Paper CC-18

Windows® PowerShell™ Commands and Scripts for SAS® Programmers

Adeline J. Wilcox, Veterans Health Administration, Office of Informatics and Analytics

Abstract

With Windows PowerShell, power users can gain at least some of the productivity at the hands of UNIX/Linux/Mac OS X command-line programmers. While PowerShell is designed for system administrators, SAS®programmers, particularly those with command line experience, can work more productively by learning even a few PowerShell commands. I will give examples of both commands and scripts. Among commands that can be quickly executed without reaching for the mouse, I will show how to move groups of similarly named files from one folder or server to another and use the `Select-String` cmdlet, similar to the UNIX grep command, to find all SAS programs within a folder that contain a SAS data set name of interest. I will describe the PowerShell script I wrote to create a file containing names of all specifications documents stored within a directory tree. Another script will show how a SAS program may be executed iteratively by passing environment variable values assigned within the PowerShell script to the SAS program. Object-oriented, PowerShell is hard to learn. But if you learn even a couple of commands, you'll be a more powerful computer user.

Introduction

PowerShell is a command-line scripting environment for Microsoft Windows. Based on the IEEE POSIX 1003.2 standard, PowerShell bears some resemblance to UNIX shells (Payette). The Microsoft .NET Framework interfaces PowerShell with the Windows operating system. Cmdlets, with Verb-Noun names like `Copy-Item`, access the .NET Framework. Unlike text-based UNIX shells, PowerShell uses an object pipeline (Ford).

Starting PowerShell

If you don't find a PowerShell icon waiting for you, click Start > All Programs > Accessories > Windows PowerShell > Windows PowerShell.

Moving, Copying and Deleting Many Similarly Named Files

This command moves all of the files named `fy030010.zip`, `fy040010.zip`, `fy050010.zip`, `fy060010.zip`, and `fy060010.zip` to directory named `overthar`. Using the regular expression `[3-7]` powerfully moves many files with one commmand.

```
mv fy0[3-7]0010.zip F:\overthar\.
```

Substituting `cp` for `mv` copies the files instead of moving them. Similarly, substituting `rm` deletes all the files. In this example, I've used the PowerShell cmdlet alias `mv` familiar to UNIX users for the cmdlet named `Move-Item`.

Finding Code of Interest in SAS Progams

Do you need to find your old SAS program in which you used perl regular expressions? The cmdlet

```
Select-String prxmatch *.sas
```

lists all lines containing the token `prxmatch` in every SAS program with the file extension `.sas`.
Likewise, if you need to find all SAS programs reading or writing the SAS data set named `vastdata`, the cmdlet

```
Select-String -list vastdata *.sas
```

will list the first line containing the string `vastdata` in each SAS program in the current working directory.

PowerShell Execution Policy

By default, PowerShell scripts are disabled. The *Windows PowerShell Cookbook*, (Holmes) explains how to configure PowerShell. You may find you will need to discuss PowerShell execution policy with your System Administrator or enterprise IT executive.

Script for Listing PDFs Containing Programming Specs

Do you have programming specifications stored in many Word or PDF files where each file is named for the specification contained within? Would you like to create a list of these specifications? My PowerShell script listed here changes to a folder below which files are stored in several subfolders.

```
cd "F:\10Q\XXX\YYY\Dataset Questions and Scoring FY00 to Present\FY11\2Q11 Scoring"
get-childitem -recurse | out-string > F:\mine\fy11q2all.txt
cd F:\mine\
select-string -SimpleMatch pdf F:\mine\fy11q2all.txt > F:\mine\justpdf.txt
```

With the `-recurse` parameter, the `Get-Childitem` cmdlet recursively lists all files below the folder and saves the list to a text file named `fy11q2all.txt`. After changing my working folder away from specs storage to one for my personal use with the command `cd F:\mine\`, I use the grep-like cmdlet named `Select-String` to filter out all specifications files with the file extension PDF. Next, I write the `Select-String` output to a file named `F:\mine\justpdf.txt`. Finally, to produce a clean list of specification file names, I use my favorite text editor to delete blank records and any extraneous data from the file named `F:\mine\justpdf.txt`.

Script for Iteratively Executing a SAS Program

I execute my PowerShell script named `example.ps1` by typing the command `.\example.ps1` at the PowerShell prompt. Interestingly, the command `./example.ps1` also works.

On the next page you will find my PowerShell script named `.\example.ps1` listed.

The `Get-Content` cmdlet gets the identifiers listed in the file named `statnids.csv` and assigns them to the environment variable named `$collection`. If they exist, old SAS log files from previous runs are deleted. This makes debugging simpler. `foreach` loops through the identifiers, executing the SAS program named `asasprog.sas` in batch mode, once for each identifier.

Inside the SAS program, the value of the PowerShell environment variable named `$env:station` can be assigned to a SAS macro variable using a SAS statement like the one below.

```
%let sta3=%sysget(station);
```

The `Wait-Process` cmdlet prevents PowerShell from executing the next command before the SAS program completes execution.

Exit status is checked by first assigning the value of the PowerShell variable `$lastExitCode` to the variable named `$sas_exit`, then writing a message to the screen based on the value of `$sas_exit`. The value of `$lastExitCode` assigned to the SAS executable must be captured promptly before another command changes the value of `$lastExitCode`. When SAS exits with a nonzero exit status, `Select-String` cmdlets can also send SAS Warning and Error messages to the screen.

Near the end of the script, I rename the SAS log using the value of the environment variable named `$env:station`. This prevents overwriting the SAS log during the next iteration of the `foreach` loop.

```
#example.ps1
#this script can hang. If it seems stuck, hit Enter key
$collection=get-content ..\statnids.csv
foreach($i in $collection)
{
  $env:station=$i
  Write-host station is $env:station
#next IF statement seemed to prevent script from sometimes abending mysteriously
  if($env:station -eq 613)
  {
  Write-host "sleeping"
  Start-Sleep -s 120
  }
  else
  {

  }
 if ( Test-Path Z:\mine\asasprog$env:station.log )
 {
   rm Z:\mine\asasprog$env:station.log
   Write-Host asasprog$env:station.log removed
 }
 C:\sas.exe asasprog -noterminal -config X:\wilcox\sasv9.cfg | Wait-Process
 $sas_exit=$lastExitCode
 Write-Host lastExitCode value is $lastExitCode
  if ($sas_exit -eq 0)
  {
    $trouble='none'
    Write-Host $trouble
  }
  elseif ($sas_exit -eq 1)
  {
    $trouble='exit code 1'
    Write-Host $trouble
    select-string WARNING: asasprog.log
  }
  elseif ($sas_exit -eq 2)
  {
    $trouble='exit code 2'
    Write-Host $trouble
    select-string ERROR: asasprog.log
  }
  elseif ($sas_exit -gt 2)
  {
    $trouble='gt 2'
    Write-Host $trouble
  }
 mv asasprog.log asasprog$env:station.log
}
$today=get-date
write-host $today
```

Last Words

Windows PowerShell Users Groups have formed in some US cities. We need a SAS Community of PowerShell users. SAS Institute should publish a PowerShell book for SAS programmers.

Works Cited

Ford, Jerry Lee. *Microsoft Windows PowerShell 2.0 Programming.* 2nd ed. Boston.Course Technology, 2009. Print.

Holmes, Lee. *Windows PowerShell Cookbook.* Sebastopol. O'Reilly Media, 2010. Print.

Payette, Bruce *Windows PowerShell in Action.* Shelter Island, NY. Manning. 2011. Print.

Credit Notices

Disclaimer

Contact Information

Your comments and questions are valued and encouraged.

Contact the author: Adeline J. Wilcox
Department of Veterans Affairs
Washington, DC
202-266-4542
Adeline dot Wilcox at va dot gov