Full credit is given to the above companies including the OS that this PDF file was generated!

PowerShell Get-Help on command 'Sort-Object'

PS C:\Users\wahid> Get-Help Sort-Object

NAME

Sort-Object

SYNOPSIS

Sorts objects by property values.

SYNTAX

Sort-Object [[-Property] <System.Object[]>] [-CaseSensitive] [-Culture <System.String>] [-Descending] [-InputObject <System.Management.Automation.PSObject>] [-Unique] [<CommonParameters>]

DESCRIPTION

The `Sort-Object` cmdlet sorts objects in ascending or descending order based on object property values. If sort properties aren't included in a command, PowerShell uses default sort properties of the first input object. If the input object's type has no default sort properties, PowerShell attempts to compare the objects themselves. For more information, see the Notes (#notes)section.

You can sort objects by a single property or multiple properties. Multiple properties use hash tables to sort in ascending order, descending order, or a combination of sort orders. Properties are sorted as case-sensitive or case-insensitive. Use the Unique parameter to remove duplicates from the output.

PARAMETERS

-CaseSensitive <System.Management.Automation.SwitchParameter>

Indicates that the sort is case-sensitive. By default, sorts aren't case-sensitive.

-Culture <System.String>

Specifies the cultural configuration to use for sorts. Use `Get-Culture` to display the system's culture configuration.

-Descending <System.Management.Automation.SwitchParameter>

Indicates that `Sort-Object` sorts the objects in descending order. The default is ascending order.

To sort multiple properties with different sort orders, use a hash table. For example, with a hash table you can sort one property in ascending order and another property in descending order.

-InputObject <System.Management.Automation.PSObject>

To sort objects, send them down the pipeline to `Sort-Object`. If you use the InputObject parameter to submit a collection of items, `Sort-Object` receives one object that represents the collection. Because one object can't be sorted, `Sort-Object` returns the entire collection unchanged.

-Property <System.Object[]>

Specifies the property names that `Sort-Object` uses to sort the objects. Wildcards are permitted. Objects are sorted based on the property values.

If you don't specify a property, `Sort-Object` sorts based on default properties for the object type or the objects themselves.

Use commas to separate multiple properties. Multiple properties can be sorted in ascending order, descending order, or a combination of sort orders. When you specify multiple properties, the objects are sorted by the first property. If multiple objects have the same value for the first property, those objects are sorted by the second property. This process continues until there are no more specified properties or no groups of objects.

The Property parameter's value can be a calculated property. To create a calculated property, use a scriptblock or a hashtable.

Valid keys for a hash table are as follows:

- `expression` - `<string>` or `<script block>`

- `ascending` or `descending` - `<boolean>`

For more information, see about_Calculated_Properties (../Microsoft.PowerShell.Core/About/about_Calculated_Properties.md).

-Unique <System.Management.Automation.SwitchParameter>
Indicates that `Sort-Object` eliminates duplicates and returns only the unique members of the collection. The first instance of a unique value is included in the sorted output. Unique is case-insensitive. Strings that only differ by character case are considered the same. For example, character and CHARACTER.

<CommonParameters>

This cmdlet supports the common parameters: Verbose, Debug, ErrorAction, ErrorVariable, WarningAction, WarningVariable, OutBuffer, PipelineVariable, and OutVariable. For more information, see about_CommonParameters (https:/go.microsoft.com/fwlink/?LinkID=113216).

-------- Example 1: Sort the current directory by name --------

```
Get-ChildItem -Path C:\Test | Sort-Object
```

```
Directory: C:\Test

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         2/13/2019     08:55             26 anotherfile.txt
-a----         2/13/2019     13:26             20 Bfile.txt
-a----         2/12/2019     15:40         118014 Command.txt
-a----          2/1/2019     08:43            183 CreateTestFile.ps1
d-----         2/25/2019     18:25                Files
d-----         2/25/2019     18:24                Logs
-ar---         2/12/2019     14:31             27 ReadOnlyFile.txt
-a----         2/12/2019     16:24             23 Zsystemlog.log
```

The `Get-ChildItem` cmdlet gets the files and subdirectories from the directory specified by the Path parameter, `C:\Test`. The objects are sent down the pipeline to the `Sort-Object` cmdlet. `Sort-Object` doesn't specify a property so the output is sorted by the default sort property, Name .

----- Example 2: Sort the current directory by file length -----

```
Get-ChildItem -Path C:\Test -File | Sort-Object -Property Length
```

```
Directory: C:\Test

Mode                 LastWriteTime         Length Name
```

```
----                 -------------         ------ ----
-a----         2/13/2019     13:26             20 Bfile.txt
-a----         2/12/2019     16:24             23 Zsystemlog.log
-a----         2/13/2019     08:55             26 anotherfile.txt
-ar---         2/12/2019     14:31             27 ReadOnlyFile.txt
-a----          2/1/2019     08:43            183 CreateTestFile.ps1
-a----         2/12/2019     15:40         118014 Command.txt
```

The `Get-ChildItem` cmdlet gets the files from the directory specified by the Path parameter. The File parameter specifies that `Get-ChildItem` only gets file objects. The objects are sent down the pipeline to the `Sort-Object` cmdlet. `Sort-Object` uses the Length parameter to sort the files by length in ascending order.

---------- Example 3: Sort processes by memory usage ----------

```
Get-Process | Sort-Object -Property WS | Select-Object -Last 5
```

```
NPM(K)    PM(M)      WS(M)     CPU(s)      Id  SI ProcessName
 ------    -----      -----     ------      --  -- -----------
    136   193.92     217.11     889.16   87492   8 OUTLOOK
    112   347.73     297.02      95.19  106908   8 Teams
    206   266.54     323.71      37.17   60620   8 MicrosoftEdgeCP
     35   552.19     549.94     131.66    6552   8 Code
      0     1.43     595.12       0.00    2780   0 Memory Compression
```

The `Get-Process` cmdlet gets the list of processes running on the computer. The process objects are sent down the pipeline to the `Sort-Object` cmdlet. `Sort-Object` uses the Property parameter to sort the objects by WS . The objects are sent down the pipeline to the `Select-Object` cmdlet. `Select-Object` uses the Last parameter to specify the last five objects, which are the objects with the highest WS usage.

---------- Example 4: Sort HistoryInfo objects by Id ----------

```
Get-History | Sort-Object -Property Id -Descending

Id CommandLine
 -- -----------
 10 Get-Command Sort-Object -Syntax
  9 $PSVersionTable
  8 Get-Command Sort-Object -Syntax
  7 Get-Command Sort-Object -ShowCommandInfo
  6 Get-ChildItem -Path C:\Test | Sort-Object -Property Length
  5 Get-Help Clear-History -online
  4 Get-Help Clear-History -full
  3 Get-ChildItem | Get-Member
  2 Get-Command Sort-Object -Syntax
  1 Set-Location C:\Test\
```

The `Get-History` cmdlet gets the history objects from the current PowerShell session. The objects are sent down the pipeline to the `Sort-Object` cmdlet. `Sort-Object` uses the Property parameter to sort the objects by Id . The Descending parameter sorts the command history from newest to oldest.

Example 5: Use a hash table to sort properties in ascending and descending order

```
Get-Service |
    Sort-Object -Property @{Expression = "Status"; Descending = $true},
                          @{Expression = "DisplayName"; Descending = $false}

Status   Name               DisplayName
------   ----               -----------
Running  Appinfo            Application Information
Running  BthAvctpSvc        AVCTP service
Running  BrokerInfrastru... Background Tasks Infrastructure Ser...
Running  BDESVC             BitLocker Drive Encryption Service
Running  CoreMessagingRe... CoreMessaging
```

```
Running  VaultSvc          Credential Manager
Running  DsSvc             Data Sharing Service
Running  Dhcp              DHCP Client
...
Stopped  ALG               Application Layer Gateway Service
Stopped  AppMgmt           Application Management
Stopped  BITS              Background Intelligent Transfer Ser...
Stopped  wbengine          Block Level Backup Engine Service
Stopped  BluetoothUserSe... Bluetooth User Support Service_14fb...
Stopped  COMSysApp         COM+ System Application
Stopped  smstsmgr          ConfigMgr Task Sequence Agent
Stopped  DeviceInstall     Device Install Service
Stopped  MSDTC             Distributed Transaction Coordinator
```

The `Get-Service` cmdlet gets the list of services on the computer. The service objects are sent down the pipeline to the `Sort-Object` cmdlet. `Sort-Object` uses the Property parameter with a hash table to specify the property names and sort orders. The Property parameter is sorted by two properties, Status in descending order and DisplayName in ascending order. Status is an enumerated property. Stopped has a value of 1 and Running has a value of 4 . The Descending parameter is set to `$True` so that Running processes are displayed before Stopped processes. DisplayName sets the Descending parameter to `$False` to sort the display names in alphabetical order.

---------- Example 6: Sort text files by time span ----------

```
Get-ChildItem -Path C:\Test\*.txt |
    Sort-Object -Property {$_.CreationTime - $_.LastWriteTime} |
    Format-Table CreationTime, LastWriteTime, FullName

CreationTime         LastWriteTime        FullName
------------         -------------        --------
11/21/2018 12:39:01   2/26/2019 08:59:36   C:\Test\test2.txt
```

12/4/2018 08:29:41 2/26/2019 08:57:05 C:\Test\powershell_list.txt

2/20/2019 08:15:59 2/26/2019 12:09:43 C:\Test\CreateTestFile.txt

2/20/2019 08:15:59 2/26/2019 12:07:41 C:\Test\Command.txt

2/20/2019 08:15:59 2/26/2019 08:57:52 C:\Test\ReadOnlyFile.txt

11/29/2018 15:16:50 12/4/2018 16:16:24 C:\Test\LogData.txt

2/25/2019 18:25:11 2/26/2019 12:08:47 C:\Test\Zsystemlog.txt

2/25/2019 18:25:11 2/26/2019 08:55:33 C:\Test\Bfile.txt

2/26/2019 08:46:59 2/26/2019 12:12:19 C:\Test\LogFile3.txt

The `Get-ChildItem` cmdlet uses the Path parameter to specify the directory `C:\Test` and all of the `*.txt` files. The objects are sent down the pipeline to the `Sort-Object` cmdlet. `Sort-Object` uses the Property parameter with a scriptblock to determine each files time span between CreationTime and LastWriteTime .

---------- Example 7: Sort names in a text file ----------

```
# All items unsorted
Get-Content -Path C:\Test\ServerNames.txt

localhost
server01
server25
LOCALHOST
Server19
server3
localhost

# All items sorted
Get-Content -Path C:\Test\ServerNames.txt | Sort-Object
localhost
LOCALHOST
localhost
server01
```

Server19

server25

server3

Unique filtered items sorted

Get-Content -Path C:\Test\ServerNames.txt | Sort-Object -Unique

localhost

server01

Server19

server25

server3

The `Get-Content` cmdlet uses the Path parameter to specify the directory and filename. The file `ServerNames.txt` contains an unsorted list of computer names.

The `Get-Content` cmdlet uses the Path parameter to specify the directory and filename. The file `ServerNames.txt` contains an unsorted list of computer names. The objects are sent down the pipeline to the `Sort-Object` cmdlet. `Sort-Object` sorts the list in the default order, ascending.

The `Get-Content` cmdlet uses the Path parameter to specify the directory and filename. The file `ServerNames.txt` contains an unsorted list of computer names. The objects are sent down the pipeline to the `Sort-Object` cmdlet. `Sort-Object` uses the Unique parameter to remove duplicate computer names. The list is sorted in the default order, ascending.

---------- Example 8: Sort a string as an integer ----------

String sorted

Get-Content -Path C:\Test\ProductId.txt | Sort-Object

```
1
12345
1500
2
2800
3500
4100
500
6200
77
88
99999

# Integer sorted
Get-Content -Path C:\Test\ProductId.txt | Sort-Object {[int]$_}

0
1
2
77
88
500
1500
2800
3500
4100
6200
12345
99999
```

In the second example, `Get-Content` gets the contents of the file and pipes lines to the `Sort-Object` cmdlet. `Sort-Object` uses a script block to convert the strings to integers. In the sample code, `[int]` converts the

string to an integer and `$_` represents each string as it comes down the pipeline. The integer objects are sent down the pipeline to the `Sort-Object` cmdlet. `Sort-Object` sorts the integer objects in numeric order.

---------- Example 9: Sort by multiple properties ----------

```
Get-ChildItem -Path C:\Test | Sort-Object Length,Name
```

```
Directory: C:\Test

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a---          13/10/2021    22:16              2 File01.txt
-a---          13/10/2021    22:16              2 File03.txt
-a---          13/10/2021    22:18             64 File02.txt
-a---          13/10/2021    22:18             64 File04.txt
```

The `Get-ChildItem` cmdlet gets the files from the directory specified by the Path parameter. The objects are sent down the pipeline to the `Sort-Object` cmdlet. `Sort-Object` uses the Length and Name parameter to sort the files by length in ascending order. Since `File01.txt` and `File03.txt` have the same length, they're further sorted by their property Name .

Example 10: Sort hashtables by their key values with calculated properties

```
@(
    @{ name = 'a' ; weight = 7 }
    @{ name = 'b' ; weight = 1 }
    @{ name = 'c' ; weight = 3 }
    @{ name = 'd' ; weight = 7 }
) | Sort-Object -Property { $_.weight }, { $_.name } -OutVariable Sorted

$Sorted | ForEach-Object -Process { "{0}: {1}" -f $_.name, $_.weight }
```

The `{ $ .weight }` and `{ $ .name }` expressions sort the input hashtables

first by the value of their `weight` key and then by the value of their `name` key. The `Sort-Object` command uses the OutVariable parameter to store the result to a variable and display the result in the same call.

The last command iterates over the sorted hashtables to display their name and weight as a string.

REMARKS

To see the examples, type: "get-help Sort-Object -examples".

For more information, type: "get-help Sort-Object -detailed".

For technical information, type: "get-help Sort-Object -full".

For online help, type: "get-help Sort-Object -online"