How to read text files in Powershell quickly and easily

PowerShell understands the importance of text files and makes it easy to access or read them. In this article, TipsMake can show you how to read text files in PowerShell.

Text files are everywhere and you'll have to read through them, regardless of your role or job responsibilities. In fact, if you're an IT professional, you'll have to create, read or work with text files more often, as they come in a variety of formats. This flexibility has also made text files the most convenient way to identify scripts, store configuration details, etc.

PowerShell understands the importance of text files and makes it easy to access or read them. In this article, **TipsMake** can show you how to read text files in PowerShell.

Ways to read text files in Powershell

1. Read the full content
2. Read a piece of content
3. Read line by line
    1. Use Get-Content
    2. Use the StreamReader class
4. Find specific text
5. More options with Get-Content
    1. Count the number of lines in the file
    2. Choose specific line numbers at the beginning and end
    3. File constantly updated

Read the full content

Read the full content

When you want to read the entire contents of a text file, the easiest way is to use the integrated **Get-Content** function. Here is the code that allows you to do this:

```
Get-Content C:logslog01012020.txt
```

```
PS C:\WINDOWS\system32> Get-Content C:\logs\log01012020.txt
Fourscore and seven years ago our fathers brought forth on this continent a new nation, conceived in liberty and dedi
cated to the proposition that all men are created equal.

Now we are engaged in a great civil war, testing whether that nation, or any nation so conceived and so dedicated, ca
n long endure. We are met on a great battlefield of that war. We have come to dedicate a portion of that field as a f
inal resting-place for those who here gave their lives that that nation might live. It is altogether fitting and prop
er that we should do this. But, in a larger sense, we cannot dedicate - we cannot consecrate - we cannot hallow - thi
s ground. The brave men, living and dead, who struggled here have consecrated it, far above our poor power to add or
detract. The world will little note, nor long remember what we say here, but it can never forget what they did here.
It is for us the living, rather, to be dedicated here to the unfinished work which they who fought here have thus far
 so nobly advanced. It is rather for us to be here dedicated to the great task remaining before us - that from these
honored dead we take increased devotion to that cause for which they gave the last full measure of devotion - that we
 here highly resolve that these dead shall not have died in vain - that this nation shall have a new birth of freedom
 and that government of the people, by the people, for the people, shall not perish from the earth.
```

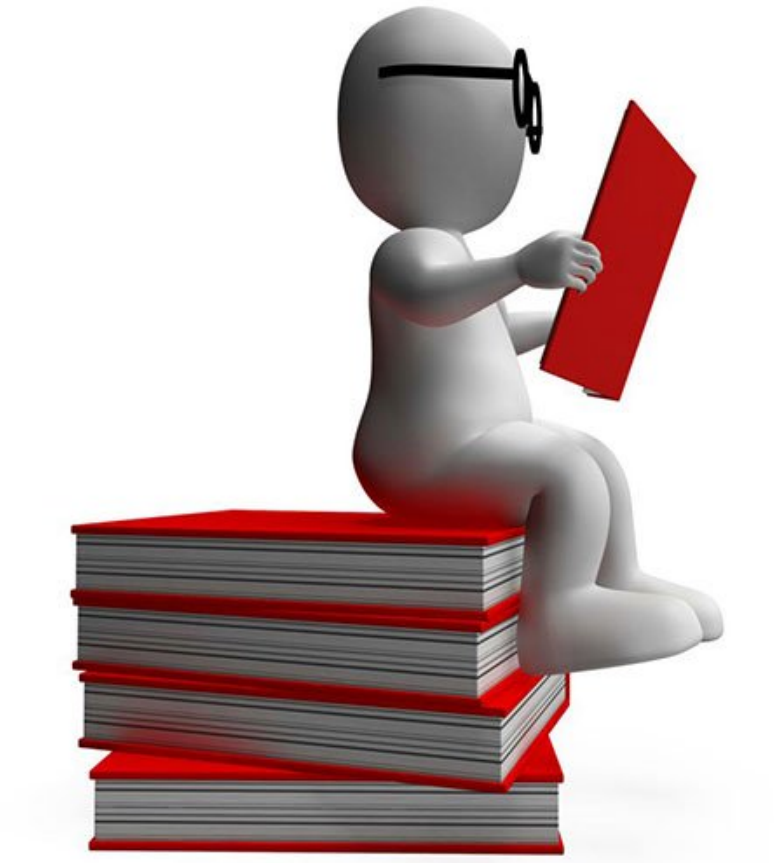
The content of this file will be displayed in the screen PowerShell ISE or Command Prompt
When you execute this command, the contents of this file will be displayed in the PowerShell ISE or Command Prompt screen, depending on where you execute.

You can also move all content to a variable and use that variable for further processing, if that's what you want your code to do.

```
$file_data = Get-Content C:logslog01012020.txt
```

You can now use this **$ file_data** variable to parse or further process.

Read a piece of content



You can select the lines you want to read
In many cases, you may just have to read a certain part of the file to get the information you want. Similar to SQL queries, you can select the lines you want to read and the code to do that is:

```
$file_data = Get-Content C:logslog01012020.txt $file_data | Select-Object -First
```

As the code shows, the first 10 lines will be stored in variables, not the entire content. You can display the contents of this variable or use it to process more.

Similarly, we can also read the last few lines:

```
$file_data = Get-Content C:logslog01012020.txt $file_data | Select-Object -Last
```

Read line by line

When you want to read a file to understand its content, you have to do it line by line and the good news is, this is possible with PowerShell. In fact, there are 2 ways to do that.

Use Get-Content

The **Get-Content** function reads every line in the text and stores them as an array, where each line is an array element. In the example above, the article used one variable to read all the content.

```
$file_data = Get-Content C:logslog01012020.txt
```

If the output of this variable, you can see it is an array. This means you can pick out specific lines using the array index. For example, if you want to read the first line, simply enter:

```
$file_data [0]
```

And this command will show the first line to you.

```
$file_data[1]
```

This command will display the second line (likewise with the next line).

Use the StreamReader class

The second option is to use a .NET class called **StreamReader.**

```
$stream_reader = New-Object System.IO.StreamReader{C:logslog01012020.txt}
```

Now you have the contents of the log file in this **$ stream_reader** variable and since it belongs to the **StreamReader** class, you can use many built-in methods to get the text you want.

As you type:

```
$stream_reader.ReadToEnd()
```

The command will output all content on your screen, similar to **Get-Content.**

To read the current line, you can use the following method:

```
$stream_reader.ReadLine()
```

But this command alone may not be useful, so you'll have to run it in a loop and read the contents of the text file, line by line.

```
$stream_reader = New-Object System.IO.StreamReader{C:logslog01012020.txt} $line_
```

The above code will start with the first line and output each line with the number of lines attached for readability. You can even use any of the available string methods on the **$ current_line** variable for further parsing.

This method is ideal for reading large files, because when you store content in a variable, it can take up too much memory and affect performance. So this is a more efficient way to get to the content you want.

Find specific text

In many cases, you want to find a specific text in a file and it is best to use the **Where-Object** cmdlet to filter the content.

```
$file_data = Get-Content C:logslog01012020.txt | Where-Object {$_ -like '*error*
```

The above code will output lines with the word '**error**' in it. Here, **$ _** is a variable representing the current line from the content, coming from **Get-Content.**

Besides **Where-Object** , you can also use the **match** and **regex** operators to find the exact text you want.

More options with Get-Content

Get-Content is a very flexible cmdlet that comes with many options. Here are some things you can do with it.

Count the number of lines in the file

You may want to know the number of lines available in the file. The code to do that is:

```
$file_data = Get-Content C:logslog01012020.txt | Measure-Object
```

Choose specific line numbers at the beginning and end

Earlier, you learned how to select the first few rows or the last few lines using the **Select-Object** cmdlet **.** You can also get similar results by using some methods integrated with **Get-Content.**

To get the first few lines, the method is:

```
$file_data = Get-Content C:logslog01012020.txt -TotalCount 3
```

This command will return the first 3 lines from the file.

```
$file_data = Get-Content C:logslog01012020.txt -Tail 3
```

This command will return the last 3 lines from the file.

File constantly updated

Suppose your log file is constantly updated and you want to look at the end of that log file to read the latest updated values. You can add the **Wait** parameter, like this:

```
$file_data = Get-Content C:logslog01012020.txt -Tail 3 -Wait
```

This command will continuously monitor the log file to find new lines added and displayed for you.