# PowerShell – Splitting large log files

Have you ever found an issue with one of your systems and gone digging for the log file, only to find out that the file is too big to open for reading? There are a number of ways you can use in the *nix world, but for the Windows folks I've created a simple script to let you split the larger file into smaller chunks for easier reading and manipulation using PowerShell.

These are your assumptions for this script to work for you:

- Must be run interactively
- You know the full path and filename of the source file (use double quotation marks to surround the file/path if there are spaces)
- You know the path to the destination (also, use double quotation marks to surround the file/path if there are spaces)
- You have a decent amount of memory (the process loads the entire file…yes…the entire file into RAM while parsing the contents
- You have some time to let it bake in the background. It's better to wait than to not be able to read the file right?
- You are perfect. There is no error handling (yet). If you make a mistake it will end the way that this gentleman's IT experience did:

> 
>
> What are you doing Dave?

To start we set some initial counters. I'm going old school on this to generate the file names for the output chunk files.

```
# Set the baseline counters
#
# Set the line counter to 0
$linecount = 0
# Set the file counter to 1. This is used for the naming of the log files
$filenumber = 1
```

Now comes the interactive part where we prompt the user (that's you) for the path and filename for the source file. I'm working on making this a GUI file picker, but for now let's just stick with what works. I figure that if you are able to parse a file for content, then I'm assuming you know how to type a file path correctly.

```
# Prompt user for the path
$sourcefilename = Read-Host "What is the full path and name of the log file to split? (e.g. D:mylogfilesmylog.txt)"

# Prompt user for the destination folder to create the chunk files
$destinationfolderpath = Read-Host "What is the path where you want to extract the content? (e.g. d:yourpath)"
```

Because I'm a nice guy, I like to let the user (still you) know that the next process will take a little

while.

```
Write-Host “Please wait while the line count is calculated. This may take a while. No really, it could take a long time.”
```

This is where the memory intensive part begins. I’m not an expert programmer, and you may have a better way to do this (feel free to add comments if you have any suggestions). I first read the file to capture a line count so that you can decide how many chunks you want to create. I’ve also assumed you don’t mind a little math.

```
# Find the current line count to present to the user before asking the new line count for chunk files
Get-Content $sourcefilename | Measure-Object | ForEach-Object { $sourcelinecount = $_.Count }

#Tell the user how large the current file is
Write-Host “Your current file size is $sourcelinecount lines long”
```

Now that you know the size of the original file, the next step is to ask how big the destination files need to be. This will request a number, but as a string, which is converted into an integer assigning a variable and using [int]$destinationfilesize to convert the string.

```
# Prompt user for the size of the new chunk files
$destinationfilesize = Read-Host “How many lines will be in each new split file?”

# the new size is a string, so we convert to integer and up
# Set the upper boundary (maximum line count to write to each file)
$maxsize = [int]$destinationfilesize
```

Let’s tell the user what our results are from the data collection up to this point and then we can begin the process.

```
Write-Host File is $sourcefilename – destination is $destinationfolderpath – new file line count will be $destinationfilesize
```

Here we go! This goes back to programming 101 by using a simple counter and a loop until we reach the threshold of that counter. The output files will be named **splitlog**(some number).**txt** and will each contain the number of lines of text that we’ve told the script from above.

Final warning! This loads the file into memory which will be pretty resource intensive but the end result is a happy, more easily managed set of files.

```
# The process reads each line of the source file, writes it to the target log file and increments the line counter. When it reaches 100000 (approximately 50 MB of text data)
$content = get-content $sourcefilename | % {
 Add-Content $destinationfolderpathsplitlog$filenumber.txt “$_”
  $linecount ++
  If ($linecount -eq $maxsize) {
    $filenumber++
    $linecount = 0
  }
}
```

Because we've eaten up lots of memory and processor we will need to free up some of the "garbage" we've left behind. It's effectively a PowerShell poop-n-scoop.

```
# Clean up after your pet
[gc]::collect()
[gc]::WaitForPendingFinalizers()
```

And there you have it! Hopefully you find it useful.

Here is the full script – http://gallery.technet.microsoft.com/PowerShell-Split-large-log-6f2c4da0

That's much better