参考：<http://blog.csdn.net/mycoolx/article/details/50251705>

第一章批处理基础

第一节常用批处理内部命令简介

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

小知识：可以在键盘上按下Ctrl+C组合键来强行终止一个批处理的执行过程。

了解了大概意思后,我们正式开始学习.先看一个简单的例子!

@echo off

echo "欢迎来到非常BAT!"

pause

把上面的3条命令保存为文件test.bat或者test.cmd然后执行,

他就会在屏幕上显示以下二行话:

欢迎来到非常BAT!

请按任意键继续. . .

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

从上面这个简单的批处理中,我们可以发现其实批处理就是运用一些含有特殊意义的符号和一些完成指定功能的命令组合而成,那么在批处理中有多少这样的特殊符号和功能命令呢？我们现在就来仔细了解一下一些最常用的!

(以下内容来源网络,请各位仔细阅读,好进入下节的实例说明)

======================================================

批处理的常见命令（未列举的命令还比较多，请查阅帮助信息）

    1、REM 和 ::

    2、ECHO 和 @

    3、PAUSE

    4、ERRORLEVEL

    5、TITLE

    6、COLOR

    7、mode 配置系统设备

    8、GOTO 和 :

    9、FIND

    10、START

    11、assoc 和 ftype

    12、pushd 和 popd

    13、CALL

    14、shift

    15、IF

    16、setlocal 与变量延迟

    17、ATTRIB  显示或更改文件属性

介绍命令

1、REM 和 ::

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

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

第一, 任何以冒号:开头的字符行, 在批处理中都被视作标号, 而直接忽略其后的所有内容。

有效标号：冒号后紧跟一个以字母数字开头的字符串，goto语句可以识别。

无效标号：冒号后紧跟一个非字母数字的一个特殊符号，goto无法识别的标号，可以起到注释作用，所以 :: 常被用作注释符号，其实 :+ 也可起注释作用。

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

行内注释格式：%注释内容%  （不常用，慎用）

2、ECHO 和 @

@字符放在命令前将关闭该命令回显，无论此时echo是否为打开状态。

echo命令的作用列举如下：

（1）打开回显或关闭回显功能

    格式:echo [{on|off }]

    如果想关闭“ECHOOFF”命令行自身的显示，则需要在该命令行前加上“@”。

（2）显示当前ECHO设置状态

    格式:echo

（3）输出提示信息

    格式：ECHO 信息内容

    上述是ECHO命令常见的三种用法，也是大家熟悉和会用的，但作为DOS命令淘金者你还应该知道下面的技巧：

（4）关闭DOS命令提示符

    在DOS提示符状态下键入ECHOOFF，能够关闭DOS提示符的显示使屏幕只留下光标，直至键入ECHO ON，提示符才会重新出现。

（5）输出空行，即相当于输入一个回车

    格式：ECHO．

    值得注意的是命令行中的“．”要紧跟在ECHO后面中间不能有空格，否则“．”将被当作提示信息输出到屏幕。另外“．”可以用，：；”／[\]＋等任一符号替代。

    命令ECHO．输出的回车，经DOS管道转向可以作为其它命令的输入，比如echo.|time即相当于在TIME命令执行后给出一个回车。所以执行时系统会在显示当前时间后，自动返回到DOS提示符状态

（6）答复命令中的提问

    格式：ECHO 答复语|命令文件名

上述格式可以用于简化一些需要人机对话的命令（如：CHKDSK／F；FORMAT Drive:；del \*.\*）的操作，它是通过DOS管道命令把ECHO命令输出的预置答复语作为人机对话命令的输入。下面的例子就相当于在调用的命令出现人机对话时输入“Y”回车：

C:>ECHO Y|CHKDSK/F

C:>ECHO Y|DEL A :\*.\*

（7）建立新文件或增加文件内容

格式：ECHO 文件内容>文件名

     ECHO 文件内容>>文件名

例如：

C:>ECHO @ECHO OFF>AUTOEXEC.BAT建立自动批处理文件

C:>ECHOC:\CPAV\BOOTSAFE>>AUTOEXEC.BAT向自动批处理文件中追加内容

C:>TYPE AUTOEXEC.BAT显示该自动批处理文件

@ECHO OFF

C:\CPAV\BOOTSAFE

（8）向打印机输出打印内容或打印控制码

格式：ECHO 打印机控制码>;PRN

     ECHO 打印内容>;PRN

下面的例子是向M－1724打印机输入打印控制码。＜Alt＞156是按住Alt键在小键盘键入156，类似情况依此类推：

C:>ECHO +156+42+116>;PRN（输入下划线命令FS＊t）

C:>ECHO[email=+155@]+155@>;PRN[/email]（输入初始化命令ESC@）

C:>ECHO.>;PRN（换行）

（9）使喇叭鸣响

C:>ECHO ^G

“^G”是在dos窗口中用Ctrl＋G或Alt＋007输入，输入多个^G可以产生多声鸣响。使用方法是直接将其加入批处理文件中或做成批处理文件调用。

这里的“^G”属于特殊符号的使用，请看本文后面的章节

3、PAUSE

PAUSE，玩游戏的人都知道，暂停的意思

在这里就是停止系统命令的执行并显示下面的内容。

例：

PAUSE

运行显示：

请按任意键继续. . .

要显示其他提示语，可以这样用：

Echo 其他提示语 & pause> nul

4、errorlevel

程序返回码

echo %errorlevel%

每个命令运行结束，可以用这个命令行格式查看返回码

用于判断刚才的命令是否执行成功

默认值为0，一般命令执行出错会设 errorlevel 为1

5、title

设置cmd窗口的标题

title 新标题        #可以看到cmd窗口的标题栏变了

6、COLOR

设置默认的控制台前景和背景颜色。

COLOR [attr]

 attr        指定控制台输出的颜色属性

颜色属性由两个十六进制数字指定 -- 第一个为背景，第二个则为

前景。每个数字可以为以下任何值之一:

    0= 黑色       8 = 灰色

    1= 蓝色       9 = 淡蓝色

    2= 绿色       A = 淡绿色

    3= 湖蓝色     B = 淡浅绿色

    4= 红色       C = 淡红色

    5= 紫色       D = 淡紫色

    6= 黄色       E = 淡黄色

    7= 白色       F = 亮白色

如果没有给定任何参数，该命令会将颜色还原到 CMD.EXE 启动时

的颜色。这个值来自当前控制台窗口、/T 开关或

DefaultColor 注册表值。

如果用相同的前景和背景颜色来执行 COLOR 命令，COLOR 命令

会将 ERRORLEVEL 设置为 1。

例如: "COLOR fc" 在亮白色上产生亮红色

7、mode 配置系统设备

配置系统设备。

串行口:　　　     MODE COMm[:] [BAUD=b][PARITY=p] [DATA=d] [STOP=s]

                                [to=on|off][xon=on|off] [odsr=on|off]

                                [octs=on|off][dtr=on|off|hs]

                               [rts=on|off|hs|tg][idsr=on|off]

设备状态:             MODE [device][/STATUS]

打印重定向:　　 MODE LPTn[:]=COMm[:]

选定代码页:　　 MODE CON[:] CP SELECT=yyy

代码页状态:　　 MODE CON[:] CP [/STATUS]

显示模式:　　     MODE CON[:] [COLS=c][LINES=n]

击键率:　             MODE CON[:][RATE=r DELAY=d]

例：

mode con cols=113 lines=15 & color 9f

此命令设置DOS窗口大小：15行，113列

8、GOTO 和 :

GOTO会点编程的朋友就会知道这是跳转的意思。

在批处理中允许以“:XXX”来构建一个标号，然后用GOTO XXX跳转到标号:XXX处，然后执行标号后的命令。

例：

if {%1}=={} goto noparms

if "%2"=="" gotonoparms

标签的名字可以随便起，但是最好是有意义的字符串啦，前加个冒号用来表示这个字符串是标签，goto命令就是根据这个冒号（:）来寻找下一步跳到到那里。最好有一些说明这样你别人看起来才会理解你的意图啊。

例：

@echo off

:start

set /a var+=1

echo %var%

if %var% leq 3 GOTO start

pause

运行显示：

1

2

3

4

10、start 命令

批处理中调用外部程序的命令（该外部程序在新窗口中运行，批处理程序继续往下执行，不理会外部程序的运行状况），如果直接运行外部程序则必须等外部程序完成后才继续执行剩下的指令

例：start explorer d:\

调用图形界面打开D盘

11、assoc 和 ftype

文件关联

assoc 设置'文件扩展名'关联，关联到'文件类型'

ftype 设置'文件类型'关联，关联到'执行程序和参数'

当你双击一个.txt文件时，windows并不是根据.txt直接判断用 notepad.exe 打开

而是先判断.txt属于 txtfile '文件类型'

再调用 txtfile 关联的命令行 txtfile=%SystemRoot%\system32\NOTEPAD.EXE %1

可以在"文件夹选项"→"文件类型"里修改这2种关联

assoc           #显示所有'文件扩展名'关联

assoc .txt      #显示.txt代表的'文件类型'，结果显示.txt=txtfile

assoc .doc      #显示.doc代表的'文件类型'，结果显示.doc=Word.Document.8

assoc .exe      #显示.exe代表的'文件类型'，结果显示.exe=exefile

ftype           #显示所有'文件类型'关联

ftype exefile   #显示exefile类型关联的命令行，结果显示exefile="%1" %\*

assoc .txt=Word.Document.8

设置.txt为word类型的文档，可以看到.txt文件的图标都变了

assoc .txt=txtfile

恢复.txt的正确关联

ftype exefile="%1" %\*

恢复 exefile 的正确关联

如果该关联已经被破坏，可以运行 command.com ，再输入这条命令

12、pushd 和 popd

切换当前目录

@echo off

c: & cd\ & md mp3       #在 C:\ 建立 mp3 文件夹

md d:\mp4               #在 D:\ 建立 mp4 文件夹

cd /d d:\mp4            #更改当前目录为 d:\mp4

pushd c:\mp3            #保存当前目录，并切换当前目录为c:\mp3

popd                    #恢复当前目录为刚才保存的d:\mp4

一般用处不大，在当前目录名不确定时，会有点帮助。（dos编程中很有用）

13、CALL

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

CALL command

调用一条批处理命令，和直接执行命令效果一样，特殊情况下很有用，比如变量的多级嵌套，见教程后面。在批处理编程中，可以根据一定条件生成命令字符串，用call可以执行该字符串，见例子。

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

调用的其它批处理程序。filename 参数必须具有 .bat 或 .cmd 扩展名。

CALL :label arguments

调用本文件内命令段，相当于子程序。被调用的命令段以标签:label开头

以命令goto :eof结尾。

另外，批脚本文本参数参照(%0、%1、等等)已如下改变:

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

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

        %~1         - 删除引号(")，扩充 %1

        %~f1        - 将 %1 扩充到一个完全合格的路径名

        %~d1        - 仅将 %1 扩充到一个驱动器号

        %~p1        - 仅将 %1 扩充到一个路径

        %~n1        - 仅将 %1 扩充到一个文件名

        %~x1        - 仅将 %1 扩充到一个文件扩展名

        %~s1        - 扩充的路径指含有短名

        %~a1        - 将 %1 扩充到文件属性

        %~t1        - 将 %1 扩充到文件的日期/时间

        %~z1        - 将 %1 扩充到文件的大小

        %~$PATH : 1 - 查找列在 PATH 环境变量的目录，并将 %1

                       扩充到找到的第一个完全合格的名称。如果环境

                       变量名未被定义，或者没有找到文件，此组合键会

                       扩充到空字符串

    可以组合修定符来取得多重结果:

       %~dp1       - 只将 %1 扩展到驱动器号和路径

       %~nx1       - 只将 %1 扩展到文件名和扩展名

       %~dp$PATH:1 - 在列在 PATH 环境变量中的目录里查找 %1，

                      并扩展到找到的第一个文件的驱动器号和路径。

       %~ftza1     - 将 %1 扩展到类似 DIR 的输出行。

    在上面的例子中，%1 和 PATH 可以被其他有效数值替换。

%~ 语法被一个有效参数号码终止。%~ 修定符不能跟 %\*使用

注意：参数扩充时不理会参数所代表的文件是否真实存在，均以当前目录进行扩展

要理解上面的知识，下面的例子很关键。

例：

@echo off

Echo 产生一个临时文件 > tmp.txt

Rem 下行先保存当前目录，再将c:\windows设为当前目录

pushd c:\windows

Call :sub tmp.txt

Rem 下行恢复前次的当前目录

Popd

Call :sub tmp.txt

pause

Del tmp.txt

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 扩充的路径指含有短名： %~s1

Echo 扩充到文件属性： %~a1

Echo 扩充到文件的日期/时间： %~t1

Echo 扩充到文件的大小： %~z1

Echo 扩展到驱动器号和路径：%~dp1

Echo 扩展到文件名和扩展名：%~nx1

Echo 扩展到类似 DIR 的输出行：%~ftza1

Echo.

Goto :eof

例：

set aa=123456

set cmdstr=echo %aa%

call %cmdstr%

pause

本例中如果不用call，而直接运行%cmdstr%，将显示结果%aa%，而不是123456

14、shift

更改批处理文件中可替换参数的位置。

SHIFT [/n]

如果命令扩展名被启用，SHIFT 命令支持/n 命令行开关；该命令行开关告诉

命令从第 n 个参数开始移位；n 介于零和八之间。例如:

   SHIFT /2

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

15、IF

IF 条件判断语句，语法格式如下：

IF [NOT] ERRORLEVEL number command

IF [NOT] string1==string2 command

IF [NOT] EXIST filename command

下面逐一介绍，更详细的分析请看后面章节。

(1) IF [NOT] ERRORLEVEL number command

IF ERRORLEVEL这个句子必须放在某一个命令的后面，执行命令后由IF ERRORLEVEL 来判断命令的返回值。

Number的数字取值范围0~255，判断时值的排列顺序应该由大到小。返回的值大于等于指定的值时，条件成立

例：

@echo off

dir c:

rem退出代码为>=1就跳至标题1处执行，>=0就跳至标题0处执行

IF ERRORLEVEL 1 goto 1

IF ERRORLEVEL 0 goto 0

Rem 上面的两行不可交换位置，否则失败了也显示成功。

:0

echo 命令执行成功！

Rem 程序执行完毕跳至标题exit处退出

goto exit

:1

echo 命令执行失败！

Rem 程序执行完毕跳至标题exit处退出

goto exit

:exit

pause

运行显示：命令执行成功！

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

string1和string2都为字符的数据，英文内字符的大小写将看作不同，这个条件中的等于号必须是两个（绝对相等的意思）

条件相等后即执行后面的command

检测当前变量的值做出判断，为了防止字符串中含有空格，可用以下格式

if [NOT] {string1}=={string2} command

if [NOT] [string1]==[string2] command

if [NOT] "string1"=="string2"command

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

if {string1}==[string2] command

(3) IF [NOT] EXIST filename command

EXIST filename为文件或目录存在的意思

echo off

IF EXIST autoexec.bat echo 文件存在！

IF not EXIST autoexec.bat echo 文件不存在！

这个批处理大家可以放在C盘和D盘分别执行，看看效果

16、setlocal 与 变量延迟

本条内容引用[英雄出品]的批处理教程：

要想进阶，变量延迟是必过的一关！所以这一部分希望你能认真看。

为了更好的说明问题，我们先引入一个例子。

例1:

@echo off

set a=4

set a=5 & echo %a%

pause

结果：4

解说：为什么是4而不是5呢？在echo之前明明已经把变量a的值改成5了？

让我们先了解一下批处理运行命令的机制：

批处理读取命令时是按行读取的（另外例如for命令等，其后用一对圆括号闭合的所有语句也当作一行），在处理之前要完成必要的预处理工作，这其中就包括对该行命令中的变量赋值。我们现在分析一下例1，批处理在运行到这句“set a=5 & echo %a%”之前，先把这一句整句读取并做了预处理——对变量a赋了值，那么%a%当然就是4了！（没有为什么，批处理就是这样做的。）

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

那么如何开启变量延迟呢？变量延迟又需要注意什么呢？举个例子说明一下：

例2:

@echo off

setlocal enabledelayedexpansion

set a=4

set a=5 & echo !a!

pause

结果：5

解说：启动了变量延迟，得到了正确答案。变量延迟的启动语句是“setlocal enabledelayedexpansion”，并且变量要用一对叹号“!!”括起来（注意要用英文的叹号），否则就没有变量延迟的效果。

分析一下例2，首先“setlocal enabledelayedexpansion”开启变量延迟，然后“set a=4”先给变量a赋值为

4，“set a=5 & echo!a!”这句是给变量a赋值为5并输出（由于启动了变量延迟，所以批处理能够感知到动态变化，即不是先给该行变量赋值，而是在运行过程中给变量赋值，因此此时a的值就是5了）。

再举一个例子巩固一下。

例3:

@echo off

setlocal enabledelayedexpansion

for /l %%i in (1,1,5) do (

set a=%%i

echo !a!

)

pause

结果：

1

2

3

4

5

解说：本例开启了变量延迟并用“!!”将变量扩起来，因此得到我们预期的结果。如果不用变量延迟会出现什

么结果呢？结果是这样的：

ECHO 处于关闭状态。

ECHO 处于关闭状态。

ECHO 处于关闭状态。

ECHO 处于关闭状态。

ECHO 处于关闭状态。

即没有感知到for语句中的动态变化。

提示：在没有开启变量延迟的情况下，某条命令行中的变量改变，必须到下一条命令才能体现。这一点也可以加以利用，看例子。

例：交换两个变量的值，且不用中间变量

@echo off

::目的：交换两个变量的值，但是不使用临时变量

::Code by JM 2007-1-24[email=CMD@XP]CMD@XP[/email]

::出处：http://www.cn-dos[**.NET**](http://lib.csdn.net/base/dotnet)/forum/viewthread.[**PHP**](http://lib.csdn.net/base/php)?tid=27078

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

17、ATTRIB  显示或更改文件属性

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]

     指定要处理的文件属性。

 /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。

例：

md autorun

attrib +a +s +h autorun

上面的命令将建立文件夹autorun，然后将其设为存档、系统、隐藏属性

第二节常用特殊符号

    1、@  命令行回显屏蔽符

    2、%  批处理变量引导符

    3、>   重定向符

    4、>>  重定向符

    5、<、>&、<& 重定向符

    6、|  命令管道符

    7、^  转义字符

    8、&  组合命令

    9、&& 组合命令

   10、||  组合命令

   11、""  字符串界定符

   12、, 逗号

   13、; 分号

   14、() 括号

   15、! 感叹号

   16、批处理中可能会见到的其它特殊标记符: （略）

       CR(0D) 命令行结束符

       Escape(1B) ANSI转义字符引导符

       Space(20) 常用的参数界定符

       Tab(09) ; = 不常用的参数界定符

       + COPY命令文件连接符

       \* ? 文件通配符

       / 参数开关引导符

       : 批处理标签引导符

废话少说，开讲了

1、@  命令行回显屏蔽符

这个字符在批处理中的意思是关闭当前行的回显。我们从前几课知道

ECHO OFF可以关闭掉整个批处理命令的回显，但不能关掉ECHOOFF这个命令，现在我们在ECHO OFF这个命令前加个@，就可以达到所有命令均不回显的要求

2、%  批处理变量引导符

这个百分号严格来说是算不上命令的，它只是批处理中的参数而已（多个%一起使用的情况除外，以后还将详细介绍）。

引用变量用%var%，调用程序外部参数用%1至%9等等

%0 %1  %2  %3 %4  %5  %6 %7  %8  %9  %\*为命令行传递给批处理的参数

%0 批处理文件本身，包括完整的路径和扩展名

%1 第一个参数

%9 第九个参数

%\* 从第一个参数开始的所有参数

参数%0具有特殊的功能，可以调用批处理自身，以达到批处理本身循环的目的，也可以复制文件自身等等。

例：最简单的复制文件自身的方法

copy %0 d:\wind.bat

小技巧：添加行内注释

%注释内容%（可以用作行内注释，不能出现重定向符号和管道符号）

为什么这样呢？此时“注释内容”其实被当作变量，其值是空的，故只起注释作用，不过这种用法容易出现语法错误，一般不用。

3、>   重定向符

输出重定向命令

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

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

比如：

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

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

4、>>  重定向符

输出重定向命令

这个符号的作用和>有点类似，但他们的区别是>>是传递并在文件的末尾追加，而>是覆盖

用法同上

同样拿1.txt做例子

使用命令：

echo hello > 1.txt

echo world >>1.txt

这时候1.txt 内容如下:

hello

world

5、<、>&、<& 重定向符

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

<，输入重定向命令，从文件中读入命令输入，而不是从键盘中读入。

@echo off

echo 2005-05-01>temp.txt

date <temp.txt

del temp.txt

这样就可以不等待输入直接修改当前日期

>&，将一个句柄的输出写入到另一个句柄的输入中。

<&，刚好和>&相反，从一个句柄读取输入并将其写入到另一个句柄输出中。

常用句柄：0、1、2，未定义句柄：3—9

1>nul 表示禁止输出正确的信息

2>nul 表示禁止输出错误信息。

其中的1与2都是代表某个数据流输入输出的地址（NT CMD 称之为句柄，MSDOS称之为设备）。

句柄0：标准输入stdin，键盘输入

句柄1：标准输出stdout，输出到命令提示符窗口（console，代码为CON）

句柄2：标准错误stderr，输出到命令提示符窗口（console，代码为CON）

其中的stdin可被<重定向，stdout可被>、>>重定向。

我们已经知道读取文本中的内容可以用for命令，但如果只需要读取第一行用for命令就有点麻烦。简单的办法如下:

@echo off

set /p str=<%0

echo %str%

pause

运行显示批处理文件自身的第一行：@echo off

6、|  命令管道符

格式：第一条命令 | 第二条命令 [| 第三条命令...]

将第一条命令的结果作为第二条命令的参数来使用，记得在unix中这种方式很常见。

例如：

dir c:\|find "txt"

以上命令是：查找C：\所有，并发现TXT字符串。

FIND的功能请用 FIND /? 自行查看

在不使format的自动格式化参数时，我是这样来自动格式化A盘的

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

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

从而达到自动输入y的目的

（这条命令有危害性，[**测试**](http://lib.csdn.net/base/softwaretest)时请慎重）

7、^  转义字符

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

比如

echo test ^>1.txt

结果则是：test > 1.txt

他没有追加在1.txt里，呵呵。只是显示了出来

另外，此转义字符还可以用作续行符号。

举个简单的例子：

@echo off

echo 英雄^

是^

好^

男人

pause

不用多说，自己试一下就明白了。

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，因为每行末尾还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，从而起到了续行的作用。

8、&  组合命令

语法：第一条命令 & 第二条命令 [& 第三条命令...]

&、&&、||为组合命令，顾名思义，就是可以把多个命令组合起来当一个命令来执行。这在批处理脚本里是允许的，而且用的非常广泛。因为批处理认行不认命令数目。

这个符号允许在一行中使用2个以上不同的命令，当第一个命令执行失败了，也不影响后边的命令执行。

这里&两边的命令是顺序执行的，从前往后执行。

比如：

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

以上命令会连续显示z,y,c盘的内容，不理会该盘是否存在

9、&& 组合命令

语法：第一条命令 && 第二条命令 [&& 第三条命令...]

用这种方法可以同时执行多条命令，当碰到执行出错的命令后将不执行后面的命令，如果一直没有出错则一直执行完所有命令

这个命令和上边的类似，但区别是，第一个命令失败时，后边的命令也不会执行

dir z:\ && dir y:\ && dirc:\

10、||  组合命令

语法：第一条命令 || 第二条命令 [|| 第三条命令...]

用这种方法可以同时执行多条命令，当一条命令失败后才执行第二条命令，当碰到执行正确的命令后将不执行后面的命令，如果没有出现正确的命令则一直执行完所有命令；

提示：组合命令和重定向命令一起使用必须注意优先级

管道命令的优先级高于重定向命令，重定向命令的优先级高于组合命令

问题：把C盘和D盘的文件和文件夹列出到a.txt文件中。看例：

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没有重定向命令的优先级高！所以这句在执行时将本行分成这两部分：dir c:\和dir d:\ > a.txt，而并不是如你想的这两部分：dir c:\ && dir d:\和> a.txt。要使用组合命令&&达到题目的要求，必须得这么写：

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

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

当然这里还可以利用&命令（自己想一下道理哦）：

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

[这个也可以用 dir c:\;d:\>>a.txt 来实现]

11、""  字符串界定符

双引号允许在字符串中包含空格，进入一个特殊目录可以用如下方法

cd "program files"

cd progra~1

cd pro\*

以上三种方法都可以进入program files这个目录

12、, 逗号

逗号相当于空格，在某些情况下“,”可以用来当做空格使

比如

dir,c:\

13、; 分号

分号，当命令相同时，可以将不同目标用；来隔离，但执行效果不变，如执行过程中发生错误，则只返回错误报告，但程序仍会执行。（有人说不会继续执行，其实测试一下就知道了）

比如：

dir c:\;d:\;e:\;z:\

以上命令相当于

dir c:\

dir d:\

dir e:\

dir f:\

如果其中z盘不存在，运行显示：系统找不到指定的路径。然后终止命令的执行。

例：dir c:\;d:\;e:\1.txt

以上命令相当于

dir c:\

dir d:\

dir e:\1.txt

其中文件e:\1.txt不存在，但e盘存在，有错误提示，但命令仍会执行。

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

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

14、() 括号

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

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

例：

命令：echo 1 & echo 2 & echo 3

可以写成：

(

echo 1

echo 2

echo 3

)

上面两种写法效果一样，这两种写法都被视为是一条命令行。

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

15、! 感叹号

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

第二章 DOS循环：for命令详解

讲FOR之前呢,咋先告诉各位新手朋友,如果你有什么命令不懂,直接在CMD下面输入:

name /? 这样的格式来看系统给出的帮助文件,比如for /? 就会把FOR命令的帮助全部显示出来!当然许多菜鸟都看不懂....所以才会有那么多批处理文章!!!!俺也照顾菜鸟,把FOR命令用我自己的方式说明下!

正式开始:

一、基本格式

FOR %%variable IN (set) DO command[command-parameters]

 %%variable  指定一个单一字母表示可替换的参数。

 (set)      指定一个或一组文件。可以使用通配符。

 command    指定对每个文件执行的命令。

 command-parameters

            为特定命令指定参数或命令行开关。

参数:FOR有4个参数 /d   /l   /r  /f   他们的作用我在下面用例子解释

现在开始讲每个参数的意思

二、参数 /d

FOR /D %%variable IN (set) DO command[command-parameters]

    如果集中包含通配符，则指定与目录名匹配，而不与文件

    名匹配。

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

这个参数主要用于目录搜索,不会搜索文件,看这样的例子

@echo off

for /d %%i in (c:\\*) do echo %%i

pause

运行会把C盘根目录下的全部目录名字打印出来,而文件名字一个也不显示!

在来一个,比如我们要把当前路径下文件夹的名字只有1-3个字母的打出来

@echo off

for /d %%i in (???) do echo %%i

pause

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

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

知道作用了,给大家个思考题目!

@echo off

for /d %%i in (window?) do echo %%i

pause

保存到C盘下执行,会显示什么呢?自己看吧! 显示：windows

/D参数只能显示当前目录下的目录名字,这个大家要注意!

三、参数 /R

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

    检查以[drive:]path 为根的目录树，指向每个目录中的

   FOR 语句。如果在 /R 后没有指定目录，则使用当前

目录。如果集仅为一个单点(.)字符，则枚举该目录树。

递归

上面我们知道,/D只能显示当前路径下的目录名字,那么现在这个/R也是和目录有关,他能干嘛呢?放心他比/D强大多了!

他可以把当前或者你指定路径下的文件名字全部读取,注意是文件名字,有什么用看例子!

请注意2点：

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

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

例：

@echo off

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

pause

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

例：

@echo off

for /r %%i in (\*.exe) do @echo %%i

pause

参数不一样了吧!这个命令前面没加那个C:\也就是搜索路径,这样他就会以当前目录为搜索路径,比如你这个BAT你把他放在d:\test目录下执行,那么他就会把D:\test目录和他下面的子目录的全部EXE文件列出来!!!

例：

@echo off

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

pause

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

@echo off

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

pause

用这条命令搜索文件真不错。。。。。。

这个参数大家应该理解了吧!还是满好玩的命令!

四、参数 /L

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

    该集表示以增量形式从开始到结束的一个数字序列。

    因此，(1,1,5) 将产生序列 1 2 3 45，(5,-1,1) 将产生

    序列 (5 4 3 21)。

使用迭代变量设置起始值 (Start#)，然后逐步执行一组范围的值，直到该值超过所设置的终止值 (End#)。/L 将通过对 Start# 与 End# 进行比较来执行迭代变量。如果 Start# 小于 End#，就会执行该命令。如果迭代变量超过 End#，则命令解释程序退出此循环。还可以使用负的 Step# 以递减数值的方式逐步执行此范围内的值。例如，(1,1,5) 生成序列 1 2 3 4 5，而 (5,-1,1) 则生成序列 (5 4 3 2 1)。语法是：

看着这说明有点晕吧!咱们看例子就不晕了!

@echo off

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

pause

保存执行看效果,他会打印从1 2 3 4 5  这样5个数字

(1,1,5)这个参数也就是表示从1开始每次加1直到5终止!

等会晕,就打印个数字有P用...好的满足大家,看这个例子

@echo off

for /l %%i in (1,1,5) do start cmd

pause

执行后是不是吓了一跳,怎么多了5个CMD窗口,呵呵!如果把那个 (1,1,5)改成 (1,1,65535)会有什么结果,我先告诉大家,会打开65535个CMD窗口....这么多你不死机算你强!

当然我们也可以把那个start cmd改成md %%i 这样就会建立指定个目录了!!!名字为1-65535

看完这个被我赋予破坏性质的参数后,我们来看最后一个参数

五、参数 /F

\迭代及文件解析

使用文件解析来处理命令输出、字符串及文件内容。使用迭代变量定义要检查的内容或字符串，并使用各种options选项进一步修改解析方式。使用options令牌选项指定哪些令牌应该作为迭代变量传递。请注意：在没有使用令牌选项时，/F 将只检查第一个令牌。

文件解析过程包括读取输出、字符串或文件内容，将其分成独立的文本行以及再将每行解析成零个或更多个令牌。然后通过设置为令牌的迭代变量值，调用 for 循环。默认情况下，/F 传递每个文件每一行的第一个空白分隔符号。跳过空行。

详细的帮助格式为：

FOR /F ["options"] %%variable IN(file-set) DO command [command-parameters]

FOR /F ["options"] %%variable IN("string") DO command [command-parameters]

FOR /F ["options"] %%variable IN('command') DO command [command-parameters]

    带引号的字符串"options"包括一个或多个

    指定不同解析选项的关键字。这些关键字为:

       eol=c           - 指一个行注释字符的结尾(就一个)

       skip=n          - 指在文件开始时忽略的行数。

       delims=xxx      - 指分隔符集。这个替换了空格和跳格键的

                          默认分隔符集。

       tokens=x,y,m-n  - 指每行的哪一个符号被传递到每个迭代

                          的 for 本身。这会导致额外变量名称的分配。m-n

                          格式为一个范围。通过nth 符号指定 mth。如果

                          符号字符串中的最后一个字符星号，

                          那么额外的变量将在最后一个符号解析之后

                          分配并接受行的保留文本。经测试，该参数最多

                          只能区分31个字段。

       usebackq        - 使用后引号（键盘上数字1左面的那个键`）。

                        未使用参数usebackq时：file-set表示文件，但不能含有空格

                            双引号表示字符串，即"string"

                            单引号表示执行命令，即'command'

                          使用参数usebackq时：file-set和"file-set"都表示文件

                          当文件路径或名称中有空格时，就可以用双引号括起来

                            单引号表示字符串，即'string'

                            后引号表示命令执行，即`command`

以上是用for /?命令获得的帮助信息，直接复制过来的。

晕惨了!我这就举个例子帮助大家来理解这些参数!

For命令例1：\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@echo off

rem 首先建立临时文件test.txt

echo ;注释行,这是临时文件,用完删除>test.txt

echo 11段 12段 13段 14段 15段 16段>>test.txt

echo 21段,22段,23段,24段,25段,26段>>test.txt

echo 31段-32段-33段-34段-35段-36段>>test.txt

FOR /F "eol=; tokens=1,3\* delims=,-" %%i in (test.txt) do echo %%i %%j %%k

Pause

Del test.txt

运行显示结果：

11段 13段 14段 15段 16段

21段 23段 24段,25段,26段

31段 33段 34段-35段-36段

请按任意键继续. . .

为什么会这样?我来解释：

eol=;          分号开头的行为注释行

tokens=1,3\*    将每行第1段,第3段和剩余字段分别赋予变量%%i，%%j，%%k

delims=,-     （减号后有一空格）以逗号减号和空格为分隔符，空格必须放在最后

For命令例2：\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@echo off

FOR /F "eol= delims=" %%i in(test.txt) do echo %%i

Pause

运行将显示test.txt全部内容，包括注释行，不解释了哈。

For命令例3：\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

另外/F参数还可以以输出命令的结果看这个例子

@echo off

FOR /F "delims=" %%i in ('netuser') do @echo %%i

pause

这样你本机全部帐号名字就出来了把扩号内的内容用两个单引号引起来就表示那个当命令执行,FOR会返回命令的每行结果,加那个"delims=" 是为了让我空格的行能整行显示出来,不加就只显示空格左边一列!

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

所谓文武之道，一张一弛，现休息一下。

第三章 FOR命令中的变量

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

先把FOR的变量全部列出来:

    ~I          - 删除任何引号(")，扩展 %I

    %~fI        - 将 %I 扩展到一个完全合格的路径名

    %~dI        - 仅将 %I 扩展到一个驱动器号

    %~pI        - 仅将 %I 扩展到一个路径

    %~nI        - 仅将 %I 扩展到一个文件名

    %~xI        - 仅将 %I 扩展到一个文件扩展名

    %~sI        - 扩展的路径只含有短名

    %~aI        - 将 %I 扩展到文件的文件属性

    %~tI        - 将 %I 扩展到文件的日期/时间

    %~zI        - 将 %I 扩展到文件的大小

    %~$PATH:I   - 查找列在路径环境变量的目录，并将 %I 扩展

                   到找到的第一个完全合格的名称。如果环境变量名

                   未被定义，或者没有找到文件，此组合键会扩展到

                   空字符串

我们可以看到每行都有一个大写字母"I",这个I其实就是我们在FOR带入的变量,我们FOR语句代入的变量名是什么,这里就写什么.

比如:FOR /F  %%z IN ('set') DO@echo %%z

这里我们代入的变量名是z那么我们就要把那个I改成z,例如%~fI改为%~fz

至于前面的%~p这样的内容就是语法了!

好开始讲解:

一、 ~I          - 删除任何引号(")，扩展 %I

这个变量的作用就如他的说明,删除引号!

我们来看这个例子:

首先建立临时文件temp.txt，内容如下

"1111

"2222"

3333"

"4444"44

"55"55"55

可建立个BAT文件代码如下:

@echo off

echo ^"1111>temp.txt

echo "2222">>temp.txt

echo 3333^">>temp.txt

echo "4444"44>>temp.txt

echo^"55"55"55>>temp.txt

rem 上面建立临时文件，注意不成对的引号要加转义字符^，重定向符号前不要留空格

FOR /F "delims=" %%i IN(temp.txt) DO echo  %%~i

pause

del temp.txt

执行后,我们看CMD的回显如下:

1111           #字符串前的引号被删除了

2222           #字符串首尾的引号都被删除了

3333"          #字符串前无引号，后面的引号保留

4444"44        #字符串前面的引号删除了，而中间的引号保留

55"55"55       #字符串前面的引号删除了，而中间的引号保留

请按任意键继续. . .

和之前temp.txt中的内容对比一下,我们会发现第1、2、5行的引号都消失了,这就是删除引号~i的作用了!

删除引号规则如下(BAT兄补充!)

1、若字符串首尾同时存在引号，则删除首尾的引号；

2、若字符串尾不存在引号，则删除字符串首的引号；

3、如果字符串中间存在引号，或者只在尾部存在引号，则不删除。

龙卷风补充：无头不删，有头连尾删。

二、 %~fI        - 将 %I 扩展到一个完全合格的路径名

看例子:

把代码保存放在随便哪个地方,我这里就放桌面吧.

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

pause

执行后显示内容如下

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

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

当我把代码中的 %%~fi直接改成%%i

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

pause

执行后就会显示以下内容：

test.bat

test.vbs

通过对比,我们很容易就看出没有路径了,这就是"将 %I 扩展到一个完全合格的路径名"的作用

也就是如果%i变量的内容是一个文件名的话,他就会把这个文件所在的绝对路径打印出来,而不只单单打印一个文件名,自己动手动实验下就知道了!

三、 %~dI        - 仅将 %I 扩展到一个驱动器号

看例子:

代码如下,我还是放到桌面执行!

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

pause

执行后我CMD里显示如下

C:

C:

我桌面就两个文件test.bat,test.vbs,%%~di作用是,如果变量%%i的内容是一个文件或者目录名,他就会把他这文件

或者目录所在的盘符号打印出来!

四、 %~pI        - 仅将 %I 扩展到一个路径

这个用法和上面一样,他只打印路径不打印文件名字

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

pause

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

五、 %~nI        - 仅将 %I 扩展到一个文件名

只打印文件名字

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

pause

六、 %~xI        - 仅将 %I 扩展到一个文件扩展名

只打印文件的扩展名

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

pause

七、 %~sI        - 扩展的路径只含有短名

打印绝对短文件名

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

pause

八、 %~aI        - 将 %I 扩展到文件的文件属性

打印文件的属性

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

pause

九、 %~tI        - 将 %I 扩展到文件的日期/时间

打印文件建立的日期

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

pause

十、 %~zI        - 将 %I 扩展到文件的大小

打印文件的大小

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

pause

上面例子中的"delims=="可以改为"delims="，即不要分隔符

十一、 %~$PATH:I - 查找列在路径环境变量的目录，并将 %I 扩展

                   到找到的第一个完全合格的名称。如果环境变量名

                   未被定义，或者没有找到文件，此组合键会扩展到

                   空字符串

这是最后一个,和上面那些都不一样,我单独说说!

然后在把这些代码保存为批处理,放在桌面。

@echo off

FOR /F "delims=" %%i IN(“notepad.exe”) DO echo  %%~$PATH:i

pause

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

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

好了,FOR的的变量就介绍到这了!

第四章批处理中的变量

批处理中的变量,我把他分为两类,分别为"系统变量"和"自定义变量"

我们现在来详解这两个变量!

一、系统变量

他们的值由系统将其根据事先定义的条件自动赋值,也就是这些变量系统已经给他们定义了值,

不需要我们来给他赋值,我们只需要调用而以!  我把他们全部列出来!

%ALLUSERSPROFILE% 本地 返回“所有用户”配置文件的位置。

%APPDATA% 本地 返回默认情况下应用程序存储数据的位置。

%CD% 本地 返回当前目录字符串。

%CMDCMDLINE% 本地 返回用来启动当前的 Cmd.exe 的准确命令行。

%CMDEXTVERSION% 系统 返回当前的“命令处理程序扩展”的版本号。

%COMPUTERNAME%  系统 返回计算机的名称。

%COMSPEC% 系统返回命令行解释器可执行程序的准确路径。

%DATE% 系统返回当前日期。使用与 date /t 命令相同的格式。由 Cmd.exe 生成。有关

date 命令的详细信息，请参阅 Date。

%ERRORLEVEL%  系统 返回上一条命令的错误代码。通常用非零值表示错误。

%HOMEDRIVE% 系统返回连接到用户主目录的本地工作站驱动器号。基于主目录值而设置。用

户主目录是在“本地用户和组”中指定的。

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

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

在“本地用户和组”中指定的。

%LOGONSERVER%  本地 返回验证当前登录会话的域控制器的名称。

%NUMBER\_OF\_PROCESSORS%  系统 指定安装在计算机上的处理器的数目。

%OS% 系统返回[**操作系统**](http://lib.csdn.net/base/operatingsystem)名称。Windows 2000 显示其操作系统为 Windows\_NT。

%PATH% 系统 指定可执行文件的搜索路径。

%PATHEXT% 系统 返回操作系统认为可执行的文件扩展名的列表。

%PROCESSOR\_ARCHITECTURE%  系统 返回处理器的芯片体系结构。值：x86 或 IA64 基于

Itanium

%PROCESSOR\_IDENTFIER% 系统 返回处理器说明。

%PROCESSOR\_LEVEL%  系统 返回计算机上安装的处理器的型号。

%PROCESSOR\_REVISION% 系统 返回处理器的版本号。

%PROMPT% 本地 返回当前解释程序的命令提示符设置。由Cmd.exe 生成。

%RANDOM% 系统 返回 0 到 32767 之间的任意十进制数字。由Cmd.exe 生成。

%SYSTEMDRIVE% 系统 返回包含 Windows serveroperating system 根目录（即系统根目录）

的驱动器。

%SYSTEMROOT%  系统 返回 Windows serveroperating system 根目录的位置。

%TEMP% 和 %TMP% 系统和用户 返回对当前登录用户可用的应用程序所使用的默认临时目录。

有些应用程序需要 TEMP，而其他应用程序则需要 TMP。

%TIME% 系统 返回当前时间。使用与 time /t 命令相同的格式。由Cmd.exe 生成。有关

time 命令的详细信息，请参阅 Time。

%USERDOMAIN% 本地 返回包含用户帐户的域的名称。

%USERNAME% 本地 返回当前登录的用户的名称。

%USERPROFILE% 本地 返回当前用户的配置文件的位置。

%WINDIR% 系统 返回操作系统目录的位置。

这么多系统变量,我们如何知道他的值是什么呢?

在CMD里输入  echo %WINDIR%

这样就能显示一个变量的值了!

举个实际例子,比如我们要复制文件到当前帐号的启动目录里就可以这样

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

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

另外还有一些系统变量,他们是代表一个意思,或者一个操作!

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

%0 这个有点特殊,有几层意思,先讲%1-%9的意思.

%1 返回批处理的第一个参数

%2 返回批处理的第二个参数

%3-%9依此推类

反回批处理参数?到底怎么个返回法?

我们看这个例子,把下面的代码保存为test.BAT然后放到C盘下

@echo off

echo %1 %2 %3 %4

echo %1

echo %2

echo %3

echo %4

进入CMD,输入cd c:\

然后输入 test.bat 我是第一个参数我是第二个参数  我是第三个参数  我是第四个参数

注意中间的空格,我们会看到这样的结果:

我是第一个参数我是第二个参数 我是第三个参数 我是第四个参数

我是第一个参数

我是第二个参数

我是第三个参数

我是第四个参数

对比下代码,%1就是”我是第一个参数”  %2就是”我是第二个参数”

怎么样理解了吧!

这些%1和%9可以让批处理也能带参数运行,大大提高批处理功能!

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

例子

@echo off

echo %\*

同样保存为test.bat 放到C盘

进入CMD,输入cd c:\

然后输入 test.bat 我是第一个参数我是第二个参数  我是第三个参数  我是第四个参数

可以看到他一次把全部参数都显示出来了

好现在开始讲那个比较特殊的%0

%0  这个不是返回参数的值了,他有两层意思!

第一层意思:返回批处理所在绝对路径

例子:

@echo off

echo %0

pause

保存为test.BAT放在桌面运行,会显示如下结果

"C:\Documents andSettings\Administrator\桌面\test.bat"

他把当前批处理执行的所在路经打印出来了,这就是返回批处理所在绝对路径的意思

第二层意思:无限循环执行BAT

例子:

@echo off

net user

%0

保存为BAT执行,他就会无限循环执行net user这条命令,直到你手动停止.

龙卷风补充：其实%0就是第一参数%1前面那个参数，当然就是批处理文件名（包括路径）。

以上就是批处理中的一些系统变量,另外还有一些变量,他们也表示一些功能,

FOR命令中的那些就是,FOR变量已经说过,就不讲了.

二、自定义变量

故名思意,自定义变量就是由我们来给他赋予值的变量

要使用自定义变量就得使用set命令了,看例子.

@echo off

set var=我是值

echo %var%

pause

保存为BAT执行,我们会看到CMD里返回一个  "我是值"

var为变量名,=号右变的是要给变量的值

这就是最简单的一种设置变量的方法了

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

例子:

@echo off

set /p var=请输入变量的值

echo %var%

pause

var变量名  =号右边的是提示语,不是变量的值

变量的值由我们运行后自己用键盘输入!

一、交互界面设计

没啥说的，看看高手设计的菜单界面吧：

@echo off

cls

title 终极多功能修复

:menu

cls

color 0A

echo.

echo                 ==============================

echo                 请选择要进行的操作，然后按回车

echo                 ==============================

echo.

echo              1.网络修复及上网相关设置,修复IE,自定义屏蔽网站

echo.

echo              2.病毒专杀工具，端口关闭工具,关闭自动播放

echo.

echo             3.清除所有多余的自启动项目，修复系统错误

echo.

echo              4.清理系统垃圾,提高启动速度

echo.

echo              Q.退出

echo.

echo.

:cho

set choice=

set /p choice=          请选择:

IF NOT "%choice%"==""SET choice=%choice:~0,1%

if /i "%choice%"=="1"goto ip

if /i "%choice%"=="2"goto setsave

if /i "%choice%"=="3"goto kaiji

if /i "%choice%"=="4"goto clean

if /i "%choice%"=="Q"goto endd

echo 选择无效，请重新输入

echo.

goto cho

只要学完本教程前面的章节，上面的程序应该能看懂了。

二、if…else…条件语句

前面已经谈到，DOS条件语句主要有以下形式

IF [NOT] ERRORLEVEL number command

IF [NOT] string1==string2 command

IF [NOT] EXIST filename command

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

增强用法中加上/I就不区分大小写了!

增强用法中还有一些用来判断数字的符号：

EQU - 等于

NEQ - 不等于

LSS - 小于

LEQ - 小于或等于

GTR - 大于

GEQ - 大于或等于

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

例如:

   IF EXIST filename (

       del filename

    )ELSE (

       echo filename missing

    )

也可写成：

if exist filename (del filename) else (echofilename missing)

但这种写法不适合命令太多或嵌套命令的使用。

三、循环语句

1、指定次数循环

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

组合命令：

FOR /L %variable IN (start,step,end) DO (

Command1

Command2

……

)

2、对某集合执行循环语句。

FOR %%variable IN (set) DO command[command-parameters]

 %%variable  指定一个单一字母可替换的参数。

 (set)      指定一个或一组文件。可以使用通配符。

 command   对每个文件执行的命令，可用小括号使用多条命令组合。

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

    检查以[drive:]path 为根的目录树，指向每个目录中的

   FOR 语句。如果在 /R 后没有指定目录，则使用当前

目录。如果集仅为一个单点(.)字符，则枚举该目录树。

同前面一样，command可以用括号来组合：

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

Command1

Command2

……

commandn

)

3、条件循环

    上面的循环结构是用for命令来实现的，for命令循环有一个缺点，就是整个循环被当作一条命令语句，涉及到变量延迟的问题。

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

例：

@echo off

set var=0

rem \*\*\*\*\*\*\*\*\*\*\*\*循环开始了

:continue

set /a var+=1

echo 第%var%次循环

if %var% lss 100 goto continue

rem \*\*\*\*\*\*\*\*\*\*\*\*循环结束了

echo 循环执行完毕

pause

例：

@echo off

set var=100

rem \*\*\*\*\*\*\*\*\*\*\*\*循环开始了

:continue

echo 第%var%次循环

set /a var-=1

if %var% gtr 0 goto continue

rem \*\*\*\*\*\*\*\*\*\*\*\*循环结束了

echo 循环执行完毕

pause

四、子程序

在批处理程序中可以调用外部可运行程序，比如exe程序，也可调用其他批处理程序，这些也可以看作子程序，但是不够方便，如果被调用的程序很多，就显得不够简明了，很繁琐。

   在windowsXP中，批处理可以调用本程序中的一个程序段，相当于子程序，这些子程序一般放在主程序后面。

子程序调用格式：

 CALL :label arguments

子程序语法：

 :label

 command1

 command2

 ......

 commandn

 goto :eof

在子程序段中，参数%0指标签:label

子过程一般放在最后，并且注意在主程序最后要加上exit或跳转语句，避免错误的进入子过程。

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。

传至子程序的参数在call语句中指定，在子程序中用%1、%2至%9的形式调用，而子程序返回主程序的数据只需在调用结束后直接引用就可以了，当然也可以指定返回变量，请看下面的例子。

子程序例1：

@echo off

call :sub return 你好

echo 子程序返回值：%return%

pause

:sub

set %1=%2

goto :eof

运行结果：你好

子程序例2：设计一个求多个整数相加的子程序

@echo off

set sum=0

call :sub sum 10 20 35

echo 数据求和结果：%sum%

pause

:sub

rem 参数1为返回变量名称

set /a %1=%1+%2

shift /2

if not "%2"=="" gotosub

goto :eof

运行结果：65

五、用ftp命令实现自动下载

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

其实可以将ftp命令组合保存为一个文本文件，然后用以下命令调用即可。

ftp -n -s:[[drive:]path]filename

上面的filename为ftp命令文件，包括登录IP地址，用户名、密码、操作命令等

例：

open 90.52.8.3   ＃打开ip

user iware       ＃用户为iware

password8848    ＃密码

bin             ＃二进制传输模式

prompt

cd tmp1         ＃切换至iware用户下的tmp1目录

pwd

lcd d:\download   ＃本地目录

mget \*          ＃下载tmp1目录下的所有文件

bye             ＃退出ftp

六、用7-ZIP实现命令行压缩和解压功能

语法格式：（详细情况见7-zip帮助文件，看得头晕可以跳过，用到再学）

7z <command> [<switch>...]<base\_archive\_name> [<arguments>...]

7z.exe的每个命令都有不同的参数<switch>,请看帮助文件

<base\_archive\_name>为压缩包名称

<arguments>为文件名称，支持通配符或文件列表

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

a： Adds files toarchive. 添加至压缩包

a命令可用参数：

  -i(Include)

  -m(Method)

  -p(Set Password)

  -r(Recurse)

 -sfx (create SFX)

  -si(use StdIn)

  -so(use StdOut)

 -ssw (Compress shared files)

  -t(Type of archive)

  -u(Update)

  -v(Volumes)

  -w(Working Dir)

  -x(Exclude)

b： Benchmark

d： Deletes files fromarchive. 从压缩包中删除文件

d命令可用参数：

  -i(Include)

  -m(Method)

  -p(Set Password)

  -r(Recurse)

  -u(Update)

  -w(Working Dir)

  -x(Exclude)

e： Extract解压文件至当前目录或指定目录

e命令可用参数：

  -ai(Include archives)

  -an(Disable parsing of archive\_name)

  -ao(Overwrite mode)

  -ax(Exclude archives)

  -i(Include)

  -o(Set Output Directory)

  -p(Set Password)

  -r(Recurse)

  -so(use StdOut)

  -x(Exclude)

  -y(Assume Yes on all queries)

l： Lists contents ofarchive.

t： Test

u： Update

x： eXtract with fullpaths用文件的完整路径解压至当前目录或指定目录

x命令可用参数：

  -ai(Include archives)

  -an(Disable parsing of archive\_name)

  -ao(Overwrite mode)

  -ax(Exclude archives)

  -i(Include)

  -o(Set Output Directory)

  -p(Set Password)

  -r(Recurse)

  -so(use StdOut)

  -x(Exclude)

  -y(Assume Yes on all queries)

七、调用VB\*\*\*\*\*\*程序

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

用法：C\*\*\*\*\*\* 脚本名称 [脚本选项...] [脚本参数...]

选项：

//B        批模式：不显示脚本错误及提示信息

//D        启用 Active Debugging

//E:engine 使用执行脚本的引擎

//H:C\*\*\*\*\*\* 将默认的脚本宿主改为C\*\*\*\*\*\*.exe

//H:W\*\*\*\*\*\* 将默认的脚本宿主改为W\*\*\*\*\*\*.exe （默认）

//I        交互模式（默认，与 //B 相对)

//Job:xxxx 执行一个 WSF 工作

//Logo     显示徽标（默认）

//Nologo   不显示徽标：执行时不显示标志

//S        为该用户保存当前命令行选项

//T:nn     超时设定秒：允许脚本运行的最长时间

//X        在调试器中执行脚本

//U        用 Unicode 表示来自控制台的重定向 I/O

“脚本名称”是带有扩展名和必需的路径信息的脚本文件名称，如d:\admin\vb\*\*\*\*\*\*s\chart.vbs。

“脚本选项和参数”将传递给脚本。脚本参数前面有一个斜杠 (/)。每个参数都是可选的；但不能在未指定脚本名称的情况下指定脚本选项。如果未指定参数，则 C\*\*\*\*\*\* 将显示 C\*\*\*\*\*\* 语法和有效的宿主参数。

八、将批处理转化为可执行文件：

由于批处理文件是一种文本文件，任何人都可以对其进行随便编辑，不小心就会把里面的命令破坏掉，所以如果将其转换成.com格式的可执行文件，不仅执行效率会大大提高，而且不会破坏原来的功能，更能将优先级提到最高。Bat2Com就可以完成这个转换工作。

小知识：在DOS环境下，可执行文件的优先级由高到低依次为.com>.exe>.bat>.cmd，即如果在同一目录下存在文件名相同的这四类文件，当只键入文件名时，DOS执行的是name.com，如果需要执行其他三个文件，则必须指定文件的全名，如name.bat。

这是一个只有5.43K大小的免费绿色工具，可以运行在纯DOS或DOS窗口的命令行中，用法：Bat2Com

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

九、时间延迟

本条参考引用[英雄]教程

什么是时间延迟？顾名思义，就是执行一条命令后延迟一段时间再进行下一条命令。

延迟的应用见下节：“模拟进度条”。

1、利用ping命令延时

例：

 @echo off

 echo 延时前：%time%

 ping /n 3 127.0.0.1 >nul

 echo 延时后：%time%

 pause

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

2、利用for命令延时

例：

 @echo off

 echo 延时前：%time%

  for/l %%i in (1,1,5000) do echo %%i>nul

 echo 延时后：%time%

 pause

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

3、利用vbs延迟函数，精确度毫秒，误差1000毫秒内

例：

   @echo off

   echo %time%

   call :delay 5000

   echo %time%

   pause

   exit

   :delay

   echo W\*\*\*\*\*\*.Sleep %1>delay.vbs

   C\*\*\*\*\*\* //B delay.vbs

   del delay.vbs

   goto :eof

运行显示：

10:44:06.45

10:44:11.95

请按任意键继续. . .

上面的运行结果显示实际延时了5500毫秒，多出来的500毫秒时建立和删除临时文件所耗费的时间。误差在一秒之内。

4、仅用批处理命令实现任意时间延迟，精确度10毫秒，误差50毫秒内

仅用批处理命令就可以实现延迟操作。

例：

@echo off

set /p delay=请输入需延迟的毫秒数：

set TotalTime=0

set NowTime=%time%

::读取起始时间，时间格式为：13:01:05.95

echo 程序开始时间：%NowTime%

:delay\_continue

set /a minute1=1%NowTime:~3,2%-100

::读取起始时间的分钟数

set /asecond1=1%NowTime:~-5,2%%NowTime:~-2%0-100000

::将起始时间的秒数转为毫秒

set NowTime=%time%

set /a minute2=1%NowTime:~3,2%-100

:: 读取现在时间的分钟数

set /asecond2=1%NowTime:~-5,2%%NowTime:~-2%0-100000

::将现在时间的秒数转为毫秒

set /aTotalTime+=(%minute2%-%minute1%+60)%%60\*60000+%second2%-%second1%

if %TotalTime% lss %delay% gotodelay\_continue

echo 程序结束时间：%time%

echo 设定延迟时间：%delay%毫秒

echo 实际延迟时间：%TotalTime%毫秒

pause

运行显示：

   请输入需延迟的毫秒数：6000

   程序开始时间：15:32:16.37

   程序结束时间：15:32:22.37

   设定延迟时间：6000毫秒

   实际延迟时间：6000毫秒

   请按任意键继续. . .

实现原理：首先设定要延迟的毫秒数，然后用循环累加时间，直到累加时间大于等于延迟时间。

误差：windows系统时间只能精确到10毫秒，所以理论上有可能存在10毫秒误差。

     经测试，当延迟时间大于500毫秒时，上面的延迟程序一般不存在误差。当延迟时间小于500毫秒时，可能有几十毫秒误差，为什么？因为延迟程序本身也是有运行时间的，同时系统时间只能精确到10毫秒。

为了方便引用，可将上面的例子改为子程序调用形式：

@echo off

echo 程序开始时间：%Time%

call :delay 10

echo 实际延迟时间：%totaltime%毫秒

echo 程序结束时间：%time%

pause

exit

::-----------以下为延时子程序--------------------

:delay

@echo off

if "%1"=="" goto :eof

set DelayTime=%1

set TotalTime=0

set NowTime=%time%

::读取起始时间，时间格式为：13:01:05.95

:delay\_continue

set /a minute1=1%NowTime:~3,2%-100

set /asecond1=1%NowTime:~-5,2%%NowTime:~-2%0-100000

set NowTime=%time%

set /a minute2=1%NowTime:~3,2%-100

set /asecond2=1%NowTime:~-5,2%%NowTime:~-2%0-100000

set /aTotalTime+=(%minute2%-%minute1%+60)%%60\*60000+%second2%-%second1%

if %TotalTime% lss %DelayTime% gotodelay\_continue

goto :eof

十、模拟进度条

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

@echo off

mode con cols=113 lines=15 &color 9f

cls

echo.

echo 程序正在初始化. . .

echo.

echo ┌──────────────────────────────────────┐

set/p= ■<nul

for /L %%i in (1 1 38) do set /p a=■<nul&ping/n 1 127.0.0.1>nul

echo  100%%

echo └──────────────────────────────────────┘

pause

解说：“set /p a=■<nul”的意思是：只显示提示信息“■”且不换行，也不需手工输入任何信息，这样可以使每个“■”在同一行逐个输出。“ping /n 0 127.1>nul”是输出每个“■”的时间间隔，即每隔多少时间输出一个“■”。

十一、特殊字符的输入及应用

开始 -> 运行 -> 输入cmd -> edit -> ctrl+p（意思是允许输入特殊字符）-> 按ctrl+a将会显示笑脸图案。

（如果要继续输入特殊字符请再次按ctrl+p，然后ctrl+某个字母）

以上是特殊字符的输入方法，选自[英雄]教程，很管用的。也就是用编辑程序edit输入特殊字符，然后保存为一文本文件，再在windows下打开此文件，复制其中的特殊符号即可。

一些简单的特殊符号可以在dos命令窗口直接输入，并用重定向保存为文本文件。

例：

C:>ECHO ^G>temp.txt

“^G”是用Ctrl＋G或Alt＋007输入，输入多个^G可以产生多声鸣响。

特殊字符的应用也很有意思，这里仅举一例：退格键

退格键表示删除左边的字符，此键不能在文档中正常输入，但可以通过edit编辑程序录入并复制出来。即“”。

利用退格键，可以设计闪烁文字效果

例：文字闪烁

@echo off

:start

set/p=床前明月光<nul

::显示文字，光标停于行尾

ping -n 0 127.0.0.1>nul

::设置延迟时间

set /p a=<nul

:: 输出一些退格符将光标置于该行的最左端（退格符的数量可以自己调整）。

set /p a=                               <nul

::输出空格将之前输出的文字覆盖掉。

set /p a=<nul

::再次输出退格符将光标置于该行的最左端，这里的退格符数量一定不能比前面的空格数少。

::否则光标不能退到最左端。

goto start

例：输出唐诗一首，每行闪动多次

@echo off

setlocal enabledelayedexpansion

set str=床前明月光 疑是地上霜 举头望明月 低头思故乡

::定义字符串str

for %%i in (%str%) do (

rem 由于str中含有空格，则以空格为分隔符将str中的每一个部分依次赋给变量%%i。

       set char=%%i

       echo.

       echo.

       for /l %%j in (0,1,5) do (

                set/p=!char:~%%j,1!<nul

  rem依次取出变量char中的每一个字符，并显示。

                ping -n 0 127.0.0.1>nul

  rem设置输出每个字符的时间延迟。

       )

call :hero %%i

)

pause>nul

exit

:hero

for /l %%k in (1,1,10) do (

ping /n 0 127.0.0.1>nul

set /p a=<nul

set /p a=                               <nul

set /p a=<nul

ping /n 0 127.0.0.1>nul

set /p a=%1<nul

)

::文字闪动

goto :eof

十二、随机数（%random%）的应用技巧

%RANDOM% 系统变量 返回 0 到 32767 之间的任意十进制数字。由Cmd.exe 生成。

2的15次方等于32768，上面的0～32767实际就是15位二进制数的范围。

那么，如何获取100以内的随机数呢？很简单，将%RANDOM%按100进行求余运算即可，见例子。

例：生成5个100以内的随机数

 @echo off

 setlocal enabledelayedexpansion

  for/L %%i in (1 1 5) do (

    set /a randomNum=!random!%%100

    echo 随机数：!randomNum!

  )

 pause

运行结果：（每次运行不一样）

随机数：91

随机数：67

随机数：58

随机数：26

随机数：20

请按任意键继续. . .

求余数运算set /a randomNum=!random!%%100中的100可以是1～32768之间的任意整数。

总结：利用系统变量%random%，求余数运算%%，字符串处理等，可以实现很多随机处理。

思考题目：生成给定位数的随机密码

解答思路：将26个英文字母或10数字以及其它特殊字符组成一个字符串，随机抽取其中的若干字符。

参考答案1：（简单）

@echo off

call :randomPassword 5 pass1 pass2

echo %pass1% %pass2%

pause

exit

:randomPassword

::---------生成随机密码

::---------%1为密码长度，%2及以后为返回变量名称

::---------for命令最多只能区分31个字段

@echo off

set password\_len=%1

if not defined password\_len goto :eof

if %password\_len% lss 1 goto :eof

set wordset=a b c d e f g h i j k l m n o pq r s t u v w x y z

set return=

set num=0

:randomPassword1

set /a num+=1

set /a numof=%random%%%26+1

for /f "tokens=%numof% delims= "%%i in ("%wordset%") do set return=%return%%%i

if %num% lss %password\_len% gotorandomPassword1

if not "%2"=="" set%2=%return%

shift /2

if not "%2"=="" gotorandomPassword

goto :eof

参考答案2：（最优）

@echo off

call :randomPassword 6 pass1 pass2 pass3

echo %pass1% %pass2% %pass3%

pause

exit

:randomPassword

::---------生成随机密码

::---------%1为密码长度，%2及以后为返回变量名称

::---------goto循环、变量嵌套、命令嵌套

@echo off

if "%1"=="" goto :eof

if %1 lss 1 goto :eof

set password\_len=%1

set return=

setwordset=abcdefghijklmnopqrstuvwxyz023456789\_

::---------------------------循环

:randomPassword1

set /a numof=%random%%%36

call setreturn=%return%%%wordset:~%numof%,1%%

set /a password\_len-=1

if %password\_len% gtr 0 gotorandomPassword1

::---------------------------循环

if not "%2"=="" set%2=%return%

shift /2

if not "%2"=="" gotorandomPassword

goto :eof

说明：本例涉及到变量嵌套和命令嵌套的应用，见后。

十三、变量嵌套与 命令嵌套

    和其它编程语言相比，dos功能显得相对简单，要实现比较复杂的功能，需要充分运用各种技巧，变量嵌套与命令嵌套就是此类技巧之一。

先复习一下前面的“字符串截取”的关键内容：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

截取功能统一语法格式为：%a:~[m[,n]]%

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

方括号表示可选，%为变量标识符，a为变量名，不可少，冒号用于分隔变量名和说明部分，符号～可以简单理解为“偏移”即可，m为偏移量（缺省为0），n为截取长度（缺省为全部）。

百分号如果需要当成单一字符，必须写成%%

以上是dos变量处理的通用格式，如果其中的m、n为变量，那么这种情况就是变量嵌套了。

比如设变量word为“abcdefghij”，变量num为“123456789”

%word:~4,1%为e，其中4可以从变量num中取值，即%num:~3,1%，写成组合形式如下：

%word:~%num:~3,1%,1% 经测试这种写法不能正确执行，写成%word:~(%num:~3,1%),1%同样不行，那么，怎么实现这种变量嵌套呢？这就必须结合命令嵌套。

什么是命令嵌套呢？简单的说，首先用一条dos命令生成一个字符串，而这个字符串是另一条dos命令，用call语句调用字符串将其执行，从而得到最终结果。

例：用call语句实现命令嵌套

@echo off

set str1=aaa echo ok bbb

echo 初始字符串：%str1%

echo 生成命令字符串如下：

echo %str1:~4,7%

echo 运行命令字符串生成最终结果为：

call %str1:~4,7%

pause

运行显示：

初始字符串：aaa echo ok bbb

生成命令字符串如下：

echo ok

运行命令字符串生成最终结果为：

ok

请按任意键继续. . .

BAT脚本编写教程

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

echo 表示显示此命令后的字符

echo off 表示在此语句后所有运行的命令都不显示命令行本身与echo off相象，但它是加在每个命令行的最前面，表示运行时不显示这一行的命令行（只能影响当前行）。

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

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

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

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

批处理文件的内容为:

命令注释：

　　　　@echo off　　　　　　　　　　　不显示后续命令行及当前命令行

　　　　dir c:"\*.\* >a.txt　　　　　　　将c盘文件列表写入a.txt

　　　　call c:"ucdos"ucdos.bat　　　　调用ucdos

　　　　echo 你好　　　　　　　　　　显示"你好"

　　　　pause 　　　　　　　　　　　　暂停,等待按键继续

　　　　rem 准备运行wps 　　　　　　　注释：准备运行wps

　　　　cd ucdos　　　　　　　　　　　进入ucdos目录

　　　　wps 　　　　　　　　　　　　　运行wps

批处理文件的参数

批处理文件还可以像[**C语言**](http://lib.csdn.net/base/c)的函数一样使用参数（相当于DOS命令的命令行参数），这需要用到一个参数表示符“%”。

%[1-9]表示参数，参数是指在运行批处理文件时在文件名后加的以空格（或者Tab）分隔的字符串。变量可以从%0到%9，%0表示批处理命令本身，其它参数字符串用%1到%9顺序表示。

例2：C:根目录下有一批处理文件名为f.bat，内容为：

@echo off

format %1

如果执行C:">f a:

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

例3：C:根目录下一批处理文件名为t.bat，内容为:

@echo off

type %1

type %2

那么运行C:">t a.txt b.txt

%1 : 表示a.txt

%2 : 表示b.txt

于是上面的命令将顺序地显示a.txt和b.txt文件的内容。

特殊命令

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

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

有三种格式:

1、if [not] "参数" =="字符串" 待执行的命令

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

例：if "%1"=="a" format a:

ke¬d02、if [not] exist [路径"]文件名 待执行的命令

如果有指定的文件，则条件成立，运行命令，否则运行下一句。

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

表示如果存在c:"config.sys文件，则显示它的内容。

3、if errorlevel <数字> 待执行的命令

很多DOS程序在运行结束后会返回一个数字值用来表示程序运行的结果(或者状态)，通过if errorlevel命令可以判断程序的返回值，根据不同的返回值来决定执行不同的命令(返回值必须按照从大到小的顺序排列)。如果返回值等于指定的数字，则条件成立，运行命令，否则运行下一句。

如if errorlevel 2 goto x2

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

二、goto 批处理文件运行到这里将跳到goto所指定的标号(标号即label，标号用:后跟标准字符串来定义)处，goto语句一般与if配合使用，根据不同的条件来执行不同的命令组。

如:

goto end

:end

echo this is the end

标号用“:字符串”来定义，标号所在行不被执行。

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

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

choice的命令语法（该语法为Windows2003中choice命令的语法，其它版本的choice的命令语法与此大同小异）：

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

描述:

    该工具允许用户从选择列表选择一个项目并返回所选项目的索引。

参数列表:

   /Cchoice    指定要创建的选项列表。默认列表是"YN"。

   /N                          在提示符中隐藏选项列表。提示前面的消息得到显示，

允许选择分大小写的选项。在默认情况下，这个工具是不分大小写的。

   /Ttimeout         做出默认选择之前，暂停的秒数。可接受的值是从0到9999。如果指定了0，就不会有暂停，默认选项会得到选择。

   /Dchoice   在 nnnn 秒之后指定默认选项。字符必须在用 /C 选项指定的一组选择中; 同时，必须用 /T 指定 nnnn。

   /Mtext                指定提示之前要显示的消息。如果没有指定，工具只显示提示。

   /?                          显示帮助消息。

  注意:ERRORLEVEL 环境变量被设置为从选择集选择的键索引。列出的第一个选择返回 1，第二个选择返回 2，等等。如果用户按的键不是有效的选择，该工具会发出警告响声。如果该工具检测到错误状态，它会返回 255 的ERRORLEVEL 值。如果用户按 Ctrl+Break 或 Ctrl+C 键，该工具会返回0的ERRORLEVEL值。在一个批程序中使用ERRORLEVEL参数时，将参数降序排列。

示例:

  CHOICE /?

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

  CHOICE /T 10 /C ync /CS /D y

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

De$O'i-c%E0   CHOICE /C ab /M "选项 1 请选择 a，选项 2 请选择 b。"

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

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

屏幕上会显示：

确认请按 Y，否请按 N，或者取消请按 C。 [Y,N,C]?

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

@echo off

choice /C dme /M "defrag,mem,end"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tc u8a7[;"0if errorlevel 3 goto end

if errorlevel 2 goto mem

if errotlevel 1 goto defrag

:defrag

c:"dos"defrag

goto end

:mem

mem

goto end

:end

echo good bye

此批处理运行后，将显示“defrag,mem,end[D,M,E]?”，用户可选择d m e ，然后if语句根据用户的选择作出判断，d表示执行标号为defrag的程序段，m表示执行标号为mem的程序段，e表示执行标号为end的程序段，每个程序段最后都以goto end将程序跳到end标号处，然后程序将显示good bye，批处理运行结束。

四、for 循环命令，只要条件符合，它将多次执行同一命令。

语法：

对一组文件中的每一个文件执行某个特定命令。

FOR %%variable IN (set) DO command[command-parameters]

%%variable  指定一个单一字母可替换的参数。

(set)      指定一个或一组文件。可以使用通配符。

command           指定对每个文件执行的命令。

command-parameters    为特定命令指定参数或命令行开关。

例如一个批处理文件中有一行:

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

则该命令行会显示当前目录下所有以bat和txt为扩展名的文件的内容。

批处理示例

1. IF-EXIST

1)首先用记事本在C:"建立一个test1.bat批处理文件，文件内容如下：

@echo off

IF EXIST "AUTOEXEC.BAT TYPE"AUTOEXEC.BAT

IF NOT EXIST "AUTOEXEC.BAT ECHO"AUTOEXEC.BAT does not exist

然后运行它：

C:">TEST1.BAT

如果C:"存在AUTOEXEC.BAT文件，那么它的内容就会被显示出来，如果不存在，批处理就会提示你该文件不存在。

2)接着再建立一个test2.bat文件，内容如下：

@ECHO OFF

IF EXIST "%1 TYPE "%1

IF NOT EXIST "%1 ECHO "%1 doesnot exist

执行:

C:">TEST2 AUTOEXEC.BAT

该命令运行结果同上。

说明：

(1) IF EXIST 是用来测试文件是否存在的，格式为

IF EXIST [路径+文件名] 命令

test2.bat文件中的%1是参数，DOS允许传递9个批参数信息给批处理文件，分别为%1~%9(%0表示test2命令本身) ，这有点象编程中的实参和形参的关系，%1是形参，AUTOEXEC.BAT是实参。

3) 更进一步的，建立一个名为TEST3.BAT的文件，内容如下：

@echo off

IF "%1" == "A" ECHOXIAO

IF "%2" == "B" ECHOTIAN

IF "%3" == "C" ECHO XIN

如果运行：

C:">TEST3 A B C

屏幕上会显示:

XIAO

TIAN

XIN

如果运行：

C:">TEST3 A B

屏幕上会显示

XIAO

TIAN

在这个命令执行过程中，DOS会将一个空字符串指定给参数%3。

2、IF-ERRORLEVEL

建立TEST4.BAT，内容如下：

@ECHO OFF

XCOPY C:"AUTOEXEC.BAT D:IF ERRORLEVEL1 ECHO 文件拷贝失败

IF ERRORLEVEL 0 ECHO 成功拷贝文件

然后执行文件:

C:">TEST4

如果文件拷贝成功，屏幕就会显示“成功拷贝文件”，否则就会显示“文件拷贝失败”。

IF ERRORLEVEL 是用来测试它的上一个DOS命令的返回值的，注意只是上一个命令的返回值，而且返回值必须依照从大到小次序顺序判断。

因此下面的批处理文件是错误的：

@ECHO OFF

XCOPY C:"AUTOEXEC.BAT D:"

IF ERRORLEVEL 0 ECHO 成功拷贝文件

IF ERRORLEVEL 1 ECHO 未找到拷贝文件

IF ERRORLEVEL 2 ECHO 用户通过ctrl-c中止拷贝操作

IF ERRORLEVEL 3 ECHO 预置错误阻止文件拷贝操作

IF ERRORLEVEL 4 ECHO 拷贝过程中写盘错误

无论拷贝是否成功，后面的：

未找到拷贝文件

用户通过ctrl-c中止拷贝操作

预置错误阻止文件拷贝操作

拷贝过程中写盘错误

都将显示出来。

以下就是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 withformat（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

3、IF STRING1 ==STRING2

建立TEST5.BAT，文件内容如下：

@echo off

IF "%1" == "A" formATA:

执行：

C:">TEST5 A

屏幕上就出现是否将A:盘格式化的内容。

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

注意：为了防止参数为空的情况，一般会将字符串用双引号（或者其它符号，注意不能使用保留符号）括起来。

如：if [%1]==[A] 或者 if %1\*==A\*

5、GOTO

建立TEST6.BAT，文件内容如下：

@ECHO OFF

IF EXIST C:"AUTOEXEC.BAT GOTO \_COPY

GOTO \_DONE

:\_COPY

COPY C:"AUTOEXEC.BAT D:"

:\_DONE

注意：

(1) 标号前是ASCII字符的冒号":"，冒号与标号之间不能有空格。

(2) 标号的命名规则与文件名的命名规则相同。

(3) DOS支持最长八位字符的标号，当无法区别两个标号时，将跳转至最近的一个标号。

6、FOR

建立C:"TEST7.BAT，文件内容如下：

@ECHO OFF

FOR %C IN (\*.BAT \*.TXT \*.SYS) DO TYPE %C

运行：

C:>TEST7

执行以后，屏幕上会将C:盘根目录下所有以BAT、TXT、SYS为扩展名的文件内容显示出来（不包括隐藏文件）。

bat命令的使用

　　一. 简单批处理内部命令简介

　　1. Echo 命令

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

　　语法

　　echo [{on　off}] [message]

　　Sample：@echo off / echo hello world

在实际应用中我们会把这条命令和重定向符号（也称为管道符号，一般用> >> ）结合来实现输入一些命令到特定格式的文件中.这将在以后的例子中体现出来。

2. @ 命令

表示不显示@后面的命令，在入侵过程中（例如使用批处理来格式化敌人的硬盘）自然不能让对方看到你使用的命令啦。

Sample：@echo off

@echo Now initializing the program,pleasewait a minite...

@format X: /q/u/autoset (format 这个命令是不可以使用/y这个参数的，可喜的是微软留了个autoset这个参数给我们，效果和/y是一样的。)

　　3. Goto 命令

　　指定跳转到标签，找到标签后，程序将处理从下一行开始的命令。

　　语法：goto label （label是参数，指定所要转向的批处理程序中的行。）

　　Sample：

　　if {%1}=={} goto noparms

　　if {%2}=={} goto noparms（如果这里的if、%1、%2你不明白的话，先跳过去，后面会有详细的解释。）

　　@Rem check parameters if null show usage

　　:noparms

　　echo Usage: monitor.bat ServerIP PortNumber

　　goto end

　　标签的名字可以随便起，但是最好是有意义的字母啦，字母前加个：用来表示这个字母是标签，goto命令就是根据这个：来寻找下一步跳到到那里。最好有一些说明这样你别人看起来才会理解你的意图啊。

4. Rem 命令

　　注释命令，在C语言中相当与/\*--------\*/,它并不会被执行，只是起一个注释的作用，便于别人阅读和你自己日后修改。

Rem Message

　　Sample：@Rem Here is the description.?

5. Pause 命令

　　运行 Pause 命令时，将显示下面的消息：

　　Press any key to continue . . .

　　Sample：

　　@echo off

　　:begin

　　copy a:\*.\* d："back

　　echo Please put a new disk into driver A

　　pause

　　goto begin

　　在这个例子中，驱动器 A 中磁盘上的所有文件均复制到d:"back中。显示的注释提示您将另一张磁盘放入驱动器 A 时，pause 命令会使程序挂起，以便您更换磁盘，然后按任意键继续处理。

6. Call 命令

从一个批处理程序调用另一个批处理程序，并且不终止父批处理程序。call 命令接受用作调用目标的标签。如果在脚本或批处理文件外使用Call，它将不会在命令行起作用。

　　语法

call [[Drive:][Path] FileName[BatchParameters]] [:label [arguments]]

参数

[Drive:}[Path] FileName

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

7. start 命令

调用外部程序，所有的DOS命令和命令行程序都可以由start命令来调用。

　　入侵常用参数：

j%MD0　　MIN 开始时窗口最小化

　　SEPARATE 在分开的空间内开始 16 位 Windows 程序

　　HIGH 在 HIGH 优先级类别开始应用程序

　　REALTIME 在 REALTIME 优先级类别开始应用程序

　　WAIT 启动应用程序并等候它结束

　　parameters 这些为传送到命令/程序的参数

执行的应用程序是 32-位 GUI 应用程序时，CMD.EXE不等应用程序终止就返回命令提示。如果在命令脚本内执行，该新行为则不会发生。

8. choice 命令   #这一个命令还不会用，上网再找找资料#

choice 使用此命令可以让用户输入一个字符，从而运行不同的命令。使用时应该加/c:参数，c:后应写提示可输入的字符，之间无空格。它的返回码为1234……

　　如: choice /c:dme defrag,mem,end

　　将显示

　　defrag,mem,end[D,M,E]?

　Sample：

Sample.bat的内容如下:

@echo off

　　choice /c:dme defrag,mem,end