

这是作者的系列网络安全自学教程，主要是关于网安工具和实践操作的在线笔记，特分享出来与博友共勉，希望你们喜欢，一起进步。前文分享了Powershell基础入门知识，涉及基础概念、管道和重定向、执行外部命令、别名用法、变量定义等。这篇文章将从Powershell条件语句、循环语句、数组、函数、字符串操作、注册表访问等方面讲解。Powershell被广泛应用于安全领域，甚至成为每一位Web安全必须掌握的技术。

本文参考了Bilibili的Hack学习老师的课程，同时也结合了作者之前的编程经验进行讲解。作者作为网络安全的小白，分享一些自学基础教程给大家，希望你们喜欢。同时，更希望你能与我一起操作深入进步，后续也将深入学习网络安全和系统安全知识并分享相关实验。总之，希望该系列文章对博友有所帮助，写文不容易，大神请飘过，不喜勿喷，谢谢！

下载地址：<https://github.com/eastmountyxz/NetworkSecuritySelf-study>

百度网盘：https://pan.baidu.com/s/1dsunH8EmOB_tIHYYXGuOeA 提取码：izeb

文章目录

一.Powershell操作符

二.Powershell条件语句

1.if条件判断

2.switch语句

三.Powershell循环语句

1.foreach循环

2.while循环

3.break和continue关键词

4.for循环

5.switch循环

四.Powershell数组

1.数组定义

2.访问数组

五.Powershell函数

1.自定义函数及调用

2.函数返回值

六.Powershell字符串及交互

1.定义文本及转义字符

2.用户交互

3.格式化字符串

4.字符串操作

七.Powershell注册表操作

前文学习:

- [网络安全自学篇] 一.入门笔记之看雪Web安全学习及异或解密示例
- [网络安全自学篇] 二.Chrome浏览器保留密码功能渗透解析及登录加密入门笔记
- [网络安全自学篇] 三.Burp Suite工具安装配置、Proxy基础用法及暴库示例
- [网络安全自学篇] 四.实验吧CTF实战之WEB渗透和隐写术解密
- [网络安全自学篇] 五.IDA Pro反汇编工具初识及逆向工程解密实战
- [网络安全自学篇] 六.OllyDbg动态分析工具基础用法及Crakeme逆向破解
- [网络安全自学篇] 七.快手视频下载之Chrome浏览器Network分析及Python爬虫探讨
- [网络安全自学篇] 八.Web漏洞及端口扫描之Nmap、ThreatScan和DirBuster工具
- [网络安全自学篇] 九.社会工程学之基础概念、IP获取、IP物理定位、文件属性
- [网络安全自学篇] 十.论文之基于机器学习算法的主机恶意代码
- [网络安全自学篇] 十一.虚拟机VMware+Kali安装入门及Sqlmap基本用法
- [网络安全自学篇] 十二.Wireshark安装入门及抓取网站用户名密码（一）
- [网络安全自学篇] 十三.Wireshark抓包原理（ARP劫持、MAC泛洪）及数据流追踪和图像抓取（二）
- [网络安全自学篇] 十四.Python攻防之基础常识、正则表达式、Web编程和套接字通信（一）
- [网络安全自学篇] 十五.Python攻防之多线程、C段扫描和数据库编程（二）
- [网络安全自学篇] 十六.Python攻防之弱口令、自定义字典生成及网站暴库防护
- [网络安全自学篇] 十七.Python攻防之构建Web目录扫描器及ip代理池（四）
- [网络安全自学篇] 十八.XSS跨站脚本攻击原理及代码攻防演示（一）
- [网络安全自学篇] 十九.Powershell基础入门及常见用法（一）

前文欣赏:

- [渗透&攻防] 一.从数据库原理学习网络攻防及防止SQL注入
- [渗透&攻防] 二.SQL MAP工具从零解读数据库及基础用法
- [渗透&攻防] 三.数据库之差异备份及Caidao利器
- [渗透&攻防] 四.详解MySQL数据库攻防及Fiddler神器分析数据包

参考文献:

- <https://www.bilibili.com/video/av66327436> [推荐B站老师视频]
- 《安全之路Web渗透技术及实战案例解析》陈小兵老师
- [https://baike.baidu.com/item/Windows Power Shell/693789](https://baike.baidu.com/item/Windows%20Power%20Shell/693789)
- <https://www.pstips.net/powershell-piping-and-routing.html>
- <https://www.pstips.net/using-the-powershell-pipeline.html>
- <https://baike.baidu.com/item/注册表/101856>
- C# 系统应用之注册表使用详解 - Eastmount

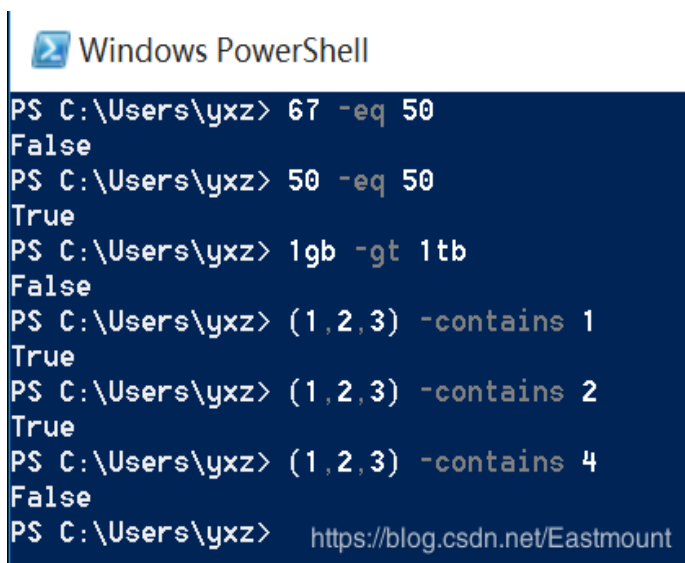
声明：本人坚决反对利用教学方法进行犯罪的行为，一切犯罪行为必将受到严惩，绿色网络需要我们共同维护，更推荐大家了解它们背后的原理，更好地进行防护。

一.Powershell操作符

常见的比较运算符包括：

- -eq 等于
- -ne 不等于
- -gt 大于
- -lt 小于
- -le 小于等于
- -contains 包含
- -notcontains 不包含

```
67 -eq 50
50 -eq 50
1gb -gt 1tb
(1,2,3) -contains 1
(1,2,3) -contains 2
(1,2,3) -contains 4
```



```
Windows PowerShell
PS C:\Users\yxz> 67 -eq 50
False
PS C:\Users\yxz> 50 -eq 50
True
PS C:\Users\yxz> 1gb -gt 1tb
False
PS C:\Users\yxz> (1,2,3) -contains 1
True
PS C:\Users\yxz> (1,2,3) -contains 2
True
PS C:\Users\yxz> (1,2,3) -contains 4
False
PS C:\Users\yxz> https://blog.csdn.net/Eastmount
```

求反运算符：

- -not

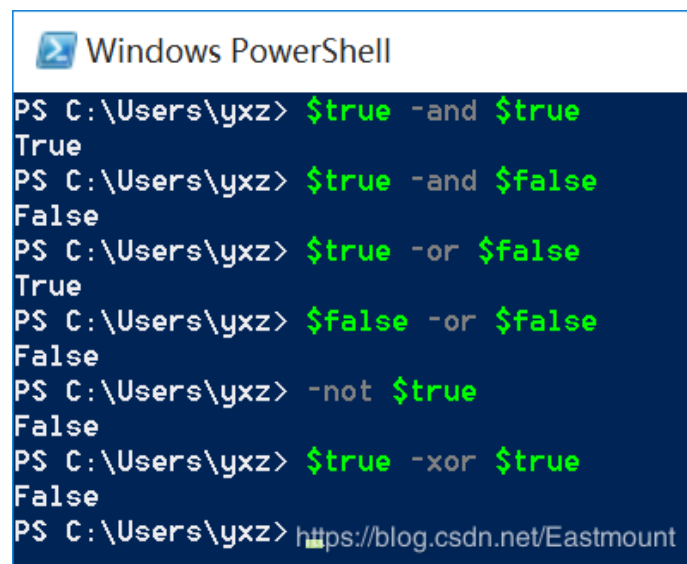
```
$a=89 -gt 50  
$a  
-not $a
```

```
PS C:\Users\yxz> $a=89 -gt 50  
PS C:\Users\yxz> $a  
True  
PS C:\Users\yxz> -not $a  
False  
PS C:\Users\yxz>
```

逻辑运算：

- -and 与运算
- -or 或运算
- -not 非运算
- -xor 异或运算

```
$true -and $true  
$true -and $false  
$true -or $false  
$false -or $false  
-not $true  
$true -xor $true
```



A screenshot of a Windows PowerShell terminal window. The title bar reads 'Windows PowerShell'. The terminal shows a series of commands and their outputs: '\$true -and \$true' returns 'True'; '\$true -and \$false' returns 'False'; '\$true -or \$false' returns 'True'; '\$false -or \$false' returns 'False'; '-not \$true' returns 'False'; '\$true -xor \$true' returns 'False'. The final prompt shows a URL: 'https://blog.csdn.net/Eastmount'.

```
PS C:\Users\yxz> $true -and $true  
True  
PS C:\Users\yxz> $true -and $false  
False  
PS C:\Users\yxz> $true -or $false  
True  
PS C:\Users\yxz> $false -or $false  
False  
PS C:\Users\yxz> -not $true  
False  
PS C:\Users\yxz> $true -xor $true  
False  
PS C:\Users\yxz> https://blog.csdn.net/Eastmount
```

比较数组和集合，从中筛选出不等于0的数字。

```
1,5,8,0,9 -ne 0
```

```
PS C:\Users\yxz> 1,5,8,0,9 -ne 0
1
5
8
9
PS C:\Users\yxz> _
```

二.Powershell条件语句

1.if条件判断

if-elseif-else条件判断，执行操作用大括号表示。

```
$num=100
if($num -gt 90) {"大于90"} else {"小于等于90"}
if($num -gt 100) {"大于100"} else {"小于等于100"}
```

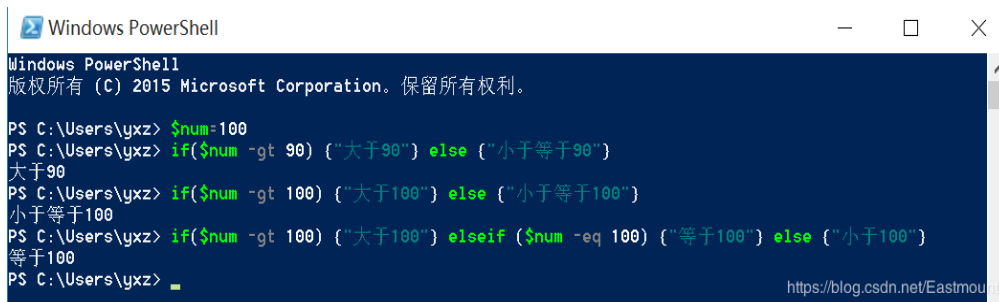


```
Windows PowerShell
版权所有 (C) 2015 Microsoft Corporation。保留所有权利。

PS C:\Users\yxz> $num=100
PS C:\Users\yxz> if($num -gt 90) {"大于90"} else {"小于等于90"}
大于90
PS C:\Users\yxz> if($num -gt 100) {"大于100"} else {"小于等于100"}
小于等于100
PS C:\Users\yxz>
```

注意，if-else中间可以增加新的判断elseif，如下所示：

```
if($num -gt 100) {"大于100"} elseif ($num -eq 100) {"等于100"} else {"小于100"}
```

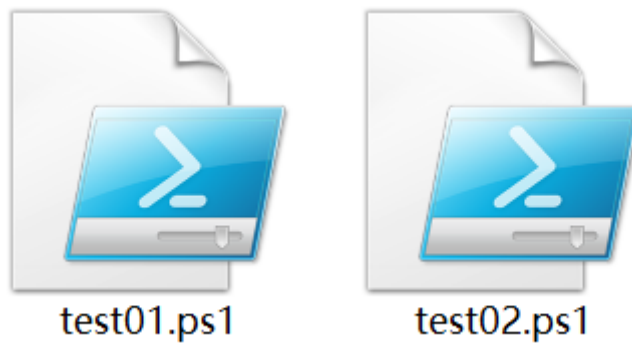


```
Windows PowerShell
版权所有 (C) 2015 Microsoft Corporation。保留所有权利。

PS C:\Users\yxz> $num=100
PS C:\Users\yxz> if($num -gt 90) {"大于90"} else {"小于等于90"}
大于90
PS C:\Users\yxz> if($num -gt 100) {"大于100"} else {"小于等于100"}
小于等于100
PS C:\Users\yxz> if($num -gt 100) {"大于100"} elseif ($num -eq 100) {"等于100"} else {"小于100"}
等于100
PS C:\Users\yxz> _
```

2.switch语句

Switch语句主要用于多种情况的判断，这里在本地创建一个test01.ps1文件，并执行该代码。



传统的if判断如下:

```
$num=56
if($num -gt 50 -and $num -lt 60)
{
    "大于50并且小于60"
}
elseif ($num -eq 50)
{
    "等于50"
}
else
{
    "小于50"
}
```


去到桌面1019文件夹, 输入“.test01.ps1”执行代码, 再打印该文件的源代码。

```
PS C:\Users\yxz> cd desktop
PS C:\Users\yxz\desktop> cd 1019
PS C:\Users\yxz\desktop\1019> .test01.ps1
大于50并且小于60
PS C:\Users\yxz\desktop\1019> type test01.ps1
$num=56
if($num -gt 50 -and $num -lt 60)
{
    "大于50并且小于60"
}
elseif ($num -eq 50)
{
    "等于50"
}
else
{
    "小于50"
}
PS C:\Users\yxz\desktop\1019>
```

switch语句如下: \$_表示对变量取值。

```
$num=56
switch($num)
{
    {$_ -lt 50} {"此数值小于50"}
    {$_ -eq 50} {"此数值等于50"}
}
```

```
{$_ -gt 50} {"此数值大于50"}  
}
```



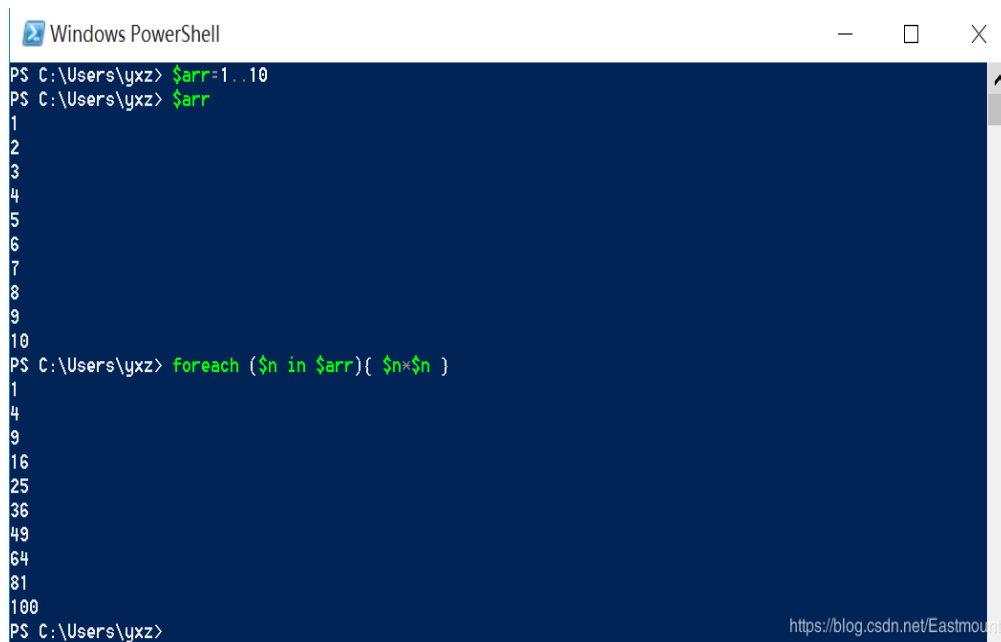
```
Windows PowerShell  
PS C:\Users\yxz\desktop\1019> .\test02.ps1  
此数值大于50  
PS C:\Users\yxz\desktop\1019> type test02.ps1  
$num=56  
switch($num)  
{  
    {$_ -lt 50} {"此数值小于50"}  
    {$_ -eq 50} {"此数值等于50"}  
    {$_ -gt 50} {"此数值大于50"}  
}  
PS C:\Users\yxz\desktop\1019>
```

三.Powershell循环语句

1.foreach循环

这里定义数组采用“\$arr=1..10”实现，表示1到10的数字，在调用foreach循环输出。

```
$arr=1..10  
foreach ($n in $arr){ $n*$n }
```



```
Windows PowerShell  
PS C:\Users\yxz> $arr=1..10  
PS C:\Users\yxz> $arr  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
PS C:\Users\yxz> foreach ($n in $arr){ $n*$n }  
1  
4  
9  
16  
25  
36  
49  
64  
81  
100  
PS C:\Users\yxz>
```

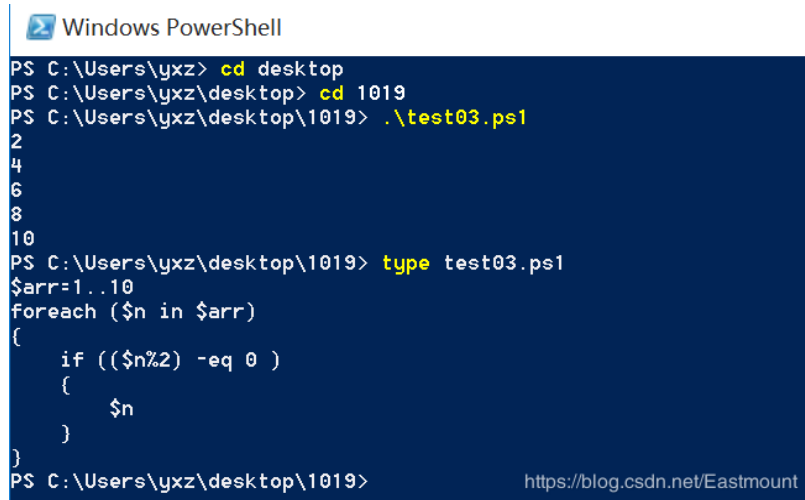
定义文件“test03.ps1”，只输出偶数内容。

```
$arr=1..10  
foreach ($n in $arr)  
{  
    if (($n%2) -eq 0 )
```

```

{
    $n
}
}

```



```

Windows PowerShell
PS C:\Users\yxz> cd desktop
PS C:\Users\yxz\desktop> cd 1019
PS C:\Users\yxz\desktop\1019> .\test03.ps1
2
4
6
8
10
PS C:\Users\yxz\desktop\1019> type test03.ps1
$arr=1..10
foreach ($n in $arr)
{
    if (($n%2) -eq 0 )
    {
        $n
    }
}
PS C:\Users\yxz\desktop\1019>

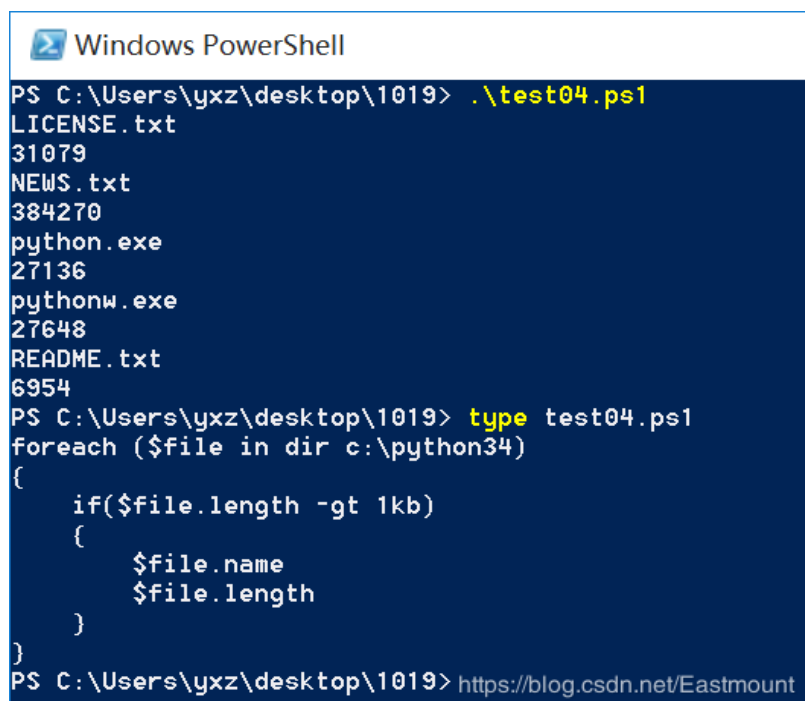
```

接着利用foreach操作文件目录，将C盘python34文件夹下的路径全部提取出来，赋值到file中输出。

```

foreach ($file in dir c:\python34)
{
    if($file.length -gt 1kb)
    {
        $file.name
        $file.length
    }
}

```

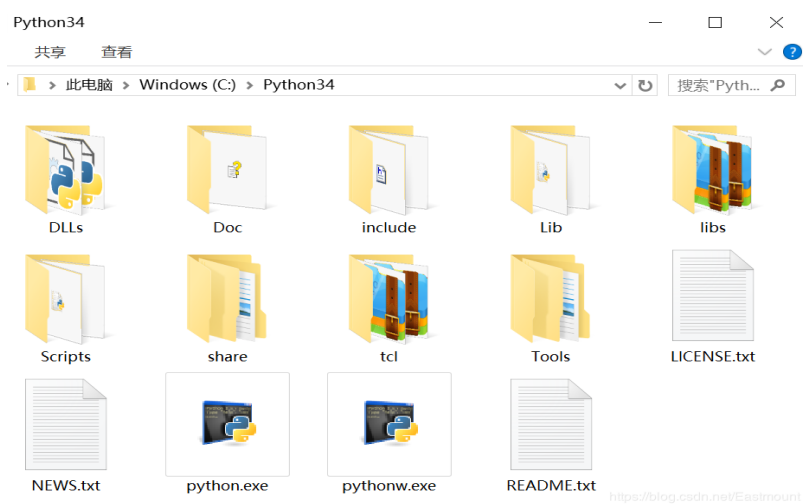


```

Windows PowerShell
PS C:\Users\yxz\desktop\1019> .\test04.ps1
LICENSE.txt
31079
NEWS.txt
384270
python.exe
27136
pythonw.exe
27648
README.txt
6954
PS C:\Users\yxz\desktop\1019> type test04.ps1
foreach ($file in dir c:\python34)
{
    if($file.length -gt 1kb)
    {
        $file.name
        $file.length
    }
}
PS C:\Users\yxz\desktop\1019>

```


原始文件内容如下所示：



也可以定义变量来指定路径

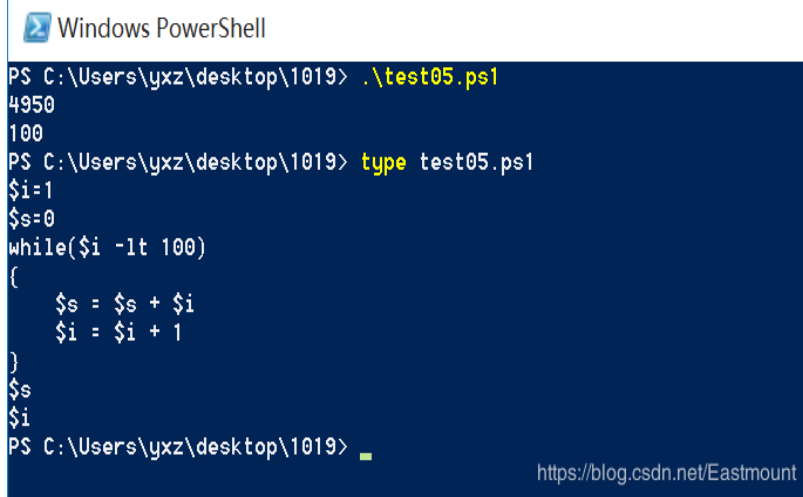
```
$path_value = dir c:\python34
foreach ($file in $path_value)
{
    if($file.length -gt 1kb)
    {
        $file.name
        $file.length
    }
}
```

2.while循环

while循环需要注意循环的终止条件，防止出现死循环，而do_while循环是先执行一次循环体，再进行判断。

下面这段代码是经典运算：1+2+3+...+99，文件名为“test05.ps1”。

```
$i=1
$s=0
while($i -lt 100)
{
    $s = $s + $i
    $i = $i + 1
}
$s
$i
```



```

Windows PowerShell
PS C:\Users\yxz\desktop\1019> .\test05.ps1
4950
100
PS C:\Users\yxz\desktop\1019> type test05.ps1
$i=1
$s=0
while($i -lt 100)
{
    $s = $s + $i
    $i = $i + 1
}
$s
$i
PS C:\Users\yxz\desktop\1019>

```

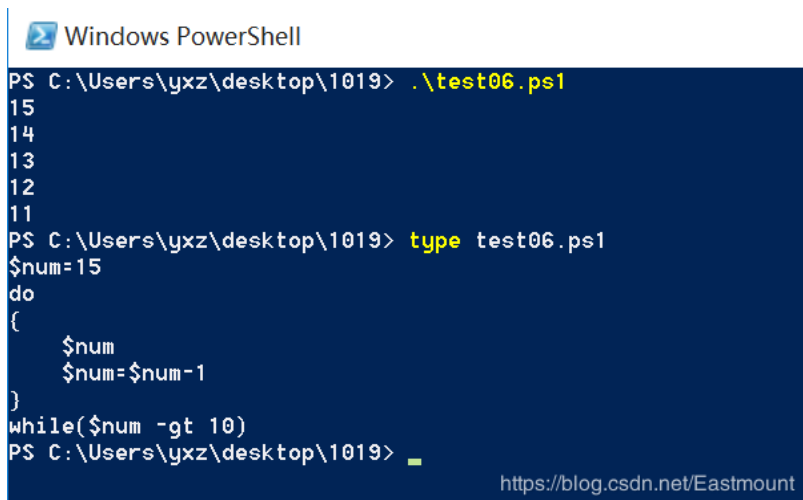
<https://blog.csdn.net/Eastmount>

do_while先执行循环体，再进行条件判断，如下所示：

```

$num=15
do
{
    $num
    $num=$num-1
}
while($num -gt 10)

```



```

Windows PowerShell
PS C:\Users\yxz\desktop\1019> .\test06.ps1
15
14
13
12
11
PS C:\Users\yxz\desktop\1019> type test06.ps1
$num=15
do
{
    $num
    $num=$num-1
}
while($num -gt 10)
PS C:\Users\yxz\desktop\1019>

```

<https://blog.csdn.net/Eastmount>

3.break和continue关键词

break跳出整个循环，停止执行；continue跳出当前循环一次，继续执行下一个判断。

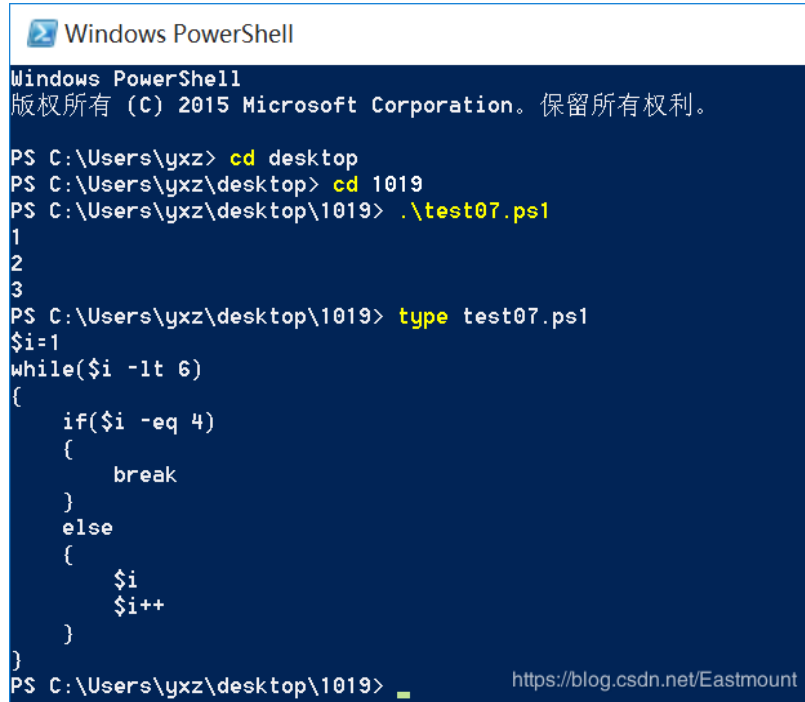
break： 下面这个代码当数值小于6继续执行，当其等于4停止循环。

```

$i=1
while($i -lt 6)
{
    if($i -eq 4)
    {
        break
    }
}

```

```
}  
else  
{  
    $i  
    $i++  
}  
}
```



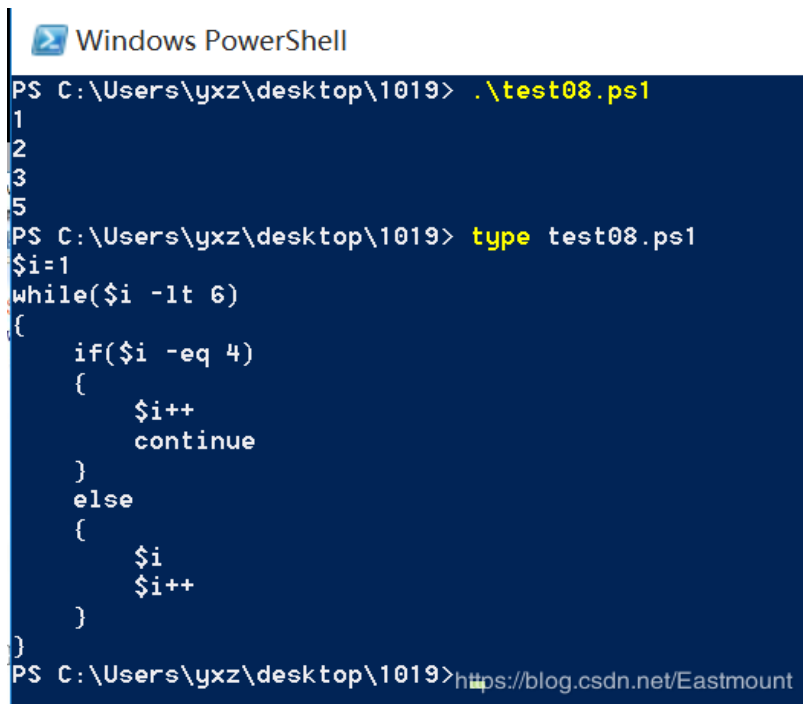
The screenshot shows a Windows PowerShell window with a dark blue background. The title bar reads 'Windows PowerShell'. The window content shows the following commands and output:

```
Windows PowerShell  
版权所有 (C) 2015 Microsoft Corporation。保留所有权利。  
  
PS C:\Users\yxz> cd desktop  
PS C:\Users\yxz\desktop> cd 1019  
PS C:\Users\yxz\desktop\1019> .\test07.ps1  
1  
2  
3  
PS C:\Users\yxz\desktop\1019> type test07.ps1  
$i=1  
while($i -lt 6)  
{  
    if($i -eq 4)  
    {  
        break  
    }  
    else  
    {  
        $i  
        $i++  
    }  
}  
PS C:\Users\yxz\desktop\1019> _
```

A URL is visible in the bottom right corner: <https://blog.csdn.net/Eastmount>

continue: 跳过了中间等于4的内容。

```
$i=1  
while($i -lt 6)  
{  
    if($i -eq 4)  
    {  
        $i++  
        continue  
    }  
    else  
    {  
        $i  
        $i++  
    }  
}
```



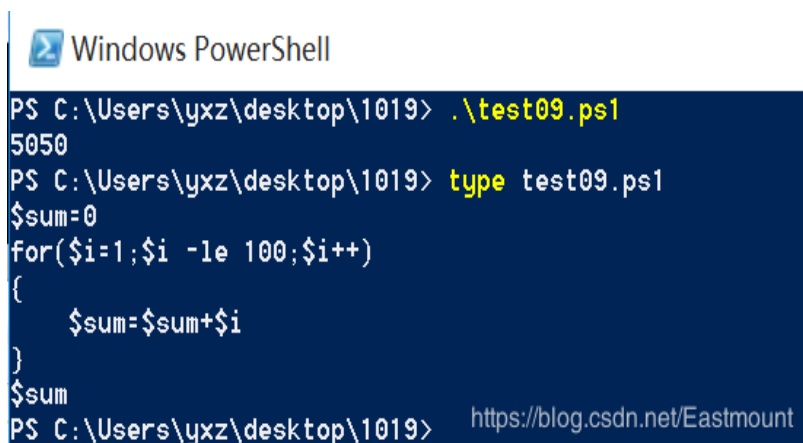
```
Windows PowerShell
PS C:\Users\yxz\desktop\1019> .\test08.ps1
1
2
3
5
PS C:\Users\yxz\desktop\1019> type test08.ps1
$i=1
while($i -lt 6)
{
    if($i -eq 4)
    {
        $i++
        continue
    }
    else
    {
        $i
        $i++
    }
}
PS C:\Users\yxz\desktop\1019>
```

4.for循环

利用for循环实现1+2+...+100的代码如下 (test09.ps1) 。

```
$sum=0
for($i=1;$i -le 100;$i++)
{
    $sum=$sum+$i
}
$sum
```

学习Powershell基础语法之后，更重要的是解决实际问题，后续作者将继续深入学习。



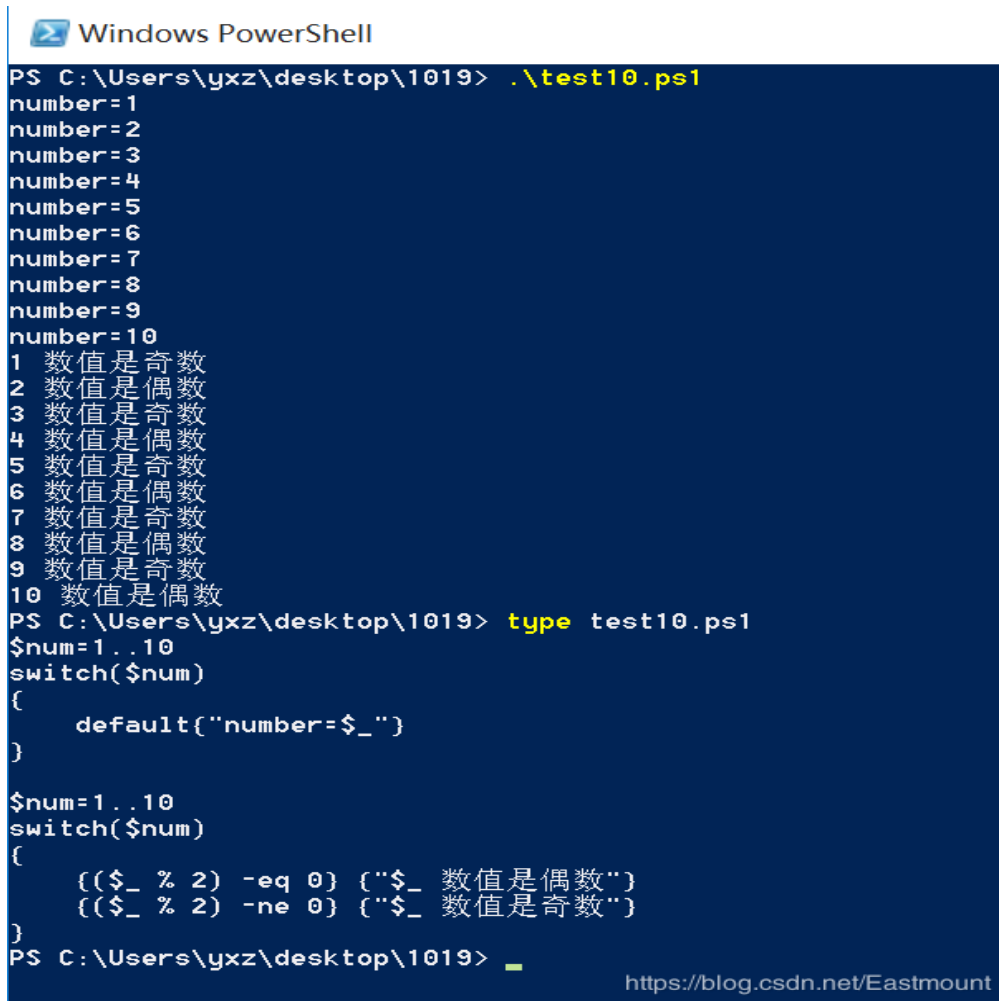
```
Windows PowerShell
PS C:\Users\yxz\desktop\1019> .\test09.ps1
5050
PS C:\Users\yxz\desktop\1019> type test09.ps1
$sum=0
for($i=1;$i -le 100;$i++)
{
    $sum=$sum+$i
}
$sum
PS C:\Users\yxz\desktop\1019>
```

5.switch循环

使用switch循环实现输出数组1到10，并进行奇数和偶数判断。

```
$num=1..10
switch($num)
{
    default{"number=$_"}
}

$num=1..10
switch($num)
{
    {($_ % 2) -eq 0} {"$_ 数值是偶数"}
    {($_ % 2) -ne 0} {"$_ 数值是奇数"}
}
```



```
Windows PowerShell
PS C:\Users\yxz\desktop\1019> .\test10.ps1
number=1
number=2
number=3
number=4
number=5
number=6
number=7
number=8
number=9
number=10
1 数值是奇数
2 数值是偶数
3 数值是奇数
4 数值是偶数
5 数值是奇数
6 数值是偶数
7 数值是奇数
8 数值是偶数
9 数值是奇数
10 数值是偶数
PS C:\Users\yxz\desktop\1019> type test10.ps1
$num=1..10
switch($num)
{
    default{"number=$_"}
}

$num=1..10
switch($num)
{
    {($_ % 2) -eq 0} {"$_ 数值是偶数"}
    {($_ % 2) -ne 0} {"$_ 数值是奇数"}
}
PS C:\Users\yxz\desktop\1019> _
```

<https://blog.csdn.net/Eastmount>

四.Powershell数组

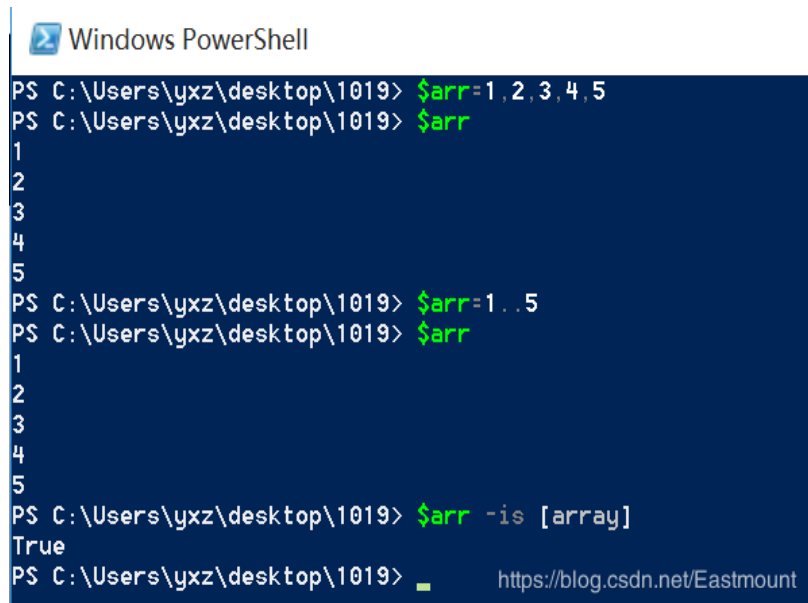
1.数组定义

数组定义一种方法是逗号隔开不同的元素，另一种是通过两个点来定义数组。

```
$arr=1,2,3,4,5
$arr=1..5
```

判断是否是一个数组，使用如下语句。

```
$arr -is [array]
```



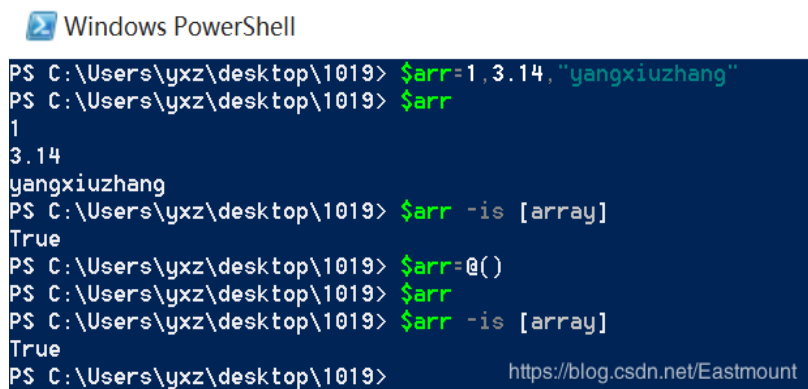
```
Windows PowerShell
PS C:\Users\yxz\desktop\1019> $arr=1,2,3,4,5
PS C:\Users\yxz\desktop\1019> $arr
1
2
3
4
5
PS C:\Users\yxz\desktop\1019> $arr=1..5
PS C:\Users\yxz\desktop\1019> $arr
1
2
3
4
5
PS C:\Users\yxz\desktop\1019> $arr -is [array]
True
PS C:\Users\yxz\desktop\1019> https://blog.csdn.net/Eastmount
```

数组可以接受不同的数值。

```
$arr=1,3.14,"yangxiuzhang"
$arr
$arr -is [array]
```

空数组定义如下：

```
$arr=@()
$arr
$arr -is [array]
```

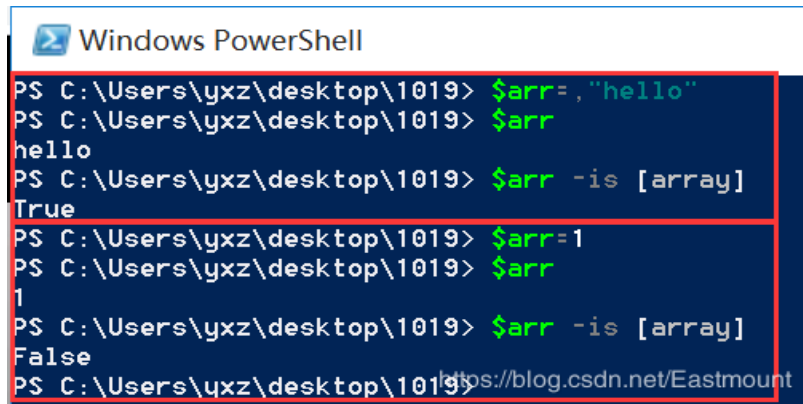


```
Windows PowerShell
PS C:\Users\yxz\desktop\1019> $arr=1,3.14,"yangxiuzhang"
PS C:\Users\yxz\desktop\1019> $arr
1
3.14
yangxiuzhang
PS C:\Users\yxz\desktop\1019> $arr -is [array]
True
PS C:\Users\yxz\desktop\1019> $arr=@()
PS C:\Users\yxz\desktop\1019> $arr
PS C:\Users\yxz\desktop\1019> $arr -is [array]
True
PS C:\Users\yxz\desktop\1019> https://blog.csdn.net/Eastmount
```

下面简单比较只有一个元素数组和变量的对比。

```
$arr="hello"
$arr
$arr -is [array]

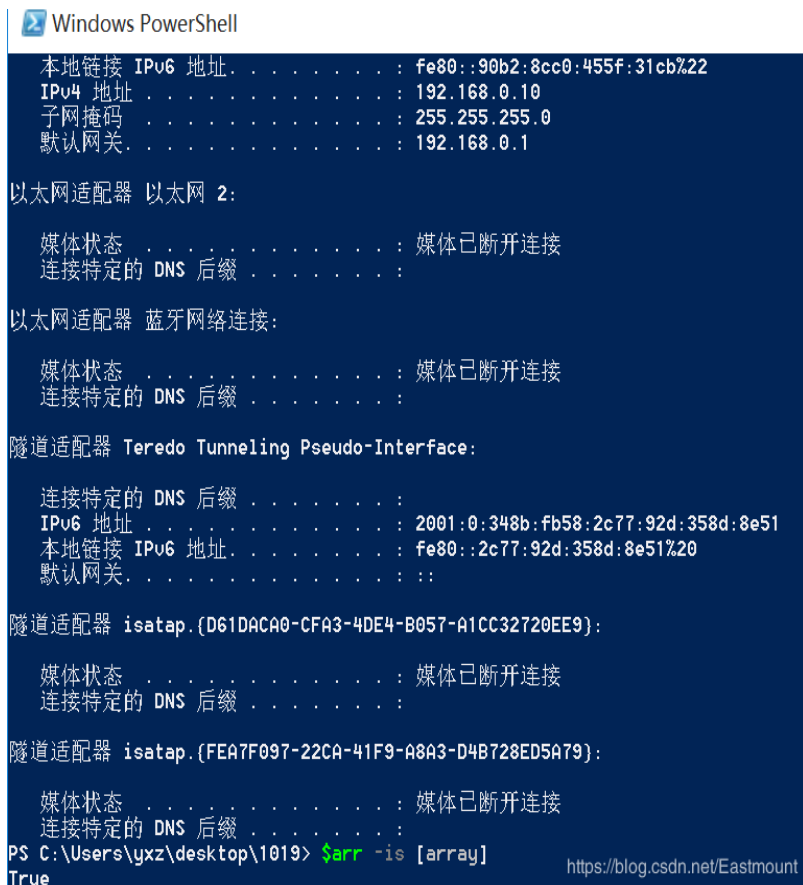
$arr=1
$arr
$arr -is [array]
```



```
Windows PowerShell
PS C:\Users\yxz\Desktop\1019> $arr="hello"
PS C:\Users\yxz\Desktop\1019> $arr
hello
PS C:\Users\yxz\Desktop\1019> $arr -is [array]
True
PS C:\Users\yxz\Desktop\1019> $arr=1
PS C:\Users\yxz\Desktop\1019> $arr
1
PS C:\Users\yxz\Desktop\1019> $arr -is [array]
False
PS C:\Users\yxz\Desktop\1019>
```

数组也可以是一个变量或命令，此时它仍然是一个数组。

```
$arr=ipconfig
$arr
$arr -is [array]
```



```
Windows PowerShell
本地链接 IPv6 地址. . . . . : fe80::90b2:8cc0:455f:31cb%22
IPv4 地址 . . . . . : 192.168.0.10
子网掩码 . . . . . : 255.255.255.0
默认网关. . . . . : 192.168.0.1

以太网适配器 以太网 2:

    媒体状态 . . . . . : 媒体已断开连接
    连接特定的 DNS 后缀 . . . . . :

以太网适配器 蓝牙网络连接:

    媒体状态 . . . . . : 媒体已断开连接
    连接特定的 DNS 后缀 . . . . . :

隧道适配器 Teredo Tunneling Pseudo-Interface:

    连接特定的 DNS 后缀 . . . . . :
    IPv6 地址 . . . . . : 2001:0:348b:fb58:2c77:92d:358d:8e51
    本地链接 IPv6 地址. . . . . : fe80::2c77:92d:358d:8e51%20
    默认网关. . . . . : ::

隧道适配器 isatap.{D61DACA0-CFA3-4DE4-B057-A1CC32720EE9}:

    媒体状态 . . . . . : 媒体已断开连接
    连接特定的 DNS 后缀 . . . . . :

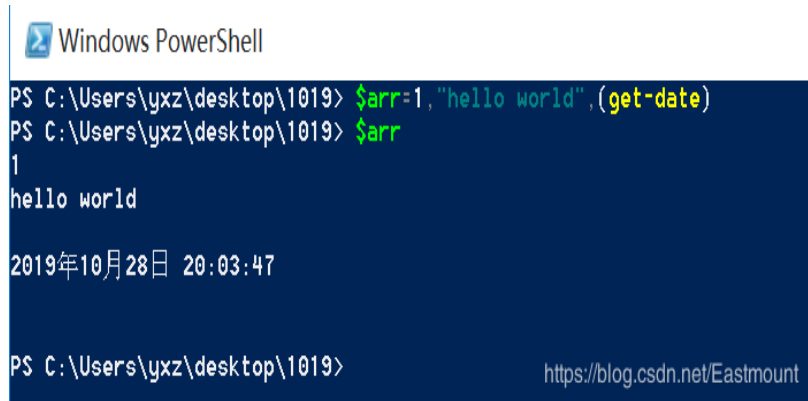
隧道适配器 isatap.{FEA7F097-22CA-41F9-A8A3-D4B728ED5A79}:

    媒体状态 . . . . . : 媒体已断开连接
    连接特定的 DNS 后缀 . . . . . :
PS C:\Users\yxz\Desktop\1019> $arr -is [array]
True
```

2.访问数组

首先定义一个多钟类型的数组。

```
$arr=1,"hello world",(get-date)
$arr
```



```
Windows PowerShell
PS C:\Users\yxz\desktop\1019> $arr=1,"hello world",(get-date)
PS C:\Users\yxz\desktop\1019> $arr
1
hello world

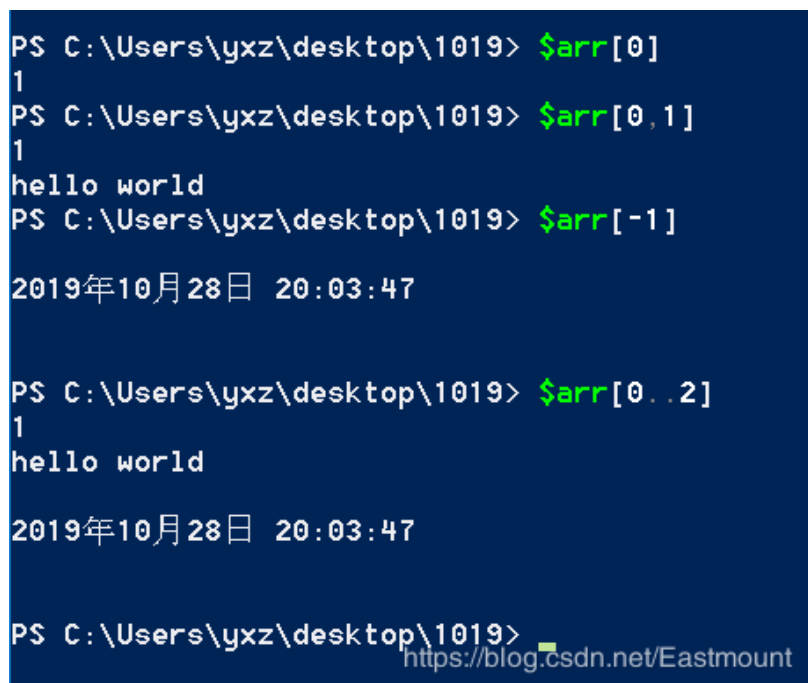
2019年10月28日 20:03:47

PS C:\Users\yxz\desktop\1019> https://blog.csdn.net/Eastmount
```

访问数组特定元素，第一个元素，获取两个元素，获取最后一个元素。

```
$arr[0]
$arr[0,1]
$arr[-1]
```

```
// 提取部分元素
$arr[0..2]
```



```
PS C:\Users\yxz\desktop\1019> $arr[0]
1
PS C:\Users\yxz\desktop\1019> $arr[0,1]
1
hello world
PS C:\Users\yxz\desktop\1019> $arr[-1]

2019年10月28日 20:03:47

PS C:\Users\yxz\desktop\1019> $arr[0..2]
1
hello world

2019年10月28日 20:03:47

PS C:\Users\yxz\desktop\1019> https://blog.csdn.net/Eastmount
```

获取数组元素大小调用count实现。

```
$num = $arr[0..2]
$num.count
```


如何将数组倒序输出呢？如下所示。

```
$arr[($arr.count)..0]
```

```
PS C:\Users\yxz\desktop\1019> $num = $arr[0..2]
PS C:\Users\yxz\desktop\1019> $num.count
3
PS C:\Users\yxz\desktop\1019> $arr[($arr.count)..0]

2019年10月28日 20:03:47
hello world
1

PS C:\Users\yxz\desktop\1019> https://blog.csdn.net/Eastmount
```

数组添加一个元素代码如下：

```
$arr=1,"hello world",(get-date)
$arr+="csdn"
$arr
$arr.count
```

```
PS C:\Users\yxz\desktop\1019> $arr=1,"hello world",(get-date)
PS C:\Users\yxz\desktop\1019> $arr+="csdn"
PS C:\Users\yxz\desktop\1019> $arr
1
hello world

2019年10月28日 20:04:54
csdn

PS C:\Users\yxz\desktop\1019> $arr.count
4
PS C:\Users\yxz\desktop\1019> https://blog.csdn.net/Eastmount
```

更多数组操作，推荐读者结合实际应用进行学习。

五.Powershell函数

1.自定义函数及调用

函数通常包括函数名、参数、函数体，下面是定义及调用一个myping函数的代码（test11.ps1）。

```
function myping()
{
    ping www.baidu.com
}
```

myping



```

Windows PowerShell
PS C:\Users\yxz\desktop\1019> .\test11.ps1

正在 Ping www.a.shifen.com [182.61.200.7] 具有 32 字节的数据:
来自 182.61.200.7 的回复: 字节=32 时间=42ms TTL=47
来自 182.61.200.7 的回复: 字节=32 时间=38ms TTL=47
来自 182.61.200.7 的回复: 字节=32 时间=39ms TTL=47
来自 182.61.200.7 的回复: 字节=32 时间=38ms TTL=47

182.61.200.7 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
    往返行程的估计时间(以毫秒为单位):
        最短 = 38ms, 最长 = 42ms, 平均 = 39ms
PS C:\Users\yxz\desktop\1019> type test11.ps1
function myping()
{
    ping www.baidu.com
}

myping
PS C:\Users\yxz\desktop\1019>
By: Eastmount CSDN
https://blog.csdn.net/Eastmount

```

同样，上面的代码可以修改为指定参数。

```

function myping($site)
{
    ping $site
}
myping www.baidu.com

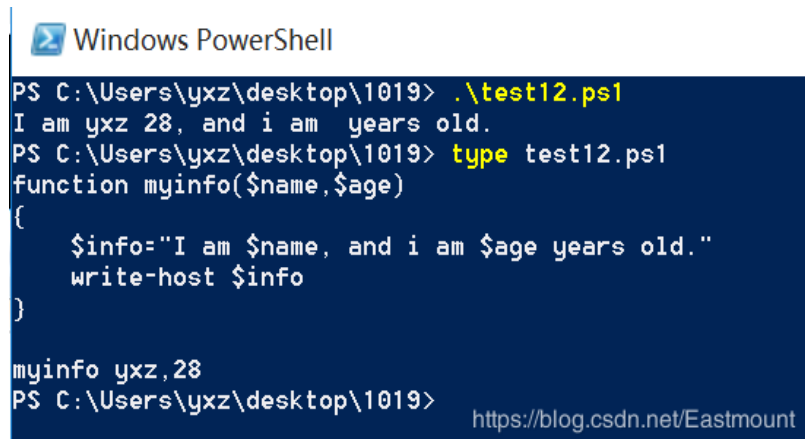
```

下面这个代码是接收两个参数并显示的功能。

```

function myinfo($name,$age)
{
    $info="I am $name, and i am $age years old."
    write-host $info
}
myinfo yxz,28

```



```

Windows PowerShell
PS C:\Users\yxz\desktop\1019> .\test12.ps1
I am yxz 28, and i am years old.
PS C:\Users\yxz\desktop\1019> type test12.ps1
function myinfo($name,$age)
{
    $info="I am $name, and i am $age years old."
    write-host $info
}

myinfo yxz,28
PS C:\Users\yxz\desktop\1019>
https://blog.csdn.net/Eastmount

```

2.函数返回值

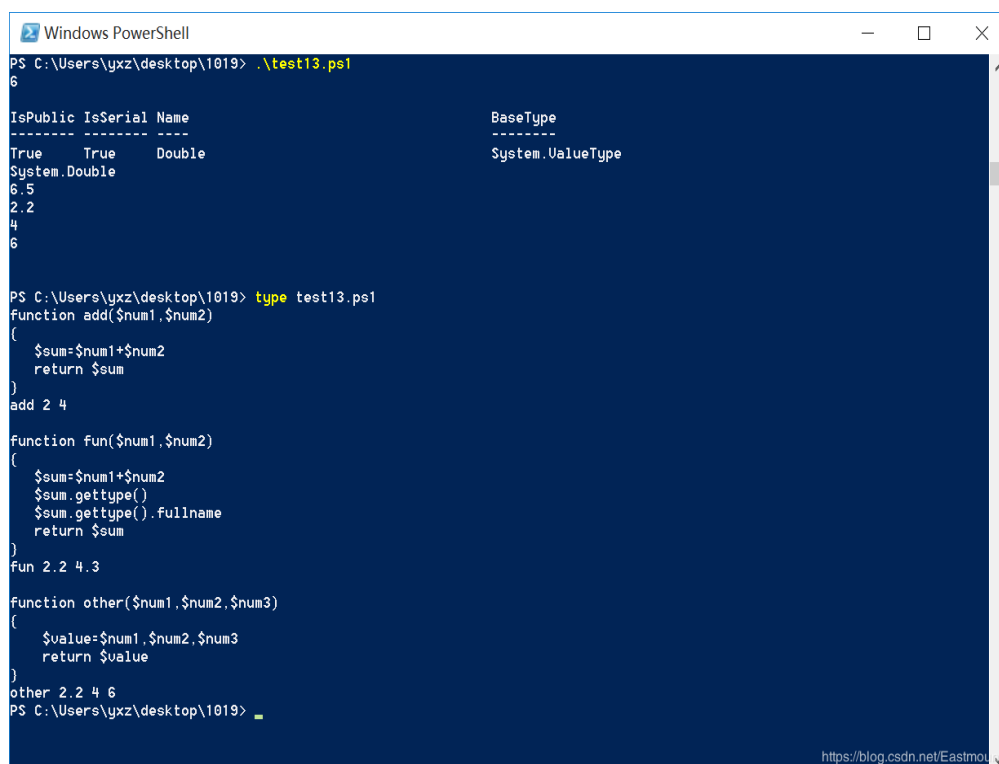
函数返回值通过return实现，可以返回多个值。下面是test13.ps1例子。

```
function add($num1,$num2)
{
    $sum=$num1+$num2
    return $sum
}
add 2 4
```

```
function fun($num1,$num2)
{
    $sum=$num1+$num2
    $sum.gettype()
    $sum.gettype().fullname
    return $sum
}
fun 2.2 4.3
```

// 多个返回值

```
function other($num1,$num2,$num3)
{
    $value=$num1,$num2,$num3
    return $value
}
other 2.2 4 6
```



```
Windows PowerShell
PS C:\Users\yxz\desktop\1019> .\test13.ps1
6
IsPublic IsSerial Name                                     BaseType
-----
True     True     Double                                     System.Double
6.5
2.2
4
6

PS C:\Users\yxz\desktop\1019> type test13.ps1
function add($num1,$num2)
{
    $sum=$num1+$num2
    return $sum
}
add 2 4

function fun($num1,$num2)
{
    $sum=$num1+$num2
    $sum.gettype()
    $sum.gettype().fullname
    return $sum
}
fun 2.2 4.3

function other($num1,$num2,$num3)
{
    $value=$num1,$num2,$num3
    return $value
}
other 2.2 4 6
PS C:\Users\yxz\desktop\1019>
```

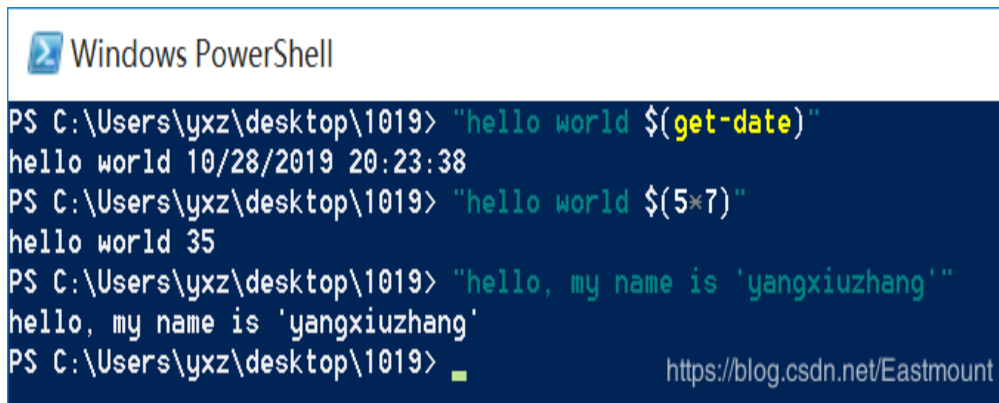
六.Powershell字符串及交互

1.定义文本及转义字符

表达式中可以定义只，如下所示。同时，单引号和双引号可以相互嵌套，这和JAVA、PHP、Python中的变量套接类似。

```
"hello world $(get-date)"
"hello world $(5*7)"
"hello, my name is 'yangxiuzhang'"
```

输出结果如下图所示：

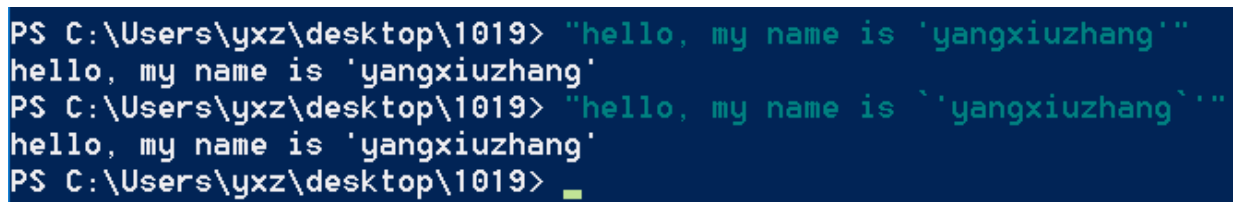


```
Windows PowerShell
PS C:\Users\yxz\desktop\1019> "hello world $(get-date)"
hello world 10/28/2019 20:23:38
PS C:\Users\yxz\desktop\1019> "hello world $(5*7)"
hello world 35
PS C:\Users\yxz\desktop\1019> "hello, my name is 'yangxiuzhang'"
hello, my name is 'yangxiuzhang'
PS C:\Users\yxz\desktop\1019> ■ https://blog.csdn.net/Eastmount
```

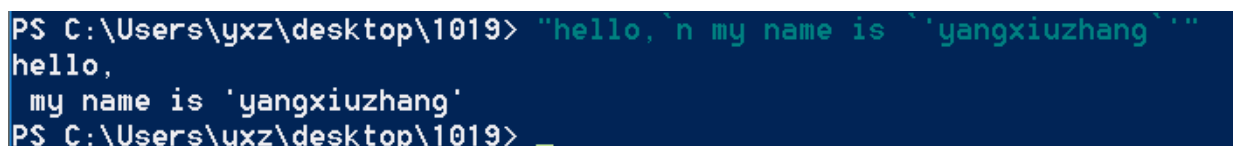
在Powershell中，转义字符不再是斜杠（\）而是（`），如下所示。

- `n 换行
- `r 回车符
- `t tab键
- `b 退格符
- `` 单引号

```
"hello,`n my name is ``'yangxiuzhang`'"
```



```
PS C:\Users\yxz\desktop\1019> "hello, my name is 'yangxiuzhang'"
hello, my name is 'yangxiuzhang'
PS C:\Users\yxz\desktop\1019> "hello, my name is ``'yangxiuzhang`'"
hello, my name is 'yangxiuzhang'
PS C:\Users\yxz\desktop\1019> ■
```



```
PS C:\Users\yxz\desktop\1019> "hello,`n my name is ``'yangxiuzhang`'"
hello,
  my name is 'yangxiuzhang'
PS C:\Users\yxz\desktop\1019> ■
```

2.用户交互

read-host 读取用户的输入。

```
$input = read-host "请输入您的姓名"  
"您好！您输入的姓名是：$input"
```

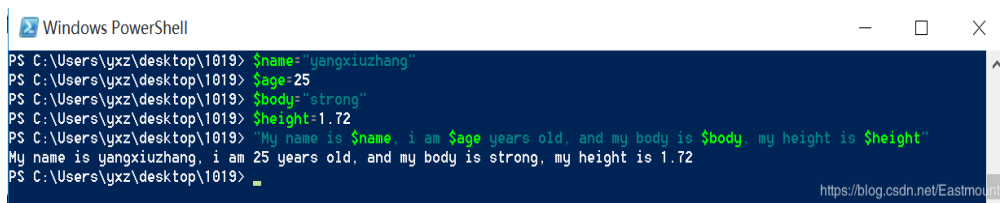


```
PS C:\Users\yxz\desktop\1019> .\test14.ps1  
请输入您的姓名：yangxiuzhang  
您好！您输入的姓名是：yangxiuzhang  
PS C:\Users\yxz\desktop\1019> type test14.ps1  
$input = read-host "请输入您的姓名"  
"您好！您输入的姓名是：$input"  
PS C:\Users\yxz\desktop\1019> █
```

3.格式化字符串

传统的多个变量输出方法：

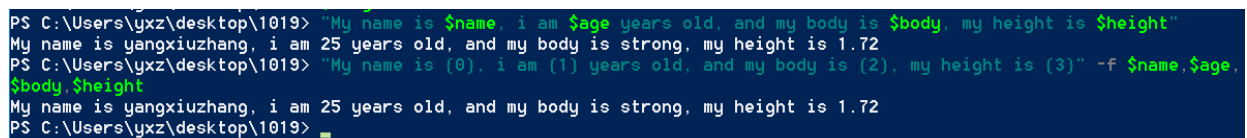
```
$name="yangxiuzhang"  
$age=25  
$body="strong"  
$height=1.72  
"My name is $name, i am $age years old, and my body is $body, my height is $height"
```



```
Windows PowerShell  
PS C:\Users\yxz\desktop\1019> $name="yangxiuzhang"  
PS C:\Users\yxz\desktop\1019> $age=25  
PS C:\Users\yxz\desktop\1019> $body="strong"  
PS C:\Users\yxz\desktop\1019> $height=1.72  
PS C:\Users\yxz\desktop\1019> "My name is $name, i am $age years old, and my body is $body, my height is $height"  
My name is yangxiuzhang, i am 25 years old, and my body is strong, my height is 1.72  
PS C:\Users\yxz\desktop\1019> █
```

格式化字符串输出方法：

```
"My name is {0}, i am {1} years old, and my body is {2}, my height is {3}"
```



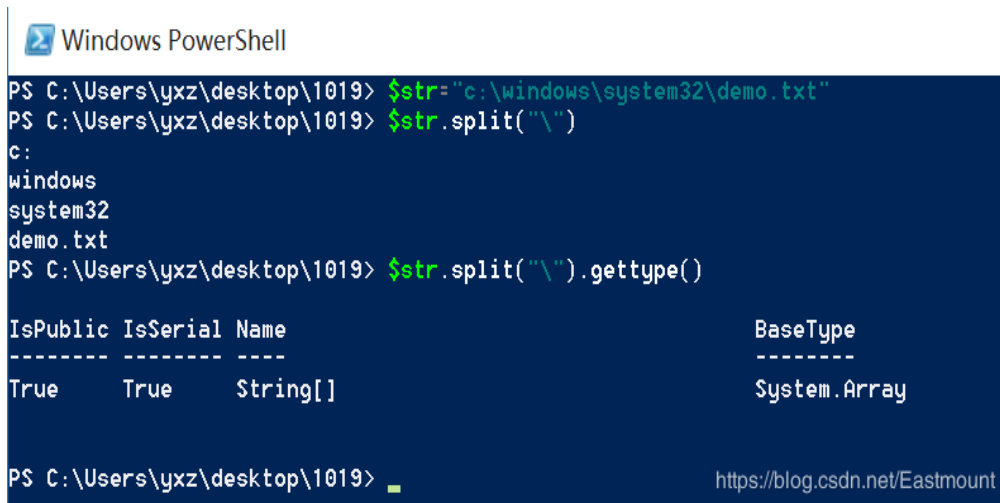
```
PS C:\Users\yxz\desktop\1019> "My name is $name, i am $age years old, and my body is $body, my height is $height"  
My name is yangxiuzhang, i am 25 years old, and my body is strong, my height is 1.72  
PS C:\Users\yxz\desktop\1019> "My name is {0}, i am {1} years old, and my body is {2}, my height is {3}" -f $name,$age,$body,$height  
My name is yangxiuzhang, i am 25 years old, and my body is strong, my height is 1.72  
PS C:\Users\yxz\desktop\1019> █
```

4.字符串操作

任何编程语言，都绕不过字符串操作，在网络安全领域，获取ip地址、URL拼接、图片或脚本文件获取等都涉及字符串操作，下面进行简单分享。

字符串分割

```
$str="c:\windows\system32\demo.txt"
$str.split("\")
//数组类型，可以通过数组下标访问
$str.split("\").gettype()
```



```
Windows PowerShell
PS C:\Users\yxz\Desktop\1019> $str="c:\windows\system32\demo.txt"
PS C:\Users\yxz\Desktop\1019> $str.split("\")
c:
windows
system32
demo.txt
PS C:\Users\yxz\Desktop\1019> $str.split("\").gettype()

IsPublic IsSerial Name                                     BaseType
-----
True     True     String[]                                           System.Array

PS C:\Users\yxz\Desktop\1019> _
```

<https://blog.csdn.net/Eastmount>

获取图片名称

```
$str="https://blog.csdn.net/Eastmount/102781411/logo.png"
$str.split("/")[-1]
```

```
PS C:\Users\yxz\Desktop\1019> $str="https://blog.csdn.net/Eastmount/102781411/logo.png"
PS C:\Users\yxz\Desktop\1019> $str.split("/")[-1]
logo.png
PS C:\Users\yxz\Desktop\1019> _
```

是否以某个字符结尾和是否包含某个字符。

```
$str.endswith("png")
$str.contains("csdn")
```

```
PS C:\Users\yxz\Desktop\1019> $str.endswith("png")
True
PS C:\Users\yxz\Desktop\1019> $str.contains("csdn")
True
PS C:\Users\yxz\Desktop\1019> _
```

字符串比较，-1表示两个字符串不一样，相等输出0。

```
$str="https://blog.csdn.net/Eastmount/102781411/logo.png"
$str.compareto("window")
$str.compareto("https://blog.csdn.net/Eastmount/102781411/logo.png")
```

```
PS C:\Users\yxz\desktop\1019> $str.compareto("window")
-1
PS C:\Users\yxz\desktop\1019> $str.compareto("https://blog.csdn.net/Eastmount/102781411/logo.png")
0
PS C:\Users\yxz\desktop\1019>
```

其他操作如下：

// 获取下标

```
$str.indexOf("s")
```

// 字符插入

```
$str.insert(4,"yxz")
```

// 字符串替换

```
$str.replace("n","N")
```

```
PS C:\Users\yxz\desktop\1019> $str.indexOf("s")
4
PS C:\Users\yxz\desktop\1019> $str.insert(4,"yxz")
httpyxzs://blog.csdn.net/Eastmount/102781411/logo.png
PS C:\Users\yxz\desktop\1019> $str.replace("n","N")
https://blog.csdN.Net/EastmouNt/102781411/logo.pNg
```

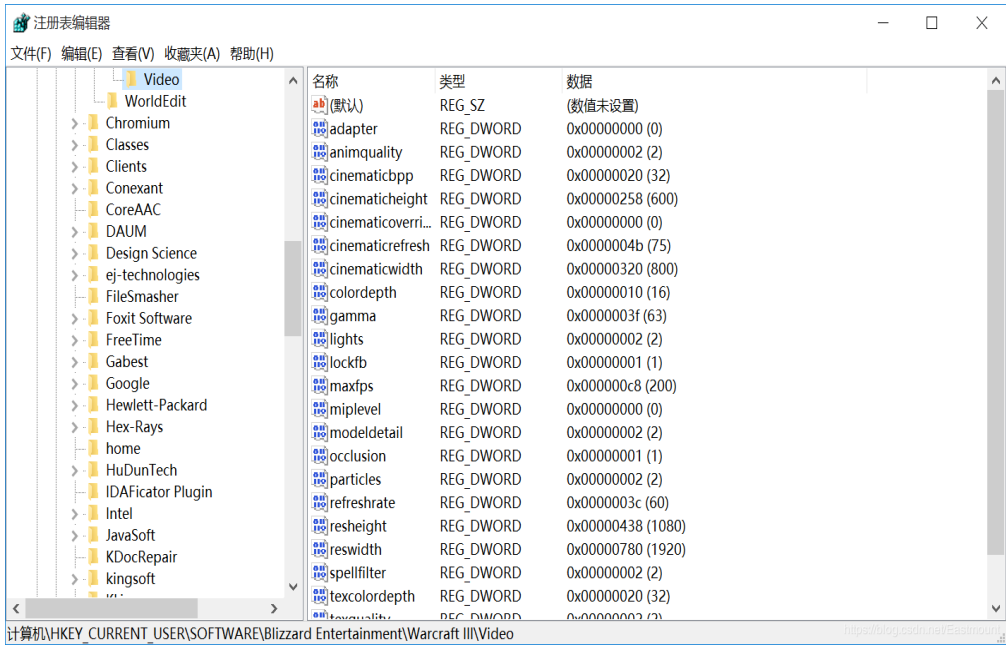
七.Powershell注册表操作

注册表（Registry，繁体中文版Windows操作系统称之为登录档）是Microsoft Windows 中的一个重要的数据库，用于存储系统和应用程序的设置信息。早在Windows 3.0推出OLE技术的时候，注册表就已经出现。随后推出的Windows NT是第一个从系统级别广泛使用注册表的操作系统。但是，从Microsoft Windows 95操作系统开始，注册表才真正成为Windows用户经常接触的内容，并在其后的操作系统中继续沿用至今。

在CMD中输入regedit即可打开注册表，如下图所示。

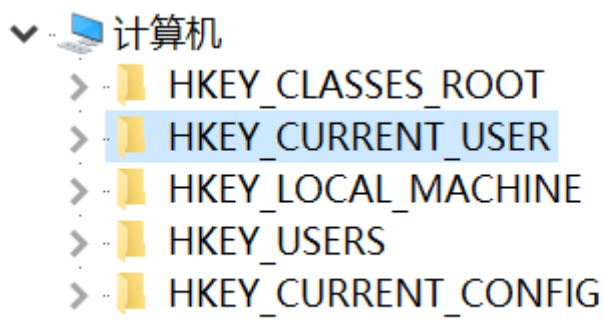
```
C:\Users\yxz>regedit

C:\Users\yxz>_
```



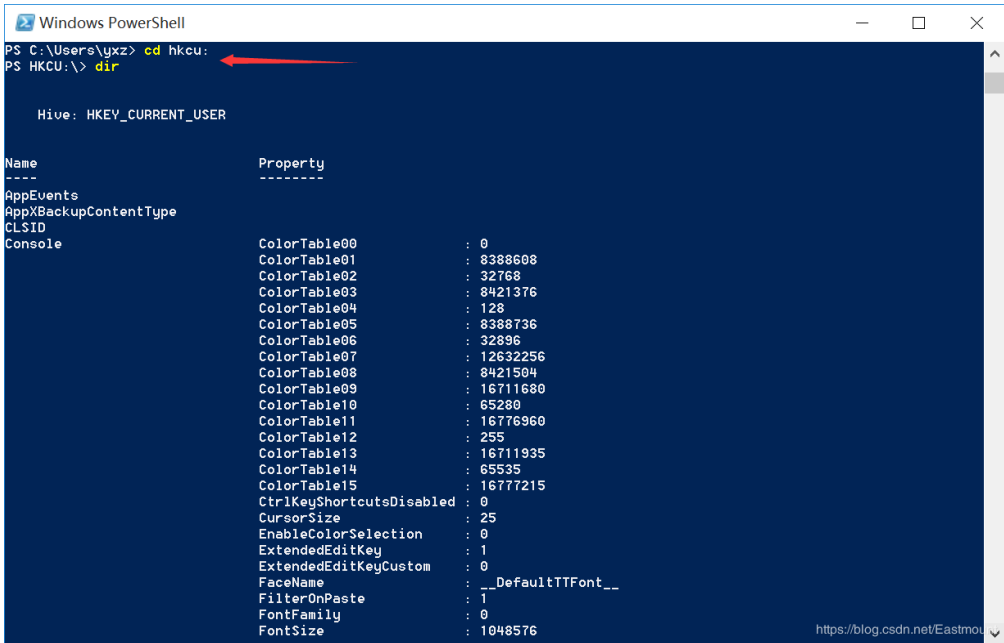
注册表图形化界面显示如下，包括各种程序的配置信息，不能随便修改它，很容易造成系统故障。

- HKEY_CLASSES_ROOT：定义文档的类型\类以及与类型关联的信息以及COM组件的配置数据
- HKEY_CURRENT_USER：包含当前登录到Windows的用户的配置信息
- HKEY_LOCAL_MACHINE：包含与计算机相关的配置信息,不管用户是否登录
- HKEY_USERS：包含有关默认用户配置的信息
- HKEY_CURRENT_CONFIG：包含有关非用户特定的硬件的配置信息



在Powershell中显示注册表指令如下：

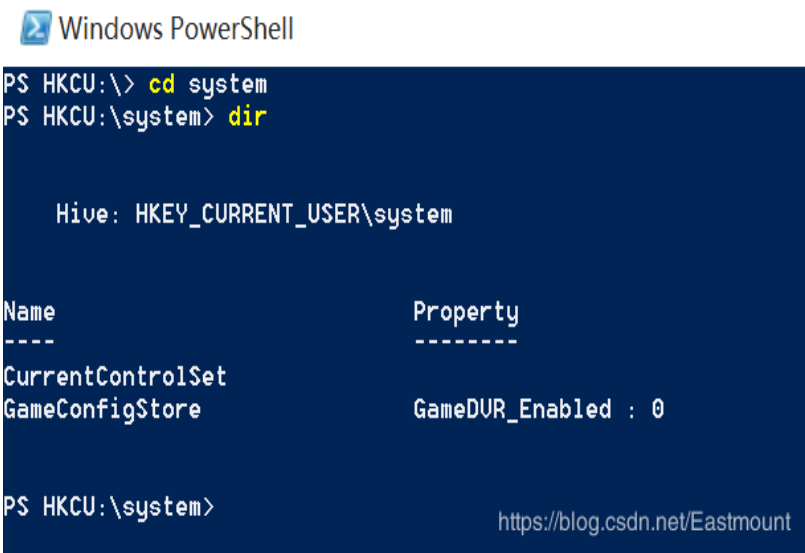
```
cd hkcu:
dir
```

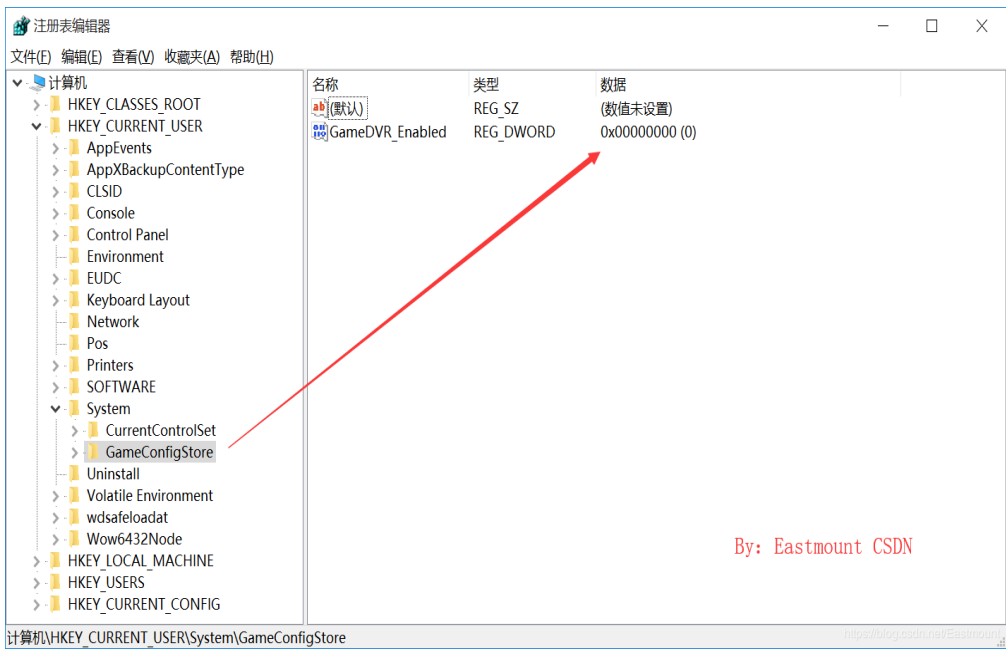
对应注册表图形界面。



cd system
dir



对应图形界面。



其他访问也类似。

cd HKLM:

```

Windows PowerShell
PS C:\Users\yxz> cd HKLM:
PS HKLM:\> dir
dir : 不允许所请求的注册表访问权。
所在位置 行:1 字符: 1
+ ~~~~
+ CategoryInfo          : PermissionDenied: (HKEY_LOCAL_MACHINE\BCD00000000:String) [Get-ChildItem], SecurityExcep
tion
+ FullyQualifiedErrorId : System.Security.SecurityException,Microsoft.PowerShell.Commands.GetChildItemCommand

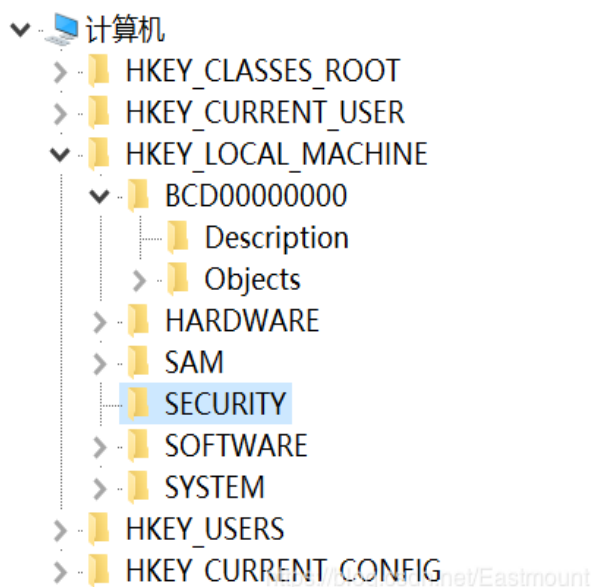
Hive: HKEY_LOCAL_MACHINE

Name                           Property
----                           -
HARDWARE
SAM
dir : 不允许所请求的注册表访问权。
所在位置 行:1 字符: 1
+ ~~~~
+ CategoryInfo          : PermissionDenied: (HKEY_LOCAL_MACHINE\SECURITY:String) [Get-ChildItem], SecurityExcep
tion
+ FullyQualifiedErrorId : System.Security.SecurityException,Microsoft.PowerShell.Commands.GetChildItemCommand

SOFTWARE
SYSTEM

```

对应图形界面:



读取键值

`get-itemproperty`

```

PS HKLM:\> get-itemproperty

位于命令管道位置 1 的 cmdlet Get-ItemProperty
请为以下参数提供值:
Path[0]: 

```

设置键值

`set-itemproperty`

由于注册表不能随便修改，很容易造成系统故障，后续随着作者深入学习，了解更多网络安全中Powershell及注册表工作再来分享，希望读者喜欢该系列文章。

作者作为网络安全的小白，分享一些自学基础教程给大家，希望你们喜欢。同时，更希望你能与我一起操作深入进步，后续也将深入学习网络安全和系统安全知识并分享相关实验。总之，希望该系列文章对博友有所帮助，写文不容易，大神请飘过，不喜勿喷，谢谢！共勉~

(By:Eastmount 2019-10-29 中午12点 <http://blog.csdn.net/eastmount/>)