How to create files and folders easily with Powershell

The good news is that you can use PowerShell to manage files easily. In this article, TipsMake will focus on how you can create files and folders in PowerShell.

File management is an essential part of everyday work. You will have to manage access to shared folders, back up them regularly, as well as perform other standard operations on files and folders in the Windows system. The good news is that you can use PowerShell to manage files easily.

In fact, this tool can reduce the amount of manual work involved in file management. In this article, **TipsMake** will focus on how you can create files and folders in PowerShell.

How to create files and folders using Powershell?

1. Create a file
    1. Create the file after double checking the file name
2. Overwrite an existing file
3. Write data to new file
4. Create a directory
    1. Use original Windows PowerShell
    2. Using the .NET framework class
    3. Use the Object
    4. Use the md function

Create a file

To create a new file, use the **New** cmdlet. Generally, this cmdlet is used to create any type of object in PowerShell. All you have to do is specify the type of object you want to create in this cmdlet.

This is how you can create files or folders in PowerShell with the **New-Item** command **.**

```
New-Item -Path 'SharedTestFoldertestfile1.txt' - ItemType File
```

The above command will create a new file called **testfile1.txt.** Here, **Itemtype** is a file, so this command will create one for you and leave the content empty. In other words, this command will create a blank text file for you.

Create the file after double checking the file name

Now, if you already have a file with the same name, it will be overwritten. So make sure you double check the file name before creating. Another option is to have a script automatically check if another file exists with the same name and will only create the file if no other file is named there.

An example of such a script is shown below.

```
$filenames = Get-ChildItem SharedTestFolder*.txt | select -expand fullname $filer
```

The above script will check if a file with the same name exists in the same directory and if it does, it will exit. If not, a new file will be created.

Overwrite an existing file

Earlier, the article showed you how to avoid overwriting an existing file. But sometimes, you have to overwrite a file and can do it using the **Force** parameter **.**

```
New-Item -Path 'SharedTestFoldertestfile1.txt' - ItemType File -Force
```

This command will force to create a file and overwrite the existing file, because you are telling the system that you know what you do and do not need any warning or prompt about the overwritten file. Although this sounds like a good idea, there's always a chance that you'll get an error and you could lose the data contained in existing files. This is why it is not recommended to abuse this conversion parameter, and if you must do so, be very careful.

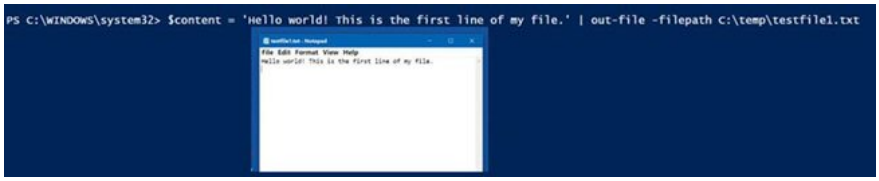
Write data to new file

In most cases, you don't need an empty file because it's pointless. Instead, you want a file containing data and it is best to have the option to add content to it regularly.

The good news is that PowerShell has an integrated command called **Out-File** to create files and write data to them at the time of creation.

```
$content = 'Hello world! This is the first line of my file.' | out-file -filepath
```

This will create a file named **textfile1.txt** in the **C: Temp** directory and it will contain your message.



Write data to new file
So there are different ways to create a file that includes the considerations you should keep in mind.

Next, we will move on to the directory creation section.

Create a directory

There are 4 ways to create folders in PowerShell and the choice depends on your programming needs.

Use original Windows PowerShell

Perhaps the easiest way to create a directory is to use the original Windows PowerShell command. You can also use the same command used in creating files to create folders and, as you anticipated, simply change the value of **ItemType** :

```
New-Item -Path 'SharedTestFolder' - ItemType Directory
```

This command returns an object called **DirectoryInfo that** has the directory mode, the last recording time, and the length of the name.

You can even omit the **Path** parameter and can specify the path name directly. The result will be the same.

```
New-Item 'SharedTestFolder' - ItemType Directory
```

Be sure to include the **ItemType** value **.**

Using the .NET framework class

The next option is to create a directory using the **system.io** namespace and the static method **CreatDirectory** in the **Directory** class **.**

The command you can use is:

```
[system.io.directory]::CreateDirectory("SharedTestFolder")
```

Warning : This is the .NET framework class and it is not considered a good way to use it in a Windows PowerShell environment. The choice is up to you and this command will help create a directory. Like the previous option, the command will return a **DirectoryInfo** object with all directory details.

Use the Object

The third way is to use the Object method from Scripting.FileSystemObject. Note that this is the same method used in VBScript, which has the advantage of being quick and relatively easy to use to create directories.

```
$newdirectory = new-object -ComObject scripting.filesystemobject $newdirectory.C
```

This command will return an object containing the directory path and other information.

Use the md function

The last option is to create a directory using the **md** function **.** Many people find this convenient because they have used it a lot in the Command Prompt to create different folders. Also, the syntax is quite simple and you don't need to mention **ItemType** , because **md** alone is enough to tell the system that you are already creating the directory.

The command to use is:

```
md SharedTestFolder
```

This command also returns a **DirectoryInfo** object **.**