

Find and remove duplicate files with PowerShell

<https://4sysops.com/archives/find-and-remove-duplicate-files-with-powershell/>

You can use PowerShell to find and remove duplicate files that are only wasting valuable storage. Once you identify duplicate files, you can move them to another location, or you might even want to permanently remove all duplicates.

Contents

1. [Data loss warning](#)
2. [Get-FileHash cmdlet](#)
3. [Find duplicate files based on the hash](#)
4. [Find duplicate files based on length and hash](#)
5. [Removing duplicate files](#)

Admins who need to manage file servers and storage understand the hassle that duplicate files may cause. When storage disks start running out of space, admins tend to look for various tools to find and delete duplicate files that unnecessarily waste storage. However, PowerShell provides all you need to deal with duplicate files.

Data loss warning

The procedure described in this post involves the removal of files, which could cause accidental data loss. Therefore, it is highly recommended to create a good backup of the source directory before proceeding.

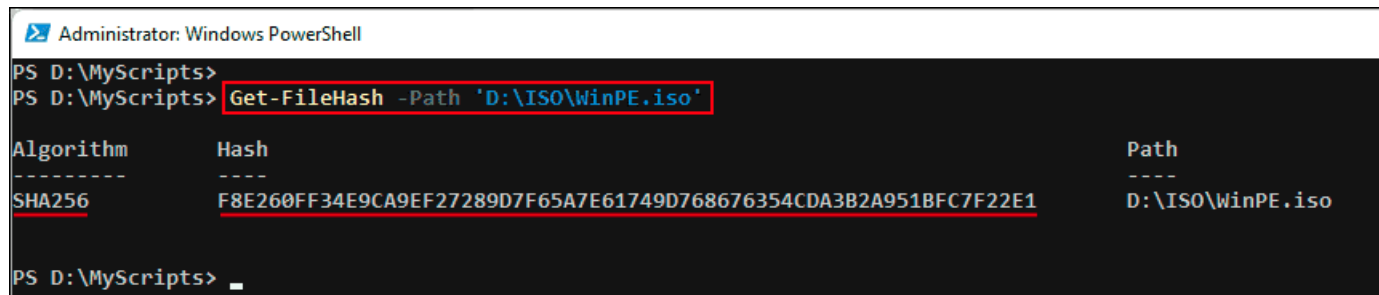
Get-FileHash cmdlet

PowerShell offers the *Get-FileHash* cmdlet to compute the hash (or checksum) of one or more files. This hash can be used to uniquely identify a file. In this post, we will use the hash value to identify duplicate files. The syntax of the command is as follows:

```
Get-FileHash -Path [file_path] -Algorithm [hashing_algorithm]
```

To calculate the hash of a single file, you can run the command shown below:

```
Get-FileHash -Path 'D:\ISO\WinPE.iso' -Algorithm SHA512
```



```
Administrator: Windows PowerShell
PS D:\MyScripts>
PS D:\MyScripts> Get-FileHash -Path 'D:\ISO\WinPE.iso'

Algorithm      Hash                                                                 Path
-----
SHA256         F8E260FF34E9CA9EF27289D7F65A7E61749D768676354CDA3B2A951BFC7F22E1  D:\ISO\WinPE.iso

PS D:\MyScripts> _
```

Calculate the hash or checksum of a file using PowerShell

When no hashing algorithm is specified, SHA256 is used by default, which is sufficient in most cases. The hash is used to verify file integrity, particularly for files downloaded from the Internet.

The good thing about *Get-FileHash* is that it accepts the input from the pipeline, which allows us to use it with the *Get-ChildItem* cmdlet to compute the hash of each item returned by *Get-ChildItem*.

Find duplicate files based on the hash

The following command gets all the files in a folder recursively, passes them to *Get-FileHash* to calculate their hash, and finally groups them based on matching hash values.

```
$srcDir = "D:\MyScripts"
Get-ChildItem $srcDir -File -Recurse `
| Get-FileHash | Group -Property Hash `
| where {$_.Count -gt 1} | foreach { $_.Group | select Path, Hash}
```

```

Administrator: Windows PowerShell
PS D:\MyScripts>
PS D:\MyScripts> $srcDir = "D:\MyScripts"
PS D:\MyScripts>
PS D:\MyScripts> Get-ChildItem $srcDir -File -Recurse `
>> | Get-FileHash | Group -Property Hash `
>> | where { $_.Count -gt 1 } | foreach { $_.Group | select Path, Hash }

Path                                     Hash
----                                     -
D:\MyScripts\Convert-JPG2PDF.ps1        E6D4C054EB8F859992B59C78D8553C4811AE2D3AEFC835D8E24C059143D1C303
D:\MyScripts\Convert-JPG2PDF.psm1      E6D4C054EB8F859992B59C78D8553C4811AE2D3AEFC835D8E24C059143D1C303
D:\MyScripts\New-PsPing.ps1            EA85FDC211EB2209FECE67673D53E945FD9200BCF16151D281255DB1C7FA9197
D:\MyScripts\New-PsPing.psm1          EA85FDC211EB2209FECE67673D53E945FD9200BCF16151D281255DB1C7FA9197
D:\MyScripts\Unlock-BLDrive.ps1       ABB0D2A6B9AF44EB80B6FAE5AC3405E15CD412BE9BDAE5853056885894EFA787
D:\MyScripts\Unlock-BLDrives.ps1     ABB0D2A6B9AF44EB80B6FAE5AC3405E15CD412BE9BDAE5853056885894EFA787

PS D:\MyScripts>

```

The output is grouped by the Hash property returned by 'Get-FileHash' cmdlet, indicating there are three duplicate files in the specified source directory

Find duplicate files with PowerShell

The command has one serious flaw, though. It works well for a few small files, but you will run into trouble if the source directory contains many large files. The hash computation is a resource-intensive operation, and the aforementioned command computes it for each file, regardless of its size. The command will therefore take ages to find duplicates when used against a directory with a large number of large files. In the next section, we further optimize this command.

Find duplicate files based on length and hash

A necessary condition for duplicate files is that their size must match, which means that files where the size does not match cannot be duplicates. You probably know the *Length* property of the *Get-ChildItem* PowerShell cmdlet. Because the *Length* value is retrieved from the directory, no computation is required. The trick is to only compute the hashes of files having the same length because we already know that files with different lengths can't be duplicates. In this way, the overall time of the command is significantly reduced. This is accomplished by the PowerShell commands below:

```



```

```

Administrator: Windows PowerShell
PS D:\MyScripts>
PS D:\MyScripts> $srcDir = "D:\ISO Files"
PS D:\MyScripts>
PS D:\MyScripts> Get-ChildItem -Path $srcDir -File -Recurse | Group -Property Length `
>> | where { $_.Count -gt 1 } | select -ExpandProperty Group | Get-FileHash `
>> | Group -Property Hash | where { $_.count -gt 1 } | foreach { $_.Group | select Path, Hash }

Path                                     Hash
----                                     -
D:\ISO Files\debian-amd64-netinst-elastix.iso DD39B430E678CB366C19055E090E04E3A00C55E1D530B019E0971D6AD6F9B78A
D:\ISO Files\debian-amd64.iso              DD39B430E678CB366C19055E090E04E3A00C55E1D530B019E0971D6AD6F9B78A
D:\ISO Files\PBX-in-a-Flash - 20220510.iso 752F53DAA0CECC242B26C4568189AC91ED37237C94DC80FA5CCE9907F345ACB6
D:\ISO Files\PBX-in-a-Flash.iso           752F53DAA0CECC242B26C4568189AC91ED37237C94DC80FA5CCE9907F345ACB6
D:\ISO Files\VMware-ESXi-6.5.0-20190702001-RTL8168.iso 9FA26C0513ADCABFBCD7431F55EA7B9247587652FD1F9D30DAFCDA3409A8757F
D:\ISO Files\VMware-ESXi-6.5.0.iso        9FA26C0513ADCABFBCD7431F55EA7B9247587652FD1F9D30DAFCDA3409A8757F
D:\ISO Files\Windows PE.iso              F8E260FF34E9CA9EF27289D7F65A7E61749D768676354CDA3B2A951BFC7F22E1
D:\ISO Files\WinPE.iso                   F8E260FF34E9CA9EF27289D7F65A7E61749D768676354CDA3B2A951BFC7F22E1

```

Fast way to find duplicate files with PowerShell

With the help of the *Group* cmdlet, we first group the files that match in size, and then pass those files to *Get-FileHash* to calculate their hash. The output is identical to the previous command, but we needed significantly less execution time to get the result. The screenshot below shows a performance comparison of both solutions:

```

Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
Untitled1.ps1* X
1 $srcDir = "D:\ISO Files"
2
3 Measure-Command {Get-ChildItem $srcDir -File -Recurse | Get-FileHash | Group -Property Hash `
4 | where { $_.Count -gt 1 } | foreach { $_.Group | select Path, Hash }
5 }
6
7 Measure-Command {Get-ChildItem -Path $srcDir -File -Recurse | Group -Property Length | where { $_.Count -gt 1 } `
8 | select -ExpandProperty Group | Get-FileHash | Group -Property Hash | where { $_.count -gt 1 } `
9 | foreach { $_.Group | select Path, Hash }
10 }

Days          : 0
Hours         : 0
Minutes       : 5
Seconds       : 52
Milliseconds  : 188
Ticks         : 3521881796
TotalDays     : 0.0040762520787037
TotalHours    : 0.0978300498888889
TotalMinutes  : 5.86980299333333
TotalSeconds  : 352.1881796
TotalMilliseconds : 352188.1796

PS C:\Windows\system32> Measure-Command {Get-ChildItem -Path $srcDir -File -Recurse | Group -Property Length `
| where { $_.Count -gt 1 } | select -ExpandProperty Group | Get-FileHash
| Group -Property Hash | where { $_.count -gt 1 } | foreach { $_.Group | select Path, Hash }
}

Days          : 0
Hours         : 0
Minutes       : 0
Seconds       : 36
Milliseconds  : 50
Ticks         : 360505371
TotalDays     : 0.000417251586805556
TotalHours    : 0.0100140380833333
TotalMinutes  : 0.600842285
TotalSeconds  : 36.0505371
TotalMilliseconds : 36050.5371

```

Using Measure Command to compare the performance of two commands

You can see that the second command took just **36** seconds, whereas the first one took about **5** minutes. This difference could increase significantly, depending on the size and number of duplicate files in your source directory.

To speed it up even further, you could use the MD5 algorithm with the *GetFileHash* cmdlet, which is way faster than the default (SHA256). MD5 isn't good for hashing secret information, but we are only using it here to compare files. The [hash-collision](#) risk is relatively higher in MD5, but it doesn't make much difference unless you're comparing billions of files.

Removing duplicate files

Now that we have come up with the right command to identify duplicate files in PowerShell, we need to handle them in a suitable manner. If you are working with important files, it is not recommended to delete duplicates straight away. Instead, you can move them to another directory, probably on a different drive, with enough free space. This is very easy with our new command. We just need to pipe the output of the above command and pass it to the *Move-Item* cmdlet. See the updated code shown below.

```

# Define source directory
$srcDir = "D:\ISO Files"
# Define destination directory
$targetDir = "E:\DuplicateFiles\$(Get-Date -Format 'yyyyMMdd')"
# Create destination directory
if(!(Test-Path -PathType Container $targetDir)){ New-Item -ItemType Directory -Path $targetDir | Out-Null }
# Move duplicate files to a different location
Get-ChildItem -Path $srcDir -File -Recurse | group -Property Length | where { $_.Count -gt 1 } `
| select -ExpandProperty Group | Get-FileHash | group -Property Hash `
| where { $_.Count -gt 1 } | foreach { $_.Group | select -Skip 1 } `
| Move-Item -Destination $targetDir -Force -Verbose

```

```

Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
Find-DuplicateFilesV2.ps1 X
1 # Define source directory
2 $srcDir = "D:\ISO Files"
3
4 # Define destination directory
5 $targetDir = "E:\DuplicateFiles\$(Get-Date -Format 'yyyyMMdd')"
6
7 # Create destination directory
8 if(!(Test-Path -PathType Container $targetDir)){ New-Item -ItemType Directory -Path $targetDir | Out-Null }
9
10 # Move duplicate files to a different location
11 Get-ChildItem -Path $srcDir -File -Recurse | group -Property Length | where { $_.Count -gt 1 } `
12 | select -ExpandProperty Group | Get-FileHash | group -Property Hash `
13 | where { $_.Count -gt 1 } | foreach { $_.Group | select -skip 1 } `
14 | Move-Item -Destination $targetDir -Force -Verbose
15
16 Get-ChildItem $targetDir

```

PS C:\Windows\system32> D:\MyScripts\Find-DuplicateFilesV2.ps1
VERBOSE: Performing the operation "Move File" on target "Item: D:\ISO Files\net55_r8168_8.045a_napi_bundle.zip Destination: E:\DuplicateFiles\20221026\net55_r8168_8.045a_napi_bundle.zip".
VERBOSE: Performing the operation "Move File" on target "Item: D:\ISO Files\VMware-ESXi-6.5.0.iso Destination: E:\DuplicateFiles\20221026\VMware-ESXi-6.5.0.iso".
VERBOSE: Performing the operation "Move File" on target "Item: D:\ISO Files\winPE.iso Destination: E:\DuplicateFiles\20221026\winPE.iso".
VERBOSE: Performing the operation "Move File" on target "Item: D:\ISO Files\win_PE.iso Destination: E:\DuplicateFiles\20221026\win_PE.iso".

Directory: E:\DuplicateFiles\20221026

Mode	LastWriteTime	Length	Name
-a----	9/30/2021 2:52 PM	1131668	net55_r8168_8.045a_napi_bundle.zip
-a----	9/30/2021 3:31 PM	333762560	VMware-ESXi-6.5.0.iso
-a----	2/24/2022 11:23 AM	349011968	winPE.iso
-a----	2/24/2022 11:23 AM	349011968	win_PE.iso

Find and move duplicate files to another drive automatically using PowerShell

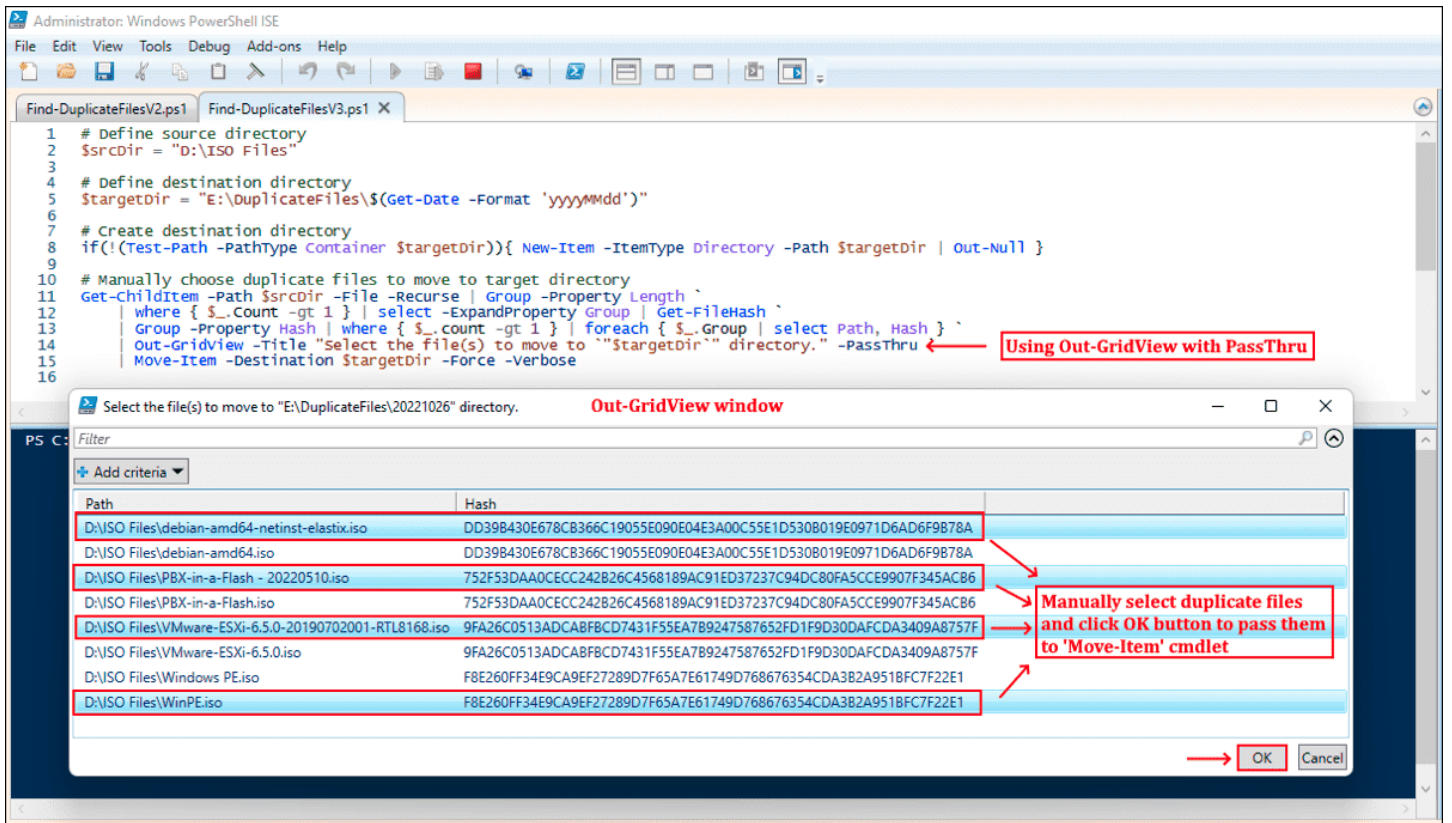
The only change here is that I used *-Skip 1* to leave one file in the source directory while moving other duplicates to the specified target directory. Once moved, you can manually review them later on and remove them, if necessary. If you're working on a huge source directory with millions of files, it is a good idea to avoid using the *-verbose* parameter with *Move-Item*.

If your directory contains a handful of files, you may want to manually select which file to move and which to leave in the source directory. You can modify the above code as follows:

```

# Define source directory
$srcDir = "D:\ISO Files"
# Define destination directory
$targetDir = "E:\DuplicateFiles\$(Get-Date -Format 'yyyyMMdd')"
# Create destination directory
if(!(Test-Path -PathType Container $targetDir)){ New-Item -ItemType Directory -Path $targetDir | Out-Null }
# Manually choose duplicate files to move to target directory
Get-ChildItem -Path $srcDir -File -Recurse | Group -Property Length `
| where { $_.Count -gt 1 } | select -ExpandProperty Group | Get-FileHash `
| Group -Property Hash | where { $_.count -gt 1 } | foreach { $_.Group | select Path, Hash } `
| Out-GridView -Title "Select the file(s) to move to `"$targetDir`" directory." -PassThru `
| Move-Item -Destination $targetDir -Force -Verbose

```



Find and move duplicate files to another drive manually using PowerShell

Here, we are using the `Out-GridView` cmdlet with the `-PassThru` switch to display duplicate files in a new window, so you can manually control what to leave in the source directory and what to move. To select multiple files, you would press and hold the `Ctrl` key while using the mouse to click through the files. When you click OK, the selected files will be moved to the target directory.

Furthermore, if you already have a good backup of your source directory and want to remove the duplicate files permanently, use the following code instead:

```

# Define source directory
$srcDir = "D:\ISO Files"
# Permanently delete duplicate files; use with caution
Get-ChildItem -Path $srcDir -File -Recurse | group -Property Length | where { $_.Count -gt 1 } `
| select -ExpandProperty Group | Get-FileHash | group -Property Hash `
| where { $_.Count -gt 1 } | foreach { $_.Group | select -Skip 1 } `
| Remove-Item -Force -Verbose

```



```
Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
Find-DuplicateFilesV4.ps1* X
1 # Define source directory
2 $srcDir = "D:\ISO Files"
3
4 # Permanently delete duplicate files; use with caution
5 Get-ChildItem -Path $srcDir -File -Recurse | group -Property Length | where { $_.Count -gt 1 } `
6 | select -ExpandProperty Group | Get-FileHash | group -Property Hash
7 | where { $_.Count -gt 1 } | foreach { $_.Group | select -skip 1 } `
8 | Remove-Item -Force -Verbose
9
10 Get-ChildItem $srcDir
Remove duplicate files directly

PS C:\windows\system32> D:\MyScripts\Find-DuplicateFilesV4.ps1
VERBOSE: Performing the operation "Remove File" on target "D:\ISO Files\net55_r8168-8.045a_napi_bundle.zip".
VERBOSE: Performing the operation "Remove File" on target "D:\ISO Files\VMware-ESXi-6.5.0.iso".
VERBOSE: Performing the operation "Remove File" on target "D:\ISO Files\winPE.iso".
VERBOSE: Performing the operation "Remove File" on target "D:\ISO Files\win_PE.iso".

PS C:\windows\system32> Get-ChildItem $srcDir

Directory: D:\ISO Files

Mode                LastWriteTime         Length Name
----                -
-a----            3/28/2022  4:54 PM      2085617664 AlmaLinux-8.5-x86_64-minimal.iso
-a----            7/30/2022  1:38 PM      396361728  debian-amd64.iso
-a----           10/20/2018  9:32 AM      1354811392 HBCD_PE_x64.iso
-a----           11/8/2012  10:38 PM        623890432 Hiren's BootCD 15.2.iso
-a----            4/20/2022  10:53 AM      2994323456 kali-linux-2022.1-installer-amd64.iso
-a----            9/30/2021  2:52 PM        1131668   net55-r8168-8.045a-napi-offline_bundle.zip
-a----            8/3/2022  2:51 PM      396361728  PBX-in-a-Flash.iso
-a----            4/7/2022  2:28 PM      5044094976 SERVER_2022_EVAL_x64.iso
-a----           1/6/2022   4:06 PM      3116482560 ubuntu-21.10-desktop-amd64.iso
-a----            9/30/2021  3:31 PM      333762560  VMware-ESXi-6.5.0-boot.iso
-a----            2/24/2022  11:23 AM      349011968  windows PE.iso
-a----            7/16/2022  9:31 AM      4493541376 WINDOWS_10_ENT.iso
-a----            8/13/2022  12:44 PM      4600823808 WINDOWS_10_PRO.iso
-a----            9/2/2022   6:01 PM      4444651520 WIN_11_ENT_21H2.iso
```

Find and remove duplicate files with PowerShell

You can see in the screenshot that four duplicate files were deleted successfully, and in the end, our source directory has only unique files.