CNT 4603: System Administration
Spring 2014

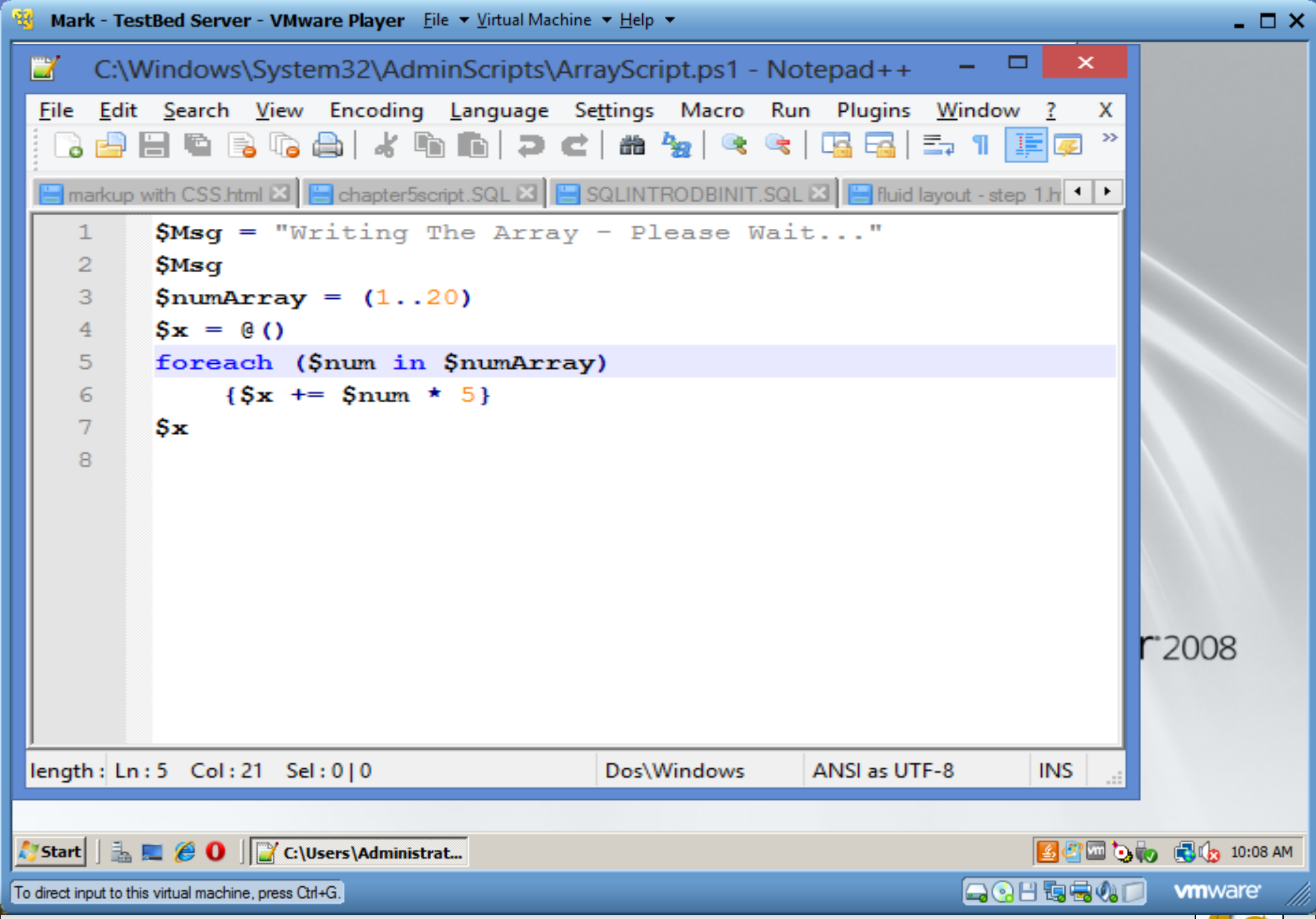
Scripting – Windows PowerShell – Part 2

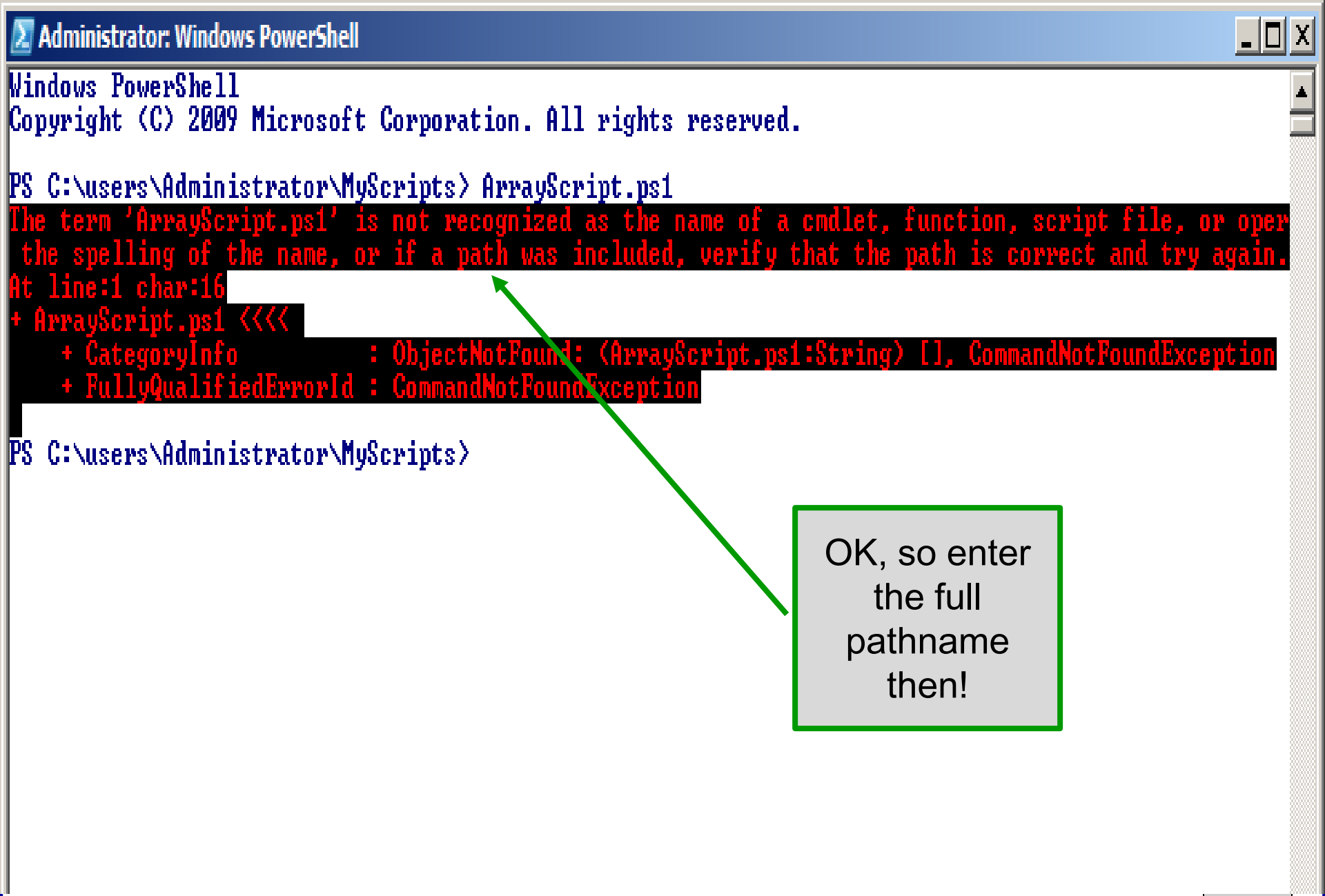
Instructor : Dr. Mark Llewellyn
markl@cs.ucf.edu
HEC 236, 4078-823-2790
http://www.cs.ucf.edu/courses/cnt4603/spr2014

Department of Electrical Engineering and Computer Science
Computer Science Division
University of Central Florida

Writing And Executing Scripts In PowerShell

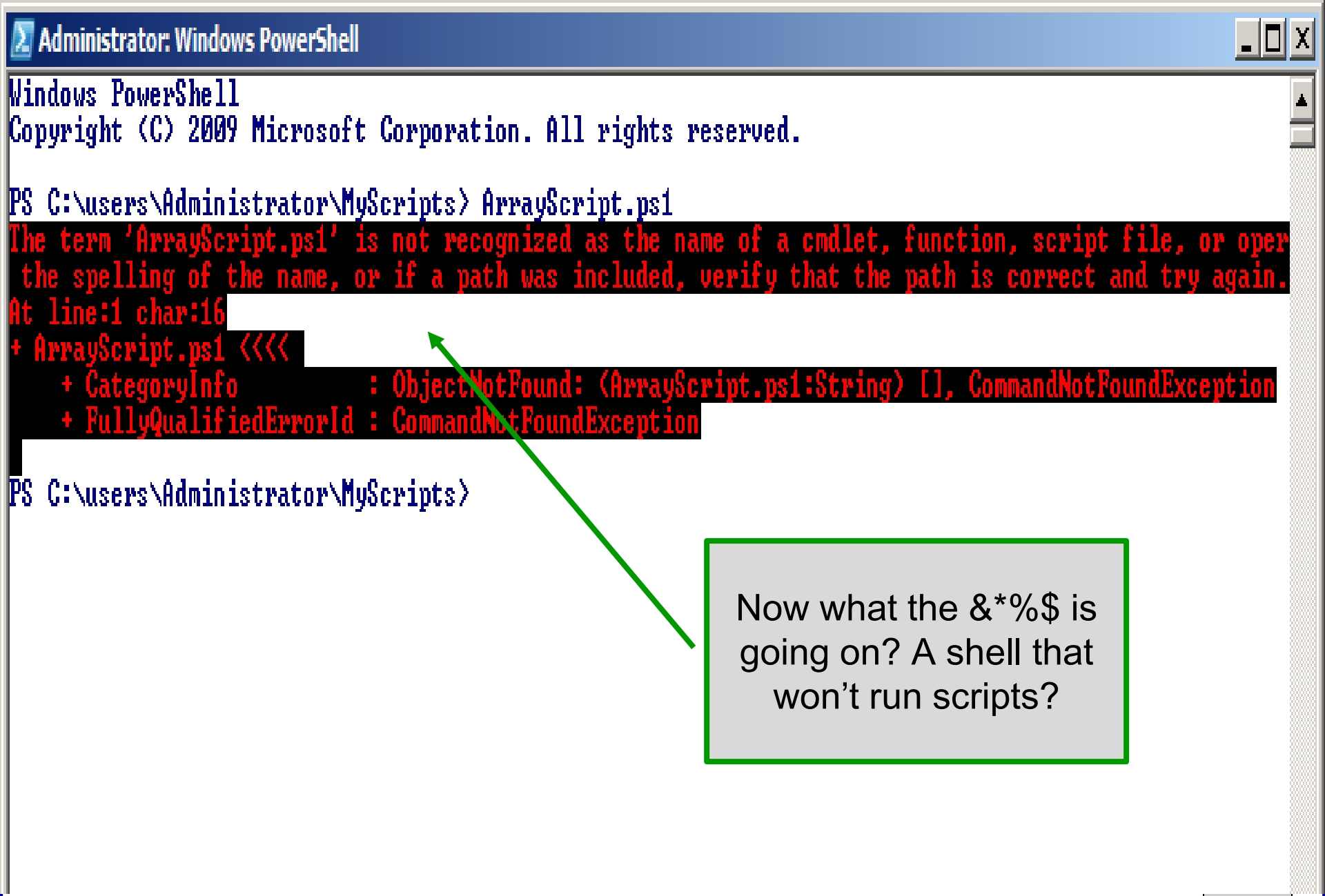
- Create the following PowerShell script in a text editor like Notepad or NotePad++.
- Save this script in the Administrator/MyScripts folder we created in the last set of notes. Save the script with the name `ArrayScript.ps1`. (In Notepad++, the PowerShell extension `.ps1` is a predefined extension.) 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:


Mark - TestBed Server - VMware Player
File
Virtual Machine
Help
C:\Windows\System32\AdminScripts\ArrayScript.ps1 - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ? X
markup with CSS.html
chapter5script.SQL
SQLINTRODBINIT.SQL
fluid layout - step 1.h
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
8
length : Ln : 5 Col : 21 Sel : 0 | 0
Dos\Windows
ANSI as UTF-8
INS
Start
C:\Users\Administrat...
10:08 AM
To direct input to this virtual machine, press Ctrl+G.
vmware


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 oper
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
PS C:\users\Administrator\MyScripts>
OK, so enter the full pathname then!

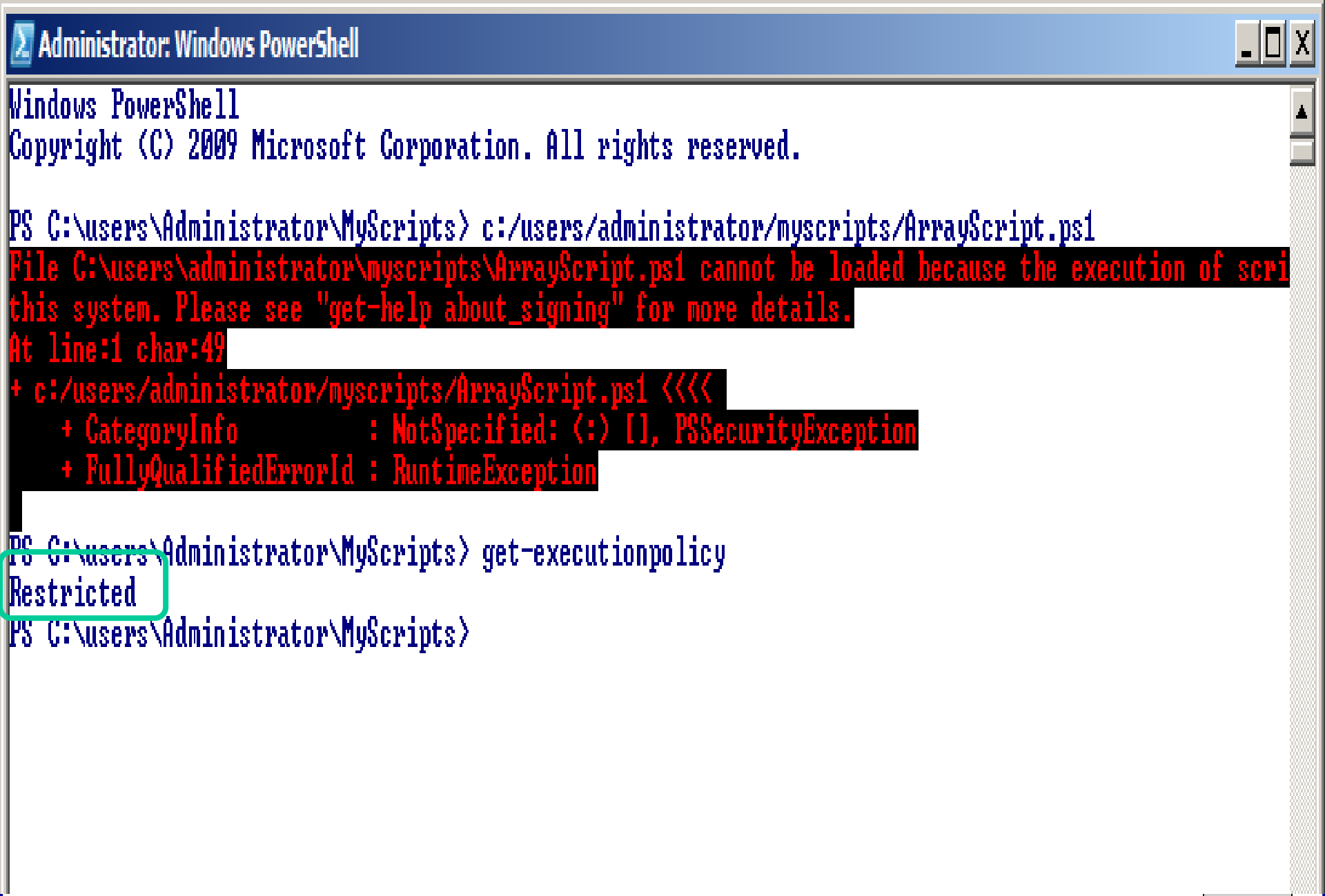
Writing And Executing Scripts In PowerShell

- 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.


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 oper
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
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> c:/users/administrator/myscripts/ArrayScript.ps1
File C:\users\administrator\myscripts\ArrayScript.ps1 cannot be loaded because the execution of scri
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> get-executionpolicy
Restricted
PS C:\users\Administrator\MyScripts>

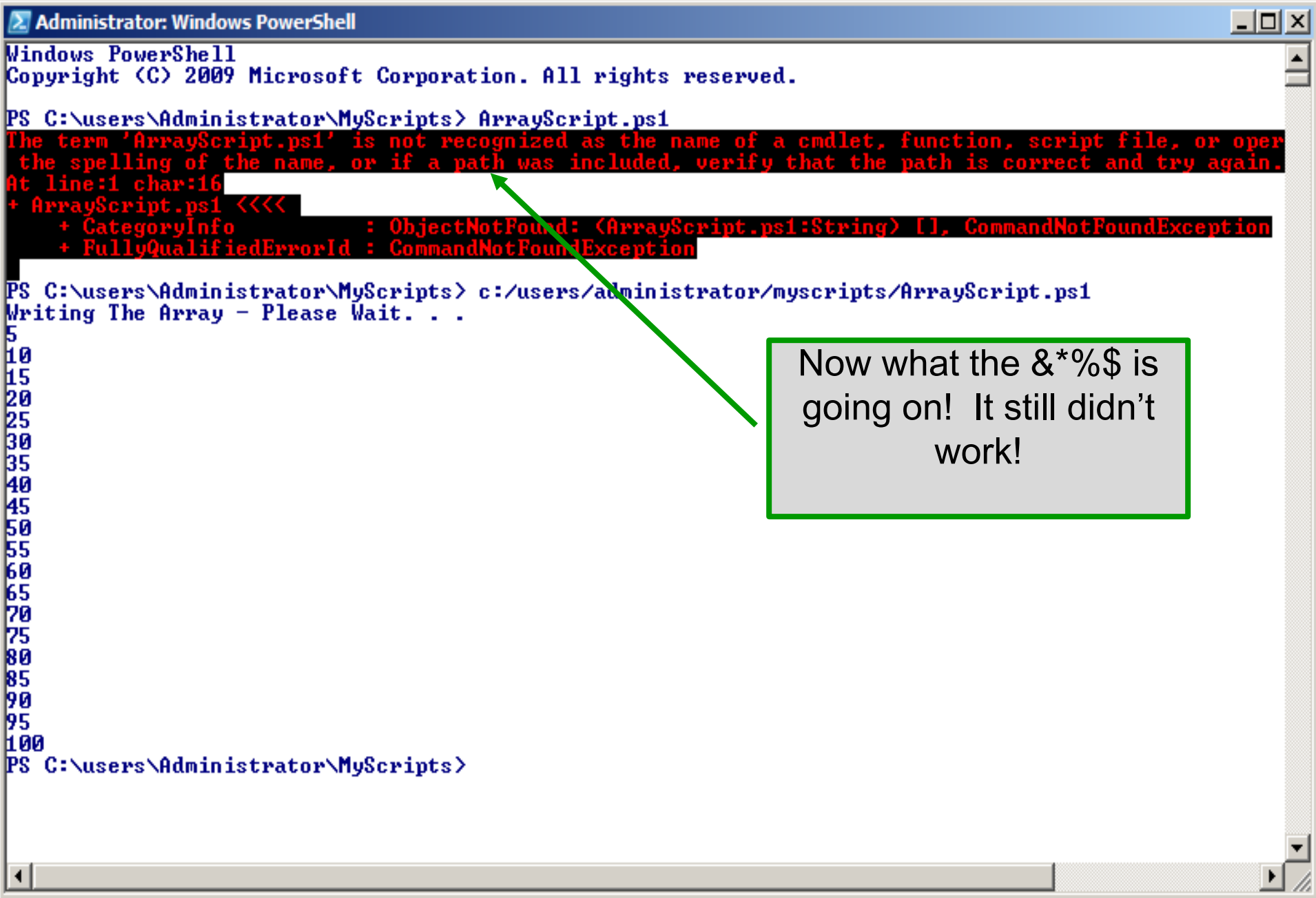
Writing And Executing Scripts In PowerShell

- While this security setting might seem a bit severe, 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> c:/users/administrator/myscripts/ArrayScript.ps1
File C:\users\administrator\myscripts\ArrayScript.ps1 cannot be loaded because the execution of scri
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> get-executionpolicy
Restricted
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.

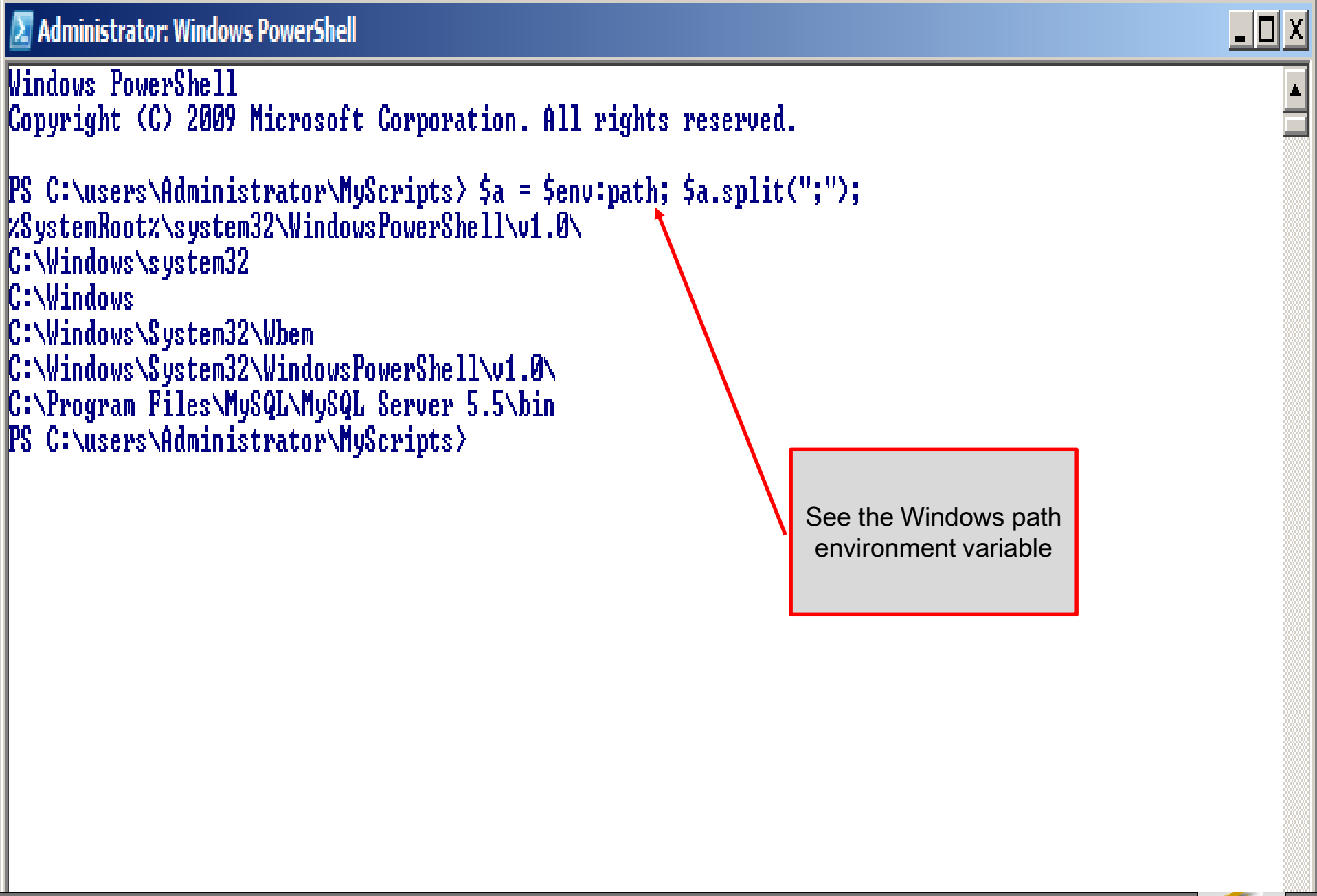

Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.
PS C:\users\Administrator\MyScripts> ArrayScript.ps1
The term 'ArrayScript.ps1' is not recognized as the name of a cmdlet, function, script file, or oper
the spelling of the name, or if 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
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>
Now what the &*%$ is going on! It still didn't work!

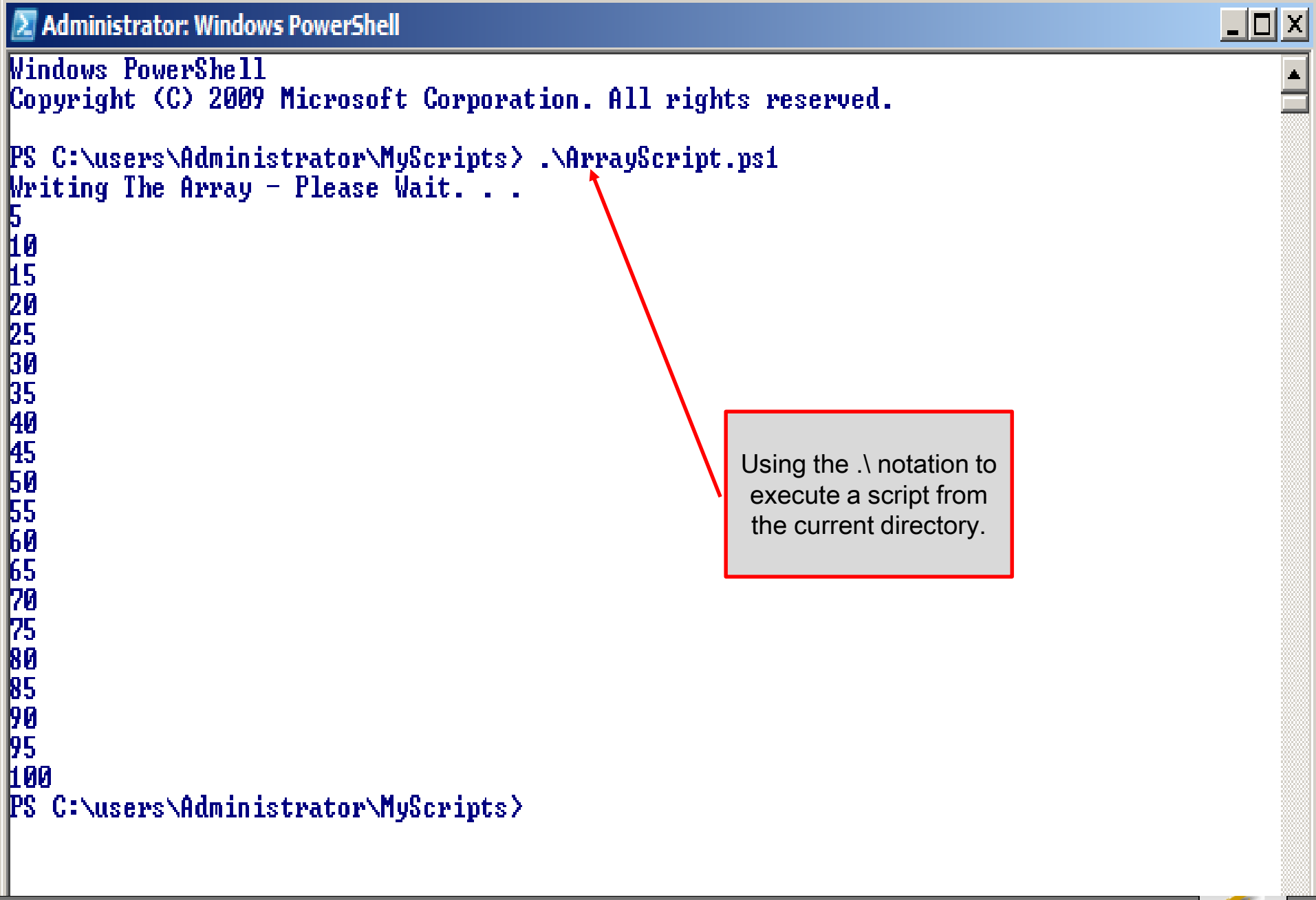
Writing And Executing Scripts In PowerShell

- As you can see from the previous slide, PowerShell does not run scripts without a fully specified pathname!
- 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.
- See the next two pages for illustrations of this.

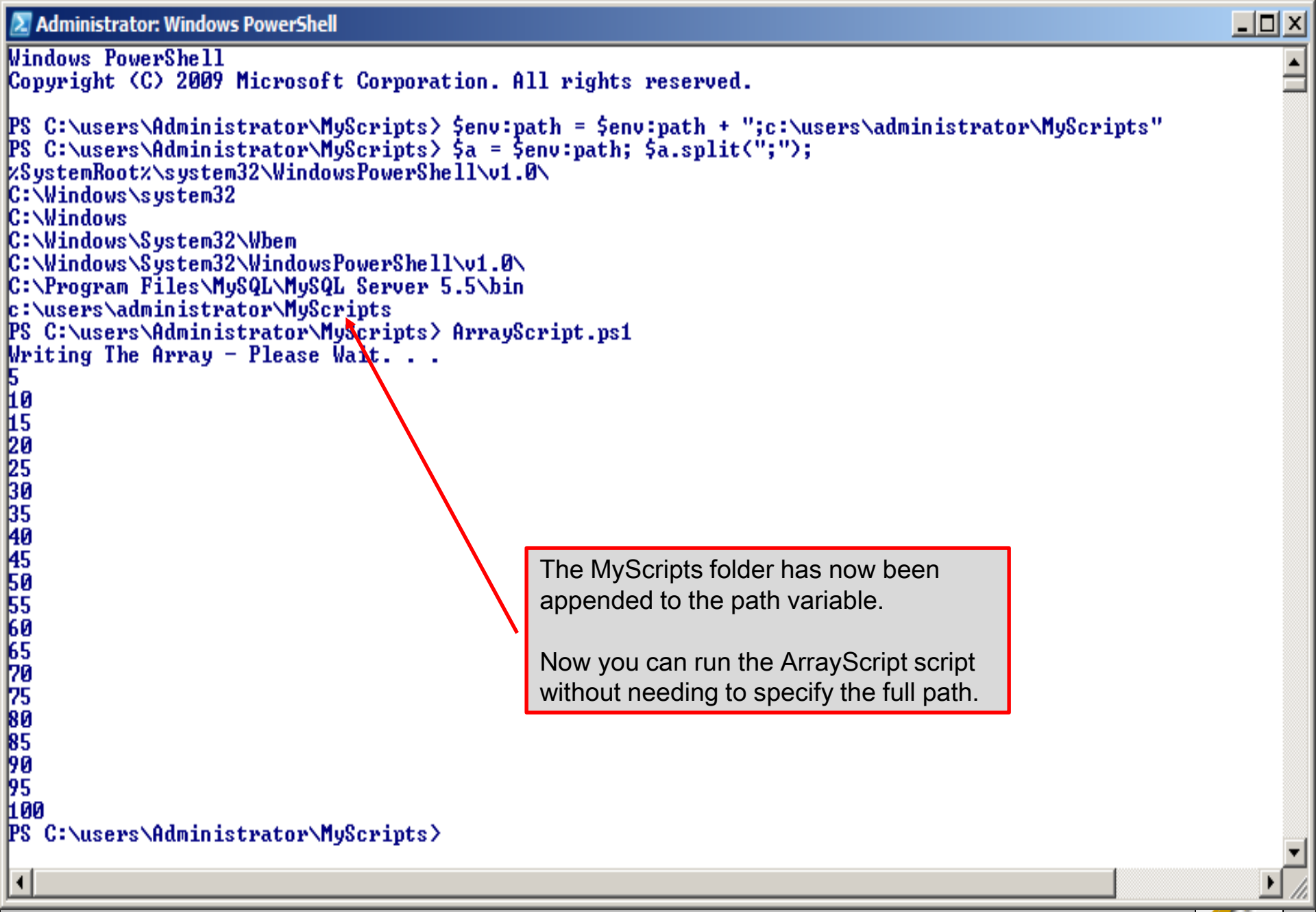

Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.
PS C:\users\Administrator\MyScripts> $a = $env:path; $a.split(";");
%SystemRoot%\system32\WindowsPowerShell\v1.0\
C:\Windows\system32
C:\Windows
C:\Windows\System32\Wbem
C:\Windows\System32\WindowsPowerShell\v1.0\
C:\Program Files\MySQL\MySQL Server 5.5\bin
PS C:\users\Administrator\MyScripts>
See the Windows path environment variable


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

- As your first real system administrator project with PowerShell. We'll modify your path environment variable.
- 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"
```

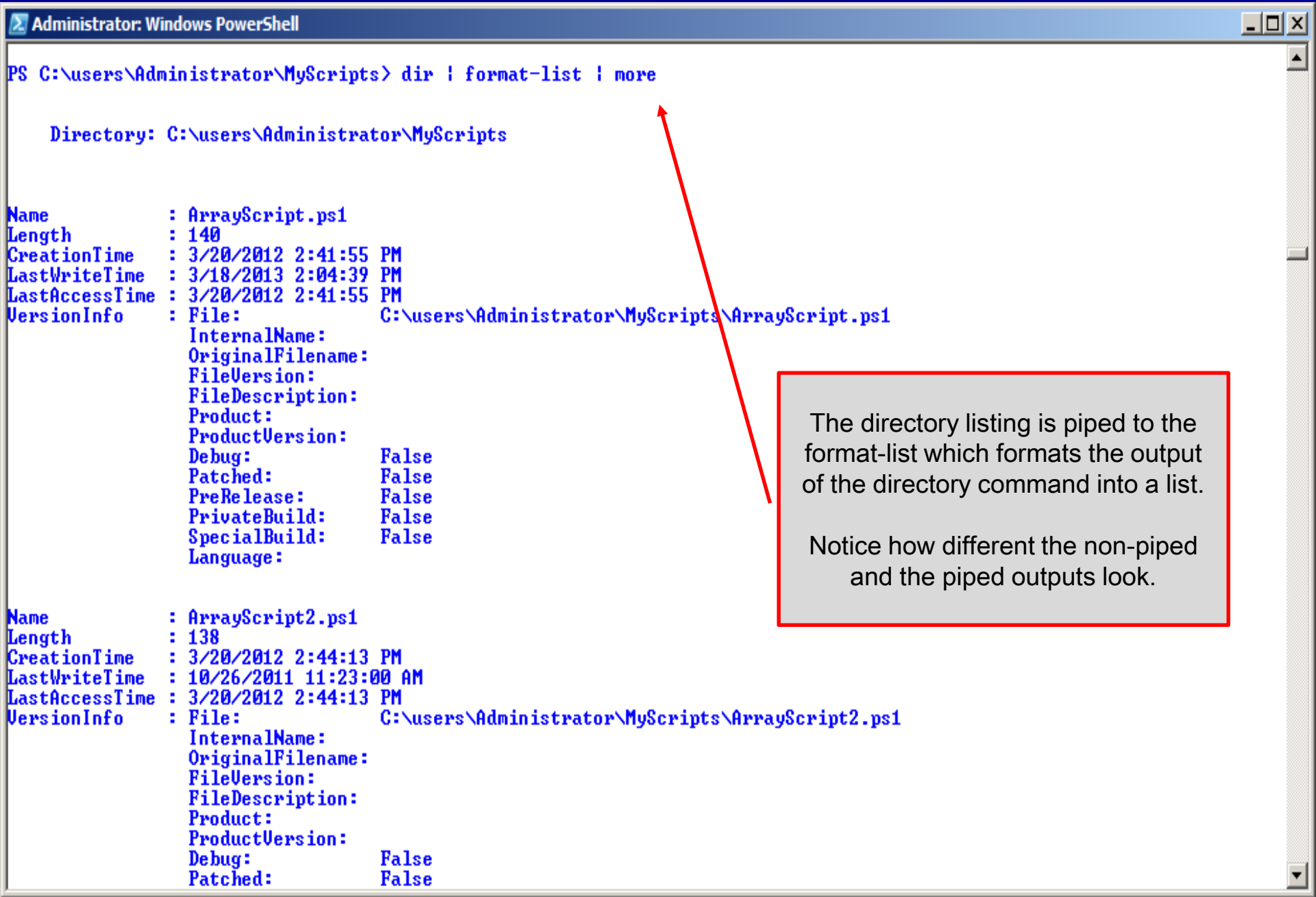

Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.
PS C:\users\Administrator\MyScripts> $env:path = $env:path + ";c:\users\administrator\MyScripts"
PS C:\users\Administrator\MyScripts> $a = $env:path; $a.split(";");
%SystemRoot%\system32\WindowsPowerShell\v1.0\
C:\Windows\system32
C:\Windows
C:\Windows\System32\Wbem
C:\Windows\System32\WindowsPowerShell\v1.0\
C:\Program Files\MySQL\MySQL Server 5.5\bin
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.
Now you can run the ArrayScript script without needing to specify the full path.

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 and 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

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

PS C:\users\Administrator\MyScripts> dir


    Directory: C:\users\Administrator\MyScripts


Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
-a---          3/18/2013   2:04 PM        140 ArrayScript.ps1
-a---         10/26/2011  11:23 AM        138 ArrayScript2.ps1
-a---          3/18/2013   2:06 PM        150 ArrayScript3.ps1
-a---         10/26/2011  11:23 AM        138 ArrayScript4.ps1
-a---          3/29/2012   3:17 PM       3252 Get-StoppedService.ps1
-a---          11/9/2011  12:41 PM       1572 listStoppedService.ps1
-a---          3/29/2012   3:17 PM       3252 Project.ps1
-a---          11/9/2011   2:37 PM       1629 Project9.ps1
-a---          11/2/2011   9:46 AM        855 PS-Part4-p19.ps1
-a---          11/2/2011  12:21 PM       1505 PS-Part4-p27.ps1
-a---          11/7/2011  12:15 PM        658 PS-Part4-p35.ps1
-a---          11/7/2011   2:42 PM        663 PS-Part4-p36.ps1
-a---          11/8/2011   1:32 PM        847 PS-Part4-p41.ps1
-a---          11/8/2011   2:27 PM        910 PS-Part4-p48.ps1
-a---          11/8/2011   7:25 PM        987 PS-Part4-p50.ps1
-a---          11/9/2011   9:39 AM       3258 PS-Part5-p15-invalidsignature.ps1
-a---          11/9/2011   9:48 AM       1737 PS-Part5-p15-pristine.ps1
-a---           4/3/2013   1:29 PM       3289 PS-Part5-p15-signed.ps1
-a---           4/3/2013   1:52 PM       3287 PS-Part5-p15-signedV2.ps1
-a---          11/9/2011   9:43 AM       3136 PS-Part5-p15.ps1
-a---         11/14/2011  12:34 PM       1197 PS-Part6-p15.ps1
-a---         11/14/2011  12:45 PM        990 PS-Part6-p17.ps1
-a---         11/16/2011  11:35 AM       2054 PS-Part6-p29.ps1
-a---         11/14/2011   8:51 AM       1144 PS-Part6-p3.ps1
-a---         11/16/2011  12:01 PM        969 PS-Part6-p33.ps1
-a---         11/16/2011   1:04 PM       1196 PS-Part6-p37.ps1
-a---          4/4/2012   2:10 PM       2253 PS-Part6-p7.ps1
-a---         11/16/2011  12:01 PM       4846 s-processes.txt
-a---         10/31/2011   3:09 PM         96 server.csv
-a---         10/31/2011  12:59 PM        408 temp.ps1
-a---          11/2/2011  12:14 PM        110 textfile.csv
```

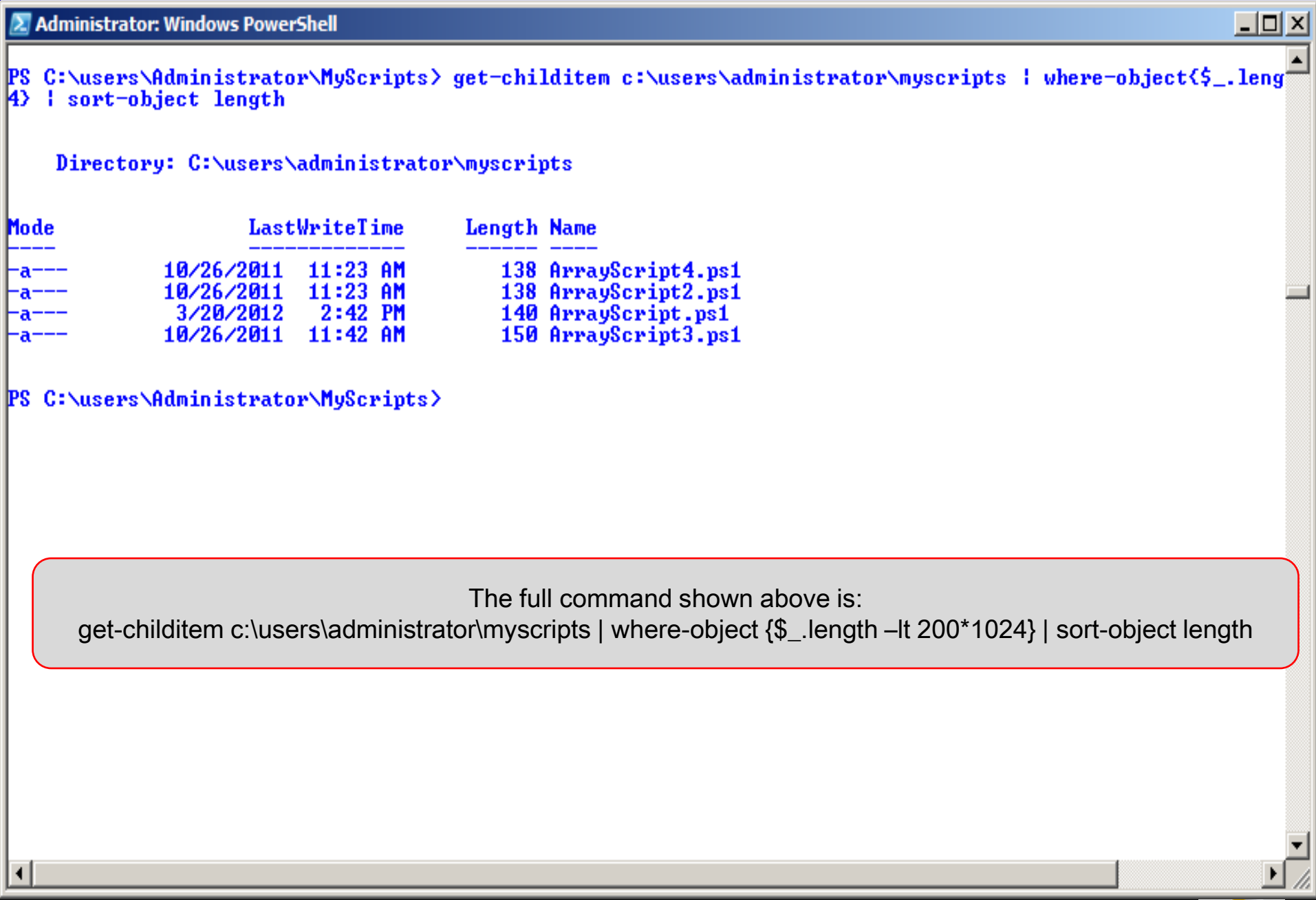



The directory listing is piped to the format-list which formats the output of the directory command into a list.

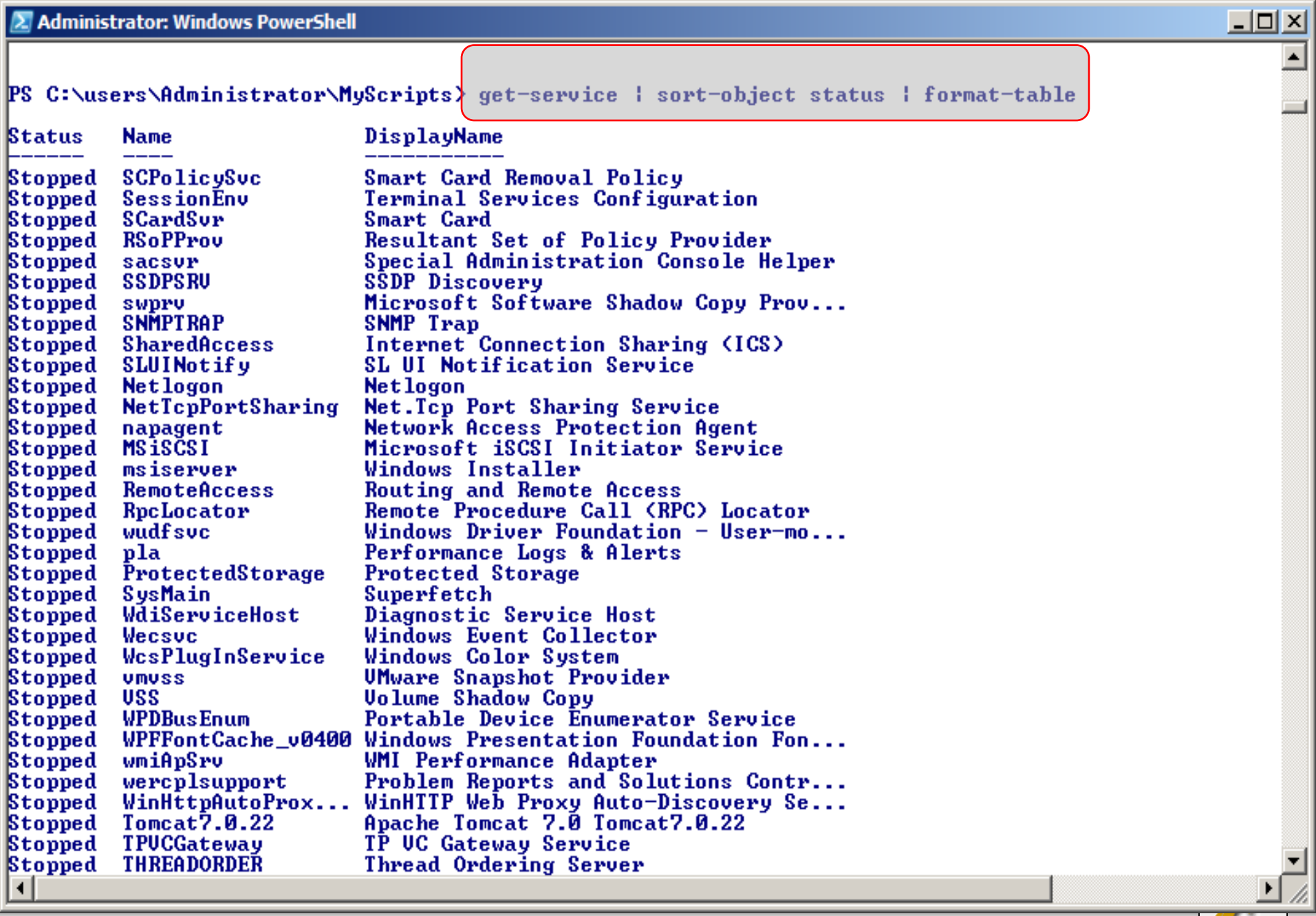
Notice how different the non-piped and the piped outputs look.

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, pipes this set to the `sort-object` cmdlet to perform a sort based on the service's status and finally pipes this result to the `format-table` cmdlet to display the results in a table based format. This example is shown on page 23.



The full command shown above is:
get-childitem c:\users\administrator\myscripts | where-object {$_.length –lt 200*1024} | sort-object length


Administrator: Windows PowerShell
PS C:\users\Administrator\MyScripts> get-service | sort-object status | format-table
Status Name DisplayName
Stopped SCPolicySvc Smart Card Removal Policy
Stopped SessionEnv Terminal Services Configuration
Stopped SCardSvr Smart Card
Stopped RSoPProv Resultant Set of Policy Provider
Stopped sacsvr Special Administration Console Helper
Stopped SSDPSRV SSDP Discovery
Stopped swprv Microsoft Software Shadow Copy Prov...
Stopped SNMPTRAP SNMP Trap
Stopped SharedAccess Internet Connection Sharing (ICS)
Stopped SLUINotify SL UI Notification Service
Stopped Netlogon Netlogon
Stopped NetTcpPortSharing Net.Tcp Port Sharing Service
Stopped napagent Network Access Protection Agent
Stopped MSiSCSI Microsoft iSCSI Initiator Service
Stopped msiserver Windows Installer
Stopped RemoteAccess Routing and Remote Access
Stopped RpcLocator Remote Procedure Call (RPC) Locator
Stopped wudfsvc Windows Driver Foundation - User-mo...
Stopped pla Performance Logs & Alerts
Stopped ProtectedStorage Protected Storage
Stopped SysMain Superfetch
Stopped WdiServiceHost Diagnostic Service Host
Stopped Wecsvc Windows Event Collector
Stopped WcsPlugInService Windows Color System
Stopped vmvss VMware Snapshot Provider
Stopped VSS Volume Shadow Copy
Stopped WPDBusEnum Portable Device Enumerator Service
Stopped WPFFontCache_v0400 Windows Presentation Foundation Fon...
Stopped wmiApSrv WMI Performance Adapter
Stopped wercplsupport Problem Reports and Solutions Contr...
Stopped WinHttpAutoProx... WinHTTP Web Proxy Auto-Discovery Se...
Stopped Tomcat7.0.22 Apache Tomcat 7.0 Tomcat7.0.22
Stopped TPVCGateway TP VC Gateway Service
Stopped THREADORDER Thread Ordering Server

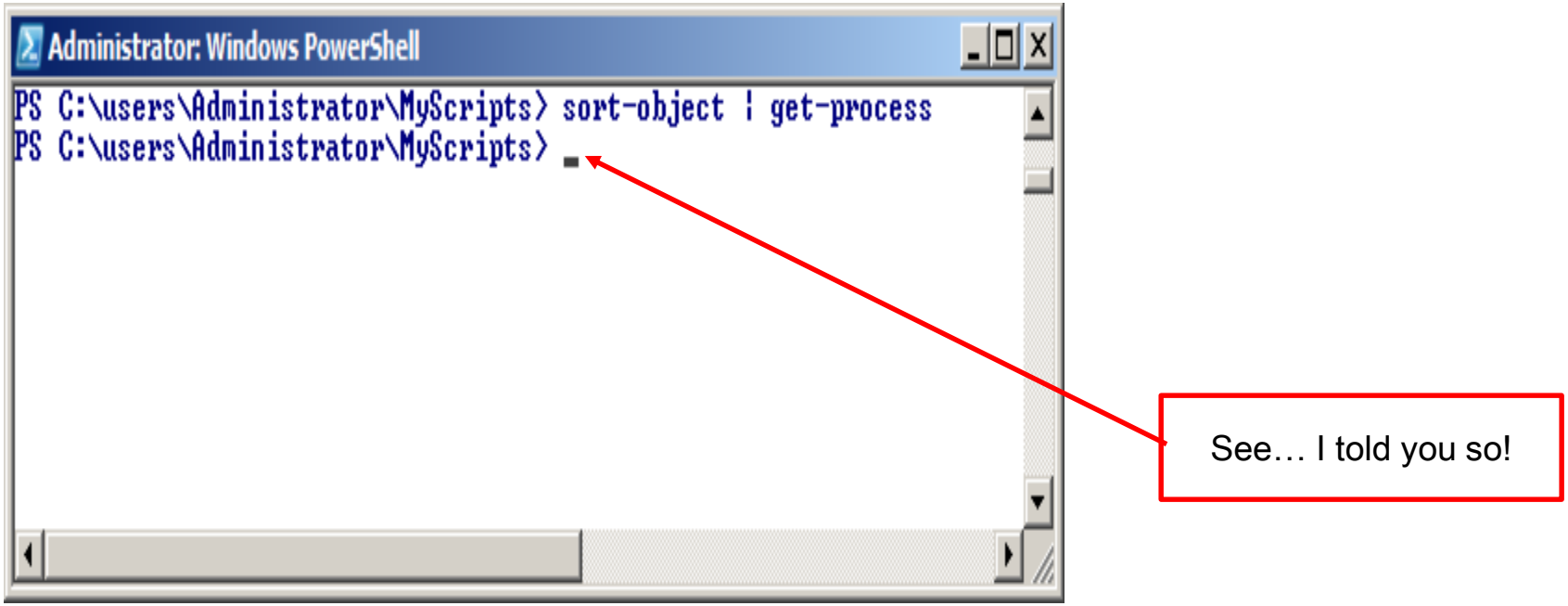
Pipelining In PowerShell

- 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.



Pipelining In PowerShell

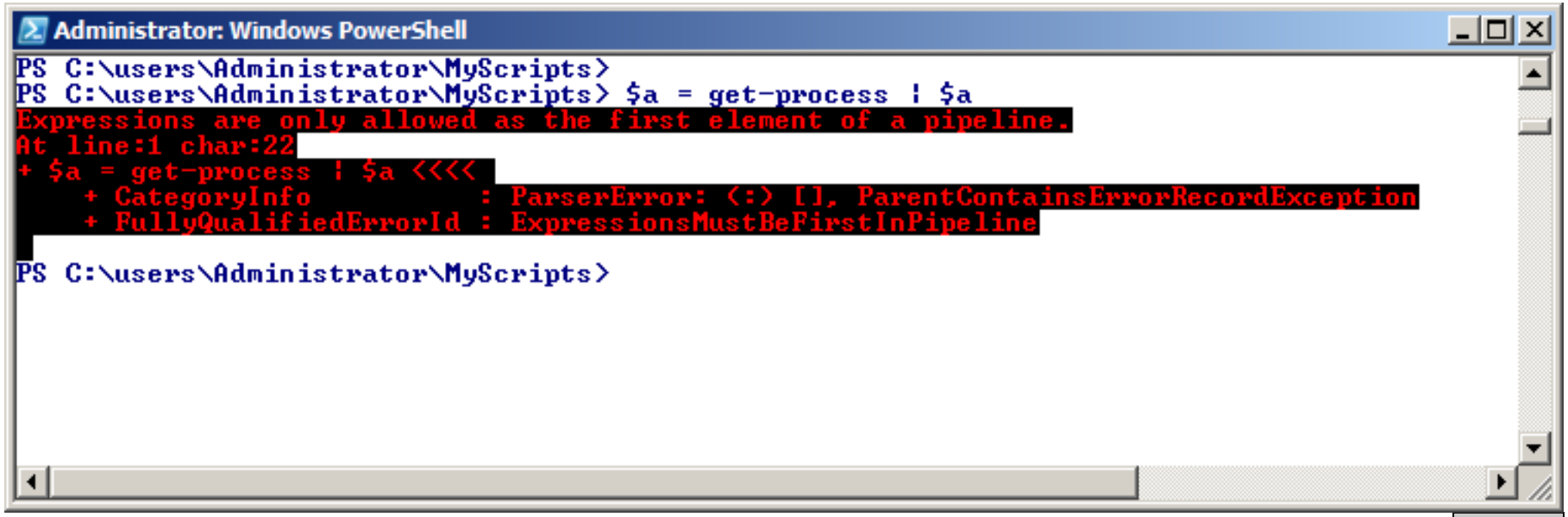
- For the most part, and there are some exceptions to the rule, for pipelines to work correctly, you first acquire something (a collection, an object, whatever) and then hand that data over the pipeline.
  - One exception to the rule would be the following situation where $a represents a variable that 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.

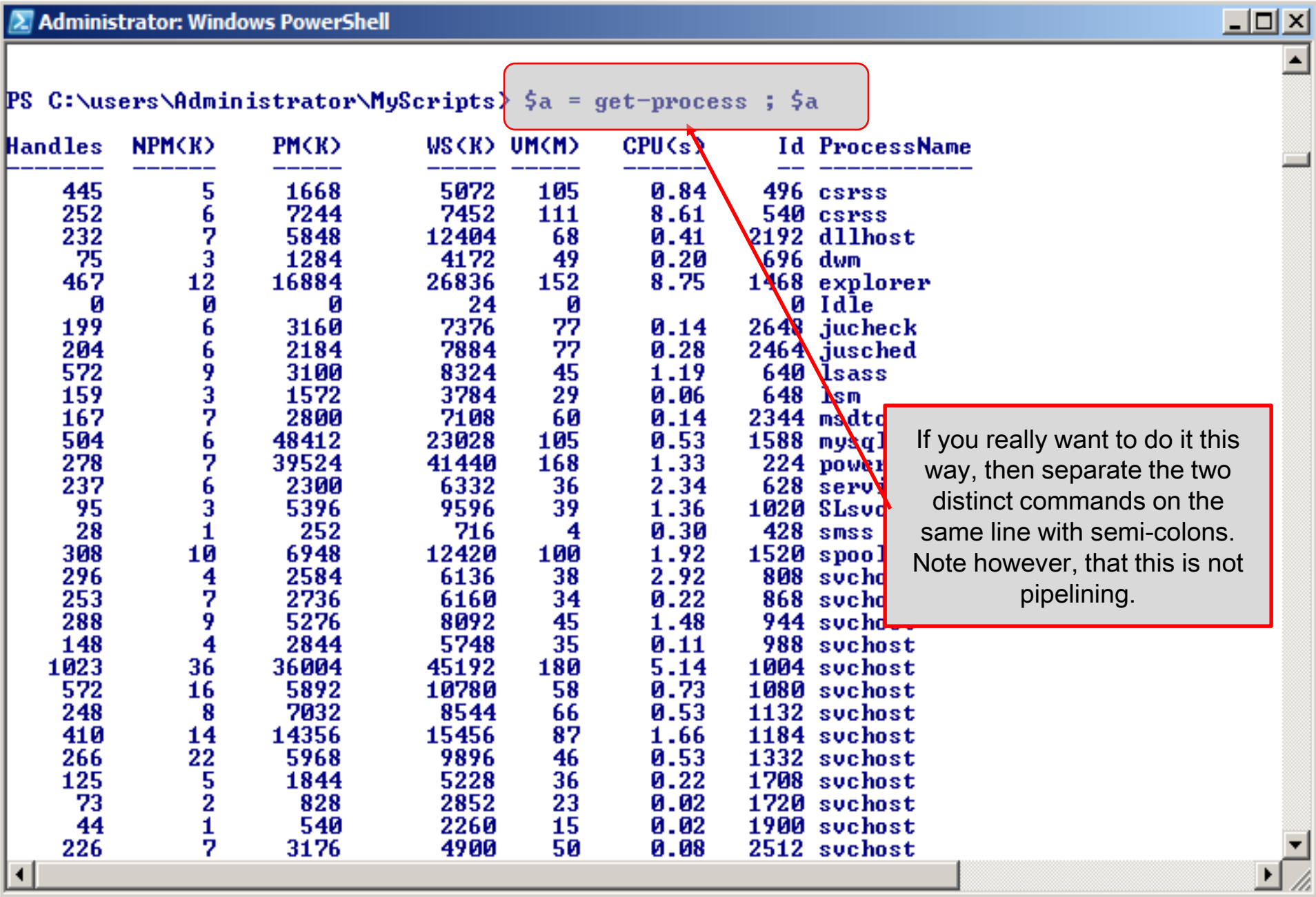


Pipelining In PowerShell

- 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 commands, 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> $a = get-process
PS C:\users\Administrator\MyScripts> $a

Handles  NPM(K)    PM(K)      WS(K) VM(M)   CPU(s)     Id ProcessName
-------  ------    -----      ----- -----   ------     -- -----------
    445       5     1668       5072   105       0.84    496 csrss
    255       6     7244       7452   111       8.20    540 csrss
    232       7     5848      12404    68       0.41   2192 dllhost
     75       3     1284       4172    49       0.20    696 dwm
    472      13    17020      26860   153       8.75   1468 explorer
      0       0        0         24     0                 0 Idle
    199       6     3160       7376    77       0.14   2648 jucheck
    204       6     2184       7884    77       0.28   2464 jusched
    572       9     3100       8324    45       1.19    640 lsass
    159       3     1572       3784    29       0.06    648 lsm
    167       7     2800       7108    60       0.14   2344 msdtc
    504       6    48412      23028   105       0.52   1588 mysqld
    489       7    38764      40428   168       1.25    224 powershell
    237       6     2300       6332    36       2.34    628 services
     95       3     5396       9596    39       1.36   1020 SLsvc
     28       1      252        716     4       0.30    428 smss
    308      10     6948      12416   100       1.92   1520 spoolsv
    296       4     2584       6136    38       2.92    808 svchost
    253       7     2736       6160    34       0.22    868 svchost
    290       9     5276       8092    45       1.47    944 svchost
    148       4     2844       5748    35       0.11    988 svchost
   1023      36    36004      45196   180       5.14   1004 svchost
    573      16     5892      10780    58       0.73   1080 svchost
    248       8     7032       8544    66       0.53   1132 svchost
    413      13    14316      15448    87       1.66   1184 svchost
    268      22     5996       9908    47       0.53   1332 svchost
    125       5     1844       5228    36       0.22   1708 svchost
     73       2      828       2852    23       0.02   1720 svchost
     44       1      540       2260    15       0.02   1900 svchost
    226       7     3176       4900    50       0.08   2512 svchost
```



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
445 5 1668 5072 105 0.84 496 csrss
252 6 7244 7452 111 8.61 540 csrss
232 7 5848 12404 68 0.41 2192 dllhost
75 3 1284 4172 49 0.20 696 dwm
467 12 16884 26836 152 8.75 1468 explorer
0 0 0 24 0 0 Idle
199 6 3160 7376 77 0.14 2648 jucheck
204 6 2184 7884 77 0.28 2464 jusched
572 9 3100 8324 45 1.19 640 lsass
159 3 1572 3784 29 0.06 648 lsm
167 7 2800 7108 60 0.14 2344 msdtc
504 6 48412 23028 105 0.53 1588 mysql
278 7 39524 41440 168 1.33 224 power
237 6 2300 6332 36 2.34 628 servi
95 3 5396 9596 39 1.36 1020 SLsvc
28 1 252 716 4 0.30 428 smss
308 10 6948 12420 100 1.92 1520 spool
296 4 2584 6136 38 2.92 808 svcho
253 7 2736 6160 34 0.22 868 svcho
288 9 5276 8092 45 1.48 944 svcho
148 4 2844 5748 35 0.11 988 svchost
1023 36 36004 45192 180 5.14 1004 svchost
572 16 5892 10780 58 0.73 1080 svchost
248 8 7032 8544 66 0.53 1132 svchost
410 14 14356 15456 87 1.66 1184 svchost
266 22 5968 9896 46 0.53 1332 svchost
125 5 1844 5228 36 0.22 1708 svchost
73 2 828 2852 23 0.02 1720 svchost
44 1 540 2260 15 0.02 1900 svchost
226 7 3176 4900 50 0.08 2512 svchost
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> $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         24     0                0 Idle
    490       0        0       1852     4                4 System
    298       7    37508      39460   168     1.45     224 powershell
     28       1      252        716     4     0.30     428 smss
    445       5     1668       5072   105     0.84     496 csrss
    254       6     7244       7452   111     8.67     540 csrss
    100       4     1148       3932    41     0.20     548 wininit
    118       3     1228       4356    32     0.39     580 winlogon
    237       6     2300       6332    36     2.34     628 services
    570       9     3100       8324    45     1.19     640 lsass
    159       3     1572       3784    29     0.06     648 lsm
     75       3     1284       4172    49     0.20     696 dwm
    296       4     2584       6136    38     2.92     808 svchost
    247       7     2832       7596    73     0.27     816 taskeng
    255       7     2764       6172    35     0.22     868 svchost
    288       9     5276       8092    45     1.48     944 svchost
    148       4     2844       5748    35     0.11     988 svchost
   1025      36    36004      45196   180     5.14    1004 svchost
     95       3     5396       9596    39     1.36    1020 SLsvc
    571      16     5920      10792    58     0.73    1080 svchost
    248       8     7032       8544    66     0.53    1132 svchost
    408      13    14316      15448    87     1.66    1184 svchost
    268      22     5996       9908    47     0.53    1332 svchost
    137       5     1860       5876    53     0.13    1428 taskeng
    470      13    16952      26848   152     8.75    1468 explorer
    308      10     6860      12396   100     1.92    1520 spoolsv
    504       6    48412      23028   105     0.53    1588 mysqld
    125       5     1844       5228    36     0.22    1708 svchost
     73       2      828       2852    23     0.02    1720 svchost
```