# 基本概念

**1、PS1文件**

一个PowerShell脚本[1] 其实就是一个简单的文本文件，这个文件包含了一系列PowerShell命令，每个命令显示为独立的一行，对于被视为PowerShell脚本的文本文件，它的文件名需要使用.PS1扩展。

**2、执行权限**

为防止[恶意脚本](http://baike.baidu.com/view/335609.htm)的执行，PowerShell有一个执行策略，默认情况下，这个执行策略被设为受限的（Restricted），意味着PowerShell脚本无法执行，你可以使用下面的cmdlet命令确定当前的执行策略：

Get-ExecutionPolicy 你可以选择使用的执行策略有：

Restricted -[脚本](http://baike.baidu.com/view/54.htm)不能运行。 RemoteSigned - 本地创建的[脚本](http://baike.baidu.com/view/54.htm)可以运行，但从网上下载的脚本不能运行（除非它们拥有由受信任的发布者签署的数字签名）。 AllSigned – 仅当[脚本](http://baike.baidu.com/view/54.htm)由受信任的发布者签名才能运行。 Unrestricted –[脚本](http://baike.baidu.com/view/54.htm)执行不受限制，不管来自哪里，也不管它们是否有签名。

你可以使用下面的cmdlet命令设置PowerShell的执行策略：

Set-ExecutionPolicy <policy name>

**3、**[**运行脚本**](http://baike.baidu.com/view/513191.htm)

如果你想从命令行运行一个可执行文件，多年来一个永恒不变的方法是，在命令行转到该执行文件所在的位置，然后键入该执行文件的名称，但这个古老的方法现在却不能适用于PowerShell可执行[脚本](http://baike.baidu.com/view/54.htm)了。

如果你想执行一个PowerShell[脚本](http://baike.baidu.com/view/54.htm)，通常必须键入完整的路径和文件名，例如，假设你要运行一个名为a.ps1的脚本，你可以键入：

C:\Scripts\aps1 最大的例外是，如果PowerShell[脚本文件](http://baike.baidu.com/view/1295748.htm)刚好位于你的系统目录中，那么你可以直接在[命令提示符](http://baike.baidu.com/view/756438.htm)后键入脚本文件名即可运行，如：

.\a.ps1 注意前面需要加上.\，这和Linux下执行Shell脚本的方法如出一辙。

4、管道

管道的作用是将一个命令的输出作为另一个命令的输入，两个命令（或cmdlet）之间只需要用管道符号（|）连接即可。

为了帮助你了解管道是如何工作的，我们以一个例子进行说明，假设你想创建运行在服务器上的进程列表，并按进程的ID号进行排序，可以使用Get-Process cmdlet命令获得进程列表，但默认情况下列表不会排序，如果将这个cmdlet命令的输出用管道输送给Sort-Object ID命令，进程列表将会按进程ID号进行排序，如：

Get-Process | Sort-Object ID

**5、**[**变量**](http://baike.baidu.com/view/296689.htm)

虽然可以使用管道将一个命令的输出输送给另一个命令，但管道本身也是有限制的，当你用管道从一个命令向另一个命令传递输出结果时，输出结果立即被使用，但有时候，你可能需要保存输出结果一段时间，以便以后可以使用（或重用），这个时候管道就应该下场，轮到变量上场了。

人们很容易将[变量](http://baike.baidu.com/view/296689.htm)想象成一个仓库，但在PowerShell中，变量可以保存命令的完整输出，例如，假设你想保存服务器处于运行中的进程列表，你可以将它赋给一个变量，如：

$a = Get-Process 在这里，[变量](http://baike.baidu.com/view/296689.htm)被命名为$a，如果你想使用这个变量，只需要简单地调用它的名称即可，例如，键入$a便可在[屏幕](http://baike.baidu.com/view/207233.htm)上打印变量的内容。

你可以将多个用管道连接的命令的最终输出赋给一个[变量](http://baike.baidu.com/view/296689.htm)，只需要用一对小括号将命令括起来即可，例如，假设你想按进程ID对运行中的进程进行排序，然后将结果输出给一个变量，你可以使用下面这个命令：

$a = (Get-Process | Sort-Object ID)

**6、@符号**

通过使用@符号，你可以将列表内容转换成一个[数组](http://baike.baidu.com/view/209670.htm)，例如，下面的代码创建了一个名为$Procs的变量，它包含多行文本内容（一个数组）：$procs = @{name="explorer","svchost"}

使用[变量](http://baike.baidu.com/view/296689.htm)时你也可以使用@符号，为了确保它作为[数组](http://baike.baidu.com/view/209670.htm)而不是单个值处理，例如，下面的代码将在我前面定义的变量上运行Get-Process cmdlet命令：

Get-Process @procs[Windows](http://baike.baidu.com/view/4821.htm)将显示[Windows](http://baike.baidu.com/view/4821.htm)[资源管理器](http://baike.baidu.com/view/108140.htm)和Svchost使用的所有进程，注意[变量](http://baike.baidu.com/view/296689.htm)前使用的@符号，而不是常见的$符号。

**7、Split**

Split操作符根据你指定的字符拆分一个文本字符串，例如，假设你想将一个句子拆分成一个单词组成的一个[数组](http://baike.baidu.com/view/209670.htm)，你可以使用下面的命令做到：

"This is a test" -split " " 拆分后的结果如下：

This is a test

**8、Join**

就像Split可以将一个文本字符串拆分成多块一样，Join的操作则是逆向的，将多个独立的块连接成一个整体，例如，下面这行代码将会创建一个文本字符串，由我的名字和姓氏组成：

"Brien","Posey" -join " " 命令末尾双引号之间的空格告诉Windows在两个文本字符串之间插入一个空格。

**9、**[**断点**](http://baike.baidu.com/view/278359.htm)

运行一个新创建的PowerShell[脚本](http://baike.baidu.com/view/54.htm)时，如果脚本有Bug，会遇到意想不到的后果，保护自己的一个方法是在脚本的关键位置插入[断点](http://baike.baidu.com/view/278359.htm)，这样你就可以确保脚本正常运行先，然后再处理可能存在的问题。

插入断点最简单的方法是根据[行号](http://baike.baidu.com/view/239028.htm)插入，例如，假设你要在第10行插入一个断点，可以使用下面的命令：

New-PSBreakpoint -Script C:\Scripts\a.ps1 -Line 10 你也可以将[断点](http://baike.baidu.com/view/278359.htm)绑定到变量上，如果你希望你的[脚本](http://baike.baidu.com/view/54.htm)任何时候都可以修改a$的内容，可以使用下面的命令：

New-PSBreakpoint -Script C:\scripts\a.ps1 -variables a 注意，我在[变量](http://baike.baidu.com/view/296689.htm)名后并没有包括美元符号。

可以和PSBreakpoint一起使用的动词包括New，Get，Enable，Disable和Remove。

**10、Step**

调试一个[脚本](http://baike.baidu.com/view/54.htm)时，有时可能需要逐行[运行脚本](http://baike.baidu.com/view/513191.htm)，这时你可以使用Step-Into cmdlet命令，它会使脚本一行一行地执行，不管有没有设置断点，如果你想从这种步进式运行模式退出来，使用Step-Out cmdlet命令即可，但需要注意的是，使用Step-Out cmdlet命令后，断点仍然有效。

顺便说一句，如果你的[脚本](http://baike.baidu.com/view/54.htm)使用了函数，你可能对Step-Out cmdlet更感兴趣，Step-Out的工作方式和Step-Into一样，不过，如果调用了一个函数，Windows不会逐步执行，整个函数将会一次性执行。

**11 脚本**

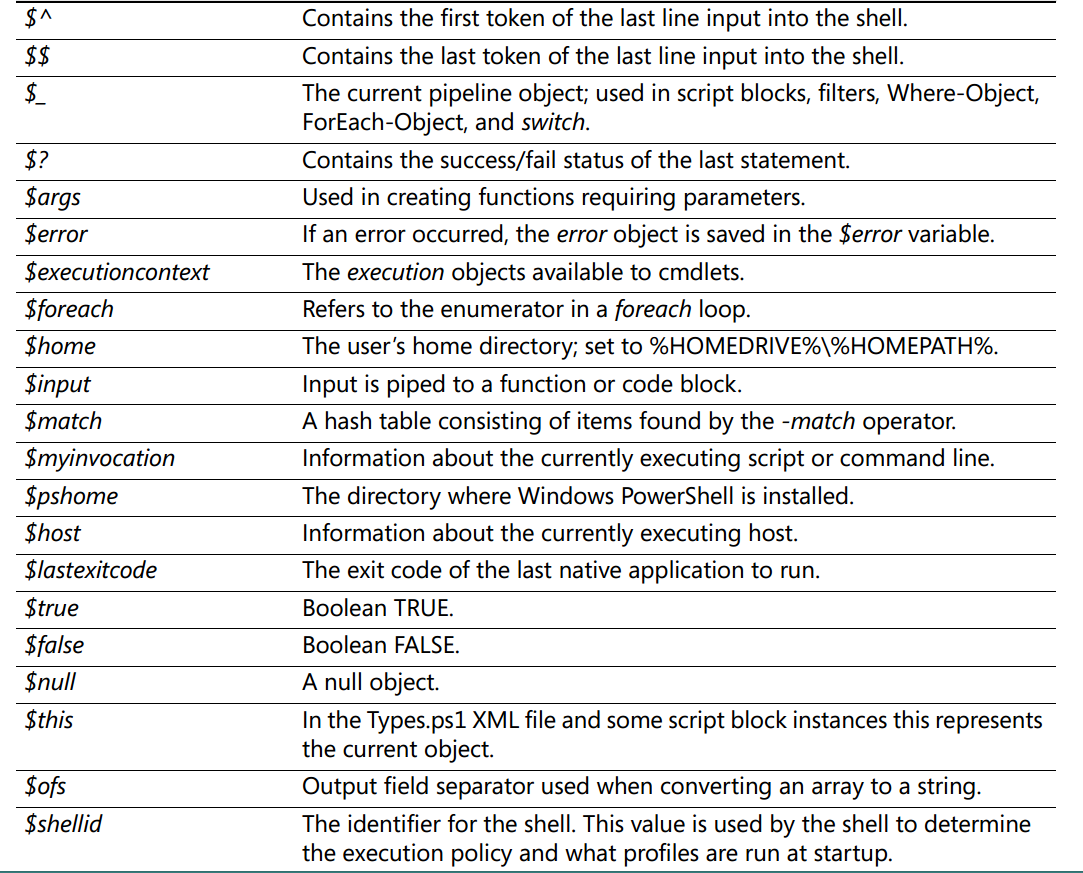
一系列命令的集合

**12 在powershell外执行脚本(注意 dot + space 和 & + ‘ )**

powershell.exe -noexit . f:\powershell\1.ps1

powershell.exe -noexit &'f:\power shell\1.ps1'

# 保留的关键变量值



# 常用命令：

查看帮助 ： get-help

查看命令 ： get-command

查找命令：get-command –verb xxx

# 日志

1：查看应用程序日志并导出

Get-eventlog application > f:\applog.txt

2:查看前十条应用程序日志

Get-eventlog application –logname application –newest 10

Or

(Get-eventlog application)[0]

# 服务

1：查询当前运行的服务并输出文件(txt、xml、csv)

Get-service | where-object {$\_.status –eq “running”} | out-file –flie “f:\mysercice.txt”

Get-service | where-object {$\_.status –eq “running”} | Export-Clixml -path “f:\mysercice.txt”

Get-service | where-object {$\_.status –eq “running”} | Export-Csv -path “f:\mysercice.txt”

2:停止服务

Stop-service servicename

3:启动一个服务

start-service –name servicename