COMP-10051 PowerShell Help - By: Anthony Adams

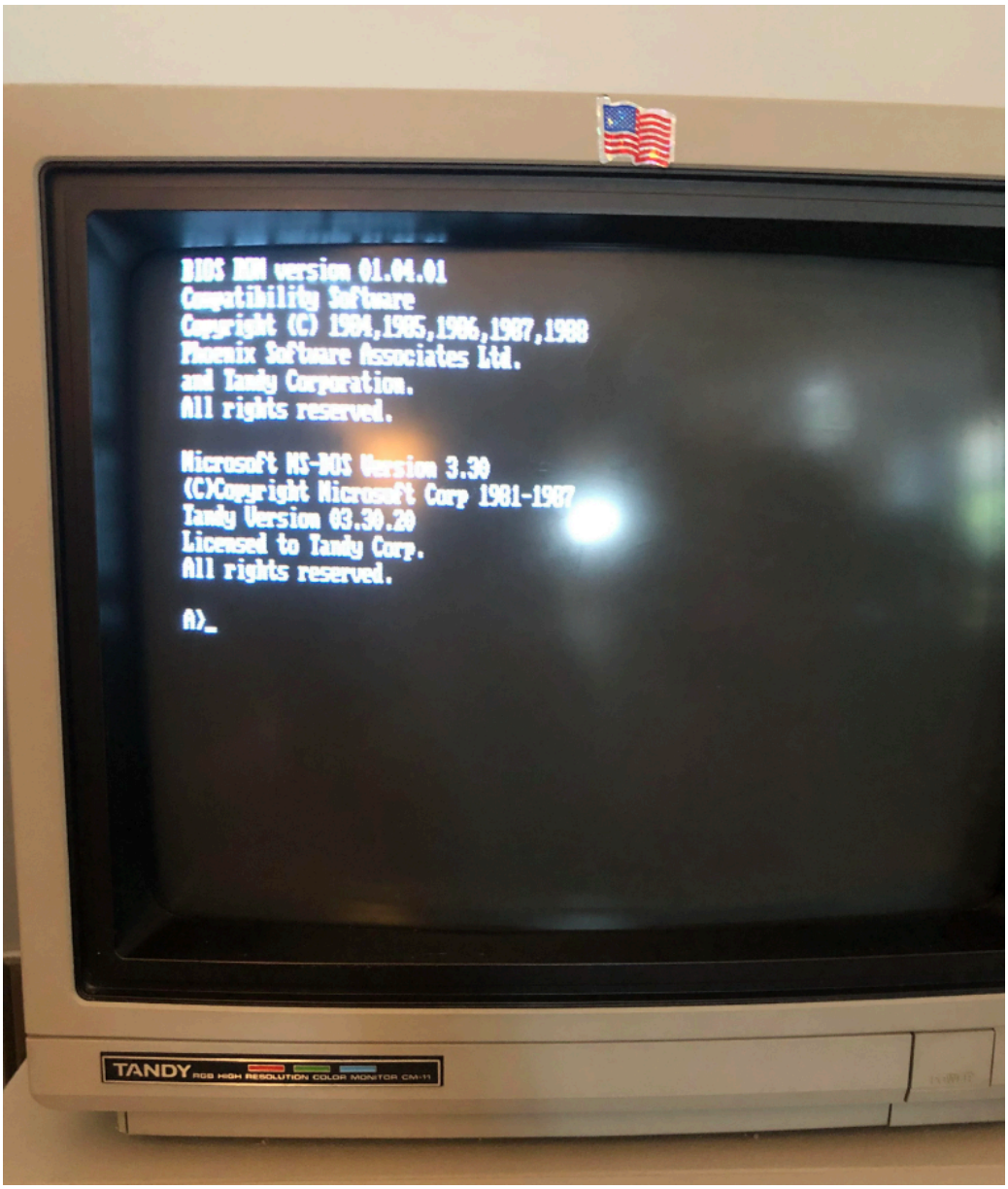
Help

Every shell has some sort of built in help system. You just have to know how to access it. Pretty much they are all similar to some degree.

Before the internet. This there was no way to just google stuff quick. The first computer I learned on when I was a kid. When it started it just booted up with a DOS command prompt and when it loaded, all it had on the screen was:

C:\

Again, there was no way to google what to do, you had to either already know, or consult the manual/help and start learning commands. Also when you forget, you had to consult the manual/help.



Help In various shells

Help in PWSH is similar to Linux or Mac's man (manual)
for example: man curl

will return the manual for curl. Which we will have to press

the q button to exit from after using it.

In other shells you can also type a command followed by -h to pull up a quick help for example: curl -h

Both of the above will work if you are using PWSH on Mac or linux. They may also work on windows, particularly the “man” command to bring up a manual. It depends on what commands you are using, if you are using a native cmdlet for PWSH.

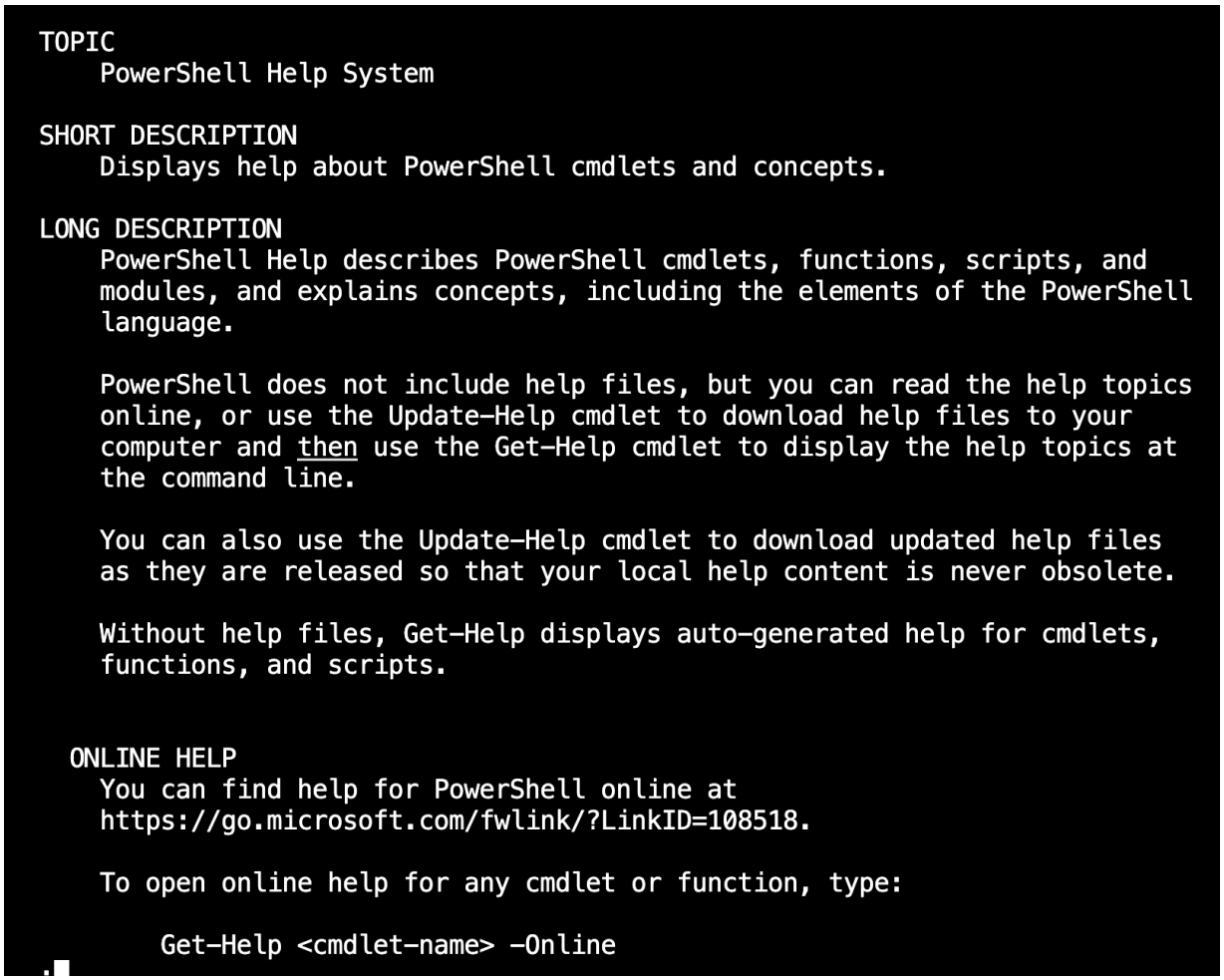
Help In Powershell

More specifically to powershell and it’s own native cmdlet’s:

-PowerShell has an abundant help system
-Every cmdlet & alias has a help file
-Similar to man in linux
-Easy to access
-Updatable

General Help:
Help

This will bring up general help



Press the q key to quit

You can bring up the same thing by using the command: get-help

Which will load it in the prompt:

```
PS /Users/maximus> get-help

TOPIC
    PowerShell Help System

SHORT DESCRIPTION
    Displays help about PowerShell cmdlets and concepts.

LONG DESCRIPTION
    PowerShell Help describes PowerShell cmdlets, functions, scripts, and
    modules, and explains concepts, including the elements of the PowerShell
    language.

    PowerShell does not include help files, but you can read the help topics
    online, or use the Update-Help cmdlet to download help files to your
    computer and then use the Get-Help cmdlet to display the help topics at
    the command line.

    You can also use the Update-Help cmdlet to download updated help files
    as they are released so that your local help content is never obsolete.

    Without help files, Get-Help displays auto-generated help for cmdlets,
    functions, and scripts.


  ONLINE HELP
```

The next level of this is the help for specific commands, similar to the example from earlier where we return the help for curl.

Get-help (command-name)

For example: get-help export-csv

```
PS /Users/maximus> get-help export-csv

NAME
    Export-Csv

SYNTAX
    Export-Csv [[-Path] <string>] [[-Delimiter] <char>] -InputObject <psobject> [-LiteralPath <string>] [-Force] [-NoClobber] [-Encoding <Encoding>] [-Append]
    [-IncludeTypeInformation] [-NoTypeInformation] [-QuoteFields <string[]>] [-UseQuotes {Never | Always | AsNeeded}] [-NoHeader] [-WhatIf] [-Confirm]
    [<CommonParameters>]

    Export-Csv [[-Path] <string>] -InputObject <psobject> [-LiteralPath <string>] [-Force] [-NoClobber] [-Encoding <Encoding>] [-Append] [-UseCulture]
    [-IncludeTypeInformation] [-NoTypeInformation] [-QuoteFields <string[]>] [-UseQuotes {Never | Always | AsNeeded}] [-NoHeader] [-WhatIf] [-Confirm]
    [<CommonParameters>]


ALIASES
    epcsv


REMARKS
    Get-Help cannot find the Help files for this cmdlet on this computer. It is displaying only partial help.
        -- To download and install Help files for the module that includes this cmdlet, use Update-Help.
        -- To view the Help topic for this cmdlet online, type: "Get-Help Export-Csv -Online" or
           go to https://go.microsoft.com/fwlink/?LinkID=2096608.
```

Note that when we get help for the command, we get some details:

Name of the command
Syntax of the command
Aliases of the command
Remarks of the command

You may also see:

Description of the command
Related Links for the command

Get help gives all documentation in one output.

It can be used with wild cards “*” which is like saying everything (which can be relative).

For example:

get-help * will pull up all the get helps for all the commands which you can browse through.

```
PS /Users/maximus> get-help *

Name                                Category  Module                    Synopsis
----                                --------  ------                    --------
foreach                             Alias                               ForEach-Object
%                                   Alias                               ForEach-Object
where                               Alias                               Where-Object
?                                   Alias                               Where-Object
clc                                 Alias                               Clear-Content
cli                                 Alias                               Clear-Item
clp                                 Alias                               Clear-ItemProperty
clv                                 Alias                               Clear-Variable
cpi                                 Alias                               Copy-Item
cvpa                                Alias                               Convert-Path
dbp                                 Alias                               Disable-PSBreakpoint
ebp                                 Alias                               Enable-PSBreakpoint
epal                                Alias                               Export-Alias
epcsv                               Alias                               Export-Csv
fl                                  Alias                               Format-List
ft                                  Alias                               Format-Table
fw                                  Alias                               Format-Wide
gal                                 Alias                               Get-Alias
gbp                                 Alias                               Get-PSBreakpoint
gc                                  Alias                               Get-Content
gci                                 Alias                               Get-ChildItem
gcm                                 Alias                               Get-Command
```

Whereas:

Get-help *help* will return all the help files specific to the help command.

```
PS /Users/maximus> get-help *help*

Name                                Category  Module                    Synopsis
----                                --------  ------                    --------
help                                Function
Get-Help                            Cmdlet    Microsoft.PowerShell.Core …
Save-Help                           Cmdlet    Microsoft.PowerShell.Core …
Update-Help                         Cmdlet    Microsoft.PowerShell.Core …
```

Note how it lists the:

Name of the cmdlet
Category of the cmdlet
Module of the cmdlet
Synopsis of the cmdlet

Adding Parameters

Get help can be used with parameters

For example: get-help get-help

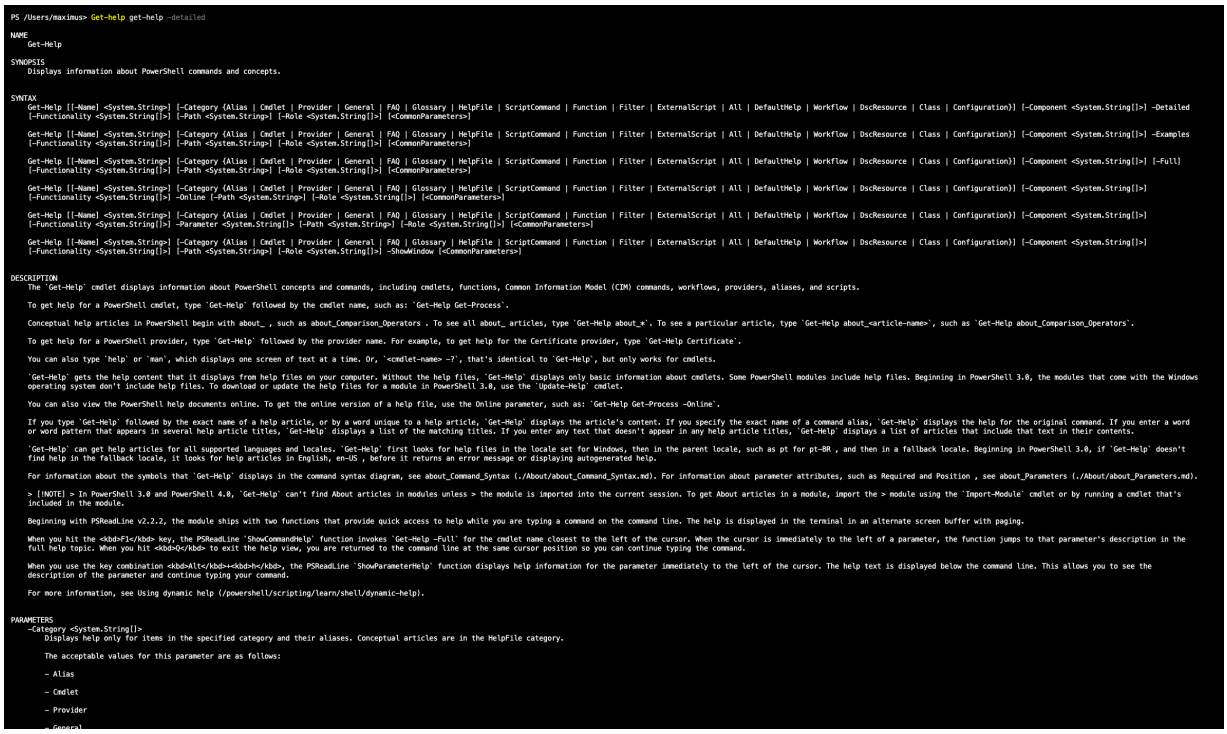
This command will get help on, you guessed it, the get help command.

It will display different ways to get help.

```
PS /Users/maximus> get-help get-help

NAME
    Get-Help

SYNOPSIS
    Displays information about PowerShell commands and concepts.


SYNTAX
    Get-Help [[-Name] <System.String>] [-Category {Alias | Cmdlet | Provider | General | FAQ | Glossary | HelpFile | ScriptCommand | Function | Filter | ExternalScript |
System.String[]>] [<CommonParameters>]

    Get-Help [[-Name] <System.String>] [-Category {Alias | Cmdlet | Provider | General | FAQ | Glossary | HelpFile | ScriptCommand | Function | Filter | ExternalScript |
System.String[]>] [<CommonParameters>]

    Get-Help [[-Name] <System.String>] [-Category {Alias | Cmdlet | Provider | General | FAQ | Glossary | HelpFile | ScriptCommand | Function | Filter | ExternalScript |
stem.String[]>] [<CommonParameters>]

    Get-Help [[-Name] <System.String>] [-Category {Alias | Cmdlet | Provider | General | FAQ | Glossary | HelpFile | ScriptCommand | Function | Filter | ExternalScript |
stem.String[]>] [<CommonParameters>]

    Get-Help [[-Name] <System.String>] [-Category {Alias | Cmdlet | Provider | General | FAQ | Glossary | HelpFile | ScriptCommand | Function | Filter | ExternalScript |
m.String>] [-Role <System.String[]>]
    [<CommonParameters>]

    Get-Help [[-Name] <System.String>] [-Category {Alias | Cmdlet | Provider | General | FAQ | Glossary | HelpFile | ScriptCommand | Function | Filter | ExternalScript |
ing[]>] -ShowWindow [<CommonParameters>]


DESCRIPTION
    The `Get-Help` cmdlet displays information about PowerShell concepts and commands, including cmdlets, functions, Common Information Model (CIM) commands, workflows, p

    To get help for a PowerShell cmdlet, type `Get-Help` followed by the cmdlet name, such as: `Get-Help Get-Process`.

    Conceptual help articles in PowerShell begin with about_ , such as about_Comparison_Operators . To see all about_ articles, type `Get-Help about_*`. To see a particul

    To get help for a PowerShell provider, type `Get-Help` followed by the provider name. For example, to get help for the Certificate provider, type `Get-Help Certificat

    You can also type `help` or `man`, which displays one screen of text at a time. Or, `<cmdlet-name> -?`, that's identical to `Get-Help`, but only works for cmdlets.

    `Get-Help` gets the help content that it displays from help files on your computer. Without the help files, `Get-Help` displays only basic information about cmdlets.
the help files for a module in PowerShell 3.0, use
    the `Update-Help` cmdlet.

    You can also view the PowerShell help documents online. To get the online version of a help file, use the Online parameter, such as: `Get-Help Get-Process -Online`.

    If you type `Get-Help` followed by the exact name of a help article, or by a word unique to a help article, `Get-Help` displays the article's content. If you specify
elp` displays a list of the matching titles. If you
    enter any text that doesn't appear in any help article titles, `Get-Help` displays a list of articles that include that text in their contents.

    `Get-Help` can get help articles for all supported languages and locales. `Get-Help` first looks for help files in the locale set for Windows, then in the parent loca
ish, en-US , before it returns an error message or
    displaying autogenerated help.

    For information about the symbols that `Get-Help` displays in the command syntax diagram, see about_Command_Syntax (./About/about_Command_Syntax.md). For information

    > [!NOTE] > In PowerShell 3.0 and PowerShell 4.0, `Get-Help` can't find About articles in modules unless > the module is imported into the current session. To get Abo

    Beginning with PSReadLine v2.2.2, the module ships with two functions that provide quick access to help while you are typing a command on the command line. The help i

    When you hit the <kbd>F1</kbd> key, the PSReadLine `ShowCommandHelp` function invokes `Get-Help -Full` for the cmdlet name closest to the left of the cursor. When the
u are returned to the command line at the same cursor
    position so you can continue typing the command.

    When you use the key combination <kbd>Alt</kbd>+<kbd>h</kbd>, the PSReadLine `ShowParameterHelp` function displays help information for the parameter immediately to t

    For more information, see Using dynamic help (/powershell/scripting/learn/shell/dynamic-help).


RELATED LINKS
    Online Version: https://learn.microsoft.com/powershell/module/microsoft.powershell.core/get-help?view=powershell-7.3&WT.mc_id=ps-gethelp
    about_Command_Syntax
    about_Comment_Based_Help
    Get-Command
    Supporting Updatable Help
    Update-Help
    Writing Comment-Based Help Topics
    Writing Help for PowerShell Cmdlets

REMARKS
    To see the examples, type: "Get-Help Get-Help -Examples"
    For more information, type: "Get-Help Get-Help -Detailed"
    For technical information, type: "Get-Help Get-Help -Full"
    For online help, type: "Get-Help Get-Help -Online"
```

We can ask for more details (-detailed)

Get-help get-help -detailed



See the difference? When we ask it to return help on get-help with the param of -details we get more info.

-Gives Help with the added:
-Details on each parameter that we can use with the command.
-Details on common parameters used with the command.
-And much much more.

We can also ask for examples (-examples)
Get-help export-csv -examples

```
PS /Users/maximus> get-help export-csv -examples

NAME
    Export-Csv

SYNOPSIS
    Converts objects into a series of character-separated value (CSV) strings and saves the strings to a file.


    ------ Example 1: Export process properties to a CSV file ------

    Get-Process -Name WmiPrvSE |
        Select-Object -Property BasePriority,Id,SessionId,WorkingSet |
        Export-Csv -Path .\WmiData.csv -NoTypeInformation
    Import-Csv -Path .\WmiData.csv

    BasePriority Id    SessionId WorkingSet
    ------------ --    --------- ----------
    8            976   0         20267008
    8            2292  0         36786176
    8            3816  0         30351360
    8            8604  0         15011840
    8            10008 0         8830976
    8            11764 0         14237696
    8            54632 0         9502720

    The `Get-Process` cmdlet gets the Process objects. The Name parameter filters the output to include only the WmiPrvSE process objects. The process objects are sent down the pipeline to
    the `Select-Object` cmdlet. `Select-Object` uses the Property parameter to select a subset of process object properties. The process objects are sent down the pipeline to the `Export-Csv`
    cmdlet. `Export-Csv` converts the process objects to a series of CSV strings. The Path parameter specifies that the `WmiData.csv` file is saved in the current directory. The
    NoTypeInformation parameter removes the #TYPE information header from the CSV output and is not required in PowerShell 6. The `Import-Csv` cmdlet uses the Path parameter to display the
    file located in the current directory.
    ---- Example 2: Export processes to a comma-delimited file ----

    Get-Process | Export-Csv -Path .\Processes.csv -NoTypeInformation
    Get-Content -Path .\Processes.csv

    "Name","SI","Handles","VM","WS","PM","NPM","Path","Parent","Company","CPU","FileVersion", ...
    "ApplicationFrameHost","4","511","2203597099008","35364864","21979136","30048", ...

    The `Get-Process` cmdlet gets Process objects. The process objects are sent down the pipeline to the `Export-Csv` cmdlet. `Export-Csv` converts the process objects to a series of CSV
    strings. The Path parameter specifies that the `Processes.csv` file is saved in the current directory. The NoTypeInformation parameter removes the #TYPE information header from the CSV
    output and is not required in PowerShell 6. The `Get-Content` cmdlet uses the Path parameter to display the file located in the current directory.
    -- Example 3: Export processes to a semicolon delimited file --

    Get-Process | Export-Csv -Path .\Processes.csv -Delimiter ';' -NoTypeInformation
    Get-Content -Path .\Processes.csv

    "Name";"SI";"Handles";"VM";"WS";"PM";"NPM";"Path";"Parent";"Company";"CPU";"FileVersion"; ...
    "ApplicationFrameHost";"4";"509";"2203595321344";"34807808";"21770240";"29504"; ...

    The `Get-Process` cmdlet gets Process objects. The process objects are sent down the pipeline to the `Export-Csv` cmdlet. `Export-Csv` converts the process objects to a series of CSV
    strings. The Path parameter specifies that the `Processes.csv` file is saved in the current directory. The Delimiter parameter specifies a semicolon to separate the string values. The
    NoTypeInformation parameter removes the #TYPE information header from the CSV output and is not required in PowerShell 6. The `Get-Content` cmdlet uses the Path parameter to display the
    file located in the current directory.
    Example 4: Export processes using the current culture's list separator

    (Get-Culture).TextInfo.ListSeparator
    Get-Process | Export-Csv -Path .\Processes.csv -UseCulture -NoTypeInformation
    Get-Content -Path .\Processes.csv

    "Name","SI","Handles","VM","WS","PM","NPM","Path","Parent","Company","CPU","FileVersion", ...
    "ApplicationFrameHost","4","511","2203597099008","35364864","21979136","30048", ...

    The `Get-Culture` cmdlet uses the nested properties TextInfo and ListSeparator and displays the current culture's default list separator. The `Get-Process` cmdlet gets Process objects.
    The process objects are sent down the pipeline to the `Export-Csv` cmdlet. `Export-Csv` converts the process objects to a series of CSV strings. The Path parameter specifies that the
    `Processes.csv` file is saved in the current directory. The UseCulture parameter uses the current culture's default list separator as the delimiter. The NoTypeInformation parameter
    removes the #TYPE information header from the CSV output and is not required in PowerShell 6. The `Get-Content` cmdlet uses the Path parameter to display the file located in the current
    directory.
    ------ Example 5: Export processes with type information ------

    Get-Process | Export-Csv -Path .\Processes.csv -IncludeTypeInformation
    Get-Content -Path .\Processes.csv

    #TYPE System.Diagnostics.Process
    "Name","SI","Handles","VM","WS","PM","NPM","Path","Company","CPU","FileVersion", ...
    "ApplicationFrameHost","4","507","2203595001856","35139584","20934656","29504", ...

    The `Get-Process` cmdlet gets Process objects. The process objects are sent down the pipeline to the `Export-Csv` cmdlet. `Export-Csv` converts the process objects to a series of CSV
    strings. The Path parameter specifies that the `Processes.csv` file is saved in the current directory. The IncludeTypeInformation includes the #TYPE information header in the CSV output.
    The `Get-Content` cmdlet uses the Path parameter to display the file located in the current directory.
    ------ Example 6: Export and append objects to a CSV file ------

    $AppService = (Get-Service -DisplayName *Application* | Select-Object -Property DisplayName, Status)
    $AppService | Export-Csv -Path .\Services.csv -NoTypeInformation
    Get-Content -Path .\Services.Csv
    $WinService = (Get-Service -DisplayName *Windows* | Select-Object -Property DisplayName, Status)
    $WinService | Export-Csv -Path .\Services.csv -NoTypeInformation -Append
    Get-Content -Path .\Services.Csv
```

-Bypasses all the basic topic save Name and Synopsis.
-Gives only examples on the use of the command.

We can ask for full detail regarding each parameter use (-full)
Get-help export-csv -full

```
PS /Users/maximus> get-help export-csv -full

NAME
    Export-Csv

SYNOPSIS
    Converts objects into a series of character-separated value (CSV) strings and saves the strings to a file.


SYNTAX
    Export-Csv [[-Path] <System.String>] [[-Delimiter] <System.Char>] [-Append] [-Encoding {ASCII | BigEndianUnicode | BigEndianUTF32 | OEM | Unicode | UTF7 | UTF8 | UTF8BOM | UTF8NoBOM | UTF32}] [-Force]
    [-IncludeTypeInformation] -InputObject <System.Management.Automation.PSObject> [-LiteralPath <System.String>] [-NoClobber] [-NoTypeInformation] [-QuoteFields <System.String[]>] [-UseQuotes
    <Microsoft.PowerShell.Commands.BaseCsvWritingCommand+QuoteKind>] [-Confirm] [-WhatIf] [<CommonParameters>]

    Export-Csv [[-Path] <System.String>] [-Append] [-Encoding {ASCII | BigEndianUnicode | BigEndianUTF32 | OEM | Unicode | UTF7 | UTF8 | UTF8BOM | UTF8NoBOM | UTF32}] [-Force] [-IncludeTypeInformation] -InputObject
    <System.Management.Automation.PSObject> [-LiteralPath <System.String>] [-NoClobber] [-NoTypeInformation] [-QuoteFields <System.String[]>] [-UseCulture] [-UseQuotes
    <Microsoft.PowerShell.Commands.BaseCsvWritingCommand+QuoteKind>] [-Confirm] [-WhatIf] [<CommonParameters>]


DESCRIPTION
    The `Export-CSV` cmdlet creates a CSV file of the objects that you submit. Each object is a row that includes a character-separated list of the object's property values. You can use the `Export-CSV` cmdlet to create
    spreadsheets and share data with programs that accept CSV files as input.

    Do not format objects before sending them to the `Export-CSV` cmdlet. If `Export-CSV` receives formatted objects the CSV file contains the format properties rather than the object properties. To export only selected
    properties of an object, use the `Select-Object` cmdlet.


PARAMETERS
    -Append <System.Management.Automation.SwitchParameter>
        Use this parameter so that `Export-CSV` adds CSV output to the end of the specified file. Without this parameter, `Export-CSV` replaces the file contents without warning.

        This parameter was introduced in Windows PowerShell 3.0.

        Required?                    false
        Position?                    named
        Default value                False
        Accept pipeline input?       False
        Accept wildcard characters?  false

    -Delimiter <System.Char>
        Specifies a delimiter to separate the property values. The default is a comma (`,`). Enter a character, such as a colon (`:`). To specify a semicolon (`;`), enclose it in quotation marks.

        Required?                    false
        Position?                    1
        Default value                comma (,)
        Accept pipeline input?       False
        Accept wildcard characters?  false

    -Encoding <System.Text.Encoding>
        Specifies the encoding for the exported CSV file. The default value is `utf8NoBOM`.

        The acceptable values for this parameter are as follows:

        - `ascii`: Uses the encoding for the ASCII (7-bit) character set.

        - `bigendianunicode`: Encodes in UTF-16 format using the big-endian byte order.

        - `bigendianutf32`: Encodes in UTF-32 format using the big-endian byte order.

        - `oem`: Uses the default encoding for MS-DOS and console programs.

        - `unicode`: Encodes in UTF-16 format using the little-endian byte order.

        - `utf7`: Encodes in UTF-7 format.

        - `utf8`: Encodes in UTF-8 format.

        - `utf8BOM`: Encodes in UTF-8 format with Byte Order Mark (BOM)

        - `utf8NoBOM`: Encodes in UTF-8 format without Byte Order Mark (BOM)

        - `utf32`: Encodes in UTF-32 format.


        Beginning with PowerShell 6.2, the Encoding parameter also allows numeric IDs of registered code pages (like `-Encoding 1251`) or string names of registered code pages (like `-Encoding "windows-1251"`). For more
        information, see the .NET documentation for Encoding.CodePage (/dotnet/api/system.text.encoding.codepage?view=netcore-2.2).
        > [!NOTE] > UTF-7 * is no longer recommended to use. As of PowerShell 7.1, a warning is written if you > specify `utf7` for the Encoding parameter.


        Required?                    false
        Position?                    named
        Default value                UTF8NoBOM
        Accept pipeline input?       False
        Accept wildcard characters?  false

    -Force <System.Management.Automation.SwitchParameter>
        This parameter allows `Export-Csv` to overwrite files with the Read Only attribute.

        When Force and Append parameters are combined, objects that contain mismatched properties can be written to a CSV file. Only the properties that match are written to the file. The mismatched properties are discarded.

        Required?                    false
        Position?                    named
        Default value                False
        Accept pipeline input?       False
        Accept wildcard characters?  false

    -IncludeTypeInformation <System.Management.Automation.SwitchParameter>
        When this parameter is used the first line of the CSV output contains `#TYPE` followed by the fully qualified name of the object type. For example, `#TYPE System.Diagnostics.Process`.

        This parameter was introduced in PowerShell 6.0.

        Required?                    false
        Position?                    named
        Default value                False
        Accept pipeline input?       False
        Accept wildcard characters?  false

    -InputObject <System.Management.Automation.PSObject>
        Specifies the objects to export as CSV strings. Enter a variable that contains the objects or type a command or expression that gets the objects. You can also pipe objects to `Export-CSV`.

        Required?                    true
```

-Gives basic help topics.
-Gives full details regarding each parameter and its use.
-Gives examples using the command.
-Difference between full and detailed are subtle.

Easier / Quicker Help

Help is an alias to get-help

Get-help *help*

```
PS /Users/maximus> get-help *help*

Name                              Category  Module                    Synopsis
----                              --------  ------                    --------
help                              Function
Get-Help                          Cmdlet    Microsoft.PowerShell.Core Displays information about PowerShell commands and concepts.
Save-Help                         Cmdlet    Microsoft.PowerShell.Core Downloads and saves the newest help files to a file system directory.
Update-Help                       Cmdlet    Microsoft.PowerShell.Core Downloads and installs the newest help files on your computer.
about_Updatable_Help              HelpFile
about_Comment_Based_Help          HelpFile
```

See how when we ask for help on the wild card for all help, it returns this list whereas the 1st entry is a function, thats the alias to get-help.

- It's easier and quicker to type
- Pauses ever screen in the console
- Uses like get-help

Help –Detailed Get-ChildItem

Is identical to

Get-help –Detailed Get-ChildItem

-In fact Help runs the Get-Help command but pauses the screen.

-No difference to the end user except it pauses so you can read the document.

-In the ISE environment there is no difference in the output of the commands, both scroll to the end of the document.

Updating Help

You can update help using:

Update-help

- Help system was modularized in PowerShell 3.0 to assure end user they had the newest help topics

- Windows PowerShell 3.0 does not come with help files

- Use the updatable help to install help

- If help files for a cmdlet, function or workflow are not installed on the computer, Get-Help displays auto generated help

-Prompts you to download the help files or read online as well

-Auto-generated help includes syntax and aliases, and remarks that explain how to use the Updatable Help cmdlets and to access the online help topics.

Saving Help

You can save help files to any location locally on your machine using:

Save-help

- Downloads the newest help files and saves them to a location you specify

- Allows you to update help on:

- Computers with no internet access

- Multiple computers

- Works just like Update-Help, except that it saves the downloaded cabinet (.cab) files, instead of extracting the help files from the .cab files

Installing without Internet

You can install help on computers with no internet access

To install help on computers with no internet access:

Use Save-Help –DestinationPath <Location to Save> To download the files to a shared folder in your network

Use

Update-Help –SourcePath <Files Location> To update help on any computer or multiple computers