Advanced Scripting   
Error Handling

Last Updated: 9/28/2021 8:22 AM Version 1  
Document Prepared for: CIT361 Student

# Name Zach Lucas ID 895649438

# Instructions

Save a copy of this document. Answer all questions directly in this document. You will save and upload this completed document as your homework submission. Nonterminating error are errors that will not cause the script to quit executing. Nonterminating errors can be handled with the built-in common parameters ErrorAction and ErrorVariable. By default, the errors are output to the error stream and added to the $error collection.

# Overview

Understanding Error handling is the key to solid code. You will explore nonterminating errors in this exercise.

# Requirements

PowerShell

# Setup

* Create some files to work with. Enter the command:  
  1..10|ForEach-Object{New-Item -Type File -Name "File$($\_\*2).demo"}  
  This will create the files: File2.demo, File4.demo, File6.demo, File8.demo, File10.demo, File12.demo, File14.demo, File16.demo, File18.demo, File20.demo in the current directory.
* Verify with the command:  
  dir \*.demo

# Task 1—$Error

Explore the nonterminating errors and the $Error collection. Whenever an error occurs in PowerShell PowerShell adds the error to the $Error collection. $Error has a max size of $MaximumErrorCount.

## Steps

1. Start a new PowerShell session.
2. The count property holds the number of errors in the $Error collection:  
   $Error.Count
   1. How many errors are in $Error? 3
3. Delete the file file2.demo  
   Remove-Item file2.demo
   1. What is the count of the $error collection? 3
4. Try to delete the file again, this should cause an error.  
   Remove-Item file2.demo
   1. What is the count of the $error collection? 4
5. Delete the file file3.demo  
   Remove-Item file3.demo
   1. What is the count of the $error collection? 5
6. View the $error collection  
   $Error
   1. Which error is listed first? That file3 doesn’t exist and therefore cannot be removed
7. As you can see the latest error is the first one in the list. You can access the most recent error using the 0 index, Enter:  
   $Error[0]
   1. Which error is displayed? The latest error.

# Task 2—ErrorAction and Error Variable

The common parameters of ErrorAction and ErrorVariable can be used to control PowerShells nonterminating error behavior.

## Steps

1. Use ErrorAction to change the behavior of errors for a specific cmdlet. Enter the following command (remember there is no file5.demo:  
   Remove-Item file5.demo -ErrorAction SilentlyContinue
   1. What was the result? Click or tap here to enter text.
   2. Was the error placed in the $Error collection? Click or tap here to enter text.
2. Use an error variable to capture the error from a specific command.   
   Remove-Item file7.demo -ErrorAction SilentlyContinue -ErrorVariable DelError
   1. What was returned? Nothing
   2. View the contents of $DelError  
      $DelError
      1. What does it contain? An error that file5.demo doesn’t exist
   3. Does $Error contain the error also? It does
3. Now delete file4.demo this should be successful.  
   del file4.demo -ErrorAction SilentlyContinue -ErrorVariable DelError
   1. What does $DelError contain? Nothing
   2. What is the count of $DelError? 0
   3. What does that tell you about the ErrorVariable when there are no errors? That it follows the most recent command, if it is good it will have nothing, if it has an error, it will contain was would be displayed.
4. The $ErrorActionPreference variable contains the default value for ErrorAction. If you don’t specify an ErrorAction for a cmdlet the value in $ ErrorActionPreference will be used.
   1. What is the value of $ ErrorActionPreference? “Continue”

# Task 3—The Error Object

When an error is encountered the text of the error is written to the error stream, but an error object is stored in the $Error collection.

## Steps

1. Exploring the Error Object
   1. Get the error object type.   
      $Error[0].gettype()
   2. What object type is stored in the $Errors collection?
   3. View the members of the object:  
      $error[0]|Get-Member
   4. For details see <https://docs.microsoft.com/en-us/powershell/scripting/developer/cmdlet/interpreting-errorrecord-objects>

# Task 4—Multiple Errors

Multiple non-terminating errors may occur during the execution of a cmdlet. Let’s see what happens.

## Steps

1. Write a command that deletes the files file10.demo – file16.demo using a foreach loop.
   1. What is the count of $Error? ErrorRecord
   2. Enter the command  
      10..16 |ForEach-Object {del "File$\_.demo" -ErrorAction Continue -ErrorVariable delerror}
   3. Did you see more than one error message? I saw 3 errors
   4. What is the count of $Error? 9
   5. How many new errors were added? 3
   6. What is in $DelError? Nothing
   7. Explain? Because we said to delete all the files within the range of 10..16, We originally created files that were even, it threw errors for file11, 13, and 15 but not 16 and $DelError keeps the most recent execution. Deleteing file 16 was successful and $DelError has nothing in it.
2. Explore multiple errors in a single cmdlet. To get ready for this you need to mark a couple of our remaining files as read only. To do that you will set the itemproperty. You should have 4 demo files left. Verify with  
    dir \*.demo
   1. You should see something similar to this.  
      Mode LastWriteTime Length Name  
      ---- ------------- ------ ----  
      -a---- 5/11/2020 10:39 AM 0 file18.demo  
      -a---- 5/11/2020 10:39 AM 0 file20.demo  
      -a---- 5/11/2020 10:16 AM 0 File6.demo  
      -a---- 5/11/2020 10:16 AM 0 File8.demo
   2. Mark files 6 and 8 as readonly:  
      Set-ItemProperty -Path file6.demo -Name IsReadOnly -Value $true  
      Set-ItemProperty -Path file8.demo -Name IsReadOnly -Value $true
   3. Verify with dir \*.demo you results should look like this:

Mode LastWriteTime Length Name  
---- ------------- ------ ----  
-a---- 5/11/2020 10:39 AM 0 file18.demo  
-a---- 5/11/2020 10:39 AM 0 file20.demo  
-ar--- 5/11/2020 10:16 AM 0 File6.demo  
-ar--- 5/11/2020 10:16 AM 0 File8.demo

* 1. Now delete all the demo files  
     del \*.demo -ErrorVariable delerror -Verbose
  2. How many items were added to $Error? Nothing was added
  3. What is in $DelError? Nothing

# Wrap-up

1. Get rid of the remaining demo files. You can delete files marked as readonly with the -force option. Enter:  
   del \*.demo -force

# Deliverable

Upload this document with completed answers to i-learn.