CNT 4603: System Administration
Spring 2011

Scripting – Windows PowerShell – Part 2

Instructor : Dr. Mark Llewellyn
markl@cs.ucf.edu
HEC 236, 4078-823-2790
http://www.cs.ucf.edu/courses/cnt4603/spr2011

Department of Electrical Engineering and Computer Science
University of Central Florida

Writing And Executing Scripts In PowerShell

- Create the following PowerShell script in a text editor like NotePad++.
- Save this script in the Administrator/MyScripts folder we created in the last set of notes. Save the script with the name `ArrayScript.ps1`. Don't worry about understanding the syntax yet, we'll get to that later.
- Once you've created the script, start PowerShell and at the prompt enter the name of the script.
- You should see screen as it appears on the next page:

Writing And Executing Scripts In PowerShell

C:\Users\Administrator\MyScripts\ArrayScript.ps1 - Notepad++

File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?

config.php | functionWithReturnValue.php | default arguments V2.php

```
1  $Msg = "Writing The Array - Please Wait. . ."
2  $Msg
3  $numArray = (1..20)
4  $x = @()
5  foreach ($num in $NumArray)
6      {$x += $num * 5}
7  $x
```

Ln : 1 Col : 4 Sel : 0 | Dos\Windows | ANSI | INS

Administrator: Windows PowerShell

```
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator\MyScripts> ArrayScript.ps1
The term 'ArrayScript.ps1' is not recognized as the name of a cmdlet, function, script file, or operable program. Chec
 the spelling of the name, or if a path was included, verify that the path is correct and try again.
At line:1 char:16
+ ArrayScript.ps1 <<<<
    + CategoryInfo          : ObjectNotFound: (ArrayScript.ps1:Str                    Exception
    + FullyQualifiedErrorId : CommandNotFoundException

PS C:\Users\Administrator\MyScripts> ArrayScript.ps1
The term 'ArrayScript.ps1' is not recognized as the name of a cmdl                    e, or operable program. Chec
 the spelling of the name, or if a path was included, verify that the path is correct and try again.
At line:1 char:16
+ ArrayScript.ps1 <<<<
    + CategoryInfo          : ObjectNotFound: (ArrayScript.ps1:String) [], CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException


Suggestion [3,General]: The command ArrayScript.ps1 was not found, but does exist in the current location. Windows Pow
Shell doesn't load commands from the current location by default. If you trust this command, instead type ".\ArrayScri
.ps1". See "get-help about_Command_Precedence" for more details.
PS C:\Users\Administrator\MyScripts>
```

OK, so enter the full pathname then!

Writing And Executing Scripts In PowerShell

- PowerShell does not load scripts from the default directory automatically, so as the previous screen shot illustrates, you need to specify the full pathname to the script.
- Do this and you should see the screen as it appears on the next page.

Select Administrator: Windows PowerShell

```
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator\MyScripts> ArrayScript.ps1
The term 'ArrayScript.ps1' is not recognized as the name of a cmdlet, function, script file, or operable program. Chec
 the spelling of the name, or if a path was included, verify that the path is correct and try again.
At line:1 char:16
+ ArrayScript.ps1 <<<<
    + CategoryInfo          : ObjectNotFound: (ArrayScript.ps1:Str
    + FullyQualifiedErrorId : CommandNotFoundException

PS C:\Users\Administrator\MyScripts> ArrayScript.ps1
The term 'ArrayScript.ps1' is not recognized as the name of a cmdl                    program. Chec
 the spelling of the name, or if a path was included, verify that
At line:1 char:16
+ ArrayScript.ps1 <<<<
    + CategoryInfo          : ObjectNotFound: (ArrayScript.ps1:Str
    + FullyQualifiedErrorId : CommandNotFoundException


Suggestion [3,General]: The command ArrayScript.ps1 was not found, but does exist in the current location. Windows Pow
Shell doesn't load commands from the current location by default. If you trust this command, instead type ".\ArrayScri
.ps1". See "get-help about_Command_Precedence" for more details.
PS C:\Users\Administrator\MyScripts> c:\users\administrator\myscripts\ArrayScript.ps1
File C:\users\administrator\myscripts\ArrayScript.ps1 cannot be loaded because the execution of scripts is disabled on
this system. Please see "get-help about_signing" for more details.
At line:1 char:49
+ c:\users\administrator\myscripts\ArrayScript.ps1 <<<<
    + CategoryInfo          : NotSpecified: (:) [], PSSecurityException
    + FullyQualifiedErrorId : RuntimeException

PS C:\Users\Administrator\MyScripts>
```

Now what the &*%$ is going on? A shell that won't run scripts?

Writing And Executing Scripts In PowerShell

- The security settings built into PowerShell include something called the "execution-policy".
- The execution-policy determines how (or if) PowerShell runs scripts.
- By default, PowerShell's execution policy is set to **`Restricted`**; that means that scripts – including those you write yourself – won't run!
- To verify the execution policy settings run the cmdlet `get-executionpolicy`. This is shown on the next page.

Administrator: Windows PowerShell

```
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator\MyScripts> get-executionpolicy
Restricted
PS C:\Users\Administrator\MyScripts> _
```

Writing And Executing Scripts In PowerShell

- The security settings built into PowerShell include something called the "execution-policy".
- The execution-policy determines how (or if) PowerShell runs scripts.
- By default, PowerShell's execution policy is set to **`Restricted`**; that means that scripts – including those you write yourself – won't run!
- To verify the execution policy settings run the cmdlet `get-executionpolicy`. This is shown on the next page.

Administrator: Windows PowerShell

```
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator\MyScripts> get-executionpolicy
Restricted
PS C:\Users\Administrator\MyScripts>
```

Writing And Executing Scripts In PowerShell

- While this security setting might seem a bit severe, but nevertheless that's what it is. So, we need to reset the execution policy.
- To do this, run the cmdlet `set-executionpolicy`.
- To configure PowerShell to run any script you write yourself – without question – but to run scripts downloaded from the Internet only if those scripts have been signed by a trusted publisher, set the execution policy to **`RemoteSigned`**.
- **`AllSigned`** requires all scripts to be signed by a trusted publisher and **`Unrestricted`** allows all scripts to be executed .
- Use the cmdlet to set the policy to **`RemoteSigned`**.

Administrator: Windows PowerShell

```
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator\MyScripts> set-executionpolicy remotesigned

Execution Policy Change
The execution policy helps protect you from scripts that you do not trust. Changing the execution policy might expose
 you to the security risks described in the about_Execution_Policies help topic. Do you want to change the execution
policy?
[Y] Yes  [N] No  [S] Suspend  [?] Help (default is "Y"): y
PS C:\Users\Administrator\MyScripts> _
```

Writing And Executing Scripts In PowerShell

- Now that you've gotten the execution policy set, you can finally execute the ArrayScript script as we tried to do earlier.
- The next page illustrates the execution, finally!, of our script.
- If you want to be able to execute scripts without providing the full pathname to the script, you'll need to modify your path.
- The following command will retrieve your Windows PATH environment variable and display it in a readable fashion.

```
$a = $env:path; $a.split(";");
```

- Note that you can also use the .\ notation to execute a script from within the current directory if you don't want to mess around with your path environment variable.

Administrator: Windows PowerShell

```
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator\MyScripts> c:\users\administrator\myscripts\ArrayScript.ps1
Writing The Array - Please Wait. . .
5
10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
100
PS C:\Users\Administrator\MyScripts> _
```

Select Administrator: Windows PowerShell

```
PS C:\Users\Administrator\MyScripts>
PS C:\Users\Administrator\MyScripts> $a = $env:path; $a.split(";");
%SystemRoot%\system32\WindowsPowerShell\v1.0\
C:\Windows\system32
C:\Windows
C:\Windows\System32\Wbem
C:\Windows\System32\WindowsPowerShell\v1.0\
PS C:\Users\Administrator\MyScripts>
```

See the Windows path environment variable

Administrator: Windows PowerShell

```
PS C:\Users\Administrator\MyScripts> .\ArrayScript.ps1
Writing The Array - Please Wait. . .
5
10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
100
PS C:\Users\Administrator\MyScripts>
```

Using the .\ notation to execute a script from the current directory.

Writing And Executing Scripts In PowerShell

- If you want to modify your path environment variable, you can do so using PowerShell.
- Let's add the MyScripts folder that we created earlier to the path environment.
- The command for this is:

```
$env:path = $env:path + ";c:\users\administrator\MyScripts"
```

Administrator: Windows PowerShell

```
100
PS C:\Users\Administrator\MyScripts> $a = $env:path; $a.split(";");
%SystemRoot%\system32\WindowsPowerShell\v1.0\
C:\Windows\system32
C:\Windows
C:\Windows\System32\Wbem
C:\Windows\System32\WindowsPowerShell\v1.0\
PS C:\Users\Administrator\MyScripts>
PS C:\Users\Administrator\MyScripts>
PS C:\Users\Administrator\MyScripts> $env:path = $env:path + ";c:\users\administrator\MyScripts"
PS C:\Users\Administrator\MyScripts> $a = $env:path; $a.split(";");
%SystemRoot%\system32\WindowsPowerShell\v1.0\
C:\Windows\system32
C:\Windows
C:\Windows\System32\Wbem
C:\Windows\System32\WindowsPowerShell\v1.0\
c:\users\administrator\MyScripts
PS C:\Users\Administrator\MyScripts> ArrayScript.ps1
Writing The Array - Please Wait. . .
5
10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
100
PS C:\Users\Administrator\MyScripts>
```

The MyScripts folder has now been appended to the path variable.

Pipelining In PowerShell

- When you start writing more elaborate scripts in PowerShell (as well as many other scripting languages), you'll eventually realize the benefits of pipelining.

- Its certainly true that not all scripts will need to use a pipeline, however, many will and knowing how to setup an work a pipeline will allow you to create very efficient scripts.

- Unlike like an oil or water pipeline, that is designed to move a liquid from one place to another; a PowerShell pipeline would more closely resemble an assembly line. We're not moving something from one point to another, but rather start with one thing and transform it into something else as it moves along the pipeline. Look at the example on the next page.

Administrator: Windows PowerShell

```
PS C:\Users\Administrator\MyScripts> dir

    Directory: C:\Users\Administrator\MyScripts

Mode                LastWriteTime     Length Name
----                -------------     ------ ----
-a---         3/28/2011  12:51 PM        136 ArrayScript.ps1

PS C:\Users\Administrator\MyScripts> dir | format-list

    Directory: C:\Users\Administrator\MyScripts


Name           : ArrayScript.ps1
Length         : 136
CreationTime   : 3/25/2011 4:01:27 PM
LastWriteTime  : 3/28/2011 12:51:32 PM
LastAccessTime : 3/25/2011 4:01:27 PM
VersionInfo    : File:             C:\Users\Administrator\MyScripts\ArrayScript.ps1
                 InternalName:
                 OriginalFilename:
                 FileVersion:
                 FileDescription:
                 Product:
                 ProductVersion:
                 Debug:            False
                 Patched:          False
                 PreRelease:       False
                 PrivateBuild:     False
                 SpecialBuild:     False
                 Language:


PS C:\Users\Administrator\MyScripts>
```

The directory listing is piped to the format-list which formats the output of the directory command into a list.

Notice how different the non-piped and the piped outputs look.

Pipelining In PowerShell

- Now let's look at a couple of somewhat more practical/useful examples.

- The first uses the cmdlet `get-childitem` to retrieve a list of all the items in the `myScripts` folder. We'll pipe this output to the `where-object` cmdlet that will filter out any item greater than 200KB in size, and then pipe this result set to the `sort-object` cmdlet. This is shown on page 22.

- The second example gets the services on the server, pipe this set to the `sort-object` cmdlet to sort based on the service's status and finally pipes this result to the `format-table` cmdlet to see the results in a table based format. This examples is shown on page 23.

Administrator: Windows PowerShell

```
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator\MyScripts> get-childitem c:\users\administrator\myscripts | where-object {$_.length -lt 20
 | sort-object length


    Directory: C:\users\administrator\myscripts


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
-a---         3/28/2011  12:51 PM        136 ArrayScript.ps1
-a---         3/30/2011   2:18 PM        177 ArrayScript2.ps1
-a---         3/30/2011   2:19 PM        198 ArrayScript3.ps1


PS C:\Users\Administrator\MyScripts> _
```

Administrator: Windows PowerShell

```
PS C:\Users\Administrator\MyScripts>
PS C:\Users\Administrator\MyScripts>
PS C:\Users\Administrator\MyScripts> get-service | sort-object status | format-table

Status   Name               DisplayName
------   ----               -----------
Stopped  MMCSS              Multimedia Class Scheduler
Stopped  UmRdpService       Terminal Services UserMode Port Red...
Stopped  lltdsvc            Link-Layer Topology Discovery Mapper
Stopped  upnphost           UPnP Device Host
Stopped  msiserver          Windows Installer
Stopped  TrustedInstaller   Windows Modules Installer
Stopped  UI0Detect          Interactive Services Detection
Stopped  MSiSCSI            Microsoft iSCSI Initiator Service
Stopped  IPBusEnum          PnP-X IP Bus Enumerator
Stopped  VSS                Volume Shadow Copy
Stopped  hkmsvc             Health Key and Certificate Management
Stopped  idsvc              Windows CardSpace
Stopped  vds                Virtual Disk
Stopped  SSDPSRV            SSDP Discovery
Stopped  KeyIso             CNG Key Isolation
Stopped  SCardSvr           Smart Card
Stopped  ProtectedStorage   Protected Storage
Stopped  RasAuto            Remote Access Auto Connection Manager
Stopped  RSoPProv           Resultant Set of Policy Provider
Stopped  TBS                TPM Base Services
Stopped  swprv              Microsoft Software Shadow Copy Prov...
Stopped  RpcLocator         Remote Procedure Call (RPC) Locator
Stopped  SysMain            Superfetch
Stopped  RemoteAccess       Routing and Remote Access
Stopped  sacsvr             Special Administration Console Helper
Stopped  Tomcat7            Apache Tomcat 7
Stopped  napagent           Network Access Protection Agent
Stopped  Netlogon           Netlogon
Stopped  Themes             Themes
Stopped  pla                Performance Logs & Alerts
Stopped  NetTcpPortSharing  Net.Tcp Port Sharing Service
Stopped  THREADORDER        Thread Ordering Server
Stopped  CertPropSvc        Certificate Propagation
Stopped  clr_optimizatio... Microsoft .NET Framework NGEN v2.0....
Stopped  wmiApSrv           WMI Performance Adapter
Stopped  Browser            Computer Browser
Stopped  CscService         Offline Files
Stopped  SLUINotify         SL UI Notification Service
Stopped  COMSysApp          COM+ System Application
Stopped  SessionEnv         Terminal Services Configuration
Stopped  SharedAccess       Internet Connection Sharing (ICS)
Stopped  Appinfo            Application Information
Stopped  wudfsvc            Windows Driver Foundation - User-mo...
```

Pipelining In PowerShell

- As you can see, its fairly easy to take advantage of pipelining in PowerShell.
- However, you do need to use caution. Not everything can be pipelined. You can't pipeline something unless it makes sense to use a pipeline.
- In the previous example, it makes sense to pipeline the service information to the `sort-object` cmdlet, since sort-object can pretty much sort anything. It also makes sense to pipe the sorted information to `format-table` because it can format just about any information and display it as a table.
- What would this command do?

```
Sort-object | get-process
```

Pipelining In PowerShell

- Answer: Absolutely nothing! Since `sort-object` is designed to sort things and here it has nothing to sort, so it will pass an empty result set to the `get-process` cmdlet which will do nothing.

```
Select Administrator: Windows PowerShell
Running  lmhosts            TCP/IP NetBIOS Helper
Running  RemoteRegistry     Remote Registry
Running  RpcSs              Remote Procedure Call (RPC)
Running  ProfSvc            User Profile Service
Running  RasMan             Remote Access Connection Manager
Running  SamSs              Security Accounts Manager
Running  SENS               System Event Notification Service
Running  ShellHWDetection   Shell Hardware Detection
Running  Schedule           Task Scheduler
Running  seclogon           Secondary Logon
Running  mysql              mysql
Running  Netman             Network Connections
Running  MpsSvc             Windows Firewall
Running  MSDTC              Distributed Transaction Coordinator
Running  netprofm           Network List Service
Running  PlugPlay           Plug and Play
Running  PolicyAgent        IPsec Policy Agent
Running  NlaSvc             Network Location Awareness
Running  nsi                Network Store Interface Service

PS C:\Users\Administrator\MyScripts> sort-object | get-process
PS C:\Users\Administrator\MyScripts>
```

See… I told you so!

Pipelining In PowerShell

- For the most part, and there are some exceptions to the rule, for pipelines to work correctly, you first acquire something (a collection, an object, whatever) and then hand that data over the pipeline.
  - One exception to the rule would be the following situation where $a represents a variable that contains a collection of data. You could sort the data in $a and sidestep the pipeline altogether with a command like:
    `sort-object  -inputobject  $a`
- When you do hand data over the pipeline, make sure that there is a cmdlet waiting for it on the other side.
- The example on the next page illustrates a common pipelining mistake.

Pipelining In PowerShell

- Suppose you entered a command like this:

```
$a = get-process | $a
```

- While it might look ok; you're thinking that will assign the output of the `get-process` cmdlet to the variable `$a` and then display `$a`. Instead you're going to get an error.

```
Administrator: Windows PowerShell
PS C:\Users\Administrator\MyScripts>
PS C:\Users\Administrator\MyScripts>
PS C:\Users\Administrator\MyScripts>
PS C:\Users\Administrator\MyScripts>
PS C:\Users\Administrator\MyScripts>
PS C:\Users\Administrator\MyScripts> $a = get-process
PS C:\Users\Administrator\MyScripts> $a = get-process | $a
Expressions are only allowed as the first element of a pipeline.
At line:1 char:22
+ $a = get-process | $a <<<<
    + CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
    + FullyQualifiedErrorId : ExpressionsMustBeFirstInPipeline

PS C:\Users\Administrator\MyScripts> _
```

Pipelining In PowerShell

- Pipelines are used to string multiple commands into a single command, with data being passed from one portion of the pipeline to the next.

- Furthermore, as that data gets passed from one section of the pipe to another it gets transformed in some way: filtered, sorted, grouped, formatted, whatever.

- In the invalid command on the previous page, we didn't pass any data. We've really got two separate commands here: we want to use the `get-process` cmdlet to return information about the processes running on the server and them without transforming that data in any way, we want to display the information. Since they are two separate command, they should be on two separate lines as shown on the next page.

Administrator: Windows PowerShell

```
PS C:\Users\Administrator\MyScripts>
PS C:\Users\Administrator\MyScripts>
PS C:\Users\Administrator\MyScripts> $a = get-process
PS C:\Users\Administrator\MyScripts> $a

Handles  NPM(K)    PM(K)      WS(K) VM(M)   CPU(s)     Id ProcessName
-------  ------    -----      ----- -----   ------     -- -----------
    597       5     1568       1012    99     1.70    444 csrss
    225       5     2488       3920   106    17.07    484 csrss
     73       3     1084        208    42     0.09   3940 dwm
    585      14    20904      14332   170    19.45   2656 explorer
    282      26    45700       4212   129     2.22    316 httpd
    128      11    41388         20    91     1.87   1484 httpd
      0       0        0         12     0                0 Idle
    202       6     3492       1124    64     0.36   4076 jucheck
    206       6     2496       1056    68     1.16   3880 jusched
    578       9     3044       3352    38     7.23    580 lsass
    155       3     1528       1212    23     0.36    588 lsm
    162       7     2684         36    58     0.09   2740 msdtc
    503       6    55652       1132   102     6.30   1620 mysqld
    146       9    18600      18080   121   128.78   3136 opera
    471       6    33272      33412   158     0.97   2840 powershell
    237       6     2008       2088    26     3.34    568 services
    128       4     7428       9784    37     7.45    948 SLsvc
     28       1      244         32     4     0.06    372 smss
    280       8     5576       1736    76    59.06   1420 spoolsv
    291       4     2488       2572    31    11.92    740 svchost
    242       7     2544       2044    28     2.77    800 svchost
    293       9     5136       4072    36   105.95    876 svchost
    145       4     2888       2424    29     1.60    916 svchost
   1056      37    48924      35676   154    54.80    936 svchost
    284      13     4096       4612    39    20.15   1020 svchost
    251       8     7052        704    61     0.53   1108 svchost
    414      13    15352       5216    73     8.13   1144 svchost
    270      22     4964       3560    42     8.66   1284 svchost
    123       5     1848         32    30     0.07   1644 svchost
     73       2      792         24    21     0.00   1656 svchost
     44       1      528        484    15     0.04   1776 svchost
     64       2     1240         16    23     0.09   2688 svchost
    228       7     3160        168    44     0.06   3464 svchost
    514       0        0         68     4                4 System
    138       5     1944       2160    47     0.40   1988 taskeng
    250       7     2900       2444    72     2.70   3404 taskeng
     93       4     3732        364    58     2.63   1444 vmsrvc
     52       2     1852       1024    52     0.77   3912 vmusrvc
     26       1      364        232    13     0.86   1744 vpcmap
    100       4     1080         16    35     0.19    492 wininit
    123       3     1260       1748    26     0.41    520 winlogon
     83       3     2436       4888    55     0.04   4052 wuauclt
```

Administrator: Windows PowerShell

```
PS C:\Users\Administrator\MyScripts> $a = get-process;  $a

Handles  NPM(K)    PM(K)      WS(K) VM(M)   CPU(s)     Id ProcessName
-------  ------    -----      ----- -----   ------     -- -----------
    591       5     1568       1012    99     1.70    444 csrss
    224       5     2488       3920   106    17.19    484 csrss
     73       3     1084        208    42     0.09   3940 dwm
    585      14    20904      14332   170    19.45   2656 explorer
    282      26    45700       4212   129     2.22    316 httpd
    128      11    41388         20    91     1.87   1484 httpd
      0       0        0         12     0                0 Idle
    202       6     3492       1124    64     0.36   4076 jucheck
    206       6     2496       1056    68     1.16   3880 jusched
    576       9     3008       3320    38     7.23    580 lsass
    153       3     1484       1196    23     0.36    588 lsm
    162       7     2684         36    58     0.09   2740 msdtc
    503       6    55652       1132   102     6.30   1620 mysqld
    146       9    18604      18084   121   128.83   3136 opera
    299       6    33272      33424   158     1.08   2840 powershell
    237       6     2008       2088    26     3.34    568 services
    128       4     7428       9784    37     7.45    948 SLsvc
     28       1      244         32     4     0.06    372 smss
    280       8     5576       1736    76    59.06   1420 spoolsv
    291       4     2488       2572    31    11.92    740 svchost
    240       7     2516       2032    27     2.77    800 svchost
    293       9     5136       4072    36   105.95    876 svchost
    145       4     2888       2424    29     1.60    916 svchost
   1054      37    48924      35672   154    54.80    936 svchost
    282      12     4068       4588    38    20.15   1020 svchost
    251       8     7052        704    61     0.53   1108 svchost
    414      13    15324       5204    73     8.13   1144 svchost
    268      22     4936       3548    42     8.66   1284 svchost
    123       5     1848         32    30     0.07   1644 svchost
     73       2      792         24    21     0.00   1656 svchost
     44       1      528        484    15     0.04   1776 svchost
     64       2     1240         16    23     0.09   2688 svchost
    228       7     3160        168    44     0.06   3464 svchost
    514       0        0         68     4                4 System
    138       5     1944       2160    47     0.40   1988 taskeng
    248       7     2880       2432    72     2.70   3404 taskeng
     93       4     3732        364    58     2.63   1444 vmsrvc
     52       2     1852       1024    52     0.77   3912 vmusrvc
     26       1      364        232    13     0.86   1744 vpcmap
    100       4     1080         16    35     0.19    492 wininit
    123       3     1260       1748    26     0.41    520 winlogon
     83       3     2436       4888    55     0.04   4052 wuauclt

PS C:\Users\Administrator\MyScripts>
```

If you really want to do it this way, then separate the two distinct commands on the same line with semi-colons. Note however, that this is not pipelining.

Pipelining In PowerShell

- Often, as some of the previous examples have illustrated, the system administrator may wish to execute some command and save the results in a variable.

- The results of a pipeline can be stored in a variable in the same manner in which the results of a single command can be stored in a variable. The previous example illustrated saving the output of the `get-process` cmdlet into a variable `$a`. (All variables in PowerShell begin with a `$`.)

- The example on the next page illustrates saving the results of a pipeline.

Administrator: Windows PowerShell

```
PS C:\Users\Administrator\MyScripts>
PS C:\Users\Administrator\MyScripts>
PS C:\Users\Administrator\MyScripts> $a = (get-process | sort-object id)
PS C:\Users\Administrator\MyScripts> $a

Handles  NPM(K)    PM(K)      WS(K) VM(M)   CPU(s)     Id ProcessName
-------  ------    -----      ----- -----   ------     -- -----------
      0       0        0         12     0                0 Idle
    514       0        0         68     4                4 System
    282      26    45700       4212   129     2.22    316 httpd
     28       1      244         32     4     0.06    372 smss
    591       5     1568       1012    99     1.70    444 csrss
    224       5     2488       3920   106    17.40    484 csrss
    100       4     1080         16    35     0.19    492 wininit
    123       3     1260       1748    26     0.41    520 winlogon
    237       6     2008       2088    26     3.34    568 services
    579       9     3044       3332    38     7.25    580 lsass
    153       3     1484       1196    23     0.36    588 lsm
    291       4     2488       2572    31    11.92    740 svchost
    240       7     2516       2036    27     2.77    800 svchost
    293       9     5136       4072    36   105.96    876 svchost
    145       4     2888       2424    29     1.60    916 svchost
   1057      37    48896      35664   153    54.80    936 svchost
    128       4     7428       9784    37     7.45    948 SLsvc
    282      12     4068       4588    38    20.18   1020 svchost
    251       8     7052        704    61     0.53   1108 svchost
    412      13    15312       5204    73     8.14   1144 svchost
    268      22     4936       3544    42     8.66   1284 svchost
    280       8     5576       1736    76    59.06   1420 spoolsv
     93       4     3732        364    58     2.63   1444 vmsrvc
    128      11    41388         20    91     1.87   1484 httpd
    503       6    55652       1132   102     6.32   1620 mysqld
    123       5     1848         32    30     0.07   1644 svchost
     73       2      792         24    21     0.00   1656 svchost
     26       1      364        232    13     0.86   1744 vpcmap
     44       1      528        484    15     0.04   1776 svchost
    138       5     1944       2160    47     0.40   1988 taskeng
    585      14    20904      14332   170    19.45   2656 explorer
     64       2     1240         16    23     0.09   2688 svchost
    162       7     2684         36    58     0.09   2740 msdtc
    431       6    30648      30864   158     1.26   2840 powershell
    146       9    18604      18084   121   128.97   3136 opera
    248       7     2880       2432    72     2.70   3404 taskeng
    228       7     3160        168    44     0.06   3464 svchost
    206       6     2496       1056    68     1.16   3880 jusched
     52       2     1852       1024    52     0.79   3912 vmusrvc
     73       3     1084        208    42     0.09   3940 dwm
     83       3     2436       4888    55     0.04   4052 wuauclt
    202       6     3492       1124    64     0.36   4076 jucheck
```