[批处理 语法详解](http://blog.itpub.net/10527166/viewspace-683275/)

【 一、什么是批处理 】

批处理文件图标　批处理(Batch)，也称为批处理脚本。顾名思义，批处理就是对某对象进行批量的处理。批处理文件的扩展名为 bat 。

目前比较常见 的批处理包含两类：DOS批处理和PS批处理。PS批处理是基于强大的图片编辑软件Photoshop的，用来批量处理图片的脚本；而DOS批处理则是基于DOS命令的，用来自动地批量地执行DOS命令以实现特定操作的脚本。这里要讲的就是DOS批处理。

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或者CMD.EXE）解释运行。类似于Unix中的Shell脚本。批处理文件具有.bat或者.cmd的扩展名，其最简单的例子，是逐行书写在命令行中会用到的各种命令。更复杂的情况，需要使用if，for，goto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，利用外部程序是必要的，这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。批处理文件，或称为批处理程序，是由一条条的DOS命令组成的普通文本文件，可以用记事本直接编辑或用DOS命令创建，也可以用DOS下的文本编辑器Edit.exe来编辑。在“命令提示”下键入批处理文件的名称，或者双击该批处理文件，系统就会调用Cmd.exe运行该批处理程序。一般情况下，每条命令占据一行；当然也可以将多条命令用特定符号（如：&、&&、**|**、**||**等）分隔后写入同一行中；还有的情况就是像if、for等较高级的命令则要占据几行甚至几十几百行的空间。系统在解释运行批处理程序时，首先扫描整个批处理程序，然后从第一行代码开始向下逐句执行所有的命令，直至程序结尾或遇见exit命令或出错意外退出。

【 二、简单批处理命令简介 】

【 **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的文件中。

【 rem 命令 】

注释命令，类似于在C语言中的/**\***--------**\***/，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。

:: 也具有rem的功能

Rem Message

Sample：

@Rem Here is the description.

【 **pause** 命令 】

暂停命令。运行 **Pause** 命令时，将显示下面的消息：

Press any key to continue. . .(或：请按任意键继续. . .)

Sample：

@**echo** off

**:begin**

**copy** G:**\***.**\*** d：\back

**echo** 请插入另一张光盘...

pause

**goto** begin

在这个例子中，驱动器 G 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光盘

盘放入驱动器 G 时，pause 命令会使程序挂起，以便您更换光盘，然后按任意键继续处理。

【 **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 ] [:label [arguments]]

参数

[Drive:}[Path] FileName  rem指定要调用的批处理程序的位置和名称。

【 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。

如：

start calc.exe

即可打开Windows的计算器。

常用参数：

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。

【 **goto** 命令 】

跳转命令。程序指针跳转到指定的标签，从标签后的第一条命令开始继续执行批处理程序。

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。

【 **set** 命令 】

显示、设置或删除变量。

显示变量：

**set** 或 set s

前者显示批处理当前已定义的所有变量及其值，后者显示所有以s开头的变量及值。

设置变量：

**set** aa**=**abcd

此句命令便可向变量aa赋值abcd。如果变量aa已被定义，则aa的值被修改为abcd；若aa尚未定义，则此句命令即可定义新的变量aa，同时为变量aa赋予初始值abcd。

删除变量：

**set** aa**=**

此句命令即可删除变量aa。若变量aa已被定义，则删除变量aa；若aa尚未定义，则此句命令为实质意义。

需要说明的是，批处理中的变量是不区分类型的，不需要像C语言中的变量那样还要区分int、float、char等。比如执行set aa**=**345后，变量aa的值既可以被视为数字345，也可以被视为字符串345。

set命令具有扩展功能，如用作交互输入、字符串处理、数值计算等，属于高级命令范畴。

【 三、批处理符号简介 】

【 回显屏蔽 @ 】

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。

@用法举例：通过运行批处理文件对比pause和@pause命令即可明了@的效果。

【 重定向1 **>**与**>>** 】

将输出信息重定向到指定的设备或文件。系统默认输出到显示器。

如：

**echo** aaaaa**>**a.txt

即可将本在显示器上显示的信息aaaaa输出到文件a.txt中，屏幕上没有任何显示。如果文件a.txt本来已经存在，该命令将首先擦除a.txt中的所有信息，然后写入信息aaaaa；若a.txt本来就不存在，该命令即可新建一个a.txt文件，并写入信息aaaaa。

**echo** aaaaa**>>**a.txt

类似于echo aaaaa**>**a.txt。区别在于：如果a.txt本已存在，**>**a.txt会擦除a.txt中的原有内容，而**>>**a.txt并不擦除原有内容，仅在a.txt文件的末尾添加信息aaaaa。a.txt不存在时，二者没有差别。

【 重定向2 **<** 】

将输入信息来源重定向为指定的设备或文件。系统默认从显示器读取输入信息。

重定向使用举例：

**=========================================**

@**echo** off

**echo** abcdefg——这是文件a.txt中的信息**>**a.txt

**echo** 请任意输入字符，以回车结束：

**set** /p ifo**=**

**cls**

**echo** 【 从屏幕获得的输入信息 】

**echo** **%ifo%**

**set** /p ifo**=<**a.txt

**echo** 【 从文件a.txt获得的输入信息 】

**echo** **%ifo%**

**pause>nul**

**=========================================**

读者观察命令与输出即可体会到重定向的功能和效果。**<**/a.txt

【 管道符号 **|** 】

将管道符号前面命令的输出结果重定向输出到管道符号后面的命令中去，作为后面命令的输入。

使用格式为：

command\_1**|**command\_2

管道符号使用举例：

**=========================================**

@**echo** off

**echo** aaaa**>**a.txt

**del** /p a.txt

**pause**

**=========================================**

@**echo** off

**echo** aaaa**>**a.txt

**echo** y**|**del /p a.txt

**pause**

**=========================================**

对比以上两个批处理执行结果，读者即可明白管道符的用法和效果。

需要说明的是，上面del命令添加开关/p只是为了让读者明白管道符号的使用方法，实际删除文件时不加/p开关即可实现无提示直接删除。

【 转义符 ^ 】

将特殊符号转化为一般符号，即剥离特殊符号的特殊地位。

特殊符号指：

**|** & **>** **<**

比如，如果我们想输出符号“**>**”，直接用命令 **echo** **>** 是不行的，必须修改为 **echo** ^**>** 。其余几个特殊符号类似需要有同样的处理。

转义字符使用举例：

**=========================================**

@**echo** off

**echo** aaaa**>**a.txt

**echo** 第一句echo执行完毕

**echo** aaaa^**>**a.txt

**echo** 第二句echo执行完毕

**pause**

**=========================================**

比较上面的两句echo，第一句echo将信息aaaa输出到了文件a.txt，而第二句echo则在直接屏幕上显示除aaaa**>**a.txt

【 逻辑命令符 】

逻辑命令符包括：

&、&&、**||**

&-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；

&&-当&&前面的命令成功执行时，执行&&后面的命令，否则不执行；

**||**-当**||**前面的命令失败时，执行**||**后面的命令，否则不执行。

使用示例

**=========================================**

@**echo** off

**echo** ^**|**^**|**

reg add HKCU /v try /f**||echo** **\*\***成功**\*\***

reg add HKCU1 /v try /f**||echo** **\*\***失败**\*\***

**echo** ^&^&

reg delete HKCU /v try /f&&echo **\*\***成功**\*\***

reg delete HKCU /v try /f&&echo **\*\***失败**\*\***

**echo** ^&

reg delete HKCU /v try /f&echo **\*\***成功**\*\***

reg delete HKCU /v try /f&echo **\*\***失败**\*\***

**pause**

**=========================================**

执行reg add或reg delete后，系统会给出执行结果；我们通过echo命令也给出了“执行结果”。对比系统和我们自己给出的结果，既可以验证逻辑命令的判断机理。

【 四、常用DOS命令 】

【 文件夹管理 】

**cd** 显示当前目录名或改变当前目录。

**md** 创建目录。

**rd** 删除一个目录。

**dir** 显示目录中的文件和子目录列表。

tree 以图形显示驱动器或路径的文件夹结构。

**path** 为可执行文件显示或设置一个搜索路径。

xcopy 复制文件和目录树。

【 文件管理 】

**type** 显示文本文件的内容。

**copy** 将一份或多份文件复制到另一个位置。

**del** 删除一个或数个文件。

**move** 移动文件并重命名文件和目录。(Windows XP Home Edition中没有)

**ren** 重命名文件。

replace 替换文件。

attrib 显示或更改文件属性。

find 搜索字符串。

fc 比较两个文件或两个文件集并显示它们之间的不同

【 网络命令 】

ping 进行网络连接测试、名称解析

ftp 文件传输

net 网络命令集及用户管理

telnet 远程登陆

ipconfig 显示、修改TCP/IP设置

msg 给用户发送消息

arp 显示、修改局域网的IP地址-物理地址映射列表

【 系统管理 】

at 安排在特定日期和时间运行命令和程序

shutdown 立即或定时关机或重启

tskill 结束进程

taskkill 结束进程(比tskill高级，但WinXPHome版中无该命令)

tasklist 显示进程列表(Windows XP Home Edition中没有)

sc 系统服务设置与控制

reg 注册表控制台工具

powercfg 控制系统上的电源设置

对于以上列出的所有命令，在cmd中输入命令**+**/**?**即可查看该命令的帮助信息。如find /**?**

【 五、语句结构 】

类似于C语言，批处理也有它的语句结构。批处理的语句结构主要有选择结构(if语句)、循环结构(for语句)等。

【 if语句(选择结构) 】

if语句实现条件判断，包括字符串比较、存在判断、定义判断等。通过条件判断，if语句即可以实现选择功能。

1、字符串比较

if语句仅能够对两个字符(串)是否相同、先后顺序进行判断等。其命令格式为：

**IF** [not] string1 compare-op string2 command1 [else command2]

其中，比较操作符compare-op有以下几类：

**==** - 等于

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

选择开关/i则不区分字符串大小写；选择not项，则对判断结果进行逻辑非。

字符串比较示例：

**===============================================**

@**echo** off

**set** str1**=**abcd1233

**set** str2**=**ABCD1234

**if** **%str1%==%str2%** (echo 字符串相同！) **else** (echo 字符串不相同！)

**if** /i **%str1%** LSS **%str2%** (echo str1^**=**str2)

**echo**.

**set** /p choice**=**是否显示当前时间？(y/n)

**if** /i **not** **%choice%** EQU n **echo** 当前时间是：**%date%** **%time%**

**pause>nul**

**===============================================**

对于最后一个if判断，当我们输入n或N时的效果是一样的，都不会显示时间。如果我们取消开关/i，则输入N时，依旧会显示时间。

另外请注意一下几个细节：1-echo str1^**=**str2；2-echo.。

2、存在判断

存在判断的功能是判断文件或文件夹是否存在。其命令格式为：

**IF** [NOT] **EXIST** filename command1 [else command2]

使用示例

**===============================================**

@**echo** off

**if exist** **%0 echo** 文件**%0**是存在的！

**if not exist** **%~df0** (

**echo** 文件夹**%~df0不存在！**

) **else echo** 文件夹**%~df0存在！**

**pause>nul**

**===============================================**

这里注意几个地方：

1-存在判断既可以判断文件也可以判断文件夹；

2-**%0**即代表该批处理的全称(包括驱动器盘符、路径、文件名和扩展类型)；

3-**%~df0是对%0**的修正，只保留了其驱动器盘符和路径，详情请参考for /**?**，属高级批处理范畴；

4-注意if语句的多行书写，多行书写要求command1的左括号必须和if在同一行、else必须和command1的右括号同行、command2的左括号必须与else同行、command1和command2都可以有任意多行，即command可以是命令集。

3、定义判断

定义判断的功能是判断变量是否存在，即是否已被定义。其命令格式为：

**IF** [not] **DEFINED** variable command1 [else command2]

存在判断举例：

**===============================================**

@**echo** off

**set** var**=**111

**if defined** var (echo var**=%var%**) **else echo** var尚未定义！

**set** var**=**

**if defined** var (echo var**=%var%**) **else echo** var尚未定义！

**pause>nul**

**===============================================**

对比可知，"**set** var**=**"可以取消变量，收回变量所占据的内存空间。

4、结果判断

使用示例

masm **%1**.asm

**if errorlevel** 1 **pause** & edit **%1**.asm

link **%1**.obj

先对源代码进行汇编，如果失败则暂停显示错误信息，并在按任意键后自动进入编辑界面；否则用link程序连接生成的obj文件。

这种用法是先判断前一个命令执行后的返回码（也叫错误码，DOS程序在运行完后都有返回码），如果和定义的错误码符合（这里定义的错误码为1），则执行相应的操作（这里相应的操作为pause & edit **%1**.asm部分）。

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

masm **%1**.asm

**if not errorlevel** 1 link **%1**.obj

**pause** & edit **%1**.asm

【 for语句(循环结构) 】

for语句可以实现类似于C语言里面的循环结构，当然for语句的功能要更强大一点，通过不同的开关可以实现更多的功能。for语句有多个开关，不同开关将会实现不同的功能。

1、无开关

无开关的for语句能够对设定的范围内进行循环，是最基本的for循环语句。

其命令格式为：

**FOR** **%%v**ariable **IN** (set) **DO** command

其中，

**%%v**ariable是批处理程序里面的书写格式，在DOS中书写为**%variable，即只有一个百分号(%**)；

set就是需要我们设定的循环范围，类似于C语言里面的循环变量；

do后面的command就是循环所执行的命令，即循环体。

无开关for语句举例：

**===============================================**

@**echo** off

**for** **%%i in** (a,"b c",d) **do echo** **%%i**

**pause>nul**

**===============================================**

2、开关/L

含开关/L的for语句，可以根据set里面的设置进行循环，从而实现对循环次数的直接控制。

其命令格式为：

**FOR** /L **%%v**ariable **IN** (start,step,end) **DO** command

其中，

start为开始计数的初始值，

step为每次递增的值，end为结束值。当end小于start时，step需要设置为负数。

含开关/L的for语句举例(创建5个文件夹)：

**===============================================**

@**echo** off

**for** /l **%%i in** (1,2,10) **do md** **%%i**

**pause**

**===============================================**

上例将新建5个文件夹，文件夹名称依次为1、3、5、7、9。可以发现，**%%i**的结束值并非end的值10，而是不大于end的一个数。

3、开关/F

含开关/F的for语句具有最强大的功能，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。

其命令格式为：

**FOR** /F ["options"] **%%v**ariable **IN** (set) **DO** command

其中，

set为("string"、'command'、file-set)中的一个；

options是(eol**=**c、skip**=**n、delims**=**xxx、tokens**=**x,y,m-n、usebackq)中的一个或多个的组合。

各选项的意义参见for /f。一般情况下，使用较多的是skip、tokens、delims三个选项。

含开关/F的for语句举例：

**===============================================**

@**echo** off

**echo** **\*\***No Options:

**for** /f **%%a in** ("1,2,10") **do echo** a**=%%a**

**echo** **\*\***Options tokens ^& delims:

**for** /f "tokens**=**1-3 delims**=**," **%%a in** ("1,2,10") **do echo** a**=%%a** b**=%%b** c**=%%c**

**pause**

**===============================================**

@**echo** off

**echo** 本文件夹里面的文件有：

**for** /f "skip**=**5 tokens**=**3**\*** delims**=** " **%%a in** ('**dir**') **do** (

**if not** "**%%a**"**==**"" **if not** "**%%b**"**==**"字节" **if not** "**%%b**"**==**"可用字节" **echo** **%%b**

)

**pause**

**===============================================**

@**echo** off

**echo** 本文件夹里面的文件有：

**dir>**c:\file.txt

**for** /f "skip**=**5 tokens**=**3**\*** delims**=** " **%%a in** (c:\file.txt) **do** (

**if not** "**%%a**"**==**"" **if not** "**%%b**"**==**"字节" **if not** "**%%b**"**==**"可用字节" **echo** **%%b**

)

**del** c:\file.txt

**pause**

**===============================================**

对于后面的两个例子，其中options里面的delims**=** 是可以删除的，因为只要添加了/F开关系统就将delims的值默认为空格。

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。本例中也可以改为4，不过文件名中有空格的文件，只能显示空格以前部分

同时我们也看到了，for语句的do后面的command也是可以分行的，只需要保证command的左括号和do在同一行就可以了。

4、开关/D或/R

含开关/D或/R的for语句是与目录或文件有关的命令，一般情况下很少使用。含开关/R的命令有时候被用于通过遍历文件夹来查找某一个文件或文件夹，故而列举此例。

含开关/R的for语句举例(文件夹遍历)：

**===============================================**

@**echo** off

setlocal enabledelayedexpansion

**FOR** /R d: **%%i IN** (.) **DO** (

**set** dd**=%%i**

**set** "dd**=!dd:~0,-1!**"

**echo** **!dd!**

)

**pause**

**exit**

**===============================================**

上例即可以罗列出D盘下的所有文件夹，其速度要比命令"tree d:"慢多了，不过其返回结果的实用性则远远超过了tree命令。

一般情况下我们不推荐通过遍历文件夹来查找文件，特别是在查找某些程序(比如QQ.exe)的位置时。推荐通过reg命令查找注册表来查找QQ的路径，以保证查找效率。

上例中也出现了几个新面孔，如setlocal、感叹号等。其中，感叹号其实就是变量百分号(**%)的强化版。之所以要用!而不用%**，是因为在for循环中，当一个变量被多次赋值时，**%dd%**所获取的仅仅是dd第一次被赋予的值；要想刷新dd的值，就必须首先通过命令"setlocal enabledelayedexpansion"来开启延迟变量开关，然后用**!dd!**来获取dd的值。

for语句是批处理里面功能最强大、使用最普遍却又最难掌握的一套命令，这也是批处理菜鸟和批处理高手最明显的一个分水岭，一旦掌握了这套命令，那么你就离批处理达人不远了！

【 六、字符串处理 】

批处理有着具有非常强大的字符串处理能力，其功能绝不低于C语言里面的字符串函数集。批处理中可实现的字符串处理功能有：截取字符串内容、替换字符串特定字段、合并字符串、扩充字符串等功能。下面对这些功能一一进行讲解。

【 1、截取字符串 】

截取字符串可以说是字符串处理功能中最常用的一个子功能了，能够实现截取字符串中的特定位置的一个或多个字符。

举例说明其基本功能：

**=========================================**

@**echo** off

**set** ifo**=**abcdefghijklmnopqrstuvwxyz0123456789

**echo** 原字符串（第二行为各字符的序号）：

**echo** **%ifo%**

**echo** 123456789012345678901234567890123456

**echo** 截取前5个字符：

**echo** **%ifo:~0,5%**

**echo** 截取最后5个字符：

**echo** **%ifo:~-5%**

**echo** 截取第一个到倒数第6个字符：

**echo** **%ifo:~0,-5%**

**echo** 从第4个字符开始，截取5个字符：

**echo** **%ifo:~3,5%**

**echo** 从倒数第14个字符开始，截取5个字符：

**echo** **%ifo:~-14,5%**

**pause**

**=========================================**

当然，上面的例子只是将字符串处理的基本功能展示出来了，还看不出字符串处理具体有什么用处。

下面这个例子是对时间进行处理。

**=========================================**

@**echo** off

**echo** 当前时间是：**%time%** 即 **%time:~0,2%**点**%time:~3,2%**分**%time:~6,2%**秒**%time:~9,2%**厘秒

**pause**

**=========================================**

【 2、替换字符串 】

替换字符串，即将某一字符串中的特定字符或字符串替换为给定的字符串。

举例说明其功能：

**=========================================**

@**echo** off

**set** aa**=**伟大的中国！我为你自豪！

**echo** 替换前：**%aa%**

**echo** 替换后：%aa:中国=中华人民共和国%

**echo** aa **=** **%aa%**

**set** "aa**=**%aa:中国=中华人民共和国%"

**echo** aa **=** **%aa%**

**pause**

**=========================================**

对于上面的例子有一点说明，对比两个echo aa **=** **%aa%**可以发现，如果要修改变量aa的内容的话，就需要将修改结果“%aa:中国=中华人民共和国%”赋值给变量aa。上面的字符串截取也有着同样的特点。

【 3、字符串合并 】

其实，合并字符串就是将两个字符串放在一起就可以了。

举例说明：

**=========================================**

@**echo** off

**set** aa**=**伟大的中国！

**set** bb**=**我为你自豪！

**echo** **%aa%%bb%**

**echo** aa**=%aa%**

**echo** bb**=%bb%**

**set** "aa**=%aa%%bb%**"

**echo** aa**=%aa%**

**pause**

**=========================================**

同样，如果要改变变量aa的内容的话，就需要将合并结果“**%aa%%bb%**”赋值给变量aa。

【 4、扩充字符串 】

“扩充”这个词汇来自于微软自己的翻译，意思就是对表示文件路径的字符串进行特殊的处理，具体功能罗列如下：

**=========================================**

~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**

**=========================================**

以上内容引用于for /**?**帮助信息。其中的I代表变量I，不过需要说明的是，不是所有的变量都能够进行扩充的，有两个条件：1、该字符串代表一个文件路径；2、变量要用%x来表示，x可取a-z A-Z 0-9共62个字符中的任意一个。

举例说明：

**=========================================**

@**echo** off

**echo** 正在运行的这个批处理：

**echo** 完全路径：**%0**

**echo** 去掉引号：**%~0**

**echo** 所在分区：**%~d0**

**echo** 所处路径：**%~p0**

**echo** 文件名：**%~n0**

**echo** 扩展名：**%~x0**

**echo** 文件属性：**%~a0**

**echo** 修改时间：**%~t0**

**echo** 文件大小：**%~z0**

**pause**

**=========================================**

其中的**%0**是批处理里面的参数，代表当前运行的批处理的完全路径。类似的还有**%1-%**9，分别代表传递来的第1-9个参数。

例子如下：

**===============================================**

@**echo** off

**set** aa**=**C:\Windows\PPP\a.btx

**call** :deal aaa **%aa%** "c c" ddd eee

**pause>nul**

**exit**

**:deal**

**echo** **%%0** **=** **%0**

**echo** **%%1** **=** **%1**

**echo** **%%2** **=** **%2**

**echo** **%%3** **=** **%3**

**echo** **%%4** **=** **%4**

**echo** **%%5** **=** **%5**

**===============================================**

其中，变量aa在之前是不可以扩充的，通过call命令并将aa作为参数传递给子函数:deal，将aa变量转换成了变量**%1，即符合%**x格式，从而可以进行字符串扩充。

至于%x中x取a-z A-Z的形式，可以复习一下for语句，for语句里面的变量就是用%x来表示的，因而可以直接进行扩充。

【 七、数值计算 】

批处理里面的数值计算功能较弱，只能够进行整型计算，忽略浮点数的小数部分；同时数值计算的范围也受限于系统位数，对于目前较为常见的32位机来说，数值计算能处理的数值范围为0x80000000h~0x7FFFFFFFh，即-2147483648~**+**2147483647。

数值计算需要使用set命令，具体格式为

“set /a expression”。

其中，

expression代表计算表达式，计算表达式跟C语言里面的表达式基本上完全一致。

set支持的运算符也跟C语言里面的一样，只是没有了増一减一。set支持的运算符及优先级排序如下：

**=========================================**

() - 分组

! ~ - - 一元运算符（逻辑非、按位非、取负）

**\*** / % - 算数运算符（乘、除得商、除得余数，即取余）

**+** - - 算数运算符（加、减）

**<<** **>>** - 逻辑移位（左移一位、右移一位）

& - 按位“与”

^ - 按位“异”

**|** - 按位“或”

**=** **\*=** /**=** %= **+=** -**=** - 赋值

&**=** ^**=** **|=** **<<=** **>>=**

, - 表达式分隔符（set可一次处理多个表达式）

**=========================================**

我们知道，批处理中取变量的值是需要用%或者!的，而在set /a 中，直接用变量名称即可取得变量的值。

另外，set支持八进制（数字前缀0）、十进制（数字无前缀）和十六进制（数字前缀0x），且支持不同进制之间的计算，如set /a a**=**123**+**0123**+**0x123，计算及显示结果为十进制。

【 八、概念方法 】

本节学习有关批处理的一些比较琐碎、但又比较有价值的一些知识，运用这些知识后，才有条件更好的实现的批处理的功能，并能使批处理摆脱黑白世界而显得更加多姿多彩。废话少说，开讲啦~

【（一）环境变量 】

所谓“环境变量”，实际上就是C语言里面的“变量”的意思。

批处理的变量可以分为两类，由系统定义的系统变量和由用户根据需要自定义的用户变量。

【 1、系统变量 】

批处理的一些变量是由操作系统事先定义好的，可以适用于任何批处理，我们称这些特殊的变量为“系统变量”。系统变量有很多个，包括硬件类、操作系统类、文件路径类、系统时间类等。要查看所有的系统变量，请新打开一个cmd窗口，输入set回车即可。对几个比较常用的变量解释如下：

ComputerName 计算机名，即右键 我的电脑--属性--计算机名 选项卡中的“完整的计算机名称”

ComSpec cmd.exe完整路径名

HomeDrive 系统分区盘符，一般都是C盘，故而HomeDrive**=**C:

**Path** 可执行文件默认搜索路径。这个东东非常重要！待会儿单独讲解…

ProgramFiles 就是系统的Program Files的路径啦，一般都是C:\Program Files，这就是安装软件时默认的安装路径了

**Prompt** 个性化设置cmd提示符的必备武器！不过，我没怎么用过~

SystemDrive 包含系统根目录的分区，其实就是HomeDrive了

SystemRoot 系统根目录路径，一般都是C:\WINDOWS

Temp、Tmp 文件、程序等可使用的临时目录，默认是C:\WINDOWS\Temp或Tmp。几乎所有的程序在运行时都会在这个目录里面“临时”写入文件。一般情况下，程序写入的临时文件都应该被该程序删除，可惜的是，大部分的程序都很健忘，导致这个文件夹占据的空间越来越大，自然也就使我们的系统增肥喽。所以，我们要把它修改到其他分区，并且时时的清理里面的临时文件。

UserName 当前用户名，即所登陆的账户名

UserProfile 当前用户的配置目录，一般都是C:\Documents and Settings\**%UserName%**。默认情况下，我们的桌面就是这个目录下面的“桌面”文件夹；我的文档就是这个目录下面的“My Documents”文件夹。所以啦，往桌面上或我的文档里面放东西就是放到这个文件夹下面了，也就是放到C盘了，重装系统时要覆盖C盘内容的，所以桌面上或我的文档里面的东西当然就会Gone with the Wind了~解决方法有两个，一是保持良好的习惯，不把重要文件放到这两个地方；二是，修改默认设置，将这两个文件夹都移到其他分区。

WinDir 操作系统路径，其实就是SystemRoot了

【 2、用户变量 】

编写批处理程序时，用户根据需要自己定义的变量称之为用户变量。用户变量类似于C语言里面的变量，仅仅在定义该变量的程序中有效。

用户变量由set命令定义，这是批处理中非常非常重要的一个操作，从而使set命令成为批处理里面使用频率最高的几个命令之一。关于set命令的使用，参考set /**?**，本教程也会在后面对其进行讲解。

【 3、变量引用 】

前面的几节课里面，我们已经看到了如何引用变量，即直接用变量名操作变量，通过"%"或"!"来获取变量的值。

其中，只有在for语句里面重复对同一变量多次赋值时才需要使用"!"，并且在使用"!"调用变量时，要首先“启用延迟环境变量扩充”，启动命令为：SetLocal EnableDelayedExpansion。

另外需要说明的是，“启用延迟环境变量扩充”后，所有的"!"都将被视为“取变量值”的特殊符号，即使用"^!"也不能输出符号"!"。若要输出"!"，则需要“停用延迟环境变量扩充”，命令为：SetLocal DisableDelayedExpansion

【（二）参数 】

跟C语言类似，在调用函数或其他批处理时可能需要传递参数。批处理的参数传递分为直接和间接两种传递参数的方法。

【 1、直接传递 】

直接传递参数，即在使用call命令时，不使用任何参数，在子函数或子批处理里面直接对主函数(也称父批处理)里面的变量进行修改。这跟汇编语言里面的参数传递方式类似。

直接传递参数举例：

**===============================================**

@**echo** off

setlocal enabledelayedexpansion

**set** var**=**aCdehiM,**?**mnrstW y

**echo** **%var%**

**call** :deal

setlocal disabledelayedexpansion

**set** var**=**%var:?=!%

**echo** **%var%**

**pause>nul**

**exit**

**:deal**

**set** tm**=!var!**

**set** var**=**

**for** **%%i in** (6,3,11,11,16,15,1,4,11,5,12,13,9,0,12,7,15,14,5,10,2,16,18,8) **do** (

**set** var**=!var!!tm:~%%i,1!**

)

**goto** :eof

**===============================================**

可以发现，当我们把变量var作为参数赋予子函数:deal后，子函数对var的值进行了修改；当子函数返回后，主函数里面的var的值就已经是子函数里面var被修改后的值了。

该例子中，使用了本节课前面讲到的setlocal enabledelayedexpansion和setlocal disabledelayedexpansion，前者保证了var在for循环里面能够根据我们的意愿进行处理，后者保证了能够正确输出符号"!"。

另外例子中还使用了命令set，利用set对字符串进行了处理。

还有一个地方使用了语句goto :eof，该语句相当于C语言里面的return或汇编语言里面的RET，即子程序返回命令。

需要说明的是，当子函数本身就在批处理文件的末尾的话，我们是可以省略这句话的，比如将此例的goto :eof删除是不会产生任何影响的。

【 2、间接传递 】

间接传递参数，即在使用call命令时，在其后面添加参数，形如

**call** {[:label][ChildBatch]} Parameter1 Parameter2 ... ParameterN

这跟C语言里面传递参数的格式类似。不同于C语言，批处理中的子函数不需要定义形参，更不需要指定参数的个数。传递过来的参数，在子函数或子批处理里面是以**%1~%**9的形式表示的，即**%1~%**9分别表示传递过来的第1~9个参数。

使用示例如下：

**===============================================**

@**echo** off

**call** :deal aaa bbb "c c" ddd eee

**pause>nul**

**exit**

**:deal**

**echo** **%%0** **=** **%0**

**echo** **%%1** **=** **%1**

**echo** **%%2** **=** **%2**

**echo** **%%3** **=** **%3**

**echo** **%%4** **=** **%4**

**echo** **%%5** **=** **%5**

**===============================================**

通过这个例子就可以清晰的看到%n参数表示法的用法。

参数列表中包含空格的依旧要用双引号(")引起来；另外，也可以看到，**%0**已经变成了子函数的标号了，而不是父批处理的文件名全称。

【 3、区别 】

这两种参数传递方法本质上是没有区别的，形式上，直接传递直接对原变量进行操作，丢失了原变量的值；间接传递则通过%n对原变量进行了简单的备份，并且通用性更强，即不限定原变量的名称。

另外，使用**%n还有一个非常大的好处，就是可以通过%**~**\***i来加强处理变量的能力。关于**%~\***i，详细内容参见for /**?**。

针对二者的差别，可以根据情况决定使用哪种传递方式：

--1、作为参数的变量名固定、且在子函数中不需要对其进行备份的情况下，使用直接传递法；

--2、若将子函数作为一个通用的程序模块，以适应于对不同变量的处理，或者作为参数的变量不需要备份时，使用间接传递法。

具体使用哪种方法，还需根据实际情况或使用习惯进行选择。

【（三）返回值 】

有些命令在执行之后将会返回一定的错误值(errorlevel)，可以通过errorlevel的值判断命令执行的状况。这点类似于C语言里面的exit(num)，num就是错误代码。

获取返回值errorlevel的方法就是，在执行命令后，立马调用返回值errorlevel，如echo **%errorlevel%**或者if **%errorlevel%==**1等命令。

errorlevel举例：

**===============================================**

@**echo** off

reg add HKCU /v try /f**>nul**

reg delete HKCU /v try /f

**if errorlevel** 0 (echo 删除成功！) **else** (echo 删除失败！)

reg delete HKCU /v try /f

**if** **%errorlevel%==**0 (echo 删除成功！) **else** (echo 删除失败！)

**pause>nul**

**===============================================**

上面例子中，由于第一成功的删除了注册表，导致第二次因为找不到注册表而宣告失败。

同时我们也看到了errorlevel的使用方法，即if **errorlevel** 0和if **%errorlevel%==**0是一样的。也许你注意到了，里面还有个笑脸呢~O(∩\_∩)O哈哈~这就是ASCII码啦，后面跟你讲啊…

一般情况下，程序或命令成功执行时，返回的errorlevel是0，错误时返回1或更高的值。当然，有些命令是没有返回值的，这点需要注意。

嗯，有没有想起前面有个类似的东西啊？对了，那就是**||**和&&了，这两个符号就是根据errorlevel的值来进行逻辑判断的~

【（四）用户交互 】

批处理，黑框白字是它最著名的特征。虽然当初DOS为人们使用计算机做出了莫大的贡献，但在Windows盛行的今天，人们已经疏远并且惧怕那个黑色的窗口了。微软为了让先天有着批量处理“体力活”能力的DOS避免“冷酷”，便于接近，特意提供了几个小命令，加强批处理的用户交互功能。

【 1、视窗 】

首先我们要DIY它的窗口。使用命令：color、mode、msg。

--1、设置窗口背景色和字体颜色by color。详细内容参见color /**?**。

--2、设置窗口大小by "MODE **CON** [COLS**=**c] [LINES**=**n]"，cols即宽度，lines即高度。

--3、GUI交互窗口by msg。详细内容参见msg /**?**。

视窗DIY举例：

**===============================================**

@**echo** off

**set** a**=**10

**set** b**=**0

**:tex**

**set** /a a**+=**3

**set** /a b**+=**1

mode **con** cols**=%a%** lines**=%b%**

**if** **%a%** lss 60 **goto** :tex

**echo** O(∩\_∩)O 圣诞快乐 O(∩\_∩)O

**set** a**=**15,a6,2e,d0,34,8b,4f,9d,5e

**for** **%%i in** (**%a%**) **do** (

ping -n 2 127.1**>nul**

color **%%i**

)

**for** **%%i in** (**%a%**,**%a%**) **do** (

ping -n 1 127.1**>nul**

color **%%i**

)

**>>**ms.txt **echo** **\***/. . . **\*** .

**>>**ms.txt **echo** .\**\*** . [] **\*** \_\_

**>>**ms.txt **echo** **\***/ . ./\~~~~~~~~~~~~'\. ^**|**◆

**>>**ms.txt **echo** \**\*** ,/,..,\,...........,\.◆

**>>**ms.txt **echo** ^**|**^**|** ..▎# ▎田 田 ▎ ^**|** ▎◆

**>>**ms.txt **echo** ^**|**^**|** ^&^&▎ ▎ ▎'^**|**'▎ o

**>>**ms.txt **echo** ^**|**^**|** ##■■■■■■■■■■〓

msg **%username%** /w /time:3600 **<**ms.txt

**del** ms.txt

**pause**

**===============================================<**/ms.txt

【 2、声音 】

呵呵，是不是注意到了批处理没有声音呐？闲话不说，直接做实验吧 ^\_^

**===============================================**

@**echo** off

**echo** 做好准备，响了啊！！

**pause**

**cls**

**echo**

**echo** 怎么样？呵呵，刺激吧~

**pause**

**cls**

mshta vbscript.:createobject("sapi.spvoice").speak("Merry Christmas and Happy New Year!")(window.close)

**pause**

**===============================================**

此处，这个就是调用vbs的方法：

vbscript.:createobject("sapi.spvoice").speak("Merry Christmas and Happy New Year!")

speak 好像不能讲中文呐~

其实不然，如果tts安装了中文引擎是可以讲中文的。

【 3、控制 】

在运行批处理的过程中，我们如何干预批处理呢？呵呵，直接点叉叉就可以结束它了！嗯，不错，不过，太野蛮了~

--1、暂停批处理：直接按键盘上的Pause键喽

--2、终止批处理：组合键Ctrl**+**C。不过，有时候它好像响应的不太积极啊…

【（五）ASCII码 】

前面的例子中，我们已经使用过一次ASCII码了，也就是那个笑脸。ASCII码是图形化的符号，可以用来点缀我们的批处理的。

在cmd窗口中我们可以通过任意一个字符的ASCII码来输入该字符，比如Ctrl**+**G、Ctrl**+**N等，字母a-z对应ASCII码的97-122。对于ASCII码大于26的字符，可以通过这个方法来输入：按住Alt键不松，通过小键盘输入ASCII码的十进制值，松开Alt键即可。

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

SET命令

显示、设置或删除 cmd.exe 环境变量。

SET命令格式为

**SET** [variable**=**[string]]

其中

variable 指定环境变量名。

string 指定要指派给变量的一系列字符串。

要显示当前环境变量，键入不带参数的 SET。

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

如果命令扩展名被启用，SET 会如下改变:

可仅用一个变量激活 **SET** 命令，等号或值不显示所有前缀匹配 SET 命令已使用的名称的所有变量的值。

例如:

**SET** P

会显示所有以字母 P 打头的变量。如果在当前环境中找不到该变量名称，SET 命令将把 ERRORLEVEL设置成 1。

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

**SET** 命令不允许变量名含有等号。

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

在 **SET** 命令中添加了两个新命令行开关:

**SET** /A expression

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

/A 命令行开关指定等号右边的字符串为被评估的数字表达式。该表达式评估器很简单并以递减的优先权顺序支持下列操作:

() - 分组

! ~ - - 一元运算符

**\*** / % - 算数运算符

**+** - - 算数运算符

**<<** **>>** - 逻辑移位

- 按位“与”

^ - 按位“异或”

**|** - 按位“或”

**=** **\*=** /**=** %= **+=** -**=** - 赋值

&**=** ^**=** **|=** **<<=** **>>=** - 赋值

, - 表达式分隔符

如果您使用任何逻辑或取余操作符，您需要将表达式字符串用引号扩起来。在表达式中的任何非数字字符串键作为环境变量名称，这些环境变量名称的值已在使用前转换成数字。如果指定了一个环境变量名称，但未在当前环境中定义，那么值将被定为零。这使您可以使用环境变量值做计算而不用键入那些 % 符号来得到它们的值。如果 **SET** /A 在命令脚本外的命令行执行的，那么它显示该表达式的最后值。该分配的操作符在分配的操作符左边需要一个环境变量名称。除十六进制有 0x 前缀， 八进制有 0 前缀的，数字值为十进位数字。因此， 0x12 与 18 和 022 相同。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。

/P 命令行开关允许将变量数值设成用户输入的一行输入。读取输入行之前，显示指定的 promptString。promptString 可以是空的。

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

环境变量替换已如下增强:

%PATH:str1=str2%

会扩展 **PATH** 环境变量，用 "str2" 代替扩展结果中的每个 "str1"。

要有效地从扩展结果中删除所有的 "str1"，"str2" 可以是空的。 "str1" 可以以星号打头；在这种情况下，"str1" 会从扩展结果的开始到 str1 剩余部分第一次出现的地方，都一直保持相配。

也可以为扩展名指定子字符串。

**%PATH:~10,5%** 会扩展 **PATH** 环境变量，然后只使用在扩展结果中从第 11 个(偏移量 10)字符开始的五个字符。

如果没有指定长度，则采用默认值，即变量数值的余数。如果两个数字(偏移量和长度)都是负数，使用的数字则是环境变量数值长度加上指定的偏移量或长度。

**%PATH:~-10%** 会提取 **PATH** 变量的最后十个字符。

**%PATH:~0,-2%** 会提取 **PATH** 变量的所有字符，除了最后两个。

终于添加了延迟环境变量扩充的支持。该支持总是按默认值被停用，但也可以通过 CMD.EXE 的 /V 命令行开关而被启用/停用。

请参阅 CMD /**?**

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

考虑到读取一行文本时所遇到的目前扩充的限制时，延迟环境变量扩充是很有用的，而不是执行的时候。以下例子说明直接变量扩充的问题:

**set** VAR**=**before

**if** "**%VAR%**" **==** "before" (

**set** VAR**=**after

**if** "**%VAR%**" **==** "after" @echo **If** you see this, it worked

)

不会显示消息，因为在读到第一个 **IF** 语句时，BOTH **IF** 语句中的 **%VAR%** 会被代替；原因是: 它包含 **IF** 的文体，IF 是一个复合语句。所以，复合语句中的 **IF** 实际上是在比较 "before" 和"after"，这两者永远不会相等。同样，以下这个例子也不会达到预期效果:

**set** LIST**=**

**for** **%i in** (**\***) **do set** LIST**=%LIST%** %i

**echo** **%LIST%**

原因是，它不会在目前的目录中建立一个文件列表，而只是将LIST 变量设成找到的最后一个文件。这也是因为 **%LIST%** 在 **FOR** 语句被读取时，只被扩充了一次；而且，那时的 LIST 变量是空的。因此，我们真正执行的 **FOR** 循环是:

**for** **%i in** (**\***) **do set** LIST**=** %i

这个循环继续将 LIST 设成找到的最后一个文件。

延迟环境变量扩充允许您使用一个不同的字符(惊叹号)在执行时间扩充环境变量。如果延迟的变量扩充被启用，可以将上面例子写成以下所示，以达到预期效果:

**set** VAR**=**before

**if** "**%VAR%**" **==** "before" (

**set** VAR**=**after

**if** "**!VAR!**" **==** "after" @echo **If** you see this, it worked

)

**set** LIST**=**

**for** **%i in** (**\***) **do set** LIST**=!LIST!** %i

**echo** **%LIST%**

如果命令扩展名被启用，有几个动态环境变量可以被扩展，但不会出现在 **SET** 显示的变量列表中。每次变量数值被扩展时，这些变量数值都会被动态计算。如果用户用这些名称中任何一个定义变量，那个定义会替代下面描述的动态定义:

**%CD%** - 扩展到当前目录字符串。

**%DATE%** - 用跟 **DATE** 命令同样的格式扩展到当前日期。

**%TIME%** - 用跟 **TIME** 命令同样的格式扩展到当前时间。

**%RANDOM%** - 扩展到 0 和 32767 之间的任意十进制数字。

**%ERRORLEVEL%** - 扩展到当前 **ERRORLEVEL** 数值。

**%CMDEXTVERSION%** - 扩展到当前命令处理器扩展名版本号。

**%CMDCMDLINE%** - 扩展到调用命令处理器的原始命令行。

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

如何延长批处理延时关机时间？

代码示例：

@**echo** off

timeout /t 3600 /nobreak

shutdown /s /f /t 0

@**echo** off

msg **%username%** /time:3600 /v /w 等待关机 **>nul**

shutdown /s /f /t 0

－－－－－－－－－－－－－－－－－－－－

通过 timeout /**?** 查询帮助如下

TIMEOUT [/T] timeout [/NOBREAK]

描述:

    这个工具接受超时参数，等候一段指定的时间(秒)或等按任意键。它还接受

    一个参数，忽视按键。

参数列表:

    /T        timeout       指定等候的秒数。有效范围从 -1 到 99999 秒。

    /NOBREAK                忽略按键并等待指定的时间。

    /**?**                      显示该帮助消息。

注意: 超时值 -1 表示无限期地等待按键。

示例:

    TIMEOUT /**?**

    TIMEOUT /T 10

    TIMEOUT /T 300 /NOBREAK

    TIMEOUT /T -1