

# How to Delete a File with PowerShell Remove-Item

Deleting files and cleaning up directories can be a time-consuming task, especially when you need to perform the same task on multiple computers regularly. But with PowerShell, we can create small scripts that will delete a file quickly and efficiently by using the Remove-Item cmdlet.

PowerShell allows us not to only delete a single file, but even an entire directory or a group of files based on specific criteria. And did you know it's also possible to only delete a file if it exists, files that are older than x days, or delete files based on a wildcard pattern?

In this article, we are going to take a look at how to delete a file in PowerShell. And how to use the different options to delete only specific files, verify the delete action, and more.

## Delete a file with PowerShell

To delete files with PowerShell we need to use the Remove-Item cmdlet. This cmdlet can delete one or more items based on the criteria. The Remove-Item cmdlet can not only be used to delete files but also for deleting folders, registry keys, variables, functions, and more. We will focus on deleting files in this article.

With the Remove-Item cmdlet, we can use the following parameters:

Parameter	Description
-Path	Specify the path of the items that need to be removed.
-LiteralPath	The exact path the item
-Include	Specifies a path element to include, for example, *.txt
-Exclude	Specifies a path element to exclude, for example, *.txt
-Recurse	Delete items in the specified location and in all subfolders
-Force	Used to delete read-only files or hidden items
-Confirm	Prompt for confirmation

So to delete a single file with PowerShell we only need to specify the full path to the file including the file name. For example, if we want to delete the readme.txt file from the folder below we can do:

```
Remove-Item -Path C:\temp\files\readme.txt
```

Note that the cmdlet won't ask for confirmation, the file will be deleted instantly. So when you are writing your PowerShell script that is going to delete one or more files, it's important to test it first. We can do this by using either the `-confirm` or `-whatif` parameter. Confirm will give a prompt if you want to delete the file or not, where what if shows what the cmdlet would do without actually deleting the item:

#### # Using Confirm

```
Remove-Item -Path C:\temp\files\readme.txt -Confirm
```

Confirm

Are you sure you want to perform this action?

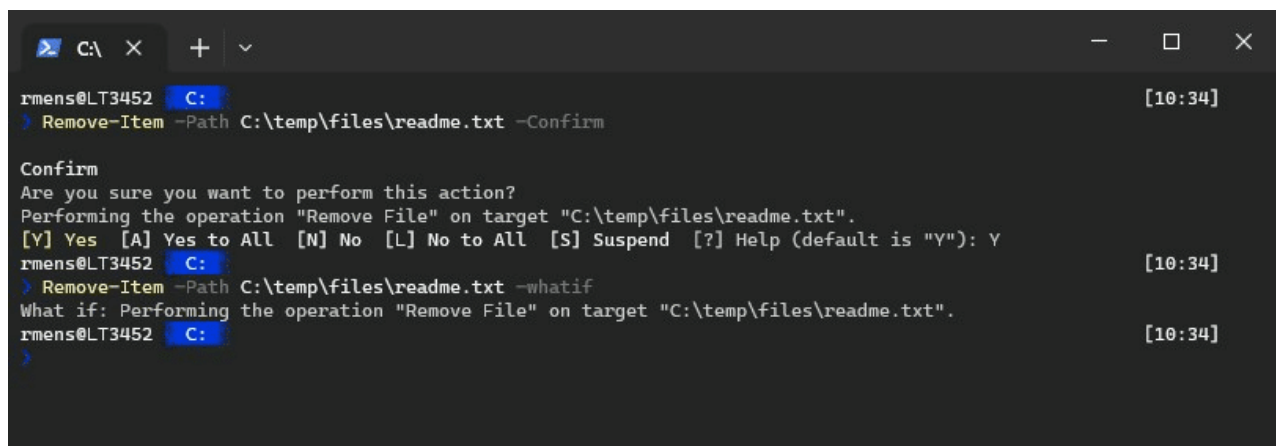
Performing the operation "Remove File" on target "C:\temp\files\readme.txt".

[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"):

#### # Or using WhatIf

```
Remove-Item -Path C:\temp\files\readme.txt -whatif
```

What if: Performing the operation "Remove File" on target "C:\temp\files\readme.txt".



```
rmens@LT3452 C: [10:34]
> Remove-Item -Path C:\temp\files\readme.txt -Confirm

Confirm
Are you sure you want to perform this action?
Performing the operation "Remove File" on target "C:\temp\files\readme.txt".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"): Y
rmens@LT3452 C: [10:34]
> Remove-Item -Path C:\temp\files\readme.txt -whatif
What if: Performing the operation "Remove File" on target "C:\temp\files\readme.txt".
rmens@LT3452 C: [10:34]
>
```

## Delete all Files in a Folder

We can use the `Remove-Item` cmdlet in PowerShell also to delete all files in a folder. Now important to note here is that the cmdlet can also remove folders. So to delete **only the files**, we are going to use a wildcard to select only the items that include a dot `.` in the name. This way only files are deleted and not any subfolders:

```
# Remove all files from the path c:\temp\files
```

```
# Tip: test the results first with the -whatif parameter
```

```
Remove-Item -Path C:\temp\files\*.*
```

Another option is to use the cmdlet `Get-ChildItem` with the parameter `-File` to get only the files and then pipe the `Remove-Item` cmdlet to remove the files with PowerShell:

```
Get-ChildItem -Path C:\temp\files\ -File | Remove-Item
```

**FREE EMAIL SERIES!**

# Level Up with PowerShell

---

5 Emails, Endless Skills



## Remove Files in Folder and SubFolders

---

To delete files not only in the folder but in the subfolder as well, you will need to use the `-Recurse` parameter. There is only one problem, you can't use `-Recurse` recombination with the files-only selection method `*.*`. Take the example below:

```
Remove-Item -Path C:\temp\files\*.* -Recurse
```

You probably expect that this will delete only the files in the folder and subfolder. However, by using the wildcard in the path, only items (files) that contain an `.` in the path name are selected. So the cmdlet won't go through any subfolders.

To remove all the files in the folder and all subfolders we will have to specify only the path of the parent folder and use the `-Include` parameter to select only the files:

```
# Delete all files from the folder c:\temp\files and it's subfolders
```

```
Remove-Item -Path C:\temp\files\ -Recurse -Include *.*
```

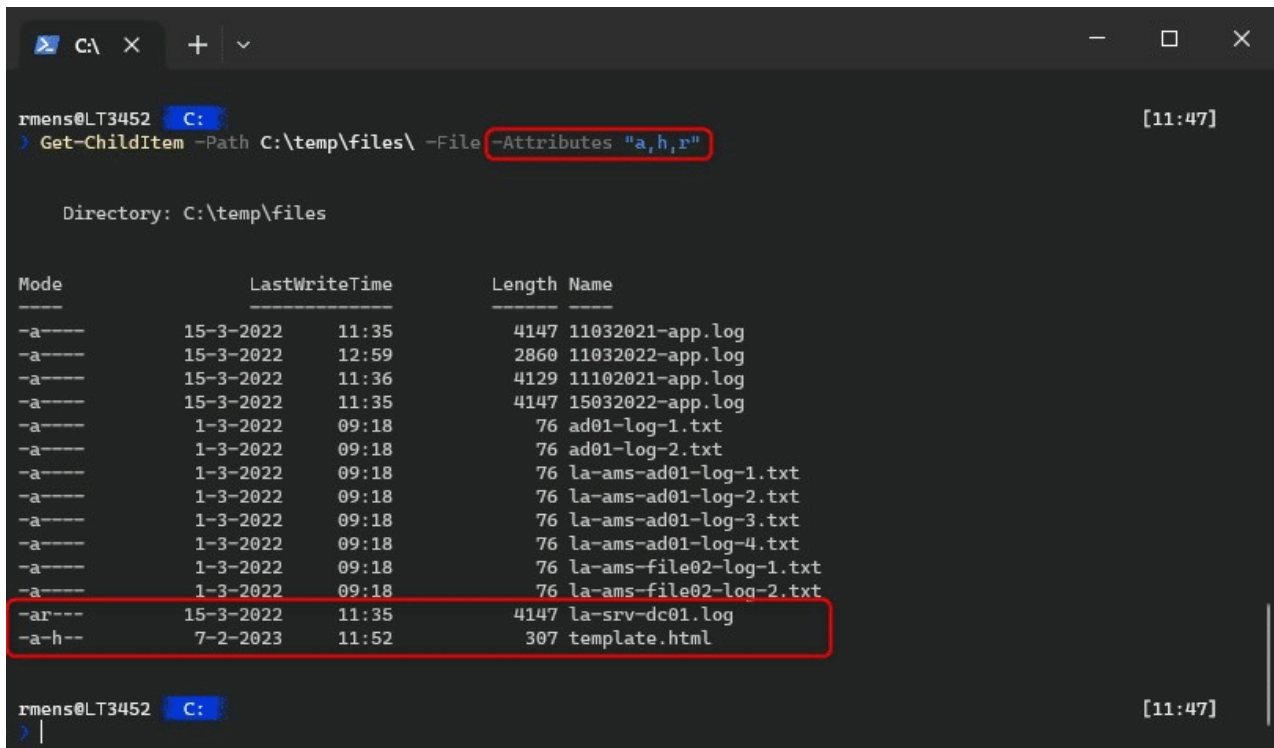
Another option is to first get the files from the folder and subfolder with the `Get-ChildItem` cmdlet and then remove them:

```
Get-ChildItem -Path C:\temp\files\ -File -Recurse | Remove-Item
```

## Delete Read-Only and Hidden Files

---

Read-Only and hidden files are protected from deletion by default. Let's take a look at the example below, the file "la-srv-dc01.log" is marked read-only and the template.html is a hidden file.



```
rmens@LT3452 C: [11:47]
> Get-ChildItem -Path C:\temp\files\ -File -Attributes "a,h,r"

Directory: C:\temp\files

Mode                LastWriteTime         Length Name
----                -
-a-----         15-3-2022   11:35         4147 11032021-app.log
-a-----         15-3-2022   12:59         2860 11032022-app.log
-a-----         15-3-2022   11:36         4129 11102021-app.log
-a-----         15-3-2022   11:35         4147 15032022-app.log
-a-----          1-3-2022    09:18           76 ad01-log-1.txt
-a-----          1-3-2022    09:18           76 ad01-log-2.txt
-a-----          1-3-2022    09:18           76 la-ams-ad01-log-1.txt
-a-----          1-3-2022    09:18           76 la-ams-ad01-log-2.txt
-a-----          1-3-2022    09:18           76 la-ams-ad01-log-3.txt
-a-----          1-3-2022    09:18           76 la-ams-ad01-log-4.txt
-a-----          1-3-2022    09:18           76 la-ams-file02-log-1.txt
-a-----          1-3-2022    09:18           76 la-ams-file02-log-2.txt
-ar-----         15-3-2022   11:35         4147 la-srv-dc01.log
-a-h-----         7-2-2023   11:52          307 template.html

rmens@LT3452 C: [11:47]
> |
```

If we would try to delete all the files in the folder, then you will get an error on the read-only file that you do not have sufficient access rights to perform this operation. The hidden file isn't even deleted at all.

So how do we delete those files? To delete the read-only and hidden files as well we will need to use the **-Force** parameter:

```
# Remove all files, including read-only and hidden
Remove-Item -Path C:\temp\files\*. * -Force
```

## PowerShell Delete File if Exists

When using the Remove-Item cmdlet inside scripts, it's a good idea to test if the file exists before you try to delete it. We can do this by using the Test-Path cmdlet. This will check if the file exists and return true or false based on the result. By using this inside an if statement we can make sure that the file is only deleted if it exists:

```
$file = C:\temp\files\readme.txt
if (Test-Path -Path $file) {
    Remove-Item -Path $file
}
```

## Using Filters to Delete Files in PowerShell

We now have looked at some of the principles to delete files with PowerShell. But besides simply deleting a single file or all the files in the folder, you will often encounter situations where you only need to delete files older than x days, or files with a specific file type.

We can use a variety of methods to select only those files and delete them with PowerShell. When working with filters always make sure that you first test and verify the selection, before you actually delete the files.

## Using the Include Filter

---

The Include parameter allows us to select all files that have a specific string or part of a string in its file name. For example, if you only want to remove the .log files from the folder, you can do:

```
# Remove all .log items
```

```
Remove-Item -Path C:\temp\files\ -Include *.log
```

The include parameter isn't limited to file extensions, you can also select files based on part of their filename. And it's also possible to specify multiple "keywords" that you want to select:

```
# Get all files that have the word app or process in their name.
```

```
Remove-Item -Path C:\temp\files\ -Include *app*, *process* -Recurse -WhatIf
```

```
# Result
```

```
What if: Performing the operation "Remove File" on target  
"C:\temp\files\subfolder\LT3452-process-errors.log".
```

```
What if: Performing the operation "Remove File" on target  
"C:\temp\files\subfolder\LT3452-process.log".
```

```
What if: Performing the operation "Remove File" on target "C:\temp\files\11032021-  
app.log".
```

```
What if: Performing the operation "Remove File" on target "C:\temp\files\11032022-  
app.log".
```

## Using the Exclude Filter

---

Following the principle of the include parameter, we can also exclude files. Include and exclude can be used together, so you could for example remove all log files except the process log files:

```
Remove-Item -Path C:\temp\files\ -Include *.log -Exclude *process* -Recurse -WhatIf
```

```
# Result
```

```
What if: Performing the operation "Remove File" on target "C:\temp\files\11032021-  
app.log".
```

```
What if: Performing the operation "Remove File" on target "C:\temp\files\11032022-  
app.log".
```

```
What if: Performing the operation "Remove File" on target "C:\temp\files\la-srv-dc01.log".
```

## Delete Files Older than x Days with PowerShell

---

A common practice when cleaning up directories is to delete only files that are older than x days. To do this, we will first need to calculate the file date. let's say we want to delete all files that are older than 30 days, we then first need to get the date from 30 days ago:

```
# Get the date from today minus 30 days
```

```
$dateTime = (Get-Date).AddDays(-30)
```

The next step is to get all files that are older than the given date. To do this we are going to compare the `lastwritetime` of the file with the `datetime` that we calculated:

```
Get-ChildItem -Path $Path -Recurse -File | Where-Object { $_.LastWriteTime -lt  
$dateTime }
```

We then only have to pipe the `Remove-Item` cmdlet behind it to actually delete the files:

```
$dateTime = (Get-Date).AddDays(-30)
```

```
$path = "C:\temp\files\"
```

```
Get-ChildItem -Path $Path -Recurse -File | Where-Object { $_.LastWriteTime -lt  
$dateTime } | Remove-Item
```

## Delete Files Larger Than

---

To delete all files that are larger than a given size, we will first need to convert the file size from Mb to kbits. With the correct size, we can then select all files that are larger than the givne size and remove them with the `Remove-Item` cmdlet:

```
$path = "C:\temp\files\"
```

```
# Set file size
```

```
$sizeInMb = 500
```

```
# Calculate actual file size
```

```
$size = $sizeInMb*1024*1024
```

```
# Delete all files that are larger then given size
```

```
Get-ChildItem -Path $Path -Recurse -File | Where-Object { $_.length -gt $size } |  
Remove-Item
```

## Wrapping Up

---

PowerShell is a great tool when it comes to managing and deleting files automatically. You can write scripts that will clean up old files every week to quickly find and delete large files from the given folder.

Keep in mind that files are deleted permanently, so make sure that you test your commands first using the parameter before you actually delete the files.

I hope you found this article useful, if you have any questions, just drop a comment below.

Did you **Liked** this **Article**?

Get the latest articles like this **in your mailbox**  
or share this article

I hate spam to, so you can unsubscribe at any time.