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

Page 1



CNT 4603: Scripting – Windows PowerShell – Part 2

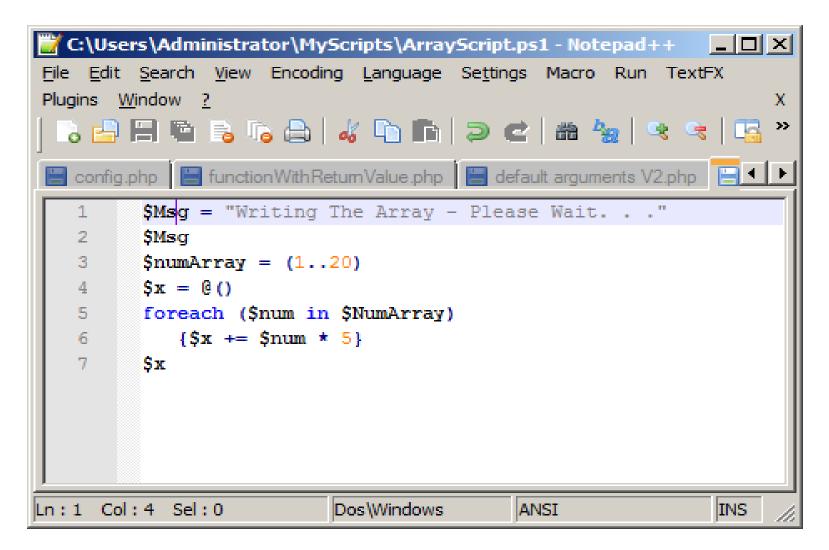
Dr. Mark Llewellyn ©

- 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.

Page 2

• You should see screen as it appears on the next page:

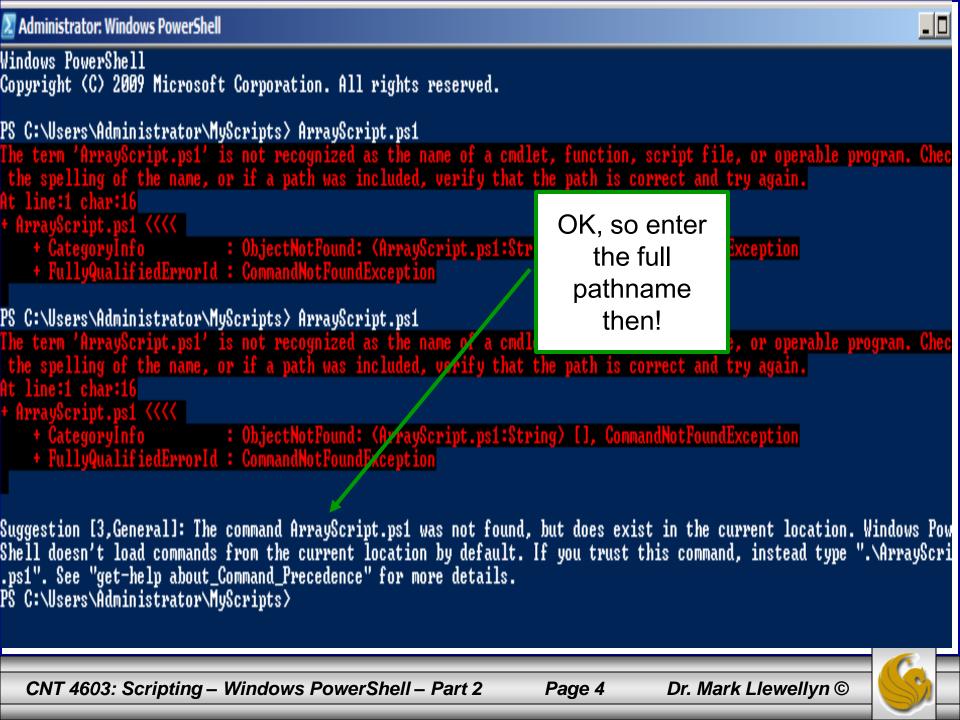




Page 3



CNT 4603: Scripting – Windows PowerShell – Part 2



- 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		_ 0
Windows PowerShell Copyright (C) 2009 Microsoft Corporation, All rights reserved.	PowerShell	
PS C:\Users\Administrator\MyScripts> ArrayScript.ps1 The term 'ArrayScript.ps1' is not recognized as the name of a cmdle the spelling of the name, or if a path was included, verify that t At line:1 char:16 + ArrayScript.ps1 <<<<	t, function, script file, or operable the path is correct and try again.	le program. Chec
<pre>+ 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 cmd1 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</pre>	Now what the &*%\$ is going on? A shell that won't run scripts?) program. Chec
+ FullyQualifiedErrorId : CommandNotFoundException Suggestion [3,General]: The command ArrayScript.ps1 was not found, Shell doesn't load commands from the current location by default. I .ps1". See "get-help about_Command_Preceder.ce" for more details. PS C:\Users\Administrator\MyScripts> c:\Lsers\administrator\myscript File C:\users\administrator\MyScripts> c:\Lsers\administrator\myscript.ps1 cannot be loa this system. Please see "get-help about_signing" for more details. At line:1 char:49 + c:\users\administrator\myscripts\ArrayScript.ps1 <<< * c:\users\administrator\myscripts\ArrayScript.ps1 <	f you trust this command, instead ty ots\ArrayScript.ps1 . <mark>ded because the execution of script</mark> :	ype ".\ArrayScri
+ CategoryInfo : NotSpecified: (:) [], PSSecurityExcep + FullyQualifiedErrorId : RuntimeException	ition .	

PS C:\Users\Administrator\MyScripts>

CNT 4603: Scripting – Windows PowerShell – Part 2

Dr. Mark Llewellyn ©

Page 6

- 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 getexecutionpolicy. This is shown on the next page.

Page 7



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

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



- 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 getexecutionpolicy. This is shown on the next page.

Page 9



Σ 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>



CNT 4603: Scripting – Windows PowerShell – Part 2



_ 8 ×

▼

- 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**.

CNT 4603: Scripting – Windows PowerShell – Part 2

Page 11 Dr. N



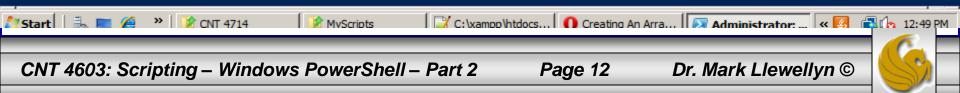
Windows PowerShell Copyright (C) 2009 Microsoft Corporation. All rights reserved.

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> _



- 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.

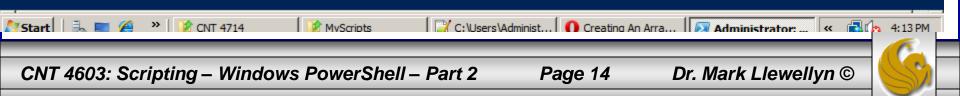
CNT 4603: Scripting – Windows PowerShell – Part 2

Page 13 Dr. Mark Llewellyn ©



PS C:\Users\Administrator\MyScripts> c:\users\administrator\myscripts\ArrayScript.ps1 Writing The Array - Please Wait. . .

- PS C:\Users\Administrator\MyScripts> 🛓

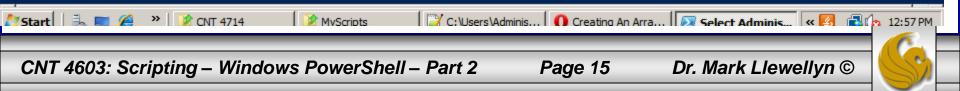


🔀 Select Administrator: Windows PowerShell

- 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. . .
```

- PS C:\Users\Administrator\MyScripts>

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



- 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"



CNT 4603: Scripting – Windows PowerShell – Part 2

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. . . 10 1225050455565778850 The MyScripts folder has now been 95 100 appended to the path PS_C:\Users\Administrator\MyScripts> _ variable. 鸄 Start 🛛 🖶 📰 🔗 🔹 🖹 🚺 CNT 4714 C: Users Adminis... O Creating An Arra... 🔽 Administrator:... « 🛐 🗐 🕼 1:07 PM MyScripts CNT 4603: Scripting – Windows PowerShell – Part 2 Page 18 Dr. Mark Llewellyn ©

- 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.

Page 19



CNT 4603: Scripting – Windows PowerShell – Part 2

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

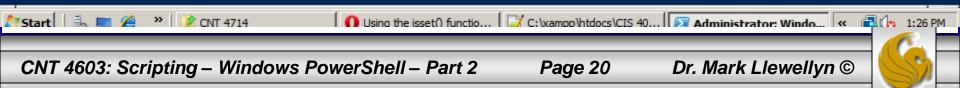
PS C:\Users\Administrator\MyScripts> dir ¦ format-list

Directory: C:\Users\Administrator\MyScripts

Name Length CreationTime LastWriteTime LastAccessTime VersionInfo	 ArrayScript.ps1 136 3/25/2011 4:01:27 3/28/2011 12:51:33 3/25/2011 4:01:27 File: InternalName: OriginalFilename: FileVersion:	2 PM PM C:\Users\Administ	trator\MyScript	s\ArrayScript.ps1
	FileDescription: FileDescription: Product: ProductVersion: Debug: Patched: PreRelease: PrivateBuild: SpecialBuild: Language:	False False False False False		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-nined

PS C:\Users\Administrator\MyScripts>

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



- 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.

Page 21





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	Last	WriteTime	Length	Name	
-a -a -a	3/28/2011 3/30/2011 3/30/2011	12:51 PM 2:18 PM 2:19 PM	177	ArrayScript.ps1 ArrayScript2.ps1 ArrayScript3.ps1	

PS C:\Users\Administrator\MyScripts> 🛓

CNT 4603: Scripting – Windows PowerShell – Part 2

Page 22

Dr. Mark Llewellyn ©

_ 8 ×

Ŧ

Z Administrator: Windows PowerShell									
PS_C:\Users\Administrator\MyScripts>									
PS C:\Users\Administrator\MyScripts} PS C:\Users\Administrator\MyScripts PS C:\Users\Administrator\MyScripts} get-service ! sort-object status ! format-table									
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 11tdsvc Link-Layer Topology Discovery Mapper									
Stopped upnphost UPnP Device Host									
Stopped msiserver Windows Installer									
Stopped TrustedInstaller Windows Modules Installer									
Stopped UIØDetect Interactive Services Detection									
Stopped MSiSCSI Microsoft iSCSI Initiator Service Stopped IPBusEnum PnP-X IP Bus Enumerator									
Stopped USS Volume Shadow Copy Stopped blesson Health Key and Contificate Management									
Stopped hkmsvc Health Key and Certificate Management									
Stopped idsvc									
Stopped vds Virtual Disk Stopped SSDPSRV SSDP Discovery									
Stopped KeyIso CNG Key Isolation Stopped SCardSvr Smart Card									
Stopped Scardsor - 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 Tomcat? 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	•								

CNT 4603: Scripting – Windows PowerShell – Part 2

Page 23



- 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.

Page 24

• What would this command do?

Sort-object | get-process

CNT 4603: Scripting – Windows PowerShell – Part 2



• 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.

Running Running Running Running Running Running Running	lmhosts RemoteRegistry RpcSs ProfSvc RasMan SamSs SENS	TCP/IP NetBIOS Helper Remote Registry Remote Procedure Call (RPC) User Profile Service Remote Access Connection Manager Security Accounts Manager System Event Notification Service	
Running Running Running Running Running Running Running	ShellHWDetection Schedule seclogon mysgl Netman MpsSvc MSDTC netprofm PlugPlay PolicyAgent NlaSvc	Shell Hardware Detection Task Scheduler Secondary Logon mysql Network Connections Windows Firewall Distributed Transaction Coordinator Network List Service Plug and Play IPsec Policy Agent Network Location Awareness Network Store Interface Service	See… I told you so!
	ers\Administrator\ ers\Administrator\	MyScripts> sort-object ¦ get-process MyScripts>	

CNT

- 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.

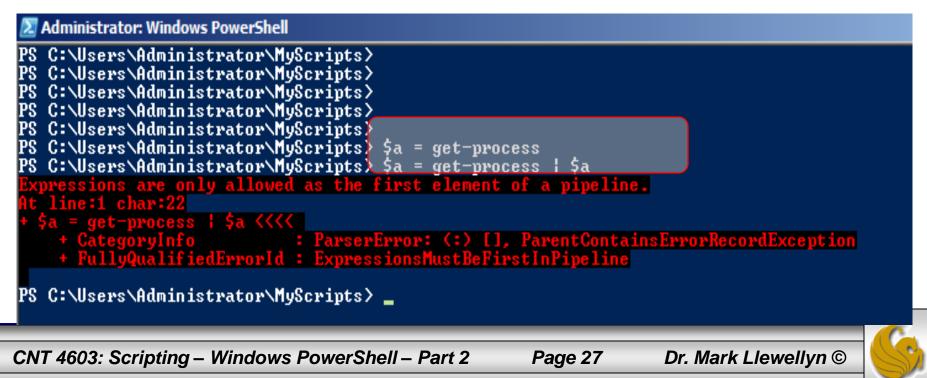




• Suppose you entered a command like this:

\$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.



- 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.

CNT 4603: Scripting – Windows PowerShell – Part 2



🗵 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)	UM(M)	CPU(s)	Id	ProcessName
597	5	1568	1012	99	1.70	444	csrss
225	5	2488	3920	106	17.07	484	CSPSS
73	5 5 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	httpd Idle
202	0 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	2	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	2	2544	2044	28	2.77	800	svchost
293	6 9 3 7 6 9 6 6 4 1 8 4 7 9 4 7 9 4 7 3 13	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	?	3160	168	44	0.06	3464	svchost
514	Ø	Ø	68	4		- 4	System
138	5	1944	2160	47	0.40	1988	taskeng
250	?	2900	2444	72	2.70	3404	taskeng
93	4	3732	364	58	2.63	1444	VMSPVC
52 26	2	1852	1024	52 13	0.77	3912	VMUSPVC
26	1	364	232	13	0.86	1744	vpcmap
100	4	1080	16	35	0.19	492	wininit
123	52127057421433	1260	1748	26	0.41	520	winlogon
83	3	2436	4888	55	0.04	4052	wuauclt
•							

CNT 4603: Scripting – Windows PowerShell – Part 2

Page 29

Dr. Mark Llewellyn ©

_ 8 ×

▼

Administrator: Windows PowerShell

10 01 00		IISCI GOOI W	19001 1 pc3.	/ 7u	acc broce	55) - 1	* •
Handles	NPMCK>	PM(K)	WS CK2	VM(M)	CPU(s)	I d	ProcessName
591	5	1568	1012	99	1.70	444	csrss
224	53	2488	3920	106	17.19	484	csrss
73	3	1084	208	42	0.09	3940	dwm 🔪
585	14	20904	14332	170	19.45	2656	explorar
282	26	45700	4212	129	2.22	316	httpd
128	11	41388	20	91	1.87	1484	httpd
Ø	Ø	Ø	12	0		Ø	Idle 🔪
202	6	3492	1124	64	0.36	4076	jucheck
206	6 9 3 7	2496	1056	68	1.16 7.23	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	0.09 6.30	1620	mysqld
146	9 6	18604	18084	121	128.83	3136	opera
299 237	6	33272	33424	158	1.08	2840	
237	6	2008	2088	26	3.34	568	
128	6 4 1 8 4 7 9 4 37	7428	9784	37	7.45	948	SLsvc
28	1	244	32	- 4	0.06 59.06	372	SMSS
280	8	5576	1736	76	59.06	1420	spoolsv
291	4	2488	2572	31	11.92		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 8.13	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	
64	2	1240	16	23	0.09	2688	svchost
228	2	3160	168	44	0.06		svchost
514	Ø	0	68	. 4		4	System
138	5	1944	2160	47	0.40	1988	taskeng
248	?	2880	2432	72	2.70	3404	taskeng
93	5212705742143	3732	364	58	2.63	1444	VMSPVC
52	2	1852	1024	52	0.77	3912	VMUSPVC
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	wuaucľt

PS_C:\Users\Administrator\MyScripts>

•

CNT 4603: Scripting – Windows PowerShell – Part 2

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.

_ 8 ×

Ŧ

Dr. Mark Llewellyn ©

Page 30

- 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.

Page 31



CNT 4603: Scripting – Windows PowerShell – Part 2

🗵 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)	UM(M)	CPU(s)	I d	ProcessName
Ø	0	 Ø	12	Ø		0	Idle
514	Ø	Ø	68	4		4	System
282	26	45700	4212	129	2.22	316	httpd
28 591		244	32	4	0.06	372	SMSS
591	5	1568	1012	99	1.70	444	CSPSS
224	5	2488	3920	106	17.40	484	CSPSS
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	1554369347	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	4 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	
73	2	792	24	21	0.00	1656	svchost
26	6 5 2 1 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	5 14	20904	14332	170	19.45	2656	
64	27	1240	16	23	0.09	2688	svchost
162	?	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	?	2880	2432	72	2.70	3404	taskeng
228	?	3160	168	44	0.06	3464	svchost
206	6 9 7 6 2 3 3	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
4							

CNT 4603: Scripting – Windows PowerShell – Part 2

Page 32

Dr. Mark Llewellyn ©

_ 8 ×

▼