
::随机取一个图标，
if not defined ttr goto :loop
set mx=!_%ttr:~,2%!
set k=%ttr:~2%
set str=!%ttr%!
set _str=!%_ttr%!

setlocal enabledelayedexpansion
for /l %%a in (0,1,3) do set/a
one=0x!_str:~%%a,1!,x=one/4+8,y=one%%4&set kk!x!.!y!=■
for /l %%a in (8,1,11) do for /l %%b in (0,1,3) do (
		if defined kk%%a.%%b (set kk%%a=!kk%%a!!kk%%a.%%b!)
else (set kk%%a=!kk%%a!    )
)
endlocal&set ebu8=%kk8%&set ebu9=%kk9%&set
ebu10=%kk10%&set ebu11=%kk11%
::对预备图标的处理

:cont
::读取按键
debug<"%~n0.bat">nul
set key=%errorlevel%
(if %paus% equ 2 goto :pause
if defined k%key% (
			if %key% equ %退% call :error %key%&exit
```

```
                    set bs=!str!&set/a bm=m,bk=k
                    set/a !k%key%!=1
                    if "%key%" equ "%转%" (for %%b in ("!ttr:~,2!!k!")
do set str=!%%~b!)
                    setlocal enabledelayedexpansion
                    (%d-v:#=3%)
                    rem  调用函数把图标点放置到总坐标空间
                    if defined err (
                              endlocal
                              set/a m=bm,k=bk
                              set str=!bs!
                    ) else (
                              (%d-e:#=20%)
                              rem  调用显示函数
                              endlocal
                    )
)

set ti=1!time:~7,1!!time:~9,2!
if !ti! lss !bti! (set /a tn=ti-bti+1000) else (set /a tn=ti-bti)
if !tn! gtr !down! (
          set/a bti=ti,n+=1
          setlocal enabledelayedexpansion
          (%d-v:#=3%)
          if defined err (
                    endlocal
                    goto :jmpout
          ) else (
                    (%d-e:#=20%)
                    endlocal
          )
)
if %key% equ %停% echo  再按暂停键继续。。。
goto :cont)
:jmpout
set/a n-=1
(%d-v:#=3%)
set m=20
for /l %%a in (20,-1,2) do for %%b in ("!ebuf:@=%%a!") do if "%%~b"
neq "■■■■■■■■■■■■" (set e!m!=%%~b&set/a m-=1)
if !m! neq 1 (
          for /l %%a in (!m!,-1,2) do set "e%%a=
```

```
"
        for /l %%a in (20,-1,2) do (
                for /l %%b in (0,1,10) do set
r%%a.%%b=!e%%a:~%%b,1!
        )
        set/a "fen=fen+(m-1)*20-10,guan=fen/300+1"
)
if !n! leq 2 (
        cls&(%d-e:#=20%)
        echo            游戏结束!
        ping -n 3 127.1 >nul
        goto :restart
)
goto :loop
:error
if %1 equ %退% echo  谢谢使用，再见。。。
ping -n 3 127.1 >nul
mode con rate=30 delay=1
goto :eof
:pause
if %key% equ %停% set paus=1
if %key% equ %退% goto :error
goto :cont
:setkey
echo;
echo  定义：  %~2
:sk_lp
ping -n 1 127.1 >nul
debug<"%~n0.bat">nul
set var=%errorlevel%
if %var% equ 0 goto :sk_lp
if defined k%var% echo;不能重复定义&goto :setkey
set k%var%=!%1!
set %1=%var%
goto :eof
```

猜拳游戏

```
@echo off
title 批处理版：猜 拳 游 戏
setlocal enabledelayedexpansion
set x=0
set y=0
set z=0
set /p t=输入想要玩的次数：
:s
set /a xyz=!x!+!y!+!z!+1
echo Round !xyz!
echo 石头、剪刀、布
set /p you=输入你的：
set /a c=%random%%%3
if %c%==0 (set com=石头) else if %c%==1 (set com=剪刀) else if
%c%==2 (set com=布)
echo.
if "%you%"=="%com%" (echo 平手！ & set /a z=!z!+1)
if "%com%"=="石头" if "%you%"=="剪刀" (echo 电脑赢！ & set /a
x=!x!+1)
if "%com%"=="剪刀" if "%you%"=="布" (echo 电脑赢！ & set /a
x=!x!+1)
if "%com%"=="布" if "%you%"=="石头" (echo 电脑赢！ & set /a
x=!x!+1)
if "%com%"=="石头" if "%you%"=="布" (echo 你赢！& set /a y=!y!+1)
if "%com%"=="剪刀" if "%you%"=="石头" (echo 你赢！ & set /a
y=!y!+1)
if "%com%"=="布" if "%you%"=="剪刀" (echo 你赢！& set /a y=!y!+1)
echo=
if !xyz!==%t% goto e
goto s
:e
set /a yz=!y!+!z!
set /a rate=%yz%*100/!xyz!
echo 玩了!xyz!次，赢了!y!次，输了!x!次，平手!z!次。胜算率为
%rate%%%！
pause
```

字母拼图游戏

```
@echo off&title 字母拼图游戏(4×4)&mode con:lines=6 cols=60
color 2A
::片头开始。。。
for /l %%i in (1,1,30) do (set/p=━<nul&for /l %%i in (1,1,100) do
echo.>nul)
set str1= xxxxxxxxxxxxxxx ╭字母拼图游戏(4×4)版╮
xxxxxxxxxxxxxxxxxxx
set str2= xxxxxxxxxx╱一款常见的游戏只不过改编成了批处理╲
xxxxxxxxxx
set str3= xxxxxxxxxxxxxxxxxx ╰不好玩不要怪我╯
xxxxxxxxxxxxxxxxxxxxx
for /l %%i in (1,1,3) do (
    for /l %%j in (0,1,50) do (call set/p=%%str%%i:~%%j,1%%<nul&for
/l %%i in (1,1,150) do echo.>nul)
    echo.
)
for /l %%i in (1,1,30) do (set/p=━<nul&for /l %%i in (1,1,100) do
echo.>nul)
pause

::正式开始
color 0f&mode con:lines=13 cols=60
setlocal enabledelayedexpansion
:restart
set a=1
set "w16=    "
for %%i in (Ａ,Ｂ,Ｃ,Ｄ,Ｅ,Ｆ,Ｇ,Ｈ,Ｉ,Ｊ,Ｋ,Ｌ,Ｍ,Ｎ,Ｏ) do (
set w!a!=%%i
set/a a+=1
)
::列出 15 个字母
for /l %%i in (1,1,16) do (
set /a rnd=!random!%%16+1
call set tmp=%%w!rnd!%%
set w!rnd!=!w%%i!
set w%%i=!tmp!
)
::字母和空格随机交换
set lastturn=无
```

```
set times=0


:loop
cls
for /l %%i in (1,1,16) do (
if "!w%%i!"==""    " set space=%%i
)
::检测空格位置
set/p= ┌──┬──┬──┬──┐  上一步:%lastturn%  总步数:%times%<nul
echo.
set/p=┃%w1%┃%w2%┃%w3%┃%w4%┃  除空格外正确格数为：%n%  目标:15<nul
echo.
set/p=├──┼──┼──┼──┤  按键:s(下)、a(左)、d(右)、w(上)<nul
echo.
set/p=┃%w5%┃%w6%┃%w7%┃%w8%┃<nul
echo.
set/p=├──┼──┼──┼──┤  游戏规则：<nul
echo.
set/p=┃%w9%┃%w10%┃%w11%┃%w12%┃        上下左右移动方块，当字母顺序全部<nul
echo.
set/p=├──┼──┼──┼──┤  正确且空格在最后一格时就成功。<nul
echo.
set/p=┃%w13%┃%w14%┃%w15%┃%w16%┃<nul
echo.
set/p= └──┴──┴──┴──┘ <nul
echo.
set/p= %wrong%<nul
echo.
set n=0
if "%w1%"=="Ａ" set/a n=n+1
if "%w2%"=="Ｂ" set/a n=n+1
if "%w3%"=="Ｃ" set/a n=n+1
if "%w4%"=="Ｄ" set/a n=n+1
if "%w5%"=="Ｅ" set/a n=n+1
if "%w6%"=="Ｆ" set/a n=n+1
if "%w7%"=="Ｇ" set/a n=n+1
if "%w8%"=="Ｈ" set/a n=n+1
```

```
if "%w9%"=="Ⅰ" set/a n=n+1
if "%w10%"=="Ｊ" set/a n=n+1
if "%w11%"=="Ｋ" set/a n=n+1
if "%w12%"=="Ｌ" set/a n=n+1
if "%w13%"=="Ｍ" set/a n=n+1
if "%w14%"=="Ｎ" set/a n=n+1
if "%w15%"=="Ｏ" set/a n=n+1
if "%n%"=="15" goto win
::检测是否完成
set "wrong="
set/p num=请输入 s/a/d/w 并按回车：
if /i "%num%"=="finish" set winx=你个作弊的家伙！！ &&goto setx
if /i "%num%"=="s" goto down
if /i "%num%"=="w" goto up
if /i "%num%"=="a" goto left
if /i "%num%"=="d" goto right
::写 finish 就直接成功了。。。
set "wrong=您输入的不是 s/a/d/w,请重新输入！！！"
goto loop
:down
if %space%==1 goto wrong_down
if %space%==2 goto wrong_down
if %space%==3 goto wrong_down
if %space%==4 goto wrong_down
set/a moveto=%space%-4
set w%space%=!w%moveto%!
set "w%moveto%=    "
set/a times+=1
set lastturn=下
goto loop

:up
if %space%==13 goto wrong_up
if %space%==14 goto wrong_up
if %space%==15 goto wrong_up
if %space%==16 goto wrong_up
set/a moveto=%space%+4
set w%space%=!w%moveto%!
set "w%moveto%=    "
set/a times+=1
set lastturn=上
```

```
goto loop

:left
if %space%==4 goto wrong_left
if %space%==8 goto wrong_left
if %space%==12 goto wrong_left
if %space%==16 goto wrong_left
set/a moveto=%space%+1
set w%space%=!w%moveto%!
set "w%moveto%=    "
set/a times+=1
set lastturn=左
goto loop

:right
if %space%==1 goto wrong_right
if %space%==5 goto wrong_right
if %space%==9 goto wrong_right
if %space%==13 goto wrong_right
set/a moveto=%space%-1
set w%space%=!w%moveto%!
set "w%moveto%=    "
set/a times+=1
set lastturn=右
goto loop

:wrong_down
set wrong=不能向下移动!!
goto loop
:wrong_up
set wrong=不能向上移动!!
goto loop
:wrong_left
set wrong=不能向左移动!!
goto loop
:wrong_right
set wrong=不能向右移动!!
goto loop

:win
echo %winx%
pause>nul
```

```
cls
echo  所用步数:%times%
echo  按任意键重新开始
pause>nul
goto restart
:setx
set a=1
for %%i in (A,B,C,D,E,F,G,H,I,J,K,L,M,N,O) do (
set w!a!=%%i
set/a a+=1
)
set "w16=    "
goto loop
```

五子棋游戏一（无禁手）

```
@echo off&setlocal enabledelayedexpansion
mode con: lines=43 cols=110
set li39=       A    B    C    D    E    F    G    H    I    J    K    L
M    N    O    P    Q    R    S
set   li0=  ┌──────────────────────────────────────────
──────────────────────┐
set   li1=A │ ┌──┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──┐
──┬──┬──┬──┬──┬──┐ │ A
set var=1
for %%a in (!li39:~5^,-1!) do (set/a var+=2&set li!var!=%%a │ ├──┼
──┼──┼──┼──┼──┼──┼──┼──┼──┼──┼──┼──┼──┼──┼──┼──┼──┼──┼──┼──
┼──┼──┤ │ %%a)
for /l %%a in (2,2,36) do (set li%%a=    │ │    │    │    │    │    │
 │    │    │    │    │    │    │    │    │    │    │    │ │ )
set li37=S │ └──┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──
┴──┴──┴──┴──┴──┴──┘ │ S
set li38=  └──────────────────────────────────────────
──────────────────────┘

set str=a b c d e f g h i j k l m n o p q r s
for %%a in (%str%) do (set/a .+=1,%%a=.&set z!.!=%%a)
set li5=!li5!     五 棋 子 人 机 对 战
set li7=!li7!           批 处 理
```

```
set li10=!li10!　　电 脑 水 平 中 等
set li31=!li31!　由 netbenton 编写完成
set li33=!li33!　棋盘设计参照了 batman
title　　批处理五子棋

set str=####################
set .=0
for /l %%a in (1,1,19) do (
set he%%a=!str!&set sh%%a=!str!
for /l %%b in (1,1,19) do set [%%a.%%b=0
)
set .=33
for /l %%a in (5,1,19) do (
set pi%%a=!str:~,%%a!&set ni%%a=!str:~,%%a!
set pi!.!=!str:~,%%a!&set ni!.!=!str:~,%%a!
set/a .-=1
)

set ●=○&set ○=●
set zhi=●
set say=say
::设置电脑 IQ
set idea=@@@@#.1 #@@@@.5 @#@@@.4 @@@#@.2 @@#@@.3
vs0 $$$$#.1 #$$$$.5 $$#$$.3 $#$$$.4 $$$#$.2 vs1 #@@@##.2
##@@@#.5 #@@#@#.3 #@#@@#.4 vs2 #@##@@#.4-5
#@@##@#.4-3 #@#@@.3-5 @#@@#.4-1 #@@@##.2-W-1
##@@@#.5-W-6 vs3
set idea=!idea! ##@@@.4-W-5 @@@##.2-W-1 @##@@#.4-5
#@##@@.3-4 #@#@#@.4-2 @#@#@#.3-5 vs4 #$$#$#.3-W-6-1
#$#$$#.4-W-1-6 ##$$$#.5-W-1-6 #$$$##.2-W-1-6 vs5
##@@##.2-5-W-6-1 #@#@#.3-w-1-5 ##$$$.W-4-5 $$$##.W-2-1
$$##$.W-2-3 $##$$.W-3-4 $#$$#.W-4-1 $#$#$.W-4-2 #$$#$.W-2-5
##$$#.W-4-W-1-5 #$$##.W-2-W-1-5 #$#$#.W-3-W-1-5 #$##$#.W-3-4
set idea=!idea! vs7 #$$$#.1-5 @@###.4-3 ###@@.3-4
###@###.3-5-W-2-6-W-1-7 vs8 ###$###.3-5 vs9 @####.4 ####@.2
#$###.3 ###$#.3
set idea=!idea! ###@#.3 #@###.3 $####.3 ####$.3 $$###.3 ###$$.3
$#$##.2 ##$#$.4 #$##$.3 $##$#.3 $###$.3 vs10
set iqam=1000000000
:restart
(
setlocal enabledelayedexpansion
```

```
for /l %%a in (0,1,39) do (echo       !li%%a!)
set li39=!li39!     reboot 重新开始,exit 退出。
set li37=!li37!             back  悔棋
set /p var=选择谁先下[ W,玩家    D,电脑    Q,退出  ]:
if /i "!var!" equ "Q" goto :quit
if /i "!var!" equ "D" (set onez=○&set towz=●&set hou=☆) else (set
onez=●&set towz=○&set hou=★)
set a!onez!=电脑&set a!towz!=玩家
)
(
set ttr=!idea:@=%onez%!&set ttr=!ttr:$=%towz%!
set idea=
for %%a in (!ttr!) do (
for /f "tokens=1,2 delims=." %%b in ("%%a") do (set %%b=%%c&set
idea=!idea! %%b)
)
set ttr=
set li27=!li27!       !onez! !a%onez%!
set li25=!li25!       !towz! !a%towz%!
set/a pos=10,poh=10&goto :getok
)
:loop
(if %zhi% equ %onez% goto :men
set .=
setlocal enabledelayedexpansion
for %%a in (!idea!) do (
set str=%%a
if "!str:~,2!" neq "vs" (
for %%b in (he sh) do (
   set
all=!%%b1!!%%b2!!%%b3!!%%b4!!%%b5!!%%b6!!%%b7!!%%b8!!%
%b9!!%%b10!!%%b11!!%%b12!!%%b13!!%%b14!!%%b15!!%%b16!!
%%b17!!%%b18!!%%b19!
   if "!all:%%a=!" neq "!all!" (
     for /l %%c in (1,1,19) do (
       if "!%%b%%c:%%a=!" neq "!%%b%%c!" set/a .+=1&set
put!.!=%%b %%c.%%a.!iqam!
     )
) )
for %%b in (pi ni) do (
   set
all=!%%b5!!%%b6!!%%b7!!%%b8!!%%b9!!%%b10!!%%b11!!%%b12
```

```
!!%%b13!!%%b14!!%%b15!!%%b16!!%%b17!!%%b18!!%%b19!!%%b
20!!%%b21!!%%b22!!%%b23!!%%b24!!%%b25!!%%b26!!%%b27!!%
%b28!!%%b29!!%%b30!!%%b31!!%%b32!!%%b33!
   if "!all:%%a=!" neq "!all!" (
     for /l %%c in (5,1,33) do (
       if "!%%b%%c:%%a=!" neq "!%%b%%c!" set/a .+=1&set
put!.!=%%b %%c.%%a.!iqam!
     )
) )
) else (
   set/a "iqam=(iqam+1)/8"
   if %%a equ vs8 if defined . goto :get
   if %%a equ vs9 if defined . goto :get



)
))
if defined . (goto :get)
echo. 已经和棋了● ● ●
pause
endlocal&goto :restart
:men
(
set/a .=lips-1&for /f "tokens=1-3" %%b in ("li!liph! !lips! !.!") do (set
%%b=!%%b:~0,%%d!%hou%!%%b:~%%c!)
set li38=!li38![%悔:~,24%]
cls
for /l %%a in (0,1,39) do (echo        !li%%a!)
for /f "tokens=1-3" %%b in ("li!liph! !lips! !.!") do (set
%%b=!%%b:~0,%%d!%zhi%!%%b:~%%c!)
set li38=%li38%
set /p user=!say:say=%error%! [列前，行后]:●
if "!user!" equ "reboot" endlocal&goto :restart
if "!user!" equ "exit" goto :quit
if "!user!" equ "back" call :悔&goto :men
set/a pos=!user:~0,1!,poh=!user:~1,2!,var=pos-1 2>nul
if not defined [!poh!.!pos! set error=输入点不存在● &goto :men
)
if "!he%poh%:~%var%,1!" neq "#" set error=该点已经有子● &goto men
goto :getok
:get
set `=
```

```
::取最佳的走法
for /l %%z in (!.!,-1,1) do (
for /f "tokens=1,2,3 delims=." %%1 in ("!put%%z!") do (
for /f "tokens=1-4" %%a in ("%%1 %%2") do (
set iqm=%%3
set vara=!%%a%%b:*%%c=!srqponmlkjihgfedcba0
for %%4 in (!%%2:-^=;!) do (
     if "%%4" equ "W" (set/a iqm=iqm/5*3) else (
   set/a var=!vara:~19,1!+%%4
   if "%%a" equ "he" (set/a poh=%%b,pos=20-var)
   if "%%a" equ "sh" (set/a poh=20-var,pos=%%b)
   if %%b lss 19 (set/a var=%%b-var+1) else (set/a var=38-%%b-var+1)
   if "%%a" equ "pi" (if %%b lss 19 (set/a pos=var,poh=%%b-var+1) else
(set/a poh=20-var,pos=%%b-19+var))
   if "%%a" equ "ni" (if %%b lss 19 (set/a pos=var,poh=19-%%b+var)
else (set/a poh=var,pos=%%b-19+var))
   if not defined R!pos!R!poh!R set /a `+=1&set ram!`!=R!pos!R!poh!R
   set/a R!pos!R!poh!R+=iqm
     )
)
)
)
)
set rmk=0
for /l %%a in (1,1,!`!) do (
for %%b in (!ram%%a!) do (
   for %%c in (!%%b!) do (
     if %%c gtr !rmk! set/a rmk=%%c,.=0
     if %%c equ !rmk! set rmz!.!=%%b&set/a .+=1
   )
) )
set/a .=!random!%%.
for /f "tokens=1,2 delims=R" %%a in ("!rmz%.%!") do (set/a
pos=%%a,poh=%%b)

rem start set r^&echo !.!^&pause^&exit

endlocal&set/a pos=%pos%,poh=%poh%
set say=say !z%pos%!!z%poh%!(%poh%)&set error=电脑最后下在：
:getok
set zhi=!%zhi%!&set win=!zhi!!zhi!!zhi!!zhi!!zhi!
(set/a piph=poh+pos-1,lips=pos*2+1,niph=19+pos-poh,liph=poh*2-1
if !piph! lss 19 (set/a pips=pos) else (set/a pips=20-poh)
```

```
if !niph! lss 19 (set/a nips=pos) else (set/a nips=poh)
for %%a in ("li!liph! !lips!" "he!poh! !pos!" "sh!pos! !poh!"
"pi!piph! !pips!" "ni!niph! !nips!") do (
for /f "tokens=1,2" %%b in (%%a) do (
   set/a .=%%c-1
   for %%d in (!.!) do (set %%b=!%%b:~0,%%d!%zhi%!%%b:~%%c!)
   if "!%%b:%win%=!" neq "!%%b!" set win=y
)
))
(set/a asc%zhi%+=1
set  悔= !z%pos%!!z%poh%!!悔!
if !win! neq y goto :loop)
for /l %%a in (0,1,39) do (echo        !li%%a!)
set/p=●       !a%zhi%! %zhi%子  第!asc%zhi%!手
● !z%pos%!!z%poh%!(%poh%)    胜出●       <nul
pause
endlocal&goto :restart
:悔
if not defined  悔  goto :eof
if "!悔:~6,1!" equ "" goto :eof
for %%a in (!悔:~^,6!) do (set str=%%a
set/a poh=!str:~-1!,pos=!str:~,1!
set/a piph=poh+pos-1,niph=19+pos-poh,liph=poh*2-1,lips=pos*2+1
if !piph! lss 19 (set/a pips=pos) else (set/a pips=20-poh)
if !niph! lss 19 (set/a nips=pos) else (set/a nips=poh)
for %%a in ( "he!poh! !pos!" "sh!pos! !poh!" "pi!piph! !pips!"
"ni!niph! !nips!") do (
for /f "tokens=1,2" %%b in (%%a) do (
     set/a .=%%c-1
     for %%d in (!.!) do (set %%b=!%%b:~0,%%d!#!%%b:~%%c!)
)
)
for /f "tokens=1,2" %%b in ("li!liph! !lips!") do (
set/a .=%%c-1
for %%d in (!.!) do (set %%b=!%%b:~0,%%d!╋!%%b:~%%c!)
))
set/a asc%zhi%-=1
set  悔=!悔:~6!
set error=你悔棋，耍赖皮！
if not defined  悔  goto :eof
set/a poh=!悔:~2,1!,pos=!悔:~1,1!,liph=poh*2-1,lips=pos*2+1
```

```
set say=say !z%pos%!!z%poh%!(%poh%)
goto :eof
:quit
taskkill /fi "WINDOWTITLE eq  批处理五子棋*" /im cmd.exe
```

五子棋游戏二（有禁手）

```
@echo off
title  五子棋大战(人机对战) BY CafeNoir Email:caruko@qq.com
mode con: lines=32 cols=90
:start
setlocal enabledelayedexpansion
::初始变量
set "tzk=11111_9999999 011110_300000 11110_2400 01111_2400
0011100_3000 11101_2000 10111_2000 11011_2300  11100_500
00111_500 010110_1650 011010_1650 10011_1550 11001_1550
10101_1590 00011000_1000 0010100_950 11000_250 00011_250
0010100_350 010010_300"
set "p_tzk=22222_9999999 022220_300000 22220_2200 02222_2200
0022200_2800 22202_2000 20222_2000 22022_2100 22200_450
00222_450 020220_1550 022020_1550 20022_1500 22002_1500
20202_1530 00022000_9000 0020200_900 22000_200 00022_200
0020200_280 020020_250"
set "C_禁手=011010_三 010110_三 011100_三 001110_三 0011100_
三 01111_四 11110_四  10111_四 11011_四  11101_四 11111_五
111111_长连"
set "P_禁手=022020_三 020220_三 022200_三 002220_三 0022200_
三 02222_四 22220_四  20222_四 22022_四  22202_四 22222_五
222222_长连"
set /a
key_a=1,key_b=2,key_c=3,key_d=4,key_e=5,key_f=6,key_g=7,key_h=8
,key_i=9,key_j=10,key_k=11,key_l=12,key_m=13,key_n=14,key_o=15
set "error1=坐标格式错误，或者坐标越界。"
set "error2=该坐标已有棋子，无法在指定坐标下子。"
set /a 步=0


::初始化棋盘
```

```
set "Display_15= ┌──┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──┐ "
for /l %%i in (2 1 14) do (
            set "Display_%%i= ├──┼──┼──┼──┼──┼──┼──┼──┼──┼──┼──┼──┼──┼──┼──┤ "
)
set "Display_1= └──┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──┘ "
for /l %%i in (1 1 15) do for /l %%j in (1 1 15) do set QP[%%i][%%j]=0
call :显示棋盘
::设置先手
:xs
set /p xs_in=请选择先手(玩家 Player：P，电脑 Computer：C）
if "!xs_in!"=="C" (set "行动方=Computerr" & set /a k1=10,k2=7 & set "先手=C" & set "qizi1=○" & set "qizi2=●")
if "!xs_in!"=="c" (set "行动方=Computer" & set /a k1=10,k2=7 & set "先手=C" & set "qizi1=○" & set "qizi2=●")
if "!xs_in!"=="p" (set "行动方=Player" & set /a k1=7,k2=10 & set "先手=P" & set "qizi1=●" & set "qizi2=○")
if "!xs_in!"=="P" (set "行动方=Player" & set /a k1=7,k2=10 & set "先手=P" & set "qizi1=●" & set "qizi2=○")
if not defined 先手 goto :xs

if "!先手!"=="C" (call :设置坐标 8 8 1) else (call :设置坐标 8 8 2 &goto :loop2)

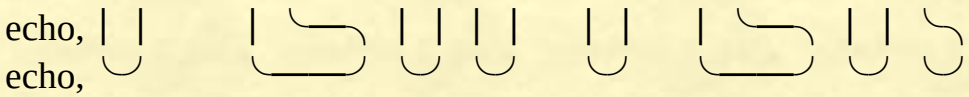
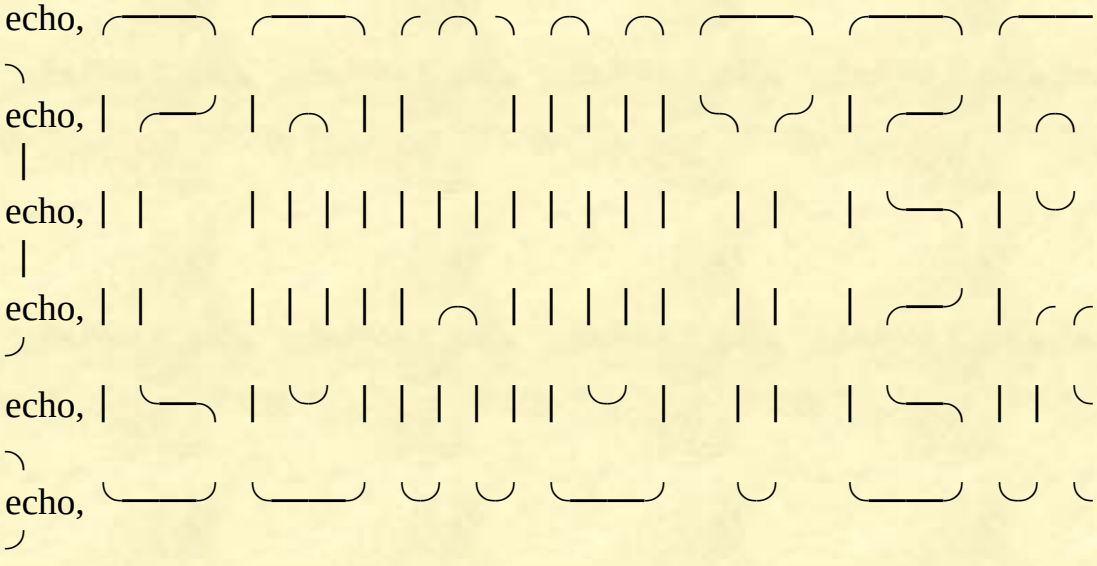
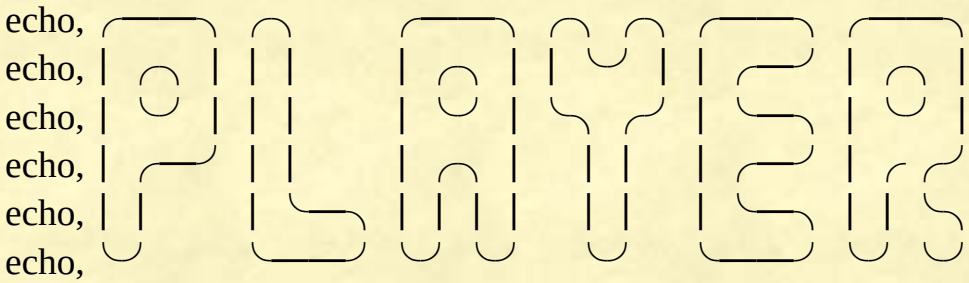
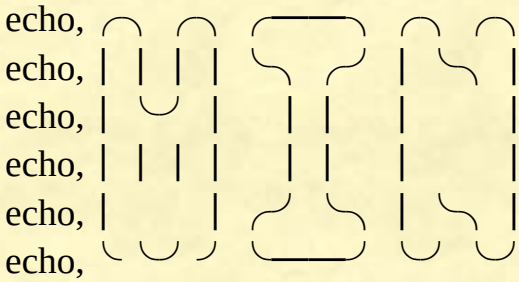
:loop
call :显示棋盘
call :玩家走
if !ERRORLEVEL! gtr 0 (
   echo,!error%ERRORLEVEL%!
   goto :loop
)
:loop2
call :显示棋盘
call :电脑智能
goto :loop

:玩家走
set "行动方=Player"
```

```
set /a zq=0
set /p P_zb=请输入坐标{A-O1-15}如"K5":
echo !p_zb!|findstr /i "^[a-oA-O][1-9]$ ^[a-oA-O]1[0-5]$ ^exit$">nul
||exit /b 1
if "!P_zb!"=="exit" exit
set p_zb=!key_%p_zb:~0,1%!,%p_zb:~1%
for /f "tokens=1,2 delims=," %%a in ("!P_zb!") do (
            if "!QP[%%a][%%b]!"=="0" (
                call :设置坐标  %%a %%b 2 || call :胜利
%ERRORLEVEL%
             ) else (
                exit /b 2
            )
)
exit /b 0



:胜利
call :显示棋盘
if "%1"=="2" (
   echo !行动方!范"!禁!"禁手，该局判负！
   if "!行动方!"=="Player" (
echo, ┌──┐ ┌──┐ ┌─┐ ┌┐ ┌┐ ┌──┐ ┌──┐ ┌──
┐
echo, │ ┌─┘ │ ┌┐ │ │ │ │ │ │ └┐ ┌┘ │ ┌─┘ │ ┌┐
 │
echo, │ │   │ │ │ │ │ │ │ │ │ │ │ │ │ └─┐ │ └┘
 │
echo, │ │   │ │ │ │ │ ┌┐ │ │ │ │ │ │ │ ┌─┘ │ ┌┐
┘
echo, │ └─┐ │ └┘ │ │ │ │ │ └┘ │ │ │ │ └─┐ │ │ └
┐
echo, └──┘ └──┘ └┘ └┘ └──┘ └┘ └──┘ └┘ └
┘
   ) else (
echo, ┌──┐ ┌┐ ┌──┐ ┌┐ ┌┐ ┌──┐ ┌──┐
echo, │ ┌┐ │ │ │ │ ┌┐ │ │ └┘ │ │ ┌─┘ │ ┌┐ │
echo, │ └┘ │ │ │ │ ┌┐ │ └┐ ┌┘ │ └─┐ │ └┘ │
echo, │ ┌─┘ │ │ │ │ ┌┐ │ │ │ │ ┌─┘ │ ┌┐ │
```

```
echo, 
echo,
    )
) else (
    if "!行动方!"=="Computer" (
echo, 
echo,
    ) else (
echo, 
echo,
    )
)
echo, 
echo,
endlocal
set /p =按任意键重新开始游戏！ <nul
pause>nul
goto :start
goto :eof
```

```
:显示棋盘
echo,　　┌──────────────────────────────────────────────
──────────┐
for /l %%i in (15 -1 2) do (
　　if %%i lss 10 (
　　　echo,　 %%i　 │ !Display_%%i! │
　　　set "Display_%%i=!Display_%%i:★=%qizi1%!"
　　) else (
　　　echo, %%i　 │ !Display_%%i! │
　　　set "Display_%%i=!Display_%%i:★=%qizi1%!"
　　)
　　echo,　　 │ │　 │　 │　 │　 │　 │　 │　 │　 │　 │　 │
 │　 │　 │　 │ │
)
echo,　 1　 │ !Display_1! │
echo,　　└- A　 B　 C　 D　 E　 F　 G　 H　 I　 J　 K　 L
 M　 N　 O -┘
goto :eof


:设置坐标 [x] [y] [1,2]
set /a QP[%1][%2]=%3,x=%1,y=%2,z=%3,cut1=x*2-2,cut2=x*2-1,最高
分=0,fens=0,步+=1,maxC=0,maxP=0
set "cr=!qizi%3!"
if "%3"=="1" set "cr=★"
set
"Display_!y!=!Display_%y%:~0,%cut1%!!cr!!Display_%y%:~%cut2%!"
set "第!步!步=%1,%2"

::判断禁手，“三四” “成五” 不算禁手，"弍""叁"看似不是活三，
两个一起构不成活四,但是跟别的活三,四搭配仍然是活三。
set "禁="
set "j="
if "!行动方:~0,1!"=="!先手!" (
　　for %%i in (!%先手%_str[%1][%2]!) do (
　　　for %%D in (!%先手%_禁手!) do (
　　　　for /f "tokens=1,2 delims=_" %%E in ("%%D") do (
　　　　　set "sstr=%%i"
　　　　　if not "!sstr!"=="!sstr:%%E=%%F!" ( set "j=%%F" )
　　　　)
```

```
        )
        set  禁=!禁!!j!
        set "j="
    )
)
if "!禁!"=="三四" set "禁="
if "!禁!"=="四三" set "禁="
if not "!禁!"=="!禁:五=!" if "!禁!"=="!禁:长连=!" set "禁="
if "!禁:~1,1!"=="" set "禁="
if defined  禁  exit /b 2

::判断是否 5 连，断定胜负
if !步! GTR 1 if "!行动方!"=="Computer" (set /a
fens=!C_soce[%1][%2]!) else (set /a fens=!P_soce[%1][%2]!)
if !fens! GEQ 9999999 exit /b 1
set "C_str[%1][%2]="
set "C_soce[%1][%2]="
set "P_soce[%1][%2]="
set "P_str[%1][%2]="

::落子点估值
for /l %%i in (1 1 4) do (
            for %%j in (%%i_0 %%i_-%%i 0_-%%i -%%i_-%%i
-%%i_+0 -%%i_+%%i 0_%%i %%i_%%i) do (
                        for /f "tokens=1,2 delims=_" %%x in ("%%j") do (
                                    set /a xx=x+%%x,yy=y+%%y
                                    for /f "tokens=1,2" %%a in ("!xx! !yy!") do
(
                                                if defined QP[%%a][%%b]
if !QP[%%a][%%b]! EQU 0 (


rem  更新双方字串并评分，没有的则重新创建，只更新  x-4 => x+4  范
围且在一条直线上的评分。
set /a a=%%a,b=%%b,c=1
if "!C_str[%%a][%%b]!"=="" (
            set /a
c_str1=c,c_str2=c,c_str3=c,c_str4=c,p_str1=2,p_str2=2,p_str3=2,p_str4=
2
            for /l %%I in (-1 -1 -4) do (
                        set /a xxx=a+%%I,yyy=b+%%I,lj1=0
```

```
                    if !xxx! LSS 1 (
                            set /a lj1=1
                            set "c_str1=x!c_str1!"
                            set "p_str1=x!p_str1!"
                    ) else (
                            for /f "tokens=1,2" %%A in ("!xxx! !b!")
do (
                                    set
"c_str1=!QP[%%A][%%B]!!c_str1!"
                                    set
"p_str1=!QP[%%A][%%B]!!p_str1!"
                            )
                    )
                    if !yyy! LSS 1 (
                            set /a lj1=1
                            set "c_str2=x!c_str2!"
                            set "p_str2=x!p_str2!"
                    ) else (
                            for /f "tokens=1,2" %%A in ("!a! !yyy!") do
(
                                    set
"c_str2=!QP[%%A][%%B]!!c_str2!"
                                    set
"p_str2=!QP[%%A][%%B]!!p_str2!"
                            )
                    )
                    if !lj1! EQU 1 (
                            set "c_str3=x!c_str3!"
                            set "p_str3=x!p_str3!"
                    ) else (
                            for /f "tokens=1,2" %%A in ("!xxx! !yyy!")
do (
                                    set
"c_str3=!QP[%%A][%%B]!!c_str3!"
                                    set
"p_str3=!QP[%%A][%%B]!!p_str3!"
                            )
                    )
                    set /a xxx=a+%%I,yyy=b-%%I,lj1=0
                    if !xxx! LSS 1 set /a lj1=1
                    if !yyy! GTR 15 set /a lj1=1
                    if !lj1! EQU 1 (
                            set "c_str4=x!c_str4!"
```

```
                        set "p_str4=x!p_str4!"
                ) else (
                        for /f "tokens=1,2" %%A in ("!xxx! !yyy!")
do (
                                set
"c_str4=!QP[%%A][%%B]!!c_str4!"
                                set
"p_str4=!QP[%%A][%%B]!!p_str4!"
                        )
                )

        )
        for /l %%I in (1 1 4) do (
                set /a xxx=a+%%I,yyy=b+%%I,lj1=0
                if !xxx! GTR 15 (
                        set /a lj1=1
                        set "c_str1=!c_str1!x"
                        set "p_str1=!p_str1!x"
                ) else (
                        for /f "tokens=1,2" %%A in ("!xxx! !b!")
do (
                                set
"c_str1=!c_str1!!QP[%%A][%%B]!"
                                set
"p_str1=!p_str1!!QP[%%A][%%B]!"
                        )
                )
                if !yyy! GTR 15 (
                        set /a lj1=1
                        set "c_str2=!c_str2!x"
                        set "p_str2=!p_str2!x"
                ) else (
                        for /f "tokens=1,2" %%A in ("!a! !yyy!") do
(
                                set
"c_str2=!c_str2!!QP[%%A][%%B]!"
                                set
"p_str2=!p_str2!!QP[%%A][%%B]!"
                        )
                )
                if !lj1! EQU 1 (
                        set "c_str3=!c_str3!x"
                        set "p_str3=!p_str3!x"
```

```
                    ) else (
                        for /f "tokens=1,2" %%A in ("!xxx! !yyy!")
do (
                            set
"c_str3=!c_str3!!QP[%%A][%%B]!"
                            set
"p_str3=!p_str3!!QP[%%A][%%B]!"
                        )
                    )
                    set /a xxx=a+%%I,yyy=b-%%I,lj1=0
                    if !xxx! GTR 15 set /a lj1=1
                    if !yyy! LSS 1 set /a lj1=1
                    if !lj1! EQU 1 (
                        set "c_str4=!c_str4!x"
                        set "p_str4=!p_str4!x"
                    ) else (
                        for /f "tokens=1,2" %%A in ("!xxx! !yyy!")
do (
                            set
"c_str4=!c_str4!!QP[%%A][%%B]!"
                            set
"p_str4=!p_str4!!QP[%%A][%%B]!"
                        )
                    )
                )
                set /a soce1=0,soce2=0,soce3=0,soce4=0
                for %%X in (!tzk!) do (
                    for /f "tokens=1,2 delims=_" %%A in ("%%X") do (
                        for /l %%W in (1 1 4) do (
                            if not
"!c_str%%W!"=="!c_str%%W:%%A=@!" if !soce%%W! LSS %%B (
                                set /a soce%%W=%%B
                                if !maxP! LSS %%B set /a
maxP=%%B
                            )
                        )
                    )
                )
                set "C_str[!a!][!b!]=!c_str1!,!c_str2!,!c_str3!,!c_str4!"
                set "C_soce[!a!][!b!]=!soce1!+!soce2!+!soce3!+!soce4!"

                set /a soce1=0,soce2=0,soce3=0,soce4=0
                for %%X in (!p_tzk!) do (
```

```
                    for /f "tokens=1,2 delims=_" %%A in ("%%X") do (
                        for /l %%W in (1 1 4) do (
                            if not
"!p_str%%W!"=="!p_str%%W:%%A=@!" if !soce%%W! LSS %%B (
                                set /a soce%%W=%%B
                                if !maxP! LSS %%B set /a
maxP=%%B
                            )
                        )
                    )
            )
            set "P_str[!a!][!b!]=!p_str1!,!p_str2!,!p_str3!,!p_str4!"
            set "P_soce[!a!][!b!]=!soce1!+!soce2!+!soce3!+!soce4!"

rem end 创建。
) else (
rem 更新字串及评分，已存在的点会加快效率。
            if %%x NEQ 0 (
                    set /a
cut1=5-%%x-1,cut2=5-%%x,tk=1,pd=%%x+%%y
                    if !pd! EQU 0 (
                        set /a cut1=cut1+30,cut2=cut2+30,tk=4
                    )
                    if %%x EQU %%y (
                        set /a cut1=cut1+20,cut2=cut2+20,tk=3
                    )
            ) else (
                    set /a cut1=15-%%y-1,cut2=15-%%y,tk=2
            )
            for /f "tokens=1,2" %%X in ("!cut1! !cut2!") do (
                    set
"C_str[%%a][%%b]=!C_str[%%a][%%b]:~0,%%X!!z!!C_str[%%a][%%
b]:~%%Y!"
                    set
"P_str[%%a][%%b]=!P_str[%%a][%%b]:~0,%%X!!z!!P_str[%%a][%%
b]:~%%Y!"
            )
            for /f "tokens=1-4 delims=+" %%A in
("!C_soce[%%a][%%b]!") do (
                    set /a
soce1=%%A,soce2=%%B,soce3=%%C,soce4=%%D
            )
            set /a soce=0,ttk=tk*10-10
```

```
		for %%T in (!ttk!) do (
			set "t_str=!C_str[%%a][%%b]:~%%T,9!"
		)
		for %%Q in (!tzk!) do (
			for /f "tokens=1,2 delims=_" %%A in ("%%Q") do (
				if not "!t_str!"=="!t_str:%%A=@!" (
					if !soce! LSS %%B set /a
soce=%%B
				)
			)
		)
		set /a soce!tk!=!soce!
		if !maxC! LSS !soce! set /a maxC=soce
		set "C_soce[%%a][%%b]=!soce1!+!soce2!+!soce3!+!soce4!"


rem  更新 player 评分
		for /f "tokens=1-4 delims=+" %%A in
("!P_soce[%%a][%%b]!") do (
			set /a
soce1=%%A,soce2=%%B,soce3=%%C,soce4=%%D
		)
		set /a soce=0
		for %%T in (!ttk!) do (
			set "t_str=!P_str[%%a][%%b]:~%%T,9!"
		)
		for %%Q in (!p_tzk!) do (
			for /f "tokens=1,2 delims=_" %%A in ("%%Q") do (
				if not "!t_str!"=="!t_str:%%A=@!" (
					if !soce! LSS %%B set /a
soce=%%B
				)
			)
		)
		set /a soce!tk!=!soce!
		if !maxP! LSS !soce! set /a maxP=soce
		set "P_soce[%%a][%%b]=!soce1!+!soce2!+!soce3!+!soce4!"

)
rem end 更新
```

```
                                        )
                                )
                        )
                )
)
if !maxC! geq !maxP! ( set /a k1=10,k2=7 ) else ( set /a k1=7,k2=10 )
exit /b 0



:电脑智能
::判断定式
::if !匹配! equ 1 (



:电脑走
set "行动方=Computer"
set /a  最高分=0,num=0
for /l %%x in (1 1 15)    do (
        for /l %%y in (1 1 15) do (
                if defined C_soce[%%x][%%y] (
                        set "m=!C_soce[%%x][%%y]!"
                        set "n=!P_soce[%%x][%%y]!"
                        set /a m=!m!,n=!n!,mn=m*!k1!+n*!k2!
                        if !最高分! EQU !mn! (
                           set /a num+=1
                           set "坐标集=!坐标集! [%%x][%%y]"
                         )
                        if !最高分! LSS !mn! (
                                set /a  最高分=!mn!,num=1
                                set "坐标集=[%%x][%%y]"
                        )
                )
        )
)
set /a get=%RANDOM%%%num+1,dd=0
```

```
for %%i in (!坐标集!) do (
		set /a dd+=1
		if !dd! equ !num! (
			for /f "tokens=1,2 delims=[]" %%x in ("%%i") do (
				call :设置坐标  %%x %%y 1
			)
		 )
)
goto :eof
```

:::下面是定式图谱及算法
x 代表第二手棋子位置，然后 x 周围 2*2 的位置可以用[方向,距离]来表示位置。
而 第二手棋子位置可以有很多，而且可能对称，那么[方向增值,距离]可以表示相对位置。
比如这 3 步棋 [8,8] [8,9] [8,10] ，从第 3 步起，可以用[0,1]来表示相对第 2 步的位置变化。
加入第二步下在[8,9]，方向是 4,那么可以得出第三步 方向=(4+0)%16，代进 set "方位_0=ly+=jl"，可以得到第 3 步位置。
得分 100,表示黑必胜,50 表示黑优势,-100 表示必败,0 表示均势。

位置值表示图
14 15 0 1 2
13 14 0 2 3
12 12 x 4 4
11 10 8 6 5
10　9 8 7 6

```
set "方位_0=ly+=jl"
set "方位_1=lx+=1,ly+=2"
set "方位_2=lx+=jl,ly+=jl"
set "方位_3=lx+=2,ly+=1"
set "方位_4=lx=lx+jl"
set "方位_5=lx+=2,ly-=1"
set "方位_6=lx=lx+jl,ly=ly-jl"
```

```
set "方位_7=lx+=1,ly-=2
set "方位_8=ly=ly-jl"
set "方位_9=lx-=1,ly-=2"
set "方位_10=lx=lx-jl,ly=ly-jl"
set "方位_11=lx-=2,ly-=1"
set "方位_12=lx=lx-jl"
set "方位_13=lx-=2,ly+=1
set "方位_14=lx=lx-jl,ly=ly+jl"
set "方位_15=lx-=1,ly+=2"

::定式图谱  [方向增值,距离] for /f "delims=" %%i in ('set [直][!第二步!][!第三步!]..')
::1,直指开局
[直][0,1][2,1][10,1]=100
[直][0,1][3,2][11,2]=100
[直][0,1][0,2][4,1]=100
[直][0,1][4,2][4,1]=100
[直][3,2][4,1][6,1]=0
[直][3,2][6,1][4,1]=100
[直][3,2][9,2][4,1]=100
[直][3,2][10,2][4,1]=100
[直][3,2][10,2][4,2]=0
[直][4,1][6,1][7,2]=150
[直][4,1][6,2][6,1]=150
[直][4,2][5,2][7,1]=100
[直][4,2][5,2][9,2]=50
[直][4,2][10,1][2,1]=100
[直][4,2][10,1][6,1]=50
[直][4,2][10,1][3,1]=0
[直][4,2][9,1][6,1]=100
[直][4,2][3,1][6,1]=100
[直][5,2][10,1][2,1]=100
[直][5,2][10,1][4,1]=50
[直][5,2][2,1][7,2]=100
[直][5,2][2,1][4,1]=50
[直][5,2][9,1][6,1]=100
[直][5,2][4,1][6,1]=100
[直][8,2][8,3][6,1]=100
```

[直][8,2][8,3][4,1]=50
[直][8,2][5,2][10,1]=50
[直][8,2][0,1][6,1]=100
[直][8,2][2,1][10,1]=100
[直][6,2][10,1][2,1]=100
[直][6,2][10,1][10,2]=50
[直][6,2][9,1][12,1]=50
[直][6,2][2,1][2,2]=100
[直][6,2][2,1][7,2]=50
[直][8,3][2,1][7,2]=100
[直][8,3][2,1][6,1]=-50
[直][8,3][2,1][11,2]=-50
[直][8,3][2,1][5,2]=-50
[直][8,3][2,1][10,1]=-100
[直][8,3][2,1][12,1]=-100
[直][8,3][7,2][9,2]=100
[直][8,3][2,1][6,2]=100
[直][8,3][8,2][7,2]=100
[直][8,3][6,2][9,2]=100
[直][8,3][6,2][6,1]=100

::斜止图谱
::太长了,累死我了,以后慢慢补上。

五子棋游戏三

```
@echo off
title  五子棋  作者：邵鸿轩
setlocal enabledelayedexpansion
mode con cols=40 lines=5
color E0
echo.
echo  欢迎访问我的博客：
echo    http://zhiqianqiutian.blog.163.com
set /p slt1=        您确定要挑战吗？（y/n）
cls
if /i not "%slt1%"=="y" goto def
:whofirst
```

```
echo.


echo                    1、挑战者先手
echo                    2、英雄家族先手
set /p slt2=
if not "%slt2%" equ "1" if not "%slt2%" equ "2" cls&echo
● 请正确输入！！ &pause&cls&goto whofirst
cls
mode con cols=45 lines=10
:people
echo                          请选择人物
echo.
echo                  级别越高思考时间越长
echo.
echo 1、宠物                    棋力--9 级以下
echo 2、宝宝                    棋力--9~7 级
echo 3、猛虎                    棋力--6~4 级
echo 4、飞龙                    棋力--3~1 级
echo 5、英雄                    棋力--9 段
set /p slt3=
cls
if %slt3%==5 echo ● 对不起，春哥喊我回家吃面条，请谅解~~。
&pause&cls&goto people
if not "%slt3%" equ "1" if not "%slt3%" equ "2" if not "%slt3%" equ "3"
if not "%slt3%" equ "4" (
cls
echo                ● 请输入正确的人物代号！
pause
cls
goto people
)


::::::::::::::::::::::::::::::::::::::::::::::::
::以下定义一些变量，包括图形界面中的元素
::::::::::::::::::::::::::::::::::::::::::::::::
```

```
:def


set  纵轴=B C D E F G H I J K L M N


for %%i in (a %纵轴% o) do (
  set /a bridge+=1
  set ctoi%%i=!bridge!


)


set bridge=


for %%i in (a %纵轴% o) do (
  set /a bridge+=1
  set itoc!bridge!=%%i


)


:start









for %%i in (a %纵轴% o) do (
       for /l %%j in (1,1,15) do (
  set my%%i%%j=*
       )


)


set A1= ┏
```

```
set A15=┐
set O1= └
set O15=┘


for /l %%i in (2,1,14) do set a%%i=┬
for /l %%i in (2,1,14) do set o%%i=┴
for %%i in (%纵轴%) do set %%i1= ├&set %%i15=┤
for %%i in (%纵轴%) do (
       for /l %%j in (2,1,14) do (
  set %%i%%j=┼
       )
)


set round=1
set preinput=
mode con cols=61 lines=40
::::::::::::::::::::::::::::::::::::::::::::::
::以下为交替落子模块
::::::::::::::::::::::::::::::::::::::::::::::
call:graph


:play


:blackinput
if %round% geq 100 goto nowin
if /i "%slt1%"=="y" if "%slt2%"=="2" call:ai ● b&goto whiteinput
set /p input=第%round%手，请黑方输入
if /i "%input%"=="restart" goto start
if "!my%input%!"=="*" (
set %input%=●
set my%input%=b
set /a round+=1
) else (
echo 输入有误，请重新输入!
goto blackinput)
call:graph
call:judge %input%
```

```
:whiteinput
if %round% geq 100 goto nowin
if /i "%slt1%"=="y" if "%slt2%"=="1" call:ai ○ w&goto play
set /p input=第%round%手，请白方输入
if /i "%input%"=="restart" goto start
if "!my%input%!"=="*" (
set %input%=○
set my%input%=w
set /a round+=1
) else (
echo 输入有误，请重新输入！
goto whiteinput)
call:graph
call:judge %input%


goto play


::::::::::::::::::::::::::::::::::::::::::::::
::以下为判断胜负模块
::::::::::::::::::::::::::::::::::::::::::::::
:judge
set tmp=%1
set var1=!tmp:~0,1!
set var2=!tmp:~1!
set /a flag1=ctoi%var1%-1+var2
set /a flag2=var2-ctoi%var1%+1
rem 下面定义四个判断要用到的指标，分别代表横纵斜四个方向的落子情况
set judgeheng=
set judgezong=
set judgeyszx=
set judgezsyx=
rem 下面定义上述变量的具体值


rem 横向
      for /l %%j in (1,1,15) do (
 set judgeheng=!judgeheng!!my%var1%%%j!
```

```
	)
rem  纵向
	for %%j in (a %纵轴% o) do (
  set judgezong=!judgezong!!my%%j%var2%!
	)
rem  从右上到左下
	for %%i in (a %纵轴% o) do (
		call set judgeyszx=!judgeyszx!%%my%%i!flag1!%%
		set /a flag1-=1
	)
rem  从左上到右下
	for %%i in (a %纵轴% o) do (
		call set judgezsyx=!judgezsyx!%%my%%i!flag2!%%
		set /a flag2+=1
	)
rem  将上述四个指标串起来并进行判断
set judge=!judgeheng!*!judgezong!*!judgeyszx!*!judgezsyx!


	(echo !judge!|find "bbbbb">nul)&&goto blackwin
	(echo !judge!|find "wwwww">nul)&&goto whitewin
set score=
goto :eof
::::::::::::::::::::::::::::::::::::::::::::::::
::以下胜负已分
::::::::::::::::::::::::::::::::::::::::::::::::
:blackwin
set %input%=★
call :graph
echo ● 黑第%round%手胜！
pause
goto start


:whitewin
set %input%=☆
call :graph
echo ● 白第%round%手胜！
pause
goto start
```

```
:nowin
echo ● 百手和棋!
pause
goto start
::::::::::::::::::::::::::::::::::::::::::::
::以下为绘图过程
::::::::::::::::::::::::::::::::::::::::::::
:graph
cls
echo                                  输入 RESTART 重新开局
set /p =    <nul
for /l %%i in (1,1,8) do set /p=%%i      <nul
for /l %%i in (9,1,15) do set /p=%%i    <nul
::echo.
for %%i in (A %纵轴%) do (
        set /p =%%i<nul
        for /l %%j in (1,1,14) do (
  set /p =!%%i%%j!━<nul
        )
        echo !%%i15!
        for /l %%j in (1,1,15) do (
  set /p =  ┃  <nul
        )
        echo.
)
set /p =O<nul
for /l %%i in (1,1,14) do set /p =!O%%i!━<nul
echo !O15!
echo  输入时先字母后数字。如 H8。
goto :eof
::::::::::::::::::::::::::::::::::::::::::::
::以下是电脑下棋思路
::::::::::::::::::::::::::::::::::::::::::::
:ai
if %round% equ 1 set input=h8&goto ainext
set aitmp=%input%
set aivar1=%aitmp:~0,1%
set aivar2=%aitmp:~1%


:loop
```

```
set /a aiflag1=ctoi%aivar1%+%random%%%3-%random%%%3
set    aiflag2=!itoc%aiflag1%!


set /a aivar=aivar2+%random%%%3-%random%%%3


if "!my%aiflag2%%aivar%!"=="*" (
set input=%aiflag2%%aivar%
)else goto loop


set /a aic1=ctoi%aivar1%+1
set /a aic2=ctoi%aivar1%-1
set aic1=!itoc%aic1%!
set aic2=!itoc%aic2%!
set /a aii1=aivar2+1
set /a aii2=aivar2-1
set aistr=bw
set air=!aistr:%2=!


call :analyse %aivar1%%aii1% %air%
call :analyse %aivar1%%aii2% %air%
call :analyse %aic1%%aivar2% %air%
call :analyse %aic2%%aivar2%    %air%
call :analyse %aic1%%aii1% %air%
call :analyse %aic1%%aii2% %air%
call :analyse %aic2%%aii1% %air%
call :analyse %aic2%%aii2% %air%


if not defined preinput goto ainext
set aitmp=%preinput%
set aivar1=!aitmp:~0,1!
set aivar2=!aitmp:~1!
set /a aic1=ctoi%aivar1%+1
set /a aic2=ctoi%aivar1%-1
set aic1=!itoc%aic1%!
set aic2=!itoc%aic2%!
set /a aii1=aivar2+1
set /a aii2=aivar2-1
call :analyse %aivar1%%aii1% %2
```

```
call :analyse %aivar1%%aii2% %2
call :analyse %aic1%%aivar2% %2
call :analyse %aic2%%aivar2% %2
call :analyse %aic1%%aii1% %2
call :analyse %aic1%%aii2% %2
call :analyse %aic2%%aii1% %2
call :analyse %aic2%%aii2% %2


:ainext
set %input%=%1
set my%input%=%2
set /a round+=1
set preinput=%input%
call:graph
call:judge %input%
goto :eof
::::::::::::::::::::::::::::::::::::::::::::
::分析评价模块
::::::::::::::::::::::::::::::::::::::::::::
:analyse
if not "!my%1!"=="*" goto :eof
set Atmp=%1
set Avar1=!Atmp:~0,1!
set Avar2=!Atmp:~1!
set /a Aflag1=ctoi%Avar1%-1+Avar2
set /a Aflag2=Avar2-ctoi%Avar1%+1


set judgeheng=
set judgezong=
set judgeyszx=
set judgezsyx=




      for /l %%j in (1,1,15) do (
 set judgeheng=!judgeheng!!my%Avar1%%%j!
      )


      for %%j in (a %纵轴% o) do (
```

```
  set judgezong=!judgezong!!my%%j%Avar2%!
    )


    for %%i in (a %纵轴% o) do (
        call set judgeyszx=!judgeyszx!%%my%%i!Aflag1!%%
        set /a Aflag1-=1
    )


    for %%i in (a %纵轴% o) do (
        call set judgezsyx=!judgezsyx!%%my%%i!Aflag2!%%
        set /a Aflag2+=1
    )


set judge=!judgeheng!*!judgezong!*!judgeyszx!*!judgezsyx!


set Astr=%2


    (echo !judge!|find "%Astr%">nul)&&(if "%score%" leq "1" set
score=1&set input=%1)
    if "%slt3%" geq "2" (echo !judge!|find
"%Astr%%Astr%">nul)&&(if "%score%" leq "2" set score=2&set
input=%1)
    if "%slt3%" geq "3" (echo !judge!|find
"%Astr%%Astr%%Astr%">nul)&&(if "%score%" leq "3" set
score=3&set input=%1)
    if "%slt3%" equ "4" (echo !judge!|find
"%Astr%%Astr%%Astr%%Astr%">nul)&&(if "%score%" leq "4" set
score=4&set input=%1)


goto :eof
```

人鬼过河

```
@echo off
title  人鬼过河
color e9
echo r 代表人，g 代表鬼。
echo  船只可以装两个人/鬼，或一人一鬼。
echo  任何一边鬼都不能比人多，
echo  否则人会被吃掉哦。
echo S 重来，E 退出。
echo.
:ready
set fx=A→B
set sidea=rrrggg
set sideb=
:start
echo =======now========
echo  船  A:%sidea%
echo ------------------
echo.
echo  河
echo.
echo  船:%fx%
echo.
echo ------------------
echo  船  B:%sideb%
echo.
:input
set /p boat=请输入：
if /i "%boat%"=="s" goto ready
if /i "%boat%"=="e" exit
if "%boat%"=="" echo  没人开船了。&&goto input
if not "%boat:~2,1%"=="" echo  不能装那么多。&&goto input
set bt1=%boat:~0,1%
set bt2=%boat:~1,1%
if /i not %bt1%==r if /i not %bt1%==g echo  你输错了。&&goto input
if /i not "%bt2%"=="r" if /i not "%bt2%"=="g" if not "%bt2%"=="" echo
你输错了。&&goto input
goto %fx%
:A→B
```

```
echo %sidea%|find /i "%bt1%"&gt;nul||echo Side A 没有%bt1%。
&&goto input
if not "%bt2%"=="" echo %sidea%|find /i "%bt2%"&gt;nul||echo Side A
没有%bt2%。&&goto input
if "%bt1%"=="%bt2%" echo %sidea%|find /i "%boat%"&gt;nul||echo
Side A 没有两个%bt1%。&&goto input
if /i %bt1%==r set sidea=%sidea:~1%&& set sideb=r%sideb%
if /i %bt1%==g set sidea=%sidea:~0,-1%&& set sideb=%sideb%g
if /i "%bt2%"=="r" set sidea=%sidea:~1%&& set sideb=r%sideb%
if /i "%bt2%"=="g" set sidea=%sidea:~0,-1%&& set sideb=%sideb%g
call :check
set fx=B→A
cls
goto start
:B→A
echo %sideb%aaa|find /i "%bt1%"&gt;nul||echo Side B 没有%bt1%。
&&goto input
if not "%bt2%"=="" echo %sideb%|find /i "%bt2%"&gt;nul||echo Side B
没有%bt2%。&&goto input
if "%bt1%"=="%bt2%" echo %sideb%|find /i "%boat%"&gt;nul||echo
Side B 没有两个%bt1%。&&goto input
if /i %bt1%==r set sideb=%sideb:~1%&& set sidea=r%sidea%
if /i %bt1%==g set sideb=%sideb:~0,-1%&& set sidea=%sidea%g
if /i "%bt2%"=="r" set sideb=%sideb:~1%&& set sidea=r%sidea%
if /i "%bt2%"=="g" set sideb=%sideb:~0,-1%&& set sidea=%sidea%g
call :check
set fx=A→B
cls
goto start
:check
if "%sideb%"=="rrrggg" cls&&echo You win!&&goto end
set ars=0
set ags=0
set brs=0
set bgs=0
if "%sidea:~0,1%"=="r" set /a ars+=1 &gt;nul
if "%sidea:~0,1%"=="g" set /a ags+=1 &gt;nul
if "%sidea:~1,1%"=="r" set /a ars+=1 &gt;nul
if "%sidea:~1,1%"=="g" set /a ags+=1 &gt;nul
if "%sidea:~2,1%"=="r" set /a ars+=1 &gt;nul
if "%sidea:~2,1%"=="g" set /a ags+=1 &gt;nul
if "%sidea:~3,1%"=="r" set /a ars+=1 &gt;nul
```

```
if "%sidea:~3,1%"=="g" set /a ags+=1 &gt;nul
if "%sidea:~4,1%"=="r" set /a ars+=1 &gt;nul
if "%sidea:~4,1%"=="g" set /a ags+=1 &gt;nul
if "%sidea:~5,1%"=="r" set /a ars+=1 &gt;nul
if "%sidea:~5,1%"=="g" set /a ags+=1 &gt;nul
if %ags% gtr %ars% if %ars% gtr 0 (
cls
echo Side A 有人被吃掉了。
goto end
)
if "%sideb:~0,1%"=="r" set /a brs+=1 &gt;nul
if "%sideb:~0,1%"=="g" set /a bgs+=1 &gt;nul
if "%sideb:~1,1%"=="r" set /a brs+=1 &gt;nul
if "%sideb:~1,1%"=="g" set /a bgs+=1 &gt;nul
if "%sideb:~2,1%"=="r" set /a brs+=1 &gt;nul
if "%sideb:~2,1%"=="g" set /a bgs+=1 &gt;nul
if "%sideb:~3,1%"=="r" set /a brs+=1 &gt;nul
if "%sideb:~3,1%"=="g" set /a bgs+=1 &gt;nul
if "%sideb:~4,1%"=="r" set /a brs+=1 &gt;nul
if "%sideb:~4,1%"=="g" set /a bgs+=1 &gt;nul
if %bgs% gtr %brs% if %brs% gtr 0 (
cls
echo Side B 有人被吃掉了。
goto end
)
goto :EOF
:end
set /p restart=再来？(Y,N)
if /i "%restart%"=="y" cls&&goto ready
```

篮球飞人

```
@echo off&setlocal enabledelayedexpansion
mode con cols=71 lines=9
for /l %%a in (1,1,8) do (
echo/
echo\
echo.
echo
**********************************************************************
*
echo **************************篮球飞人火爆版
*******************************
echo
**********************************************************************
*
color cf
ping -n 1 127.1>nul
color fc
ping -n 1 127.1>nul
cls
)
color f1
cls
echo
^|
echo
^|
echo
＝^|--^|
echo
 V'    ^|
echo
^|
echo                          o
^|
echo                        /^|\o
^|
echo
_____________/_\_____________________________________________________^|
ping -n 1 127.1>nul
cls
echo
^|
echo
^|
echo
＝^|--^|
```

```
echo
 V'   ^|
echo
^|
echo                          o
^|
echo                         /^|\
^|
echo
_________________/_^>_o_____________________________________________
^|
ping -n 1 127.1>nul
cls
echo
^|
echo
^|
echo
=^|--^|
echo
 V'   ^|
echo
^|
echo                              o
^|
echo                             /^|\o
^|
echo
_____________________^>_\_____________________________________________
^|
ping -n 1 127.1>nul
cls
echo
^|
echo
^|
echo
=^|--^|
echo
 V'   ^|
echo
^|
echo                              o
^|
echo                             /^|\
^|
echo
_____________________/_^>_o_____________________________________________
^|
```

```
ping -n 1 127.1>nul
cls
echo
^|
echo
^|
echo
＝^|--^|
echo
 V'   ^|
echo
^|
echo                                    o
^|
echo                                  /^|-o
^|
echo
_____________________________^>\_____________________________________
^|
ping -n 1 127.1>nul
cls
echo
^|
echo
^|
echo
＝^|--^|
echo
 V'   ^|
echo
^|
echo                                     o
^|
echo                                   /^|\
^|
echo
_____________________________/_^>_o_____________________________________
^|
ping -n 1 127.1>nul
cls
echo
^|
echo
^|
echo                                罚球线到了!
＝^|--^|
echo
 V'   ^|
echo
```

```
^|
echo                                    o
^|
echo                                   /^|-o
^|
echo
_____________________________________^>^>_____________________________________
__^|
ping -n 2 127.1>nul
cls
echo
^|
echo
^|
echo
=^|--^|
echo                                      o
V'   ^|
echo                                    o/
^|
echo                                   /^|
^|
echo                                    ^>\
^|
echo
_____________________________________________________________________________^
|
ping -n 1 127.1>nul
cls
echo
^|
echo
^|
echo                                       o
=^|--^|
echo                                      o^|
V'   ^|
echo                                     /^|
^|
echo                                     / ^>
^|
echo
^|
echo
_____________________________________________________________________________^
|
ping -n 1 127.1>nul
cls
echo
^|
```

```
echo                                                    o
^|
echo                                                   o^|
=^|--^|
echo                                                  /^|
V'   ^|
echo                                                   ^>\
^|
echo
^|
echo
^|
echo
_____________________________________________________________________________^
|
ping -n 1 127.1>nul
cls
echo                                                      o
^|
echo                                                    o/
^|
echo                                                   /^|
=^|--^|
echo                                                   / ^>
V'   ^|
echo
^|
echo
^|
echo
^|
echo
_____________________________________________________________________________^
|
ping -n 1 127.1>nul
cls
echo                                                        o
^|
echo                                                      o/
^|
echo                                                     /^|
=^|--^|
echo                                                     / \       V
'   ^|
echo
^|
echo
^|
echo
```

```
^|
echo
_____________________________________________________________________________^
|
ping -n 1 127.1>nul
cls
echo
^|
echo                                                                        o
^|
echo                                                                       o/
=^|--^|
echo                                                                    /^|
V'   ^|
echo                                                                    //
^|
echo
^|
echo
^|
echo
_____________________________________________________________________________^
|
ping -n 1 127.1>nul
cls
echo
^|
echo
^|
echo
=^|--^|
echo
φ'   ^|
echo
o    ^|
echo
-^|\   ^|
echo
^>^>   ^|
echo
_____________________________________________________________________________^
|
ping -n 1 127.1>nul
cls
echo
^|
echo
^|
echo
=^|--^|
```

```
echo
 V'   ^|
echo                                                                    o
^|
echo
o ^|
echo
/^|\^|
echo
_____________________________________________________________________^>
^>^|
ping -n 1 127.1>nul
cls
echo
^|
echo
^|
echo
＝^|--^|
echo
 V'   ^|
echo
^|
echo                                                                   o
o    ^|
echo
/^|- ^|
echo
_____________________________________________________________________/_\_^|
ping -n 1 127.1>nul
cls
echo
^|
echo
^|
echo
＝^|--^|
echo
 V'   ^|
echo                                                          帅气吧？呵呵~~~
^|
echo
o     ^|
echo                                                                     o
/^|\    ^|
echo
_____________________________________________________________________/_\__^|
cls
for /l %%a in (1,1,8) do (
```

```
echo/
echo\
echo.
echo
**************************************************************************
*
echo **************************GAME
OVER**********************************
echo
**************************************************************************
*
color cf
ping -n 1 127.1>nul
color fc
ping -n 1 127.1>nul
cls
)
pause>nul
```

数独

```
@echo off
Setlocal Enabledelayedexpansion
color 3e&title 数字九宫格游戏
echo.&echo 欢迎使用本绿色免安装版九宫格数字游戏, 在进入游戏前请详细阅读此说明,
echo.&echo 进入游戏会出现以下的画面,第一行和第一列分别为横坐标和纵坐标,仅作参考,
echo.&echo 游戏规则为: 用数字替换各个符号, 使每行和每列都出现 1~9 这 9 个数字,
echo.&echo 并且 9 个 3^*3 的区域里也要包含 1~9 这 9 个数字.(左图^=^=^&gt;右图)
echo.&echo 玩法: 比如下图中要把 4 填在第 3 行第 1 列的位置, 输入 431 然后回车即可.
echo.&echo 请按任意键进入游戏, 祝你玩得愉快..... ^^_^^
echo.&echo 1 2 3 4 5 6 7 8 9
echo.
echo. 1 6 8 7 _ / 1 - + 2 6 8 7 4 9 1 5 3 2
echo. 2 9 3 - 5 8 _ / + 7 9 3 1 5 8 2 4 6 7
echo. 3 / 5 _ 3 + 7 1 8 - 4 5 2 3 6 7 1 8 9
echo. 4 / 6 4 - 1 + _ 9 8 5 6 4 2 1 3 7 9 8
```

```
echo. 5 2 9 8 7 5 + / _ - 2 9 8 7 5 6 3 1 4
echo. 6 7 - _ 9 4 + 2 5 / 7 1 3 9 4 8 2 5 6
echo. 7 _ 4 9 6 + 5 / - 1 3 4 9 6 7 5 8 2 1
echo. 8 - _ 5 8 / + 6 4 3 1 7 5 8 2 9 6 4 3
echo. 9 _ / 6 - 3 4 9 + 5 8 2 6 1 3 4 9 7 5
pause&gt;nul&cls
set "cod= +-_/\#"&set "st= 1 2 3 4 5 6 7 8 9"&set
"color_=12345689abcde"
set "kong= "
set "echo_=好 极 了, 完 全 正 确, 恭 喜 你, 继 续 努 力 喔 "
for /l %%a in (1 1 80) do (set "tui=!tui!◘ ")
:again
color 3e&set "input=4"
for /l %%a in (1 1 9) do (
for /l %%b in (1 1 9) do (set "_%%a%%b="&set "%%a%%b=")
)
echo.&echo 1~6 为难度选择,4 为默认的难度
echo.&set /p input=请输入:
cls
if %input% gtr 6 (echo.&echo  选择错误&goto :again)
if %input% lss 1 (echo.&echo  选择错误&goto :again)
for /l %%i in (1 1 9) do echo.
echo  正在生成游戏,请稍候...
:st
for /l %%i in (1 1 9) do (set "mor%%i="&set "%%i=")
for /l %%i in (1 1 9) do call:lp %%i
for /l %%a in (1 1 9) do (set "moniuming%%a=!mor%%a!")
:lp3
set /a "num1=%random%%%9+1"
if defined %num1% goto :lp3
set "%num1%=god"
for /f "tokens=%num1%" %%g in ("!mor%line%!") do (
set /a "num2+=1","_%line%%num1%=god"
call set "%line%%num1%=%%cod:~!num2!,1%%"
call call set
"mor%line%=%%%%mor%line%:%%g=%%cod:~!num2!,1%%%%%%
"
)
if %dfc% lss %input% (set /a "dfc+=1"&goto :lp3)
for /l %%a in (1 1 9) do (set "%%a=")
set /a "num3+=1","num2=0"
if %num3% lss 9 (set /a "line+=1"&set "dfc=1"&goto :lp3)
```

```
cls&echo %st%&echo.
for /l %%a in (1 1 9) do (call echo %%a %%mor%%a%%)
:lp4
echo.&echo  请按照"替换数字,行,列"(不包括引号)的顺序连续输入数字
echo  或者 ans 显示答案(默认),new 开启新游戏,out 退出本游戏
:lp5
color 3e&echo.&set "ent=ans"&set /p ent=请输入:
if /i "%ent%"=="out" (
cls
for /l %%a in (1 1 9) do echo.
echo  谢谢使用本游戏,再见.....
ping -n 4 127.1&gt;nul&exit
)
if /i "%ent%"=="ans" (
for /l %%l in (1 1 9) do (echo !moniuming%%l!)
goto :again
)
if /i "%ent%"=="new" cls&goto :again
set "ma=%ent:~,1%"&set "mb=%ent:~1,1%"&set "mc=%ent:~2,1%"
if not defined _%mb%%mc% (
color fc&echo.&echo  第%mb%行第%mc%列为非替换位置
ping -n 3 127.1&gt;nul&goto :lp5
)
for %%a in (!mor%mb%!) do (
if "%%a"=="%ma%" (
color fc&echo.&echo  第 %mb% 行已经存在 "%ma%"
ping -n 3 127.1&gt;nul&goto :lp5
)
)
for /l %%a in (1 1 9) do (
if "%ma%"=="!%%a%mc%!" (
color fc&echo.&echo  第 %mc% 列已经存在 "%ma%"
ping -n 3 127.1&gt;nul&goto :lp5
)
)
for /f "tokens=%mc%" %%a in ("!mor%mb%!") do (
set "mor%mb%=!mor%mb%:%%a=%ma%!"&set
"%mb%%mc%=%ma%"
)
cls
echo %st%&echo.
```

```
for /l %%a in (1 1 9) do (echo %%a !mor%%a!)
for /l %%a in (1 1 9) do (if "!mor%%a!"=="!moniuming%%a!" (set /a
right+=1))
if "%right%"=="9" (
call :lp6
set "right=0"&goto :again
)
set "right=0"
goto :lp4
:lp
for /l %%i in (1 1 9) do set ".!random!!random!!random!=%%i"
for /f "tokens=1,2 delims==" %%i in ('set .') do (set "str=%%j !str!"&set
"%%i=")
for /l %%i in (1 1 9) do set "%1%%i="
set "var=%str%"&set/a
lie=1,hx=1,hy=%1-1,lx=1,ly=3,a=%1%%3,dfc=1,line=1,num2=0,num3=
0
if %1 geq 4 (
if %1 leq 6 (set/a hx=4) else (set hx=7)
)
:lp0
if %lie% geq 4 (
if %lie% leq 6 (set/a lx=4,ly=6) else (set/a lx=7,ly=9)
)
for %%a in (!var!) do (
for /l %%i in (1 1 %1) do (
if %%a equ !%%i%lie%! (set var=!var:%%a=!&goto lp0)
)
if %a% neq 1 (
for /l %%j in (%hx% 1 %hy%) do (
for /l %%k in (%lx% 1 %ly%) do (
if %%a equ !%%j%%k! set "var=!var:%%a=!"&goto lp0
)
)
)
set "%1!lie!=%%a"
if !lie! lss 9 (set/a lie+=1&set "str=!str:%%a=!"&set "var=!str!"&goto
lp0)
)
set "str="&set "lie="
if "!%19!"=="" set/a ttl+=1&if !ttl! gtr 40 (set "ttl="&goto st) else goto lp
for /l %%i in (1 1 9) do (set "mor%1=!mor%1!!%1%%i! ")
set "ttl="&goto :eof
```

```
:lp6
echo.
for /l %%a in (1 1 40) do (
set "show=!echo_:~0,%%a!"
set /a "col1=!random!%%13","col2=!random!%%13"
if not "!col1!"=="!col2!" call color
%%color_:~!col1!,1%%%%color_:~!col2!,1%%
set /p=!tui!!kong!!show!&lt;nul
ping -n 1 127.1&gt;nul
)
echo.
```

---

24 点游戏一

24 点游戏的难点在于对小数的处理。

```
@echo off
setlocal enableextensions enabledelayedexpansion
title  作者:liuzhaonan11
echo  批处理算 24 点
echo.
:loop
set /p "n=输入四个数用空格分开: "
set /a "lzn=0"

for %%i in (!n!) do (
    set /a "lzn+=1"
    set /a "lz!lzn!z=%%i"
    set /a "lz!lzn!m=1"
    set /a "lz!lzn!s=%%i"
)

call:liuzhaonan11 lz
if not errorlevel 1 echo  结果:  没有答案
goto loop

::原理说明:使用有理数进行计算,选择算式中最先结合的两数和它们之间的运算,并递归.

:liuzhaonan11
```

```
setlocal

set "f1=+"
set "f2=-"
set "f3=*"
set "f4=/"
for /l %%i in (1,1,!%1n!) do (
    for /l %%j in (1,1,!%1n!) do (
        if %%i neq %%j (
            for /l %%k in (1,1,4) do (
                setlocal
                if %%k leq 2 (
                    set /a
"z=%1%%jz*%1%%im!f%%k!%1%%jm*%1%%iz"
                    set /a "m=%1%%jm*%1%%im"
                )
                if %%k==3 (
                    set /a "z=%1%%jz*%1%%iz"
                    set /a "m=%1%%jm*%1%%im"
                )
                if %%k==4 (
                    set /a "z=%1%%jz*%1%%im"
                    set /a "m=%1%%jm*%1%%iz"
                )
                if !%1n!==2 (
                    set /a "t=!z!-24*!m!"
                    if !t! equ 0 (
                        if !m! neq 0 (
                            echo  结果: !%1%%js! !f%%k! !%1%%is! =
24
                            exit /b 1
                        )
                    )
                ) else (
                    set "s=(!%1%%js! !f%%k! !%1%%is!)"
                    set /a "n=0"
                    for /l %%l in (1,1,!%1n!) do (
                        if %%l neq %%i if %%l neq %%j (
                            set /a "n+=1"
                            set "%1!n!z=!%1%%lz!"
                            set "%1!n!m=!%1%%lm!"
                            set "%1!n!s=!%1%%ls!"
                        )
```

```
                    )
                    set /a "n+=1"
                    set "%1!n!z=!z!"
                    set "%1!n!m=!m!"
                    set "%1!n!s=!s!"
                    set "%1n=!n!"
                    call%0 %1
                    if errorlevel 1 exit /b 1
                )
                endlocal
            )
        )
    )
)
exit /b 0
```

---

24 点游戏二

```
@echo off
mode con: cols=65 lines=1100
color 2e
setlocal enabledelayedexpansion
set t0=24
::此处可修改上面"t0"的值以实现其他点数的求解，例如 36 点游戏就是"t0=36"。
title %t0%点游戏的求解--康康编写
set t1=  请按由大到小的顺序输入第
set t2=━━━━━━━━━━━━━━━━━━
set t3=[满足要求]
set t4=copyright@  康康
:z
cls
echo,&echo, %t2% %t4% %t2%
echo,&set/p M1=%t1%1 个数  ___
echo,&set/p M2=%t1%2 个数  ___
echo,&set/p M3=%t1%3 个数  ___
echo,&set/p M4=%t1%4 个数  ___
set/a qq=10000,kk=100
echo,&echo,&echo, --[查找程式启动中...]--
for /l %%A in (90,-2,22) do (
```

```
for /l %%B in (1,1,%%A) do (
pause>nul)
set/p=^>
)<nul
for /l %%A in (23,3,110) do (
for /l %%B in (1,1,%%A) do (
pause>nul)
set/p=^>
)<nul
echo,
for %%A in (1234,1324,1423,2314,2413,3412) do (
set a=%%A
set/a N4=M!a:~0,1!,N5=M!a:~1,1!,N2=M!a:~2,1!,N3=M!a:~3,1!
set/a f1=0
if !N5!==0 (set/a yu1=2) else (set/a yu1=!N4!%%!N5!)
for %%B in (+,-,#,/) do (
set/a f1+=1
set yu1=!yu1:~0,1!
if !f1!!yu1! leq 40 (
set BB=%%B
set BB=!BB:#=*!
set/a N1=!N4!!BB!!N5!&set S1=^(!N4!!BB!!N5!^)
if !N1! geq !N2! (set/a b1=1,b2=2,b3=3,e1=1,e2=1,e3=3)
if !N1! lss !N2! (if !N1! geq !N3! (set/a
b1=2,b2=1,b3=3,e1=2,e2=3,e3=1))
if !N1! lss !N3! (set/a b1=2,b2=3,b3=1,e1=3,e2=2,e3=2)
for %%C in (!b1!!b2!!b3!!e1!,!b1!!b3!!b2!!e2!,!b2!!b3!!b1!!e3!) do (
set c=%%C
set/a P3=N!c:~0,1!,P4=N!c:~1,1!,P2=N!c:~2,1!,e=!c:~3,1!
set/a f2=0
if !P4!==0 (set/a yu2=2) else (set/a yu2=!P3!%%!P4!)
for %%D in (+,-,#,/) do (
set/a f2+=1
set yu2=!yu2:~0,1!
if !f2!!yu2! leq 40 (
set DD=%%D
set DD=!DD:#=*!
set/a P1=!P3!!DD!!P4!
if !e!==1 set SS1=[!S1!!DD!!P4!]&set S2=!P2!
if !e!==2 set SS1=[!P3!!DD!!S1!]&set S2=!P2!
if !e!==3 set S2=!S1!&set SS1=^(!P3!!DD!!P4!^)
if !P1! geq !P2! (set/a R1=!P1!,R2=!P2!&set SR1=!SS1!&set SR2=!S2!)
else (set/a R1=!P2!,R2=!P1!&set SR1=!S2!&set SR2=!SS1!)
```

```
set/a f3=0
if !R2!==0 (set/a yu3=2) else (set/a yu3=!R1!%%!R2!)
for %%E in (+,-,#,/) do (
set/a f3+=1
set yu3=!yu3:~0,1!
if !f3!!yu3! leq 40 (
set EE=%%E
set EE=!EE:#=*!
set/a qq+=1
set/a Q=!R1!!EE!!R2!
set SS=!SR1!!EE!!SR2! = !Q!
if !Q!==%t0% (set/a kk+=1&echo,  ┌%t2%┐ &echo, !qq:~1!
┃!SS:~0,22! ├!kk:~1!─%t3%&echo,  └%t2%┘ &set SS!kk!=!SS!)
else (echo, !qq:~1! !SR1!!EE!!SR2! = !Q!)
)))))))
echo,
for %%A in
(1234,1243,1324,1342,1423,1432,2134,2143,2314,2341,2413,2431,3124,
3142,3214,3241,3412,3421,4123,4132,4213,4231,4312,4321) do (
set a=%%A
set/a N1=M!a:~0,1!,N2=M!a:~1,1!,N3=M!a:~2,1!,N4=M!a:~3,1!
for %%B in (+,-) do (
if not !N2!==0 (
set/a yu=!N3!%%!N2!
if not !yu!==0 (
set/a qq+=1
set/a
Q=^(^(!N3!*!N4!^)/!N2!^)%%B^(!N1!*!N4!^),yu=^(!N3!*!N4!^)%%!N
2!
set SS=[^(!N3!/!N2!^)%%B!N1!]*!N4! = !Q!
if !yu!==0 (set yu=) else (set yu=+[!yu!/!N2!])
if !yu!!Q!==%t0% (set/a kk+=1&echo,  ┌%t2%┐ &echo, !qq:~1!
┃!SS:~0,22! ├!kk:~1!─%t3%&echo,  └%t2%┘ &set SS!kk!=!SS!)
else (echo, !qq:~1! !SS:~0,24!!yu!)
))
if not !N1!==0 if not !N3!==0 (
set/a yu=!N2!%%!N1!,yu1=!N4!%%!N3!
if not !yu!!yu1!==00 (
set/a qq+=1
set/a fm=!N1!*!N3!
set/a
Q=^(^(!N2!*!N3!^)%%B^(!N4!*!N1!^)^)/!fm!,yu=^(^(!N2!*!N3!^)%%
```

```
B^(!N4!*!N1!^)^)%%!fm!
set SS=^(!N2!/!N1!^)%%B^(!N4!/!N3!^) = !Q!
if !yu!==0 (set yu=) else (set yu=+[!yu!/!fm!])
if !yu!!Q!==%t0% (set/a kk+=1&echo,  ┌%t2%┐ &echo, !qq:~1!
 ┃ !SS:~0,22! ┣!kk:~1!━%t3%&echo,  └%t2%┘ &set SS!kk!=!SS!)
else (echo, !qq:~1! !SS:~0,24!!yu!)
)))
if not !N2!==0 (
set/a yu=!N3!%%!N2!
if not !yu!==0 (
set/a qq+=1
set/a
Q=^(!N1!*!N4!^)-^(^(!N3!*!N4!^)/!N2!^),yu=^(!N3!*!N4!^)%%!N2!
set SS=[!N1!-^(!N3!/!N2!^)]*!N4! = !Q!
if !yu!==0 (set yu=) else (set yu=-[!yu!/!N2!])
if !yu!!Q!==%t0% (set/a kk+=1&echo,  ┌%t2%┐ &echo, !qq:~1!
 ┃ !SS:~0,22! ┣!kk:~1!━%t3%&echo,  └%t2%┘ &set SS!kk!=!SS!)
else (echo, !qq:~1! !SS:~0,24!!yu!)
)))
echo,&echo,&echo, --[查找完毕,有!kk:~1!个式子可行]--&echo,
for /l %%A in (101,1,!kk!) do (
set kk=%%A
echo, !kk:~1! !SS%%A!
)
echo,&echo, %t2% %t4% %t2%&echo,&echo,  按任意键重新开始...
pause>nul
goto z
```

汉诺塔

这是个模拟汉诺塔实际移动的程序，所以代码比较长。如果不模拟出来，那么我之前写过的汉诺塔代码短短两三行就行了。

```
@echo off
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::
::本汉诺塔游戏的规则如下:
:: 把 a 上的盘子(从上到下依次为 1.2.3...大于 1 的数字为多个重复只是为了直观的显示盘子的大小,
::并无其它意义)借住 b 逐个全部移到 c 上,在移动的过程中,不能把大的盘子放在小盘子上面.
:: 输入的规则为:第一个字符为要移动的盘子的编号,第二个字符为要移动的盘子所在的位置,
::第三个字符为盘子移动后的位置.如下所示:
:: 1
:: 222
:: 33333
:: _______ _______ _______
:: a b c
::要把编号为 1 的盘子移动到 c 上,输入"1ac"回车即可.
::
::以下为最终的目的:
:: 1
:: 222
:: 33333
:: _______ _______ _______
:: a b c
::
::在提示输入盘子个数时可以输入"h"观看游戏演示
::本游戏由于要考虑效率的关系,并没做太多错误输入的判断,所以请严格按要求输入...
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
:bgn
echo.
setlocal enabledelayedexpansion
set "ttl=9"
:agn
```

```
set input=&set /p input=请输入盘子的个数(2~7)(h 游戏演示,回车退
出):
if not defined input goto :eof
if /i "%input%"=="h" (
set "movie=movie"
set /a "input=!random!%%5+3"
goto :skp
)
if %input% lss 2 (echo.别那么弱智好不好...&goto :agn)
if %input% gtr %ttl% (echo.太多了,你想玩到猴年马月呀...&goto :agn)
:skp
cls
set /a a=1,all=1
for /l %%a in (%input% -1 1) do (
set /a n+=1,m=n*2-1,all*=2
set "space="
for /l %%i in (1 1 %%a) do (set "space=!space! ")
for /l %%x in (1 1 !m!) do (call set "code!n!=%%code!n!%%!n!")
call set "col_a!n!=!space!%%code!n!%%!space!"
call echo.%%col_a!n!%%
)
set /a m+=2,all-=1&set "space="&set "kong= "
for /l %%a in (1 1 %m%) do (
set "line=!line!_"
set "empty=!empty! "
)
set "line=!line!!kong!!line!!kong!!line!"
echo.!line!
set /a "tmp_=%m%/2"
for /l %%a in (1 1 %tmp_%) do (set "space=!space! ")
set "line2=!space!a!space!!kong!!space!b!space!!kong!!space!c"
echo.!line2!
for /l %%i in (1 1 %input%) do (
set "a%%i=ok"
set "col_b%%i=%empty%"
set "col_c%%i=%empty%"
)
if defined movie (goto :movie)
:game
echo.&set in=&set /p in=请输入(回车开始新游戏):
if not defined in (endlocal&goto :bgn)
echo.%in:~-2%|findstr /i /v "^[abc]*$" >nul
```

```
if "%errorlevel%"=="0" (
echo.目标位置不正确^^^!^^^!^^^!
goto :game
)
if "%in:~-2,1%"=="%in:~-1%" (
echo.请别开这种玩笑好不好^^^!^^^!^^^!
goto :game
)
if %in:~0,-2% gtr !input! (
echo.盘子的编号超出范围了^^^!^^^!^^^!
goto :game
)
if %in:~0,-2% lss 1 (
echo.盘子的编号超出范围了^^^!^^^!^^^!
goto :game
)
set /a "tmp_=%in:~0,-2%-1"&set "tmp_1=!in:~-1!"&set
"tmp_2=!in:~-2,1!"
if %tmp_% gtr 0 (
for /l %%a in (1 1 !tmp_!) do (
if defined !tmp_2!%%a (
echo.请先把上面的小盘子移走...
goto :game
)
if defined !tmp_1!%%a (
echo.大盘子不能放在小盘子的上面^^^!^^^!^^^!
goto :game
)
)
)
set /a "times+=1"
set "%in:~-2,1%%in:~0,-2%="
set "%in:~-1%%in:~0,-2%=ok"
call set "var=%%col_!in:~-2,1!!%in:~-2,1%!%%"
call :set_ %in:~-1% "!var!"
set "col_%in:~-2,1%!%in:~-2,1%!=%empty%"
set /a %in:~-2,1%+=1
cls
set n=0
for /l %%i in (1 1 %input%) do (
echo.!col_a%%i!!kong!!col_b%%i!!kong!!col_c%%i!
if defined c%%i (set /a n+=1)
```

```
)
echo.!line!&echo.!line2!
if "%n%"=="%input%" (
echo.&echo.恭喜你,完全正确,你用%times%步完成了这次游
戏...&echo.
if "%times%"=="%all%" (
echo.你的方法是最便捷的,太厉害了^^^!^^^!^^^!
) else (
echo.最便捷的方法只用%all%步就可以完成哦,继续努力^^^!^^^!^^^!
)
endlocal
goto :bgn
)
goto :game

:set_
for /l %%a in (%input% -1 1) do (
if "!col_%1%%a: =!"=="" (
set "col_%1%%a=%~2"
set "%1=%%a"
goto :eof
)
)
goto :eof

:movie
set "str="
call :show !input! a b c
for %%i in (!str!) do (
set "in=%%i"
call call set "var=%%%%col_!in:~-2,1!%%!in:~-2,1!%%%%%%"
call :set_ !in:~-1! "!var!"
call set "col_!in:~-2,1!%%!in:~-2,1!%%=!empty!"
set /a "!in:~-2,1!+=1"
ping -n 2 127.1 >nul
cls
for /l %%m in (1 1 !input!) do (
echo.!col_a%%m!!kong!!col_b%%m!!kong!!col_c%%m!
)
echo.!line!&echo.!line2!
)
echo.&echo.演示完毕...&endlocal&goto :bgn
```

```
:show
if "%1"=="1" (
set "str=!str!%1%2%4 "
) else (
set /a "var=%1-1"
call :show !var! %2 %4 %3
set "str=!str!%1%2%4 "
set /a "var=%1-1"
call :show !var! %3 %2 %4
)
goto :eof@echo off
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::
::本汉诺塔游戏的规则如下:
::  把 a 上的盘子(从上到下依次为 1.2.3...大于 1 的数字为多个重复只是为了直观的显示盘子的大小,
::并无其它意义)借住 b 逐个全部移到 c 上,在移动的过程中,不能把大的盘子放在小盘子上面.
::  输入的规则为:第一个字符为要移动的盘子的编号,第二个字符为要移动的盘子所在的位置,
::第三个字符为盘子移动后的位置.如下所示:
:: 1
:: 222
:: 33333
:: _______ _______ _______
:: a b c
::要把编号为 1 的盘子移动到 c 上,输入"1ac"回车即可.
::
::以下为最终的目的:
:: 1
:: 222
:: 33333
:: _______ _______ _______
:: a b c
::
::在提示输入盘子个数时可以输入"h"观看游戏演示
::本游戏由于要考虑效率的关系,并没做太多错误输入的判断,所以请严格按要求输入...
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
:bgn
```

```
echo.
setlocal enabledelayedexpansion
set "ttl=9"
:agn
set input=&set /p input=请输入盘子的个数(2~7)(h 游戏演示,回车退
出):
if not defined input goto :eof
if /i "%input%"=="h" (
set "movie=movie"
set /a "input=!random!%%5+3"
goto :skp
)
if %input% lss 2 (echo.别那么弱智好不好...&goto :agn)
if %input% gtr %ttl% (echo.太多了,你想玩到猴年马月呀...&goto :agn)
:skp
cls
set /a a=1,all=1
for /l %%a in (%input% -1 1) do (
set /a n+=1,m=n*2-1,all*=2
set "space="
for /l %%i in (1 1 %%a) do (set "space=!space! ")
for /l %%x in (1 1 !m!) do (call set "code!n!=%%code!n!%%!n!")
call set "col_a!n!=!space!%%code!n!%%!space!"
call echo.%%col_a!n!%%
)
set /a m+=2,all-=1&set "space="&set "kong= "
for /l %%a in (1 1 %m%) do (
set "line=!line!_"
set "empty=!empty! "
)
set "line=!line!!kong!!line!!kong!!line!"
echo.!line!
set /a "tmp_=%m%/2"
for /l %%a in (1 1 %tmp_%) do (set "space=!space! ")
set "line2=!space!a!space!!kong!!space!b!space!!kong!!space!c"
echo.!line2!
for /l %%i in (1 1 %input%) do (
set "a%%i=ok"
set "col_b%%i=%empty%"
set "col_c%%i=%empty%"
)
if defined movie (goto :movie)
```

```
:game
echo.&set in=&set /p in=请输入(回车开始新游戏):
if not defined in (endlocal&goto :bgn)
echo.%in:~-2%|findstr /i /v "^[abc]*$" >nul
if "%errorlevel%"=="0" (
echo.目标位置不正确^^^!^^^!^^^!
goto :game
)
if "%in:~-2,1%"=="%in:~-1%" (
echo.请别开这种玩笑好不好^^^!^^^!^^^!
goto :game
)
if %in:~0,-2% gtr !input! (
echo.盘子的编号超出范围了^^^!^^^!^^^!
goto :game
)
if %in:~0,-2% lss 1 (
echo.盘子的编号超出范围了^^^!^^^!^^^!
goto :game
)
set /a "tmp_=%in:~0,-2%-1"&set "tmp_1=!in:~-1!"&set
"tmp_2=!in:~-2,1!"
if %tmp_% gtr 0 (
for /l %%a in (1 1 !tmp_!) do (
if defined !tmp_2!%%a (
echo.请先把上面的小盘子移走...
goto :game
)
if defined !tmp_1!%%a (
echo.大盘子不能放在小盘子的上面^^^!^^^!^^^!
goto :game
)
)
)
set /a "times+=1"
set "%in:~-2,1%%in:~0,-2%="
set "%in:~-1%%in:~0,-2%=ok"
call set "var=%%col_!in:~-2,1!!%in:~-2,1%!%%"
call :set_ %in:~-1% "!var!"
set "col_%in:~-2,1%!%in:~-2,1%!=%empty%"
set /a %in:~-2,1%+=1
cls
```

```
set n=0
for /l %%i in (1 1 %input%) do (
echo.!col_a%%i!!kong!!col_b%%i!!kong!!col_c%%i!
if defined c%%i (set /a n+=1)
)
echo.!line!&echo.!line2!
if "%n%"=="%input%" (
echo.&echo.恭喜你,完全正确,你用%times%步完成了这次游
戏...&echo.
if "%times%"=="%all%" (
echo.你的方法是最便捷的,太厉害了^^^!^^^!^^^!
) else (
echo.最便捷的方法只用%all%步就可以完成哦,继续努力^^^!^^^!^^^!
)
endlocal
goto :bgn
)
goto :game

:set_
for /l %%a in (%input% -1 1) do (
if "!col_%1%%a: =!"=="" (
set "col_%1%%a=%~2"
set "%1=%%a"
goto :eof
)
)
goto :eof

:movie
set "str="
call :show !input! a b c
for %%i in (!str!) do (
set "in=%%i"
call call set "var=%%%%%col_!in:~-2,1!%%!in:~-2,1!%%%%%%"
call :set_ !in:~-1! "!var!"
call set "col_!in:~-2,1!%%!in:~-2,1!%%=!empty!"
set /a "!in:~-2,1!+=1"
ping -n 2 127.1 >nul
cls
for /l %%m in (1 1 !input!) do (
echo.!col_a%%m!!kong!!col_b%%m!!kong!!col_c%%m!
```

```
)
echo.!line!&echo.!line2!
)
echo.&echo.演示完毕...&endlocal&goto :bgn

:show
if "%1"=="1" (
set "str=!str!%1%2%4 "
) else (
set /a "var=%1-1"
call :show !var! %2 %4 %3
set "str=!str!%1%2%4 "
set /a "var=%1-1"
call :show !var! %3 %2 %4
)
goto :eof
```

素数搜索程序

```
@echo off
setlocal enabledelayedexpansion
set /p var=请输入搜索范围:
echo 3 >>Result.txt
echo 5 >>Result.txt
echo 7 >>Result.txt
set /a Num=9
:Start
set /a Num+=2
title  正在检测：%Num%
for /f %%i in (Result.txt) do (
    set /a S=%%i*%%i
    if !S! leq %Num% (
        set /a T1=%Num%%%%%i
        if !T1! EQU 0 goto :Start
    ) else (
        echo %Num%>>Result.txt
        if !var! lss !num! (goto :eof) else goto :Start
    )
)
pause
```

3D 动画

简短的 3D 效果，挺厉害的编程。这个已经超出批处理编程水平了。

```
goto 1
a
db B0 13 CD 10 68 00 A0 07 8C C8 80 C4 10 8E E0 31
db C9 BA C8 03 89 C8 EE 42 D0 F8 78 07 EE F6 E0 C1
db E8 06 EE B0 00 EE 79 08 28 C8 D0 E8 EE D0 E8 EE
db 89 CB 64 88 1F E2 DA 89 CB 01 C8 D3 C0 88 C6 C0
db FE 05 10 F2 64 12 97 FF 00 D0 EA 64 88 17 F6 D7
db 64 88 17 E2 E2 DB E3 D9 EE 80 C7 08 BF 04 02 D8
db 45 F4 57 BA B0 FF BD 60 FF BE FC 01 DF 44 D6 89
db 2C DF 04 89 14 DF 04 B1 02 D9 C3 D9 FB D9 C2 D8
db C9 D9 C4 D8 CB DE E9 D9 CB DE CA DE CB DE C2 D9
db CA E2 E6 D9 C1 DC C8 D9 C1 DC C8 DE C1 D9 FA DE
db FB D9 F3 DE 4C FC DF 1C DE 4C FC DF 5C 01 8B 34
db 8D 00 00 E0 24 40 B0 FB 74 0F C1 E6 02 8D 00 28
db E0 B0 F0 79 04 D1 E6 B0 D0 64 02 00 00 05 47 45
db 81 FD A0 00 75 93 42 83 FA 50 75 8A 5E BF 00 19
db B5 64 F3 A5 B5 C8 4E C0 3C 02 E2 FA E4 60 98 48
db 0F 85 65 FF B0 03 CD 10 29 00 C3 3C 62 61 7A 65

g
q

:1
debug<%0
```

世界编程大赛第一名作品

这是一段很牛 X 的代码，不到 1k 的程序，就有 3D 图像和声音！视觉听觉效果都很好！不愧是世界编程大赛第一名作品！

```
@echo off
more +1 %~s0|debug
e100 33 f6 bf 0 20 b5 10 f3 a5 8c c8 5 0 2 50 68 13 1 cb e 1f be a1 1 bf 0 1
e11b 6 57 b8 11 1 bb 21 13 89 7 4b 4b 48 79 f9 ad 86 e0 8b c8 bd ff ff e8 20
e134 0 3d 0 1 74 1a 7f 3 aa eb f3 2d ff 0 50 e8 f 0 5a f7 d8 8b d8 26 8a 1 aa
```

e14f 4a 75 f9 eb de cb 57 bb 21 13 8b c1 40 f7 27 f7 f5 8b fb ba 11 1 4f 4f 4a
e168 39 5 7f f9 52 8b c5 f7 25 f7 37 2b c8 95 f7 65 2 f7 37 95 2b e8 fe e fe
e181 10 79 6 c6 6 fe 10 7 46 d0 14 d1 d1 d1 e5 79 ec 5a b8 11 1 ff 7 4b 4b 48
e19b 3b d0 75 f7 5f c3 83 f7 83 a6 5d 59 82 cd b2 8 42 46 9 57 a9 c5 ca aa 1b
e1b4 4f 52 b4 92 3f ab 6e 9e a8 1d c6 3 fc e 6a e7 ae bb 5f 7b 10 b8 b4 f7 8
e1cd e2 bf 36 4e 39 9d 79 29 3f a f9 36 52 16 fb 5 e8 e5 a6 c2 e9 b0 43 d3 a3
e1e6 cf d3 fd fd cb d1 4c 5e e0 63 58 86 bb 3e 9 c1 20 bc cc 91 a3 47 81 70 b3
e1ff d6 1a 9e c2 c9 12 e7 4e ad f4 5f e3 30 e9 9 39 d7 e8 f9 f4 d2 44 e8 d7 22
e218 be e2 ce 88 25 cf 30 4a a8 29 ae 3f 47 c6 2d 85 e9 73 54 13 b e6 e0 34 65
e231 e2 50 8a 89 18 5f ce 70 99 3 5f 42 bf eb 7 ae d0 ca 5 22 8d 22 a5 b7 f0
e24a 90 81 bc 7a bc dc 5 db c0 6a 2 e5 57 38 be 60 cb ac ba a5 3b 9d f1 77 38
e263 a6 84 d1 3c af 49 d8 6a 45 a2 76 60 21 12 c0 c2 44 f2 5e bb e5 37 a9 2b
e27b ec 4a 8c 4c f2 f7 a9 58 71 2b ba 6d d6 6a e5 60 46 e0 da e5 b9 90 e5 a3
e293 f7 7f 31 60 58 f0 c4 88 10 4e 3c a3 ee 4e 11 55 8f a 92 eb db ad 7a 9c f
e2ac db 5a 28 96 da 87 ae 91 91 2d e3 5e ea df 6 95 71 67 71 40 ce d1 2e 31 6d
e2c5 c1 9c d8 6a 76 9b 4a e8 36 44 d6 76 d 30 5 ff d4 1b ac 1f 32 65 31 bf 55
e2de 26 b a4 55 e1 5d 5e 16 ed 97 48 6c 77 fb 81 86 e f9 18 bd d4 f4 8b de 1d
e2f7 ba d 47 75 3 89 4b 3e dc 27 86 1c d0 17 89 48 d1 a6 8d d4 2b 54 4e 8f b0
e310 2 e1 6b 1a 75 78 ea 21 91 13 c0 cf 78 a0 ab f3 35 c6 b4 c8 90 8d d7 45 e7
e329 c 5b a4 ba 52 10 64 f5 4a 50 b7 ec 46 22 15 23 84 30 81 5c df 61 5a 8f 67
e342 c4 63 57 6d f7 26 92 a3 1f e5 3 a5 0 54 41 8 48 7c 26 90 33 82 9c 91 b0
e35b ab 78 5d df 99 e0 b9 fc 5 36 ac d9 49 91 ab 20 a2 63 48 89 ce 5c 60 64 f0
e374 63 d9 a8 38 3b d3 e6 4c 8c 23 34 4e 20 51 93 5e 6d b4 7a 22 9b 4c f2 d3
e38c c4 f8 3 6f 47 40 f4 f8 45 9b 83 f3 83 6 31 d0 0 17 82 83 dc 67 f9 62 77
e3a5 90 3b d9 ec f3 55 96 b8 d9 db 79 55 f1 e5 8c 5e f2 e5 2e b0 b 6e e2 81 25
e3be 93 8e b5 dd 5b 46 f9 af ed 6 12 cf c9 1d f0 f7 3b 16 2d c6 58 73 8d e9 5f
e3d7 fd 5a b6 a1 94 4d 1a 8 ff eb b7 6 80 c7 86 83 b6 b9 fd 1c e0 c c3 2e a0
e3f0 2f b 3e 3 6b 29 e1 27 85 1c ea 6d df b3 a3 ed 65 4a 9a 59 3b 54 e 4b ae
e409 9e 27 f0 4d 3b c 4c 46 b7 e5 57 1b 1f 1f bb 80 86 f5 b7 ef 73 52 bf 2c c7
e422 ed a b7 81 2 f3 90 3e ee cc 6c eb f 38 1 6c 68 b1 d 45 78 b2 f f6 83 b0
e43c c4 33 df b1 d1 91 98 1e 81 a5 e2 59 9f f4 8c b6 72 8 a7 8c f6 e a3 b2 1f
e455 d9 d3 23 f0 7c 5e 5f 68 61 8b 45 da 1d 91 ec 8d 4e ea 1a 38 85 94 aa ac
e46d f2 4 f6 c4 e5 92 8e 9a 4e 83 e1 73 e8 cf 2a 5c 2b 7e f1 30 2 8a e6 28 1a
e486 3b ce bc 96 aa 7f eb 87 cd 8b 96 2d 9 59 7a a0 1a 43 62 9a 9e 4f ff 8e d9
e49f ce d6 a4 70 79 cd 65 fa 2e 92 14 29 f7 6c 74 4b 49 60 80 bb ff 41 bb 2d
e4b7 60 33 3f 98 77 9a 1 ee a6 a3 da bc ba e9 f3 72 f4 7c c3 59 2 a6 44 a4 c8
e4d0 c8 54 93 ce bd 69 bb b9 43 21 2c c4 ea 4a 5c 3f 75 60 f2 b4 91 ca 9 82 e3
e4e9 a e9 a6 20 b9 76 50 ed 47 e9 fe 6d 41 34 13 2f 28 2f 4e f4 da e 3c 78 6c
e502 b1 79 87 45 98 a4 d4 c3 b3 29 c2 4a 8b ed a6 54 e2 1b 31 62 60 ff 2c 1d
e51a 21 0 15 b2 4e 5c c 2 d 83 fa a2 f3 8a 5 12 72 4a c7 44 7c 91 d4 be b a f2
e535 70 52 fb b4 a2 df 89 de ff c4 96 73 c9 c ed d3 c9 8e 5c dc 8e d1 3b de 8c
e54e 53 a2 8b f9 e9 91 dd d6 df 6e 74 d1 dd 34 60 8f 9e 32 7f 3b ec 79 a3 83
e566 45 78 b4 2f 1c 50 7b 7a 97 b0 9d 2d c dd 8a 26 cd 7d 8c 4c 5a 8a 4c f9 a4
e57f 11 f9 2c 6c 92 e9 b5 cb 56 89 8c be f6 64 fa 25 43 fa 6f e2 c8 3a 18 a8
e597 f0 e9 f4 c2 86 e6 2b 44 67 4a b9 34 9 ed 5f 33 42 62 d4 8a 1e 5b 31 67 cd
e5b0 3d 71 6d 83 fd 36 20 69 ea 1 c3 e6 e6 de 99 aa 7 11 5b 59 8a 1f 43 83 52
e5c9 ea 5d 8c 6a 69 c7 3 eb 4e 3b 88 a5 5f b1 6e 27 5f 3 5c 28 c 9b 6c c3 f8
e5e2 e5 b9 d6 11 d6 8b fa 5c 8 c7 1 eb 45 db f3 6c 9f 16 46 61 51 ed df f bb
e5fb c0 c4 1e 64 68 98 4 79 30 94 72 df d4 cd 1f 7f 72 c6 82 2e 79 47 4e 8c 4b
e614 a2 c7 e2 36 df 76 fd a4 b6 4e db 96 40 3b 8b b5 d4 85 64 c6 0 2c ad 9d 27

e62d 14 99 82 4b bc 9 fa 94 b5 db 7c 98 eb b 13 a7 b0 79 1d 7e c5 45 aa 20 49
e646 be ff 9d 64 0 5d c ec 6 5 ad f2 38 6b ed 7a d6 b2 c7 2e 6a a6 12 4b ff 55
e660 20 3b a 77 f b9 0 9d 57 4a ad ce a4 d3 ff 1 4f fb 53 54 88 f 1 ed 4b 56
e67a 15 c8 dc 28 bf f2 72 d4 10 1f 99 42 69 9e 78 e2 47 82 93 31 d0 2d be 9f
e692 93 93 9a 1b 80 c0 10 c 53 78 a0 26 2a 96 4f 74 4b 16 c7 9c 8d ad ac fb 16
e6ab 15 c6 fd c9 a4 14 48 62 47 20 c9 41 ed 61 f8 9b f8 ff ba 39 50 65 87 ee
e6c3 bd ce 95 c0 fb a5 7e d8 cd 27 fd 2c 74 3 c1 1b 89 b9 51 d5 e3 da ef 9e 6
e6dc f0 aa a9 a7 fb 87 4c 5d cd ff 65 36 8c 73 6f 9 c6 78 9a b6 77 db df 81 68
e6f5 3b b8 ae 5d e1 af d4 e6 66 8c d6 a4 83 9f 37 3c 1 dc a2 a6 57 c2 20 1b 90
e70e 75 df cd a5 62 a5 36 79 fb 35 8a 9b b0 a0 a5 c3 37 6f 80 72 bc 52 30 8d
e726 9f 7a 64 d3 7 41 45 d8 68 97 f2 aa 1c a1 6c 7c 9d 32 7d ad 15 b1 53 e3 33
e73f 8a ed e9 49 d4 cf dc 96 22 37 36 11 9d 7f f0 4d e0 62 31 b1 c7 69 c4 79
e757 ac 20 1 e8 3c 6a 8c 32 cb 52 63 36 68 f4 10 2b 9c 21 4f df 5d 60 92 39 91
e770 e2 f9 c9 7d ca 48 3 3f 21 dd 6c f 23 2e 61 3a 9f ba c3 f9 4e 7 ea ed ef
e789 71 4a 72 3a ed 23 3d 77 b5 ed d5 1d f6 a4 99 fa ef 98 dd 2 98 80 b6 7c a3
e7a2 62 96 7b 8e bf 7b 81 9f 9a ce 3f 12 40 2e 25 db 84 16 dd 2e 86 f f4 b2 7e
e7bb 5e b4 14 6a f3 29 b1 a4 57 d5 a8 17 6f 87 a4 74 5b 9b 17 79 f1 ec 33 c8
e7d3 f0 1d b2 7e a8 4d 95 7f 5f 9 d5 1a 5a 45 f4 41 c6 d 3f eb 66 2a c0 e8 5b
e7ec 3c bd 50 ad f1 53 9d 2e 45 9a d8 7d 2c 17 a8 6e 15 48 13 39 53 ed 3d 78
e804 ad f 3a 65 a3 3e 2e fa ca 7 94 4a 1f b4 d8 7e 47 8a 8e de e7 7e 34 c1 69
e81d 7f 6a aa 66 58 18 31 24 72 13 22 34 8a 56 36 87 df c2 d 8e 3f 71 a2 5f 25
e836 8b 8d 4 78 fd c9 45 d1 55 79 c1 9f 13 84 1b c8 5 db 95 d0 7c 64 96 20 51
e84f c4 e0 5e ee 47 8a 11 ac fb 9 e0 bb 40 db 86 84 12 93 b9 c9 f2 9c 63 47 c9
e868 eb ad 1 3e fa 6d 3f a 64 5b 58 56 27 f ca 5d e0 30 bc 3e 10 5d ec 17 28
e881 85 5 51 8e 95 a3 94 3a a8 f1 96 f2 f 29 5c 97 dc 47 db 9d 6c 63 e8 e7 f0
e89a e4 a 70 f8 f1 47 54 d3 2d 32 7c ef bb 9a b4 1b 0 2b d6 dd e7 30 b a2 75
e8b3 c7 f5 d0 31 d7 d2 8a b0 ac 1c 6d 60 3a f7 c2 db 1e 6d 7 f6 8f 35 88 e5 7f
e8cc 3c 26 81 34 a0 32 a3 25 18 6e 73 b2 a0 f1 cb 86 61 e7 65 8b 76 98 19 6f
e8e4 c0 62 9b a3 cc 18 5e 40 12 97 2b d0 15 79 de 19 ea df 7a 59 2f b5 d7 39
e8fc 52 e2 6 f1 3 a0 a5 d9 1b 88 93 4d 30 c8 2d f5 db 55 ea 85 6f a 3f dc bd
e915 57 15 6a a3 a3 3e 8e ad 2d da a0 ca 75 7c 57 8b c5 cb b 1d 2c 8e c6 96 2e
e92e 6d 59 83 7d 64 72 ca 80 2e 6 a4 ff f6 f2 d5 1e 7 4 ba 34 6e 9 86 25 aa 4e
e948 e0 7f f5 32 47 3e 7c 43 d8 28 c4 1c 11 1d bd 33 3 b5 ca 13 43 34 2 b1 a0
e961 57 ed 9d 3c 23 d4 45 b2 6e 81 6e af 3e 67 90 be 59 a5 45 34 53 46 85 d1
e979 25 ee 7d cb a4 db 12 c3 aa 17 61 9a fb 66 40 76 fe 3a 69 96 c0 91 14 a7
e991 5d cc 9f f6 73 59 ee b8 55 97 20 26 ff 99 ec 72 41 b5 27 21 6e ae 8a d0
e9a9 e4 d3 da 6f c4 53 c5 f8 b3 a7 a1 5d 66 93 d8 b1 89 40 23 92 c0 90 fb cb
e9c1 e7 6b 4e 51 0 5d 57 f7 cd 1 e2 88 bf 44 9f ef c4 33 ce fa 46 46 a1 86 b
e9da 7a 84 66 66 b9 2 ec 10 c6 a1 d4 c1 18 33 b1 d1 2 18 ad 2f 53 e4 b9 33 59
e9f3 be 3c af 80 4c 8a d5 76 c 3b a7 e2 97 94 15 75 4d 17 d5 97 cf f9 4a d0 6e
ea0c bb 27 20 fc f1 f5 9 a8 df 4d b6 5d f0 1d 69 3b 76 35 82 a4 f3 56 64 39 5b
ea25 6b b3 7 e7 5 8e 82 11 22 a8 1a db c8 3e 67 4a 3 7e 72 51 d6 3d 1a 1c f6
ea3e b8 da 4b 18 8a 15 9d d0 a4 84 96 3e cd 3 f9 3a 30 f3 fb 8f 6e 2 73 eb 52
ea57 93 95 cf dc 6f 48 fb ab d2 a9 70 b4 e2 23 8d 72 86 a8 fa 78 98 1d c5 fe
ea6f 8a 51 88 2b b7 58 b0 ca ae 40 8a 33 32 75 1 6 c0 d4 b7 da 2a a7 bb ad f7
ea88 48 98 5a bc d3 d1 e6 16 97 c3 80 ab 73 ac 32 11 41 1f d 5d aa 0 dc d9 6e
eaa1 fc 30 6 ef 11 60 27 a2 5f eb 5f b9 35 8 23 4 be 10 c0 85 3e 55 b3 82 fd
eaba f7 c3 24 9f 2d 83 94 32 36 de ff 7c 87 7f 4a 80 7 2 23 cf a4 52 eb 3e 19
ead3 a0 b4 a 94 1a 40 58 d9 16 6d c0 64 c4 69 ed 60 46 65 cb df 58 38 0 51 c3
eaec ad a0 37 e4 cf ab f7 6c 24 7d 9 48 65 4a 9f 91 ad 1c 79 a4 a1 78 55 c e8

eb05 44 5b d ef 51 bd ea 2d a7 42 57 ab 3a 4f 2 b 3 19 6a 4d 72 76 5c 97 0 6c
eb1f c5 5d bc dd e7 81 cf 8d 34 38 50 3c 98 58 cc 41 aa 99 90 af fe 4e 96 77
eb37 ed 54 18 ce 2c d1 5d 34 cb 79 50 ff 28 96 44 e0 51 64 6 a8 b7 6e 8c 62 c4
eb50 66 95 81 4f 8c f6 26 ba ea 5d d2 79 b1 e4 e9 29 fc a fd b3 85 8c e6 52 dd
eb69 33 bd 5d c7 39 ef 6 ef 9e a6 6a 61 9c 9f d5 54 b4 fa a1 d4 10 9b ff 7e 33
eb82 11 52 99 c7 26 6e a1 36 8a ad ee 48 7a 2c 7f d5 b7 27 8a 6b 37 c 71 39 85
eb9b 9c ba a8 a 17 b9 d0 51 56 95 c2 3b 5 a7 31 c5 8b 5c 95 6e 4c 89 6f 17 ef
ebb4 d4 5a a 77 65 e1 49 b2 e8 72 ac 3c f0 6b 71 fa 3 c7 ca fc ad f9 55 22 ec
ebcd 58 2f 1c fa 29 cf 73 b4 ad 51 5c f8 66 70 59 5d 70 3e d1 3f c4 eb ec f1
ebe5 7 78 6a 93 67 9f 44 fc cb 5b 95 ff 74 c0 b7 42 77 26 c9 aa 8c ed 39 a2 db
ebfe 9c b3 eb 3d 4a 1e 9b 89 e4 d8 a8 27 74 ef a3 ed a5 24 5d bb ab d0 fe a1
ec16 29 ab df 75 a a6 23 0 cc f1 14 72 9b 1a 55 7e e5 d1 da 98 dc c4 cf ab 34
ec2f ba 8d de 4a 59 6 13 dd d8 44 3c e bb 56 95 ae 97 e2 3b 49 e5 9a 6b a2 53
ec48 c1 33 35 24 1b 33 17 c3 8a 8c 12 3d 3d 4e 5b 75 22 30 67 4f a0 5d 3a 78
ec60 88 a 11 35 7 b1 77 42 32 a8 c3 bb 20 fb 98 5 d6 ac e7 3a 63 35 90 93 9e
ec79 44 24 2e 1b d7 8c aa 29 53 4d d9 ab eb e6 1 56 c4 fd 54 a3 bd 14 5b b0 8f
ec92 ce be 23 24 93 c4 48 18 a3 e7 4 5 4b 78 cc 79 dd 3 56 a4 ed dd 5f 98 41
ecab 1b 68 4c c1 bb 41 c2 1e 3e 94 8e ef 28 1e b 76 e 4f 36 b1 c 6e e2 18 17
ecc4 20 fc 35 40 1f e4 6d a4 18 bb bc d5 9e ea 85 86 af af 63 d4 13 66 92 c4
ecdc 2b 69 84 ca 23 2b d3 66 81 6b 81 73 26 4 85 36 21 4c 49 44 75 64 39 16 3c
ecf5 ed e0 6d 44 75 45 30 43 68 c0 78 fc d0 17 b eb 81 3e c3 ba 1b f 4d ae c5
ed0e 55 1f c 39 12 5d 8 65 f1 34 59 de dd 98 56 17 43 38 66 49 9a eb db c1 87
ed27 51 38 cc b7 5f 98 fd 43 be 2d bb 74 f3 f8 f2 36 3d a4 34 a5 7e d2 26 cc
ed3f 84 1f ea 56 f0 80 18 69 4d 88 41 fc 56 fd 41 3b 1e e 9 27 4f f6 3b 62 4e
ed58 5a 1b 2a 4e 85 8c b2 4f 79 ef 59 4e e 73 3d bd c4 ca 60 e7 4a 47 90 b5 8
ed71 2a f0 4e dc ba 66 ae 48 2b 31 73 a2 11 c 32 ff 54 14 77 6b d6 58 4b bf ee
ed8a f6 6a bc dd 1 88 d da a9 f 81 24 c5 f8 72 9a db d5 c8 2a 80 a9 16 d7 c6
eda3 b1 91 c0 a9 95 40 b5 b3 a8 2a 28 c6 92 16 ab 54 7d f8 93 5f 3a 17 c8 45
edbb a9 f0 e0 71 23 76 53 38 a5 a1 cc d4 f1 f2 3c 2b 46 43 a1 d5 ba e d7 19 7a
edd4 c2 e1 8f 67 1d d 98 9d a1 79 9d 1b 20 7f 4d e7 bf f9 ff fe aa 28 ab 8f c
eded 4d 50 33 e3 26 fc 3c 3 3a 2b 26 12 f7 1 8f ee 97 4c e6 6 2b d9 1f a1 4a
ee06 77 44 d4 8b b7 3e 5e 2d 18 c3 54 68 99 a8 8d 92 96 9e 9d ab 33 38 ff b8
ee1e ee 78 c6 7b b5 84 95 d3 6 27 ae 5d 27 38 a 38 8e f0 1 a5 96 4b d7 9b 42
ee37 e5 6f 57 75 4c e9 78 2d 5b ec b6 d2 29 e2 a8 92 95 9c 65 2a 3e bf 8d e0
ee4f bf b3 ac c8 e 7e 13 af 88 26 7d 48 5a c7 39 29 36 d2 90 e8 3b 3 d0 61 1a
ee68 d2 e8 a8 f ba 8e a1 9f df 12 ab 54 7 23 98 de 62 af 4c 7e d4 fb 6b 2 6e
ee81 40 40 37 b7 73 f2 d8 81 be 29 d2 99 c0 73 25 1a 3c 92 75 6e bd d7 79 79
ee99 4 14 c0 4e 99 57 66 93 74 ec b0 29 7c df 61 b0 3 3a d1 c3 fa a4 f7 f 9f
eeb2 d3 f 0 b9 2a 5a 3a c5 88 25 b8 b9 cc 82 3 57 3a e1 7b 51 75 70 a6 74 1a
eecb ca cb 3 18 68 ca 77 fe 1b ad cd 68 7f 36 85 fc b7 4f a0 11 da 69 fa 79 87
eee4 d6 b9 21 dd 3e 70 db dc 84 d4 6e d1 20 4 af f6 32 a2 8e d 54 25 fe 7 54
eefd e 7a 74 4b a0 4b f7 f4 e8 74 22 e9 98 70 fb 25 2e f4 64 57 75 28 85 45 53
ef16 3a 2e e2 3c 54 36 e9 29 6 67 59 43 10 7e c1 49 cd 5e f9 97 a 58 5f 8a 11
ef2f 4f 3d 9a e2 2b 22 58 fa be fc 69 91 7a 8c 3f 77 9f c9 3b 54 26 23 93 b3
ef47 85 de ae f5 bd c5 47 4c c4 cd 5e ad bc 8f ba 31 f6 e4 70 fb 6e a7 96 d5
ef5f ad 10 80 39 43 97 4f 10 cc 1b 8f 8d cd 4c 63 4 d8 1e 85 70 41 6c a8 eb df
ef78 7f 36 c5 60 a7 12 9 16 73 fe 75 3a 2d 40 29 7d aa a 5c 2 29 23 0 a6 e5 6b
ef92 24 6d 9b 20 e5 7 cb 40 b0 38 59 9c a7 69 6a 70 d3 38 ef e2 b2 11 3e ea 2a
efab f9 2b 2e 43 1d 65 cf d6 1b ef 83 5a 5f e6 c5 62 16 ca 5e 4c a6 39 e4 53
efc3 2d 23 d2 5e 7e 15 54 8a 8 b7 3d bb 88 59 b9 9e a2 7c 42 1f a2 77 3c 5b 9

efdc 6d fa 8f 21 46 1a 3e ed ce 49 56 1d 29 2d 70 3 a7 6f 75 ac 1 87 ff 27 86
eff5 73 49 28 85 2d 97 7a 84 e 37 3d 86 10 21 4c e2 74 62 6b 51 70 8f 15 72 f3
e100e 81 b2 a9 9d 8a 63 ad 1b d5 aa 8a dc 96 3c e7 47 16 51 fc 87 50 9 b7 60
e1026 29 33 52 fb b0 df 70 c5 65 4a 60 3b c d7 a8 29 47 51 f7 8a 77 f3 99 3f
e103e 38 16 60 de 68 27 b2 24 7 62 a2 fd 40 86 b2 75 c3 3c 2f 3d fa 9 d9 a9 9a
e1057 71 3c ce 46 94 0 f9 bc 46 7f b8 2e 85 7f 7d d3 8d ea b4 63 81 59 10 bb
e106f 57 d0 b6 ab e1 83 74 1e 25 d5 73 78 18 b1 60 62 c f4 76 8d 17 d5 ed 23
e1087 23 e4 f6 32 64 5a 61 9 63 f6 92 57 d5 29 40 d6 3b ba 63 72 18 0 25 1b 7
e10a0 ee 7f 25 4a fa 6 74 19 46 e3 e8 89 7a c6 56 54 a7 43 13 4e bf 97 a5 6f
e10b8 99 2f ac 33 4d fa 58 3a 5a a a4 1a 74 62 c8 4f 3b 78 9 d7 ee 7e ee 2d 69
e10d1 30 40 ea 47 82 3b 85 8e 3 23 8f 74 4e 8 35 ab 74 4 1 57 d5 85 b1 6b 1e
e10ea f4 7d 1e d2 1e b3 fe f3 12 10 32 39 51 48 2d 6f e5 d3 a3 8c 8 8

g
rcx
fff
n1.com
w
q
pause

批处理评说

批处理的特点

1、所有变量不必预先定义，随时用到的变量随时定义。

对变量随时想用随时定义，看似随心所欲非常方便，其实不然。因为没有预先定义，计算机就无法判别我们哪些代码是变量，这就需要我们在使用过程中，时时刻刻标明哪些代码是变量，而且标明的符号还分为没有延迟变量的%var%和有延迟变量的!var!，反而更加麻烦。一个很麻烦的例子：当开启了延迟变量后，如果要引用数组元素 S!n!的值，我们不可以用!S!n!!，用%S!n!%或%S%n%%就更加大错特错了。正确的做法是先用 set /a var=S!n! 然后用!var!来表达 S!n!的值就行了，这个利用了 set /a 计算语句，后面的表达式的变量可以标明也可以不标的规则来达到目的。

2、批处理的数组的长度也是动态的，这一点挺方便。但数组的下标必须用确定的值表示，不能用类似 i+j 的计算表达式，那么我们需要表达相应的数组时，就要另外用一条表达式先计算出 i+j 的值，然后再将其作为下标，这个又比较死板。

3、批处理如果没有@echo off，那么所有命令执行都会被回显，挺烦人，造成了批处理程序几乎都是以@echo off 语句开头。我想问一下：为什么不反过来，如果没有 echo on，所有命令执行都不会被回显呢？？？

4、批处理所有数据都不必定义数据类型，所有数字都等于 C 的 long 类型，也就是-2147483648 到+2147483648。批处理不支持浮点运算，除法结果一律舍弃小数部分，这个在某些情况下会很不方便。（当然，山不转水转，我们可以通过模拟数组来实现浮点运算和高精度运算，但这是另外的话题了）

5、批处理所有数据都不必定义，这就造成了字符型数据和数据型数据混杂在一块。不过也有好处，就是读入一串数字时，我们既可以当做数字来处理，来进行大小比较和运算，也可以当做字符串分解到数组里。但批处理不区分数字和字符也有不好的地方，就是我们很难将字符和数字相互 ASCII 转换，造成了用批处理输出 ASCII 表都是个挑战！

6、批处理默认状态下不能感知数据的动态变化，需要用到数据的动态值，就要开启延迟变量，不知道 DOS 这么设计有什么意义？

7、for 命令首当其冲是因为其适用范围无人可比，而且技巧层出不穷，基本上稍微复杂的代码都会用到 for。用 for 循环时，假如循环变量不满足条件，那么循环一次都不会被执行。例如 for /l %%i in (1,1,%n%) do echo Hello!，当 n 小于等于 0 时，循环增量无法从 1 增加到 0，那么程序一次都不会被执行，所以这种情况就不必增加判断 n 值大小的情况了。

8、批处理旁门左道的用法太多，令人非常头疼。

例如一般知道用 echo 可以显示出信息，而 set 是拿来赋值和计算的。殊不知，ehco ^G 还可以发声！set /p var=%num%<nul 可以用于显示信息！再看一个例子：

```
@echo off
set /p var=
echo %var%|findstr /be [0-9.]* >nul && echo Yes||echo No
pause
```

这里的 echo %var%不用于显示功能，而用作参数功能。因为纯粹的%var%不可以被执行，找不到合适的命令来作为%var%的开头，所以选用 echo。后面的>nul，是因为我们只需要参数功能就可以了，不需要其显示功能，所以用>nul 阻止信息显示。而 findstr 在标准帮助信息里，并没有使用参数的语法格式！能想到给它参数的人，非高手不能办到！

9、为了感知变量的动态变化，批处理必须借助 setlocal enabledelayedexpansion 延迟变量，这个标示符好长哦，偶至今还不会背诵它!为什么批处理不支持变量动态变化呢？像 C 一样多方便啊？搞得基本上复杂点的程序，都要用到 setlocal enabledelayedexpansion 这个令人望而生畏的标识符。

10、批处理毕竟不是一门计算机语言，它的数据类型不够丰富，例如没有指针，于是无法制作链表、二叉树等等数据类型，那么好多东西会很难实现或者无法实现。

11、批处理用 call 可以实现函数调用，于是递归算法变得可能。但是，call 在传递参数时，最多只能从%1-%9，而且不能用数组参数，这个会令很多东西很难实现或者无法实现。

12、批处理没有真假判断，于是我们就需要自行定义数据用以真假判断。

13、批处理的数值计算是弱项中的弱项，比 C 语言耗时长 n 多倍，所以用批处理来计算数据并不是明智的选择。

14、批处理用在对操作系统的操作，实现一些特定的功能上，却远远比 C 语言简便多了。特别是网络测试时，批处理更是利器！

15、C 语言的关键字只有 35 个，扩展的 C99 标准也没超过 40 个，而批处理的常用命令超过了 100 个！加上不常用的命令，估计可以达到 150-200 个！而且每个命令有若干不同参数用法，每种用法又有许多种语法格式细节！其中的细节奇多奇复杂，稍不慎就出错，而且往往还很难查出错在哪里！然后像 set、call、findstr 等还有 n 多旁门左道用法，这个对记忆力来说是个巨大的挑战！批处理是典型的难学易忘！

出错例子：

```
@echo off
echo  请分别输入三个字符(字符之间用回车隔开)
set /p a=
set /p b=
set /p c=
echo  三个字符合并结果为
set e=          %a%
set /p   var=%e:~-7%<nul
set e=          %b%
set /p   var=%e:~-7%<nul
set e=          %c%
set /p   var=%e:~-7%<nul
echo.
pause>nul
```

上面代码很简单，实现的功能是分别读入三个变量，然后合并为同一行输出。我们分别输入“a、b、c”，或者“,、+、=”，或者其它任何字符，结果都可以合并为同一行输出，但分别输入 0-9 这十个个位数字就不行了，例如“7、8、9”，程序没有显示什么内容。解决的办法是在“<nul”前面添加一个空格。但这是为什么呢？估计没几个人可以回答！没错，批处理的细节就是这么诡异！像这样的 bug，很难被发现！批处理就是这么折磨人！

后记

2013 年冬，偶然在网上看到一些批处理代码，很有意思，于是很想学习批处理编程。在网上搜索了老半天，没有发现比较系统规范的教材，很是失望。无奈只能在论坛、博客、网页上寻找。攻克一个个命令，加上自己的研究与实践，不断前进。

网上的东西鱼龙混杂，令人学习起来磕磕碰碰，异常艰难，于是想到要写一本规范全面的教材，方便以后的广大读者学习。从开始学习批处理到写完本书，耗时半年，倾注了笔者大量的心血。本书详细讲解了大量的 DOS 命令，循序渐进，并且附有大量有趣的代码，代码全都在 WindowsXP 系统下测试通过，是目前网上最全的教程，学完本教程后差不多就是批处理高手了。

批处理博大精深，限于时间和笔者的水平，错误和不足在所难免，并且还有好多领域没来得及仔细研究，这些有待读者批评指正并完善。

作者：MHL
QQ：　1208980380
邮箱：1208980380@qq.com
2014 年 5 月 25 日于深圳市福田区