



帐号 ☒ 自

密码



门户 家园 导读 淘帖 群组 排行 签到

帖子 热搜: 一级建造师 注册会计师 中级经济师 造价工程师 新东方 造价师 初级会计

我们无专区 Windows 操作系统 PowerShell & DOS Windows PowerShell 初学者入门教程

google 的服务

呃...找不到此网站。

我们无google的服务

如果确以尝试

我们无法连接至

返回列表 1 2 1

• 过

• 检正 18

• 如墙否

[推荐] Windows PowerShell 初学者入门教程 [复制链接]

发表于 2009-12-29 09:21 | 只看该作者 ▶

1楼 电

呃...找不到此网站。

我们无法连接至

PowerShell的概念性介绍我不太想说得太多, 简单概括几点:

1. 微软Windows操作系统最初的设计目标是面向非技术用户, 甚至是对计算机不感兴趣的人群, 因此, 真正自Shell一直不是微软公司所关心的问题. 随着Windows开拓市场, 用户的反馈等, 微软意识到, 通过图形化工具管理一台计算机或许是最完美的手段, 但是随着需要管理的计算机数量提升, 就必须依靠能够有效自动化计算机配置的工具, 这是开发PowerShell的一个重要原因.

2. 为什么不使用已有的shell或扩展cmd.exe呢? 我们熟知的bash, ksh是为unix, linux系统设计的, 优化的shell, 从这些操作系统内核提供的system call来看, shell将内核的特性真正表现出来. 但是, Windows操作系统和unix, linux是完全不同的, Windows将大部分管理通过面向对象的方式进行管理, 提供了诸如COM, WMI等概念. 传统的shell输出信息都是字符串, 这样有利于统一的处理, 但是字符串也是你唯一能够操作的对象. PowerShell开发队伍为了能够为Windows提供更好的优化, 因此选择了重新创建一门新的语言.

3. PowerShell最初的蓝本是Posix的shell标准, 并借鉴了大量的Perl语言中精华, 提供了一个基于对象的脚本环境. PowerShell v1(版本1.0)希望使用者与对象交互, 但没有真正提供面向对象创建的语法(但是, 借助PowerShell的机制, 可以自己实现这套语法). 并且大量特性与C#语言相同, 因此C#到PowerShell的转换是很容易的.

4. PowerShell有一些核心的特性, 这些特性是比较复杂的, 他们的存在是为了让传统shell用户能够尽量获得相似的用户体验. 对于普通用户, 你不需要理解, 掌握这些复杂的特性, 但是对于想深入理解, 掌握PowerShell的朋友, 这些特性对你将会产生巨大的帮助.

综上所述, 如果你用bash, perl, .Net, WMI, COM, VBscript的相关知识, 那么学习PowerShell的过程就会更短. 但是对于掌握bash, ksh的用户, 最困难的步骤是对问题的处理方式, 思维方式转变. 因为PowerShell中不仅仅有字符串, 因此, 尝试使用更加丰富的手段去处理问题, 往往阻挠大家的前进, 不过只要你有信心, 那么你一定克服这些不是困难的困难. 祝大家在学习PowerShell的过程中, 能够举一反三, 从软件工程等多种角度, 更好的理解计算机科学.

下面, 简单的贴出简单的语法及解释.

```
01. PS C:\> 5 + 100
02. 105
复制代码
```

在PowerShell中, 进行数学运算相当的简单, 将需要计算的表达式输入, 结果就会输出, 这里根本不需要什么打印语句, 执行的结果不会被丢弃掉, 而是将结果输出(以后, 我们会看到如何将结果丢弃).

```
01. PS C:\> "hello world!"
02. hello world!
复制代码
```

学习任何语言, 第一个代码总会是hello world, 和上面的数学计算一样, 直接输入即可.

```
01. PS C:\> (5 + 23 * 2) / 23
```

3003

关闭 帖子 金币

大家网大学二年级

积分 1843

发消息

02. 2.21739130434783

复制代码

对于, 四则混合运算, PowerShell也支持的很好, 能够使用()来修改运算符的优先级和结合性. 这里注意, PowerShell支持几种不同的数据类型, 如: int, float, double等. 大多数时候, 我们会在后面类型转换等做一些介绍.

01. PS C:\> (2+2)*3/7 > c:\foo.txt

02. PS C:\> type c:\foo.txt

03. 1.71428571428571

复制代码

除了将运算结果输出到显示器, 我们可以将结果存储到临时文件, 然后用type命令将文件的结果取回. 怎么样? 是不是和shell很相似呢?

01. PS C:\> \$n = (2+2)*3

02. PS C:\> \$n

03. 12

04. PS C:\> \$n / 7

05. 1.71428571428571

复制代码

除了将运算结果存储到文件, 我们可以通过变量赋值, 将运算结果存储到变量中, 并使用该变量进行后续的计算.

PS C:\> \$files = dir

PS C:\> \$files[3]

Directory: Microsoft.PowerShell.Core\FileSystem::C:\

Mode LastWriteTime Length Name

d-r-- 2007-7-26 21:25 Program Files

PowerShell是一个支持对象的语言, 我们可以简单的将命令返回的结果赋值给变量, \$files中包含了但前目录下的目录项的对象集合, 使用普通数组访问语法, 就可以获得该该位置的对象. 例子中显示了Program Files目录. 注意PowerShell中的数组下标是从0开始的. 这点与.Net Common Language Runtime完全一致.

大家在学习PowerShell时, 希望大家能坚持到最后. 坚持就是胜利!!

分享到: 微信新浪微博腾讯微博QQ空间人人网豆瓣网

呃...找不到此网站。

我们无法连接至

★ 收藏 1 ❤ 分享 1 📄 淘帖 ⬆ 支持 ⬇ 反对

shell, perl, python, ...

回复

使用道具 举报

👤 楼主 | 发表于 2009-12-29 09:27 | 只看该作者

2楼

如果要学一门语言, 你就必须对语言的语法, 特性有所了解. 只要对语言的各个环境了解, 才能更好编写, 调式, 部署你的程序. 因此, 教程第二部分将会着重于如下几个方面:

1. PowerShell, Shell, 脚本语言等概念;
2. 参数绑定;
3. 类型转换.

这些基础知识, 希望大家不要轻视, 因为后面的教程将会直接引用这些概念.



312 主题

444 帖子

3003 金币

大家网大学二年级



积分1843

发消息

首先, 我们先来调查PowerShell中最重要的元素: 命令(Command). 在PowerShell中, 命令分为四类: cmdlet, function, script和native Windows commands. 可能看到这四个英文名称会有些头大, 我们来仔细了解一下.

1. Cmdlet

在PowerShell官方blog上, 有篇关于Cmdlets和API的介绍. 因为PowerShell建立在.Net上, 再加上COM, WMI, ADO, XML等技术, 使得PowerShell对于.Net应用开发人员, 服务器管理人员来说都会非常方便, 通过这些已有的接口, 即可完成大量的任务. 但是, PowerShell的设计者明确的指出Cmdlets是PowerShell的灵魂.

Cmdlet是Command-Let的缩写. cmdlet这类命令有一种统一的命名方法: Verb-Noun, 也就是 动词-名词. 如果你是一名.Net程序员, 那么编程规范中应该也会提出, 对于方法(method)的命名, 最好使用动词-名词形式配合骆驼(Camel)命名法.

查看有哪些cmdlet参考如下:

```
PS C:\> get-command -CommandType cmdlet

CommandType Name Definition
-----
Cmdlet Add-Content Add-Content [-Path] <String[]> [-Value] <Object[...
Cmdlet Add-History Add-History [[-InputObject] <PSObject[]>] [-Pass...
Cmdlet Add-Member Add-Member [-MemberType] <PSMemberTypes> [-Name]...
Cmdlet Add-PSSnapin Add-PSSnapin [-Name] <String[]> [-PassThru] [-Ve...
Cmdlet Clear-Content Clear-Content [-Path] <String[]> [-Filter <Strin...
Cmdlet Clear-Item Clear-Item [-Path] <String[]> [-Force] [-Filter ...
Cmdlet Clear-ItemProperty Clear-ItemProperty [-Path] <String[]> [-Name] <S...
```

这里我省略了绝大部分的显示(^), PowerShell Version 1.0 微软官方发布的版本包含了

```
01. PS C:\> (get-command -CommandType cmdlet).count
02. 129
复制代码
```

129确实有点多, 但是实际上在学习过程中你会发现学习他们的过程比你学习unix shell中命令更加容易些, 原因稍后为你解答.

cmdlet是可以进行扩展的, 如果你想开发自己的cmdlet, 你需要下载PowerShell SDK. 目前, 我只知道该 SDK 包含在了Windows Vista Platform SDK中. 下载该 SDK, 需要进行正版验证. 在伴随着教程的编写过程, 我也会阅读PowerShell SDK中关于PowerShell的介绍, 并尽量用最简单的语言将需要注意的内容, 传达给大家.

cmdlet的开发并不复杂, 可以使用C#作为实现语言, PowerShell SDK已经实现了好了cmdlet的一个基类, 开发的cmdlet只需要继承该基类. 通过这种开发模式, 最大的优点:

(1). 所有的cmdlet中包含了一些公共参数(common parameters), 例如: -Verbose, -Debug, -ErrorAction, -ErrorVariable, and -OutVariable. 这些参数大多用于一些脚本调试等.

(2). 所有的cmdlet中的参数(parameters)具有相似的参数, 相同的类型, 甚至具有相同的性质. 对于接受输入的某些cmdlet来说, 输入的参数就叫做-InputObject, 参数类型一般是[object[]], 而通常都会具有从管道线读取处理对象的属性.

这种一致性带来的最大好处就是, 对于一个命令, 往往你只需要关注命令的特性, 不需要记住复杂繁多的参数. 如果你熟悉unix shell下的utils core tools那么我相信, 例如awk使用-F指定一个支持正则表达式的与分隔符, cut使用-d来分割, sort使用-t进行分割对你来说是痛苦的. 在PowerShell中, 你甚至不需要对cmdlet产生的输出进行分割. 这点你会在稍后看到.

呃...找不到此网站。

我们无法连接至googleads.g.doubleclick.net的服务器。

如果确定此网址正确，您可以尝试：

cmdlet编译后的结果不是可执行文件, 而是dll文件, PowerShell启动时, 将这些命令加载. 这些命令执行效率最高, 因为他们在PowerShell时被载入PowerShell进程内.

2. 函数(function)

函数, 准确的说就是: 有名称的代码块(scriptblock). 下面就是一个简单的函数定义:

```
01. PS C:\> function Get-DayToBeijingOlympic
02. >> {
03. >> ([datetime] "2008-08-08" - [datetime]::Now).Days;
04. >> }
05. >>
06. PS C:\> Get-DayToBeijingOlympic
07. 370
```

复制代码

细节大家可以不用关心, 稍后都会讲到. 函数就是一段你输入到PowerShell的命令集合.

但是函数在被定义后, 才存在于PowerShell内存中, 当PowerShell退出时, 就会消失.

效率上, 函数在第一次调用时需要编译, 因此第一次调用速度较慢.

3. 脚本(script)

PowerShell在交互式Shell(interactive Shell)和脚本语言(script Language)之间进行了平衡, 提供了执行脚本的能力. 脚本类似于函数, 存放在文件中, 调用时由PowerShell载入内存, 编译并执行.

效率上来讲, 脚本慢于函数(function), 主要原因在于函数只会在第一次调用时被编译, 而脚本每次调用都会被编译一次. 但是编译后的执行阶段, 他们的性能是近似相等的.

4. native Windows command

我不知道应该这个东西用什么名字. 这类命令在Windows世界大量存在, 譬如: dir, findstr, del, ping. 这些命令就是Windows之前的可执行文件(非.Net命令行可执行文件). 在PowerShell中调用这些命令效率是最低的, 因为执行他们PowerShell需要创建一个新进程. 此外因为这些命令早于PowerShell, 因此他们的信息输出都是基于文本的(也可以说是字符串), 因此丧失了PowerShell对对象的处理能力. 因此, 如果你知道PowerShell的实现, 就避免使用这些命令.

讨论完命令, 我简单介绍下别名(alias):

```
01. PS C:\> (Get-Command -CommandType alias).count
02. 101
```

复制代码

你一定在想, PowerShell为什么提供了这么多别名, 难道要累死我们??

PowerShell为了方便使用windows和unix, linux的人们, 支持了两套别名, 譬如dir相对于ls, 还有unix, linux的grep等. 这样无论你使用windows还是linux都能相对来说快速的上手PowerShell.

PowerShell Version 1.0中有一个缺陷, 既定义别名时, 不允许指定参数. 这无疑是PowerShell的一个缺陷. 但是因为cmdlet中参数较少(一般都是10个以内), 因此大多数时候, 你不需要定义带有参数的别名, 如果真的要, 可以用函数来替代. 这种缺陷, 与代码块的执行等功能有冲突. 希望在后续版本能有别名功能能够更加强大.

写在最后:

如果您有幸坚持看了我的啰唆, 我非常的感激. 接下来的3, 4讲也会偏向于概念而非更多的语法. 任何事, 请相信你对解释器行为的理解, 对语言的特性的理解, 都会让你在编写该语言代码时受益匪浅. 因此, 请坚持, 努力的去学习. 我希望在这份漫长教程过程中, 带大家更多计算机历史, 发展; 不同语言, 技术的对比; 编写健壮代码(无论是像C, C#还是像sh, PowerShell)的思路. 在03, 04中, 我会介绍两个关键核心的概念(算法??), 参数绑定和类型转换.


学习一门语言, 还能学到编程的心得, 这也是我希望的.

shell, perl, python, ...

回复支持反对

举报

scripts

 楼主 | 发表于 2009-12-29 09:34 | 只看该作者

3楼

PowerShell数据类型之整型



312 444 3003
主题 帖子 金币

大家网大学二年级
👤👤👤

积分 1843
发消息

关闭

因为一些原因, 之前的教程之写了两篇, 教程将会从现在继续开始, 并且不会再发生间断, 尽量做到每周能够发两篇. 在02中已经为大家介绍了一些基本概念, 所以从03开始, 将真正接触使用PowerShell的例子.

PowerShell是建立在.Net Framework之上的, .Net Framework的好处和坏处争论不休, 多数人认为.Net Framework最大的弊端在于性能问题. 这句话的确不假, 但是目前计算机硬件无论从速度还是存储空间来说, 都为.Net Framework的推广提供了足够的硬件环境. 所以我们不需要害怕.Net Framework带来的性能损失, 相反, 我们应该能够认清.Net Framework究竟能够让我们在实现一些复杂任务中节省了多少人力.

我相信这样一句话: 我们应该学会花时间来节省时间. 这话的意思类似于: 磨刀不误砍柴工. 也就是说, 有时候我们虽然花费了时间, 但实际上我们确节省了时间. 这也是一些中型和大型企业需要完善, 甚至说琐碎的管理制度一样. 另一个例子就是计算机中的操作系统, 如果每个程序自己去操作硬件(假设是可行的), 那么每个人都需要做很多同样的事情, 不如将这些操作交给操作系统去处理, 我们利用操作系统提供的更加友善的接口, 完成更复杂的功能.

PowerShell充分利用了现有的.Net Framework环境, 虽然PowerShell的开发者也承认PowerShell中还有很多很多没有完成的部分, 但是这些部分几乎都可以通过.Net Framework来解决. 在学习PowerShell的过程中, 你将会了解更多关于.Net Framework中的知识, 能够让你完成更多用其他工具无法完成的工作.

好了, 废话差不多了, 我们先来学习最常用的数据类型: 整数. 你只要在PowerShell.exe中随便输入一些数字就创建了整数对象:

```
01. PS C:\Users\Eden> 13800138000
02. 13800138000
复制代码
```

PowerShell不需要像其他语言一样什么print之类的命令才能输出信息, 相反, 多数情况下, 你想到的结果就会展现在屏幕上, 就好像上面输出了整数: 13800138000. 你需要记住, PowerShell中任何东西都是对象(Object). 对象是一种包含属性和方法的实体. 实体可以想象每个数值就好像每个人或者动物一样, 是真实存在的物体.

你可以通过使用点"."来访问对象上的方法, 这些方法能够帮助你完成一些非常基础的操作. 我们可以使用下面的办法查看数字的类型:

```
PS C:\Users\Eden> 13800138000.GetType()
数字常量无效: 13800138000.GetType。
所在位置
行:1
字符: 19
+ 13800138000.GetType <<<< ()

PS C:\Users\Eden> (13800138000).GetType()

IsPublic IsSerial Name BaseType
-----
True True Int64 System.ValueType
```

注意, 第一种方法由于实现问题(词法分析或语法分析), PowerShell会产生错误信息. 我们可以在数值外面加上括号来避免. 我们可以看到PowerShell输出了很多信息, 大家只要注意看Name对应的Int64就好了, 它表名13800138000是一个64位的整数类型(.Net Framework中是System.Int64). 我们可以使用cmdlet: Get-Member来查看一个对象上的成员:

呃...找不到此网站。

我们无法连接至
googleads.g.doubleclick.net
的服务器。

如果确定此网址正确，您可以尝试：

关闭

关闭

关闭

```
PS C:\Users\Eden> 13800138000 | Get-Member | ft Name,MemberType -AutoSize
```

Name	MemberType
-----	-----
CompareTo	Method
Equals	Method
GetHashCode	Method
GetType	Method
GetTypeCode	Method
ToString	Method

Name表示了成员的名称, MemberType表示成员的类型. 这里我们看到了6个方法成员. 其中的Get-Type方法就是我们刚才例子中查看类型使用的, Get-Type可以返回一个表示类型的对象(过于细节了, 大家可以忽略).

因为数值13800138000已经很大了, 所以PowerShell选择了Int64来表示, 一般情况下PowerShell使用Int32表示大多数整数:

```
01. PS C:\Users\Eden> (0).GetType().FullName
02. System.Int32
复制代码
```

Int32使用4个字节表示数值, 而Int64使用8个字节表示数值, 它们的范围是有限的:

```
01. PS C:\Users\Eden> [System.Int32]::MaxValue
02. 2147483647
03. PS C:\Users\Eden> [System.Int32]::MinValue
04. -2147483648
05. PS C:\Users\Eden> [System.Int64]::MinValue
06. -9223372036854775808
07. PS C:\Users\Eden> [System.Int64]::MaxValue
08. 9223372036854775807
复制代码
```

如果两个Int32类型的数值的和或差超出了Int32的表示范围, PowerShell能够自动进行类型转换:

```
01. PS C:\Users\Eden> ([System.Int32]::MaxValue + 1).GetType().FullName
02. System.Double
复制代码
```

默认PowerShell使用双精度浮点数 (Double)来表示计算的结果, 单精度浮点数(Float)是不提倡使用的, 主要原因在于其表现精度太低, 而且现在计算机内存容量较大, 因此一般情况下, 应该避免使用单精度浮点数. 如果两个整数不能整除, PowerShell也会聪明的选择浮点数来避免精度信息的丢失:

```
01. PS C:\Users\Eden> 19 / 3
02. 6.33333333333333
03. PS C:\Users\Eden> (19 / 3).GetType().FullName
04. System.Double
复制代码
```


今天为大家介绍到这里, 在下一篇文章中, 我会为大家介绍面向对象中的一些知识. 包括类, 实例属性, 实例方法, 静态属性, 静态方法等内容. 这些内容比较难, 大家先简单了解, 慢慢通过PowerShell的学习了解面向对象的一些深入知识. PowerShell是探索.Net世界的有力工具哦~

shell, perl, python, ...

回复 支持 反对

举报

scripts

 楼主 | 发表于 2009-12-29 10:00 | 只看该作者

4楼

PowerShell面向对象基础之一

面向对象编程的引入主要是为了解决软件复杂化带来的维护等问题, 早在20世纪60年代, 面向对象就已经出现了. 虽然C语言并不支持面向对象的特性, 但是使用C语言进行开发的程序人员已经将面向对象的核心思想应用到其中, 因此我们更应该理解的是面向对象的思想, 而不需要纠缠于语言本身。

关闭

关闭

scripts



312主题

444帖子

3003金币

大家网大学二年级

👉👉👉

积分1843

发消息

PS C:\Users\Eden> [int] | Get-Member -Static | Out-String -Width 80

TypeName: System.Int32

Name	MemberType	Definition
-----	-----	-----
Equals	Method	static System.Boolean Equals(Object objA, Object o...
Parse	Method	static System.Int32 Parse(String s), static System...
ReferenceEquals	Method	static System.Boolean ReferenceEquals(Object objA,...
TryParse	Method	static System.Boolean TryParse(String s, Int32& re...
MaxValue	Property	static System.Int32 MaxValue {get;}
MinValue	Property	static System.Int32 MinValue {get;}

我们可以看到，System.Int32上有MaxValue和MinValue两个静态属性，它们指示了System.Int32类型的值域：

01. PS C:\Users\Eden> [int]::MaxValue
02. 2147483647
03. PS C:\Users\Eden> [int]::MinValue
04. -2147483648

复制代码

如果没有指定Static参数，那么Get-Member会显示对象的实例方法：

PS C:\Users\Eden> [int]::MinValue | Get-Member | Out-String -Width 80

TypeName: System.Int32

Name	MemberType	Definition
-----	-----	-----
CompareTo	Method	System.Int32 CompareTo(Int32 value), System.Int32 Comp...
Equals	Method	System.Boolean Equals(Object obj), System.Boolean Equa...
GetHashCode	Method	System.Int32 GetHashCode()
GetType	Method	System.Type GetType()
GetTypeCode	Method	System.TypeCode GetTypeCode()
ToString	Method	System.String ToString(), System.String ToString(IForm...

shell, perl, python, ...

回复支持反对

举报

楼主 | 发表于 2009-12-29 10:02 | 只看该作者

6楼

PowerShell中格式化命令和输出命令

好久不见, 我又食言了, 一直没有更新教程, 我也不想找借口, 因为我花了很多时间跑去玩大蛇无双和魔王再临.

今天, 我将为您介绍如何使用格式化和输出命令. 在PowerShell中, 负责进行格式化和输出的是三类cmdlet. 他们分别是: Format-*, Out-*和 Write-*. 顾名思义, Format-* 主要是用来对信息进行格式化操作, Out-*用于指定输出设备, 而 Write-* 为 PowerShell 宿主 (Host) 输出的更多有用信息提供了快捷访问接口.

我们要将对象输出到控制台或打印机等位置前, 我们首先必须将对象格式化成为字符串的形式. 一个对象可能包含几十种属性信息, PowerShell的开发者已经考虑到我们在日常使用时, 不会使用一个对象的所有信息, 因此默认情况下只保留该对象上最主要的几种属性. 让我们来看看进程对象默认会显示哪些属性. 首先看看进程对象究竟有多少个属性:

01. PS C:\> (Get-Process Idle | Get-Member -MemberType Property).Count
02. 51

复制代码

在上面例子, 首先使用Get-Process命令返回表示Idle进程的对象, 并通过Get-Member命令返回该对象上所有属性, 最后我们给出该对象的属性个数 (打印这些属性太地方了, 你可以把Count去掉, 检查返回的结果). 让我们看看PowerShell默认情况下会输出哪些属性吧:


```
PS C:\> Get-Process Idle
Handles NPM(K) PM(K) WS(K) VM(M) CPU(s) Id ProcessName
-----
0 0 0 16 0 0 Idle
```

正如前面所介绍的，PowerShell默认只会返回对象上一部分属性信息，究竟显示哪些属性是通过定义在ETS系统中的信息决定的。

Format族包含了：Format-Custom、Format-List、Format-Table和Format-Wide四个命令。在大多数情况下，Format-Table是PowerShell默认使用的格式器。Format-List的输出类似上面输出进程属性的格式，每个对象将会占一行。一行内包括该对象的多个主要属性，如果该属性信息太长，PowerShell就会将该信息截断。让我们看几个管理PowerShell最常用的例子：

```
PS C:\> Get-EventLog -LogName 'Windows PowerShell' -Newest 10 | Format-Table
Index Time      Type      Source      InstanceID Message
-----
1029 五月 31 21:38 Information PowerShell 400 Engine state is changed from None
to Available....
1028 五月 31 21:38 Information PowerShell 400 Engine state is changed from None
to Available....
1027 五月 31 21:38 Information PowerShell 600 Provider "Certificate" is Started. ...
1026 五月 31 21:38 Information PowerShell 600 Provider "Variable" is Started. ...
1025 五月 31 21:38 Information PowerShell 600 Provider "Registry" is Started. ...
1024 五月 31 21:38 Information PowerShell 600 Provider "Function" is Started. ...
1023 五月 31 21:38 Information PowerShell 600 Provider "FileSystem" is Started. ...
1022 五月 31 21:38 Information PowerShell 600 Provider "Environment" is Started. ...
1021 五月 31 21:38 Information PowerShell 600 Provider "Alias" is Started. ...
1020 五月 31 15:43 Information PowerShell 403 Engine state is changed from
Available to Stopp...
```

这个例子显示PowerShell最近10个日志的日志信息。Format-Table为了保证每个对象只输出一行，只会输出那些较重要的信息，一般对对象的检索，使用Format-Table就是最好的办法。

其他几个命令，您可以参照PowerShell的帮助，或我翻译的帮助信息详细学习。

看完了负责格式化对象输出信息的cmdlet，我们需要考虑该把这些信息输出到什么位置了。Out族包括了：Out-Default、Out-File、Out-Host、Out-Null、Out-Printer、Out-GridView和Out-String。

Out-Host命令，就是将显示结果返回给宿主程序，由宿主程序展现给使用者。Out-Default默认就是使用Out-Host进行输出，但是根据 PowerShell文档的描述，Out-Default用来提供一种占位符的特性，你可以在脚本完成后，重新修改Out-Default来控制整个程序的输出。

Out-Null的作用就是将输出丢弃，在很多时候，如果我们不关心命令的返回结果，而只关一条命令产生的某些影响时，我们可以将该命令的输出通过管道发送给Out-Null。例如：

关闭

关闭

scripts



312主题

444帖子

3003金币

大家网大学二年级



积分1843

发消息

```
PS C:\> New-Item TestIt -Type dir

Directory: Microsoft.PowerShell.Core\FileSystem::C:\

Mode                LastWriteTime         Length Name
----                -
d-----          2007-9-16   21:10           TestIt

PS C:\> New-Item TestIt -Type dir | Out-Null
New-Item : Item with specified name C:\TestIt already exists.
At line:1 char:9
+ New-Item <<<< TestIt -Type dir | Out-Null
PS C:\> Remove-Item TestIt
PS C:\> New-Item TestIt -Type dir | Out-Null
PS C:\>
```

我们首先创建了TestIt的目录，默认情况下，New-Item返回新创建的目录，该对象被输出到控制台上。我们接下来试图再创建一次该目录，并将输出信息丢弃。然而，由于该目录已经存在，因此创建失败，命令给出了错误提示。这里我们需要注意错误信息并没有被丢弃！接下来我们删除目录TestIt，再次执行创建命令，这次，屏幕上没有显示创建的目录信息。

最后，我们来看看Out-File。很多时候，我们需要将某种编码的字符串存储为其他编码形式。我们可以简单的使用Out-File的参数 Encoding来设置输出信息的编码格式。这些编码包括了：“Unicode”、“UTF7”、“UTF8”、“UTF32”、“ASCII”等等。有时候，我们还会遇到如果将信息写入文件中。如果该文件已经存在，我们可能希望信息继续追加到文件的末尾，或者当文件已经存在时，我们希望覆盖这个文件。通过参数Append，我们可以告诉Out-File究竟是覆盖文件，还是在文件的末尾进行追加。让我们看看简单的例子：

```
PS C:\> "我是第一行!" | Out-File append.txt
PS C:\> Get-Content C:\append.txt
我是第一行!
PS C:\> "我是第二行,但是我要试试覆盖第一个行!!" | Out-File append.txt
PS C:\> Get-Content C:\append.txt
我是第二行,但是我要试试覆盖第一个行!!
PS C:\> "我是第三行,让我跟在第二行后面吧!!!" | Out-File -Append append.txt
PS C:\> Get-Content C:\append.txt
我是第二行,但是我要试试覆盖第一个行!!
我是第三行,让我跟在第二行后面吧!!!
```

Write族的命令我将会在下一讲为大家详细介绍. 希望大家喜欢我的教程.

shell, perl, python, ...

回复支持反对举报

楼主 | 发表于 2009-12-29 10:04 | 只看该作者7楼

PowerShell中Write族cmdlet的介绍和使用

Write相关的命令大都是和宿主相互交互的，例如：输出调试信息、输出错误对象、显示进度信息、控制输出效果等。

在很多时候，我们希望能够将一些信息以特殊的颜色或背景色输出。在PowerShell的控制台中，我们可以使用Write-Host命令来实现这个功能。参数ForegroundColor用于指定字体颜色，BackgroundColor用于指定背景的颜色。例如：

```
01. PS C:\> Write-Host -BackgroundColor Black -ForegroundColor Red "Hello World!"
02. Hello World!
复制代码
```

如果你记不住有哪些颜色该怎么办呢？很简单，只需要在该参数后面输入一个不存在的颜色即可，例如：

关闭

```
PS C:\> Write-Host -BackgroundColor NoSuchColor
Write-Host : Cannot bind parameter 'BackgroundColor'. Cannot convert value "NoSuchColor" to
type "System.ConsoleColor"
due to invalid enumeration values. Specify one of the following enumeration values and try again.
The possible enumerat
ion values are "Black, DarkBlue, DarkGreen, DarkCyan, DarkRed, DarkMagenta, DarkYellow, Gray,
DarkGray, Blue, Green, Cy
an, Red, Magenta, Yellow, White".
At line:1 char:28
+ Write-Host -BackgroundColor <<<< NoSuchColor
```

我们可以看到PowerShell的错误提示信息已经包含该参数的所有合法取值。利用这个办法，在很多时候，都可以避免去查看手册或查阅其他资料来确定参数所支持的值。

Write-Debug、Write-Warning和Write-Verbose是三个非常类似的命令,它们都是向控制台输出一条消息，它们主要为了脚本编写者能够更加灵活的控制输出。传统的shell和脚本语言，经常通过向屏幕打印一些调试信息来帮助编写者排除脚本的错误。PowerShell虽然提供了强大的调试功能，还通过提供Write-Debug为大家保留了利用打印消息来调试这一经典方法的功能。

在进行拷贝操作时，如果某个已经存在的话，可能并不希望停止拷贝操作，但是期待着PowerShell能够提供一个警告信息。此时使用Write-Verbose
如果使用过压缩类软件的话，我们可能会关心压缩的时候，究竟哪些文件被打包了，这时候我们往往希望一个程序执行时，线程更加详细的消息。我们可以利用Write-Verbose来达成这个目的。

这几个Write命令实际使用频率不高, 因此我将它们放到以后专门介绍如何编写健壮脚本章节中。

如果，您正在编写一个拷贝文件或执行安装的脚本，提供一个进度条将会使用户再等待脚本执行时获得更好的体验。PowerShell的设计人员已经为我们想到了这些细节，可以使用Write-Progress来完成这样的操作。

让我们来看一个最简单的进度条模型：

```
01. PS C:\> for ($i = 0; $i -lt 100; $i++) { Write-Progress -Activity "Learning PowerShell" -Status
"Percentage: $i" -PercentComplete $i; Start-Sleep -Milliseconds 50 }
复制代码
```

你将会在屏幕上看到：

```
Learning PowerShell
Percentage: 61
[oooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
]
```

我们使用参数-Activity来为整个的进度条设置一个名称（活动的名称），这里我们定义为“Learing PowerShell”，参数-Status用于指示在整个过程中，当前执行的状态，这里我们简单的提示了当前完成的百分比。

今天就这么多了，希望大家回去自己看看Get-Help获得的帮助信息. 每个命令都有些例子, 大家仔细学习一下。

shell, perl, python, ...

回复支持反对

举报

楼主 | 发表于 2009-12-29 10:05 | 只看该作者

8楼

字符串字面值

字符串类型在PowerShell的使用中频率也非常高，对比其他shell来说，虽然PowerShell使用字符串的情况变少了，但是在功能上甚至可以说得到了增强。主要的原因可以归功于.Net Framework类库。System.String类中包含了各式各样的方法，让使用者可以方便的处理字符串。

在PowerShell中，字符串就是.Net Framework中的16-bit Unicode字符序列。因此，日常使用时，可以正确处理大部分常用的各国字符。

关闭



3124443003

主题 | 帖子 | 金币

大家网大学二年级



积分 1843

发消息

说起字符串，就必须要解释引号规则，引号规则往往总是与两个概念有关：特殊字符的保护以及变量代换。

变量代换有时候也被为变量展开或变量篡改，通过这种手段，可以简化字符串表示，使脚本的读者或维护者更加方便的理解代码。例如：

```
01. PS C:\> $a = 'Windows PowerShell'
02. PS C:\> "Hello, $a"
03. Hello, Windows PowerShell
复制代码
```

通过将变量名称替换成变量的值，使变量能够在字符串展开成原始值的过程就是变量代换。

单引号

单引号的功能有时候会让人爱不释手，因为单引号中任何字符都只表示自己。换句话说来说，单引号中不会进行变量代换，也不会对任何转义序列进行处理，你看到的“几乎”就是你输入的字符串。之所以我用“几乎”这副词来强调，是因为引号规则中 有个特例，如果需要在单引号的字符串中包括单引号该怎么办呢？为了解决这种情况，Windows PowerShell采用了类似 CSV文件中的解决方法：通过使用连续两个引号来表示一个引号。下面是几个单引号字符串的例子：

```
PS C:\> $shells = 'PowerShell' # 设置变量$shells为字符串PowerShell
PS C:\> 'What'up, $shells' # 两个连个连续的单引号表示一个单引号
What'up, $shells
PS C:\> 'What'up, $shells' # 因为单引号不匹配，引起错误
Unexpected token 'up' in expression or statement.
At line:1 char:9
+ 'What'up, <<<< $shells'
```

第一个例子中，我们使用了两个连续的单引号来表示在字符串中应该包含一个单引号。第二个例子中展示了如果缺少单引号，就会导致引号不匹配，产生错误信息。另外需要注意的是在单引号中\$shells变量没有被代换。

双引号

双引号，可以形容他是个变色龙，如果双引号包含变量，则可以随着脚本的执行，将结果动态的附加进入。双引号中允许：变量代换和转义序列。说到这里，必须要唠叨一下Windows PowerShell中怪异的转移序列。在常见的语言中，例如：C、C#、Java、Perl和Python等。转义字符都是反斜线“\”(HTML和 XML属于例外)。由于Windows操作系统历史原因，反斜线“\”一直被用作目录分隔符。如果PowerShell使用反斜线作为转义字符，就会产生两个可怕的情况：要么在PowerShell中用正斜线“/”作为分隔符，要么每次都需要连续使用两个反斜线进行转义处理。无论上面哪种情况，PowerShell要么与Windows的定义不一致，要么用户使用时就会抱怨输入太多的反斜线。最后PowerShell团队决定：转义字符选择反引号（backtick或者back quote）“`”，反引号位于数字键1的左边。转义序列如下所示：

```
`' -- Single quote
`" -- Double quote
`0 -- Null
`a -- Alert
`b -- Backspace
`f -- Form feed
`n -- New line
`r -- Carriage return
`t -- Horizontal tab
`v -- Vertical tab
```

虽然这些长相怪异的转移序列可能吓到你了，但是，相信它不会阻碍您学习PowerShell。让我们来双引号中的变量代换以及转移序列使用。

```
PS C:\> $shell = "PowerShell"
PS C:\> "Who are you?"`n$shell"
Who are you?
PowerShell
PS C:\> "`$shell is $shell"
$shell is PowerShell
PS C:\> "我们来看看重叠双引号的效果""
我们来看看重叠双引号的效果"
```

第一个例子中，我们使用换行的转移序列，也使用了变量代换。第二例子，我们使用“\$”来输出变量和变量的值。最后我们看到了与单引号相同的特性，如果连续两个双引号表示一个双引号。下面的例子，我们看看单引号和双引号的对比，以及相互包含时的特性：

```
01. PS C:\> "`'$shell is $shell'"
```

关闭

关闭

scripts

312主题

444帖子

3003金币

大家网大学二年级

👍👍👍

积分1843

发消息

02. '\$shell is PowerShell'

03. PS C:\> '\$shell is \$shell'

04. '\$shell is \$shell'

复制代码

如果大家已经吸收了前面所介绍的知识，那么这个例子的结果应该非常好解释。我们对单引号双引号做个简单的概括：1. 单引号中不支持变量代换和转义序列；双引号中允许变量代换以及转义序列。2. 根据最外层的引号决定字符串如何被处理。3. 连续重复两次最外层引号表示一个引号。只要这三条记住，引号规则就不会成为困难。

shell, perl, python, ...

回复

支持

反对

举报

👤 楼主

| 发表于 2009-12-29 10:07

| 只看该作者

9楼

PowerShell中的数组使用

在日常处理中，除了使用像“数值类型”和“字符串类型”外，还需要使用能够包含其他对象的“集合”类型（请注意，这里的集合对应英文是Collection，而非数学概念上的Set。）。大多数常见语言，都提供一些操作集合类型的语法。最基本的集合类型就是数组类型，它提供了一种下标基于0的数组对象。

首先通过几个简单的例子，来了解一下PowerShell中的数组是如何使用的。

PS C:\> 1,'a',2MB,0x100
1
a
2097152
256

上面的小例子输出了4个对象：数值“1”、字符串“a”（注意：虽然只有一个字符，但这里仍然是一个字符串）、数值“2MB”和十六进制数值“0x100”。也许您很想了解这个例子和数组究竟有什么联系，实际上这个例子清楚地解释了PowerShell中数组类型的语法。在PowerShell中，只要把不同的对象用逗号","连接起来，就可以构造出数组对象。本章节的重点在于向大家介绍数组类型的使用，因此将在后续的章节详细介绍逗号表达式","的使用。我们回到刚才的例子中，虽然这4个对象组成了PowerShell中的数组对象，但是如果没有把这个对象存储起来或者丢掉的话，PowerShell就会利用默认的格式器和输出器将它们输出到控制台上。最终我们就看到了上面四行输出结果。

现在我们知道了PowerShell中构造数组的最简单的方式，接下来让我们一起探索更多关于数组对象更多的特性：

PS C:\> \$array1 = 1,2,3,4
PS C:\> \$int = 1
PS C:\> \$array1.count
4
PS C:\> "\$array1"
1 2 3 4
PS C:\> \$int.count
PS C:\> "\$int"
1

细心的读者肯定已经发现，在第一个例子中，使用了逗号构造了一个包含四个整数的数组，并将其存储在变量\$array1中。接下来将数值1存储在变量\$int中。数组对象上有一个属性Count用来指出当前数组中的对象数量，我们可以看到\$array1中包含了4个对象。但是在\$int上确没有该属性（注意：如果引用的属性不存在，PowerShell返回\$null对象，该对象不会对控制台产生任何影响，只是简单的使提示用户继续输入命令）。

PowerShell是基于对象的shell，现在让我们来看看数组究竟是什么对象：

PS C:\> \$array1.GetType().FullName
System.Object[]

在PowerShell中调用对象上的GetType()方法，可以得到该对象的类型对象（类型也是一个对象）。而该对象上的FullName属性就是该对象表示的类型的名称。这里有点绕口令了，总之如果你想知道一个对象的类型就调用GetType()方法，再访问FullName属性就可以了。

学会了创建数组以及知道了数组包含的对象数量，现在来学习一下如何访问数组中的元素。实际上PowerShell的数组其实通过.Net类库中的数组来实现的，也就是说PowerShell在.Net类库原有的实现上，增加了一些PowerShell便捷的访问接口，而实际操作的对象还是.Net类库中的对象。使用数组时，需要大家记住数组的第一个元素的下标是“0”、数组的最后一个元素下标是数组长度减去1。例如：

PS C:\> \$array

第13页 共16页

2021/1/2 18:11

```
= 1,2,3
PS C:\> $array.Length
3
PS C:\> $array[0]
1
PS C:\> $array[2]
3
PS C:\> $array[4]
PS C:\>
```

实际上，Count是属性Length的别名，这主要是为不同对象提供一致性接口的技术。这里需要特别注意的是，当数组访问越界时，PowerShell是不会给出任何错误信息的，只会得到一个\$null对象。

如果我们已经有一个数组了，应该如何向这个数组中添加新的元素呢？在Perl中，你只需简单的使用你想使用的索引，然后存储上你想存储的数值就完成了任务。然而在PowerShell中，如果你使用类似的方法就会得到如下的结果：

```
PS C:\> $a
=
"My ThinkPad", "My PC"
PS C:\> $a[0]
My ThinkPad
PS C:\> $a[1]
My PC
PS C:\> $a[2] = "My Mac"
```

数组赋值失败，因为索引“2”超出范围。
所在位置 行:1 字符: 4
+ \$a[2 <<<<] = "My Mac"

PowerShell的错误信息指出：索引越界。解决这个的办法其实很简单，只要使用下面的代码就可以巧妙解决：

```
PS C:\> $a
=
$a
+
"My Mac"
PS C:\> $a
My ThinkPad
My PC
My Mac
PS C:\> $a[2]
My Mac
```

在PowerShell中，数组其实是一个大小固定的数据结构，如果需要向数组中添加更多的对象时，就必须使用加号“+”运算符。在执行“+”操作的过程中，PowerShell实际上进行了下述操作：

- 1) 首先创建一个新的数组，该数组的大小能够存下运算结果的所有对象；
- 2) 将第一个数组的成员拷贝到新数组中；
- 3) 将第二个数组的成员拷贝到新数组中。

通过观察这个操作过程，很容易发现：如果数组中包含元素很多，而且频繁的向数组添加对象，一定会使PowerShell执行脚本时更慢。因此，我们需要谨记：大多数时候，数组是PowerShell中的便捷的工具，但是应该避免频繁地对其进行添加操作。在.Net类库中提供了其他的数据结构来满足需要经常向数组中添加对象的需求。

shell, perl, python, ...

scripts

 楼主 | 发表于 2009-12-29 10:08 | 只看该作者

10楼

PowerShell中的HashTable的使用

哈希表 (hashtable) 有时候也被称为：“关联数组”或“字典”。哈希表可以称得上是计算机科学中最重要的数据结构之一，例如：在计算机操作系统、数据库系统、编译器、加密算法等计算机底层程序中，哈希表都发挥着重要的作用。哈希表提供以近乎常数时间开销，根据数据的键 (key) 来索引到该键对应的值 (value)。就好像使用工具书一样，我们总是通过目录中项的名称和页码，来检索我们关心的知识或信息。

在传统的Unix环境中，awk程序内置的关联数组，为数据处理提供了便捷的手段。因此掌握哈希表类型，可以使您在解决工

312

444

3003

主题

帖子

金币

大家网大学二年级



积分1843

发消息

作中的问题时候，更加得心应手。

如何声明一个哈希表对象？哈希表的创建是非常简单的，请看如下示例：

```
PS C:\> $hash = @{}
PS C:\> $hash.GetType().FullName
System.Collections.Hashtable
PS C:\> $hash.Count
0
```

示例中首先使用了"@{}"语法初始化了一个空的哈希表，并将其保存在变量\$hash中。接下来验证了该对象的类型，并查看了哈希表中元素的数量，由于只是初始化了一个空哈希表，所以返回结果是0。如果在初始化时，需要添加一些键值对到哈希表的话，应该使用什么样的语法呢？我们只要在"@{"和"}"标记之间输入以分号","分隔的键值对即可，键值对的格式：<key> = <value>。请看如下示例：

```
PS C:\> $hash = @{ "Computer Name" = "AD Server";
>> "Administrator" = "Ma Tao", "Spider Man";
>> "OS" = "Windows 2008";
>> "Installed Date" = Get-Date;
>> "Disk Size" = 5000GB
>> }
>>
PS C:\> $hash
Name                Value
----                -
OS                  Windows 2008
Disk Size           5368709120000
Installed Date      2008/7/31 23:09:57
Computer Name       AD Server
Administrator       {Ma Tao, Spider Man}
```

这是一个记录服务器信息的哈希表，哈希表的所有信息都已经可以看到。在创建哈希表的时候，键"Administrator"对应的值是一个数组的引用；类似的还有安装日期是命令Get-Date返回的对象。通过这个例子，我们可以了解到PowerShell中的哈希表可以很方便地存储各式各样的数据信息。但是我们应该如何去访问哈希表中的元素呢？在PowerShell中，有两种便捷的办法获取哈希表中存储的信息。第一种办法是类似访问对象属性的方法：

```
PS C:\> $hash.os
Windows 2008
PS C:\> $hash.Administrator
Ma Tao
Spider Man
PS C:\> $hash."Computer Name"
AD Server
```

这种方法很方便，只需要在点号后面输入键的名称即可。如果键值包含空白字符的话，可以通过使用引号来访问该键值对应的信息。如果希望能够同时索引多个键的值，这种类似访问属性的方法就无法完成了。不过PowerShell提供了另一种办法，它类似于数组访问的形式，使得我们可以同时返回哈希表中的几个元素。例如：

```
PS C:\> $hash["Disk Size", "Installed Date"]
5368709120000
2008年7月31日 23:09:57
```

shell, perl, python, ...

回复支持反对

举报

发 帖 ▾

返回列表

1

2

1 / 2 页

下一页

呃...找不到此网站。

我们无法连接至

高级模式

您需要登录后才可以回帖 [登录](#) | [注册](#)

发表回复

☐ 回帖后跳转到最后一页

[本版积分规则](#)