# IN609 Operations Engineering 1

Semester 2, 2021

# Checkpoint 3.2

# PowerShell file processing

Total Points: 8

# Learning Outcome:

1. Develop advanced shell scripts to automate system administration tasks

# Objectives:

* Develop an understanding of how to parse text files and perform basic information retrieval tasks

**Instructions:**

* Solve the following tasks based on the **integrated help** (get-help), **lecture slides, previous labs** and PowerShell **online reference**. You are not allowed to use the Internet other than using powershell online help.
* You need to complete the tasks within the allocated time.
* There are two tasks each weighing 4 points.



# Preparation

1. Open the PowerShell ISE (To do that type ‘powershell’ in the search field on your workstation
2. Select Windows **PowerShell ISE**). You can also perform this lab in the basic PowerShell environment (Windows **PowerShell**), but you may be missing the code completion facilities of the Interactive Scripting Environment (ISE).
3. You can also pin it to your taskbar for ease of access in future.

**Please write down the commands you used to achieve the tasks. Also include the screenshots of command execution**

**Task 1: Parsing Text (4 points)**

A challenging task is to parse text files, especially when the content is poorly structured (i.e. not available as XML or CSV), which is the case for many log files. However, you should still be able to extract essential information. We do this using text mining capabilities PowerShell provides out of the box.

We will be using the set of Shakespeare’s works as an example. You will find the files ‘*shakespeare.txt’* and ‘*stopwords.txt’* (this one is relevant for the later task) on *Teams->files->Week3*

The file ‘shakespeare.txt’ is quite large (for a text file), so you should use a powerful editor such as Notepad++ if you want to inspect it. In general, it is a good idea to get an understanding about file structures before you start parsing those. It will help you find structural markers (i.e. text formatting), such as titles, author names, indicators for the end of a section, etc. which you can then use to build your parser.

**Task:**

|  |
| --- |
| The task is to find the top 20 words used in his novel ‘The Tragedy of Antony and Cleopatra’ – which is only one of many in this file. |

**Hints:** If you break the problem down into sub-tasks you should be able to solve it. Look at the file content to understand its structure and think about how to approach the task. **Don’t just write everything out and hope that it works.** In most cases it won’t and you will need to take it apart to understand what went wrong.

What are the individual steps? (**Write them down in natural language before you start coding** to have a clearer picture of what needs to be done.)

Read in the content of the text file ‘shakespeare.txt`

Search for the title of the novel ‘The Tragedy of Antony and Cleopatra’

End search at `THE END`

For each word in the work, find 20 words that occur most often.; output the count

Once you have broken down the tasks, you can start coding! As before, work in small iterative steps (one functionality at a time) to ensure that your code is doing what it is supposed to do before proceeding.

**Some pointers:**

* Use the split function (…**| %{$\_.split(" ")} |**…) to tokenise the entire text based on whitespaces (“ “) in order to arrive at individual words.
* To determine the number of occurrences for each word, use the group object (… **| group |** …).
* Consult the previous lecture/lab material

Paste your code along with the screenshot of the output here:

$content = Get-Content -Path 'C:\Users\leggtc1\OneDrive - Otago Polytechnic\BIT\_Year 2\2.OPs Eng\Checkpoint Labs\shakespeare.txt'

$start = $content | Select-String -SimpleMatch "The Tragedy of Antony and Cleopatra" | ForEach-Object {$\_.LineNumber}

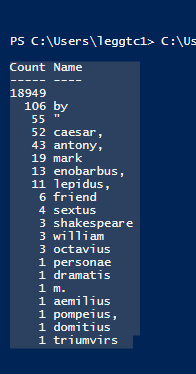
$end = $content | Select-String -SimpleMatch "High order in this great solemnity." | ForEach-Object {$\_.LineNumber}

$story = $content[$start..$end]

$words = ($story).Split(" ").ToLower() | Group-Object | Select-Object -Property Count, Name -First 20 | Sort-Object -Descending -Property Count

$words

What are some of the most frequent words?



**Task 2: Removing “stopwords” from parsed text (4 points)**

The result you have seen shows a typical challenge when parsing text. The most frequent words are words like ‘the’ and ‘and’. They don’t provide you much insight about the actual content. Search engines like Google rely on lists of stopwords to exclude such meaningless terms from the analysis, so we get more meaningful results.

For this purpose, amend your statement and add an additional filter for words that appear in the file ‘stopwords.txt’ in this week’s folder. Read the file into a variable and exclude terms that are contained in that file from processing.

(**Hint:** You can use **| where {$\_ -notin $stopwords} |** to include only terms that are not contained in the variable $stopwords.)

What is the refined PowerShell code?

$content = Get-Content -Path 'C:\Users\leggtc1\OneDrive - Otago Polytechnic\BIT\_Year 2\2.OPs Eng\Checkpoint Labs\shakespeare.txt'

$stopwords = Get-Content -Path 'C:\Users\leggtc1\OneDrive - Otago Polytechnic\BIT\_Year 2\2.OPs Eng\Checkpoint Labs\stopwords.txt'

$start = $content | Select-String -SimpleMatch "The Tragedy of Antony and Cleopatra" | ForEach-Object {$\_.LineNumber}

$end = $content | Select-String -SimpleMatch "High order in this great solemnity." | ForEach-Object {$\_.LineNumber}

$story = $content[$start..$end]

$words = ($story).Split(" ") | Group-Object | Sort-Object -Property Count -Descending

$words

What are the results now?

None