

PowerShell

Snippets/Scripts

- [Function](#)
- [If-Else Statement](#)
- [Directory Tree Size Script](#)
- [PSObject](#)
- [Folder Directory to ISO file Script](#)

Function

Create a function and return the value

```
function CombineString () {  
    [CmdletBinding()]  
    param(  
        [Parameter(Mandatory = $true)] $stringIn1,  
        [Parameter(Mandatory = $true)] $stringIn2  
    )  
  
    $newStr = $stringIn1 + " " + $stringIn2  
  
    # To return the value, simply print it  
    $newStr  
}  
  
$theNewString = CombineString -stringIn1 "Hello" -stringIn2 "World"  
Write-Host "The new string is $theNewString"
```

If-Else Statement

```
if ($computertype -eq "64bit"){  
    $result = "64bit"  
} elseif ($computertype -eq "32bit") {  
    $result = "32bit"  
} elseif ($computertype -eq "all") {  
    $result = "all"  
} else {  
    $result = "none"  
}
```

Directory Tree Size Script

Code:

```
Function Get-DirectoryTreeSize {
```

```
<#
```

```
.SYNOPSIS
```

This is used to get the file count, subdirectory count and folder size for the path specified. The output will show the current folder stats unless you specify the "AllItemsAndAllFolders" property.

Since this uses Get-ChildItem as the underlying structure, this supports local paths, network UNC paths and mapped drives.

```
.NOTES
```

Name: Get-DirectoryTreeSize

Author: theSysadminChannel

Version: 1.0

DateCreated: 2020-Feb-11

```
.LINK
```

<https://thesysadminchannel.com/get-directory-tree-size-using-powershell> -

```
.PARAMETER Recurse
```

Using this parameter will drill down to the end of the folder structure and output the filecount, foldercount and size of each folder respectively.

```
.PARAMETER AllItemsAndAllFolders
```

Using this parameter will get the total file count, total directory count and total folder size in MB for everything under that directory recursively.

```
.EXAMPLE
```

```
Get-DirectoryTreeSize "C:\Some\Folder"
```

Path	FileCount	DirectoryCount	FolderSizeInMB
------	-----------	----------------	----------------

----	-----	-----	-----
------	-------	-------	-------

C:\Some\folder	3	3	0.002
----------------	---	---	-------

.EXAMPLE

```
Get-DirectoryTreeSize "\\MyServer\Folder" -Recurse
```

Path	FileCount	DirectoryCount	FolderSizeInMB
----	-----	-----	-----
\\MyServer\Folder	2	1	40.082
.\Subfolder	1	0	26.555

.EXAMPLE

```
Get-DirectoryTreeSize "Z:\MyMapped\folder" -AllItemsAndAllFolders
```

Path	TotalFileCount	TotalDirectoryCount	TotalFolderSizeInMB
----	-----	-----	-----
Z:\MyMapped\folder	3	1	68.492

#>

```
[CmdletBinding(DefaultParameterSetName="Default")]
```

```
param(
```

```
    [Parameter(
```

```
        Position = 0,
```

```
        Mandatory = $true
```

```
    )]
```

```
    [string] $Path,
```

```
    [Parameter(
```

```
        Mandatory = $false,
```

```
        ParameterSetName = "ShowRecursive"
```

```
    )]
```

```
    [switch] $Recurse,
```

```
    [Parameter(
```

```
        Mandatory = $false,
```

```
        ParameterSetName = "ShowTopFolderAllItemsAndAllFolders"
```

```

    ])
    [switch] $AllItemsAndAllFolders
)

BEGIN {
    #Adding a trailing slash at the end of $path to make it consistent.
    if (-not $Path.EndsWith('\')) {
        $Path = "$Path\"
    }
}

PROCESS {
    try {
        if (-not $PSBoundParameters.ContainsKey("AllItemsAndAllFolders") -and -not
$PSBoundParameters.ContainsKey("Recurse")) {
            $FileStats = Get-ChildItem -Path $Path -File -ErrorAction Stop | Measure-Object -Property Length -Sum
            $FileCount = $FileStats.Count
            $DirectoryCount = Get-ChildItem -Path $Path -Directory | Measure-Object | select -ExpandProperty
Count
            $SizeMB = "{0:F3}" -f ($FileStats.Sum / 1MB) -as [decimal]

            [PSCustomObject]@{
                Path          = $Path#.Replace($Path, ".\")
                FileCount      = $FileCount
                DirectoryCount  = $DirectoryCount
                FolderSizeInMB = $SizeMB
            }
        }

        if ($PSBoundParameters.ContainsKey("AllItemsAndAllFolders")) {
            $FileStats = Get-ChildItem -Path $Path -File -Recurse -ErrorAction Stop | Measure-Object -Property
Length -Sum
            $FileCount = $FileStats.Count
            $DirectoryCount = Get-ChildItem -Path $Path -Directory -Recurse | Measure-Object | select -
ExpandProperty Count
            $SizeMB = "{0:F3}" -f ($FileStats.Sum / 1MB) -as [decimal]

            [PSCustomObject]@{
                Path          = $Path#.Replace($Path, ".\")
                TotalFileCount = $FileCount
            }
        }
    }
}

```

```

        TotalDirectoryCount = $DirectoryCount
        TotalFolderSizeInMB = $SizeMB
    }
}

if ($PSBoundParameters.ContainsKey("Recurse")) {
    Get-DirectoryTreeSize -Path $Path
    $FolderList = Get-ChildItem -Path $Path -Directory -Recurse | select -ExpandProperty FullName

    if ($FolderList) {
        foreach ($Folder in $FolderList) {
            $FileStats = Get-ChildItem -Path $Folder -File | Measure-Object -Property Length -Sum
            $FileCount = $FileStats.Count
            $DirectoryCount = Get-ChildItem -Path $Folder -Directory | Measure-Object | select -
ExpandProperty Count
            $SizeMB = "{0:F3}" -f ($FileStats.Sum / 1MB) -as [decimal]

            [PSCustomObject]@{
                Path          = $Folder.Replace($Path, ".\")
                FileCount      = $FileCount
                DirectoryCount = $DirectoryCount
                FolderSizeInMB = $SizeMB
            }
            #clearing variables
            $null = $FileStats
            $null = $FileCount
            $null = $DirectoryCount
            $null = $SizeMB
        }
    }
} catch {
    Write-Error $_.Exception.Message
}

}

END {}
}

```

Get-DirectoryTreeSize "C:\Temp" -Recurse

PSObject

Code:

```
# Dummy Data
$name = "John Doe"
$age = 29
$gender = "Male"
$desc = "Nameless Person"

# Create an object with the given properties
$newPSObject = New-Object -TypeName PSObject
$newPSObject | Add-Member -MemberType NoteProperty -Name Name -Value $name
$newPSObject | Add-Member -MemberType NoteProperty -Name Age -Value $age
$newPSObject | Add-Member -MemberType NoteProperty -Name Gender -Value $gender
$newPSObject | Add-Member -MemberType NoteProperty -Name Description -Value $desc
```

Result:

```
PS C:\> $newPSObject
```

Name	Age	Gender	Description
John Doe	29	Male	Nameless Person

Folder Directory to ISO file Script

Change the following:

Line 80 -> Source Directory

Line 81 -> Output File

```
function New-IsoFile
{
    [CmdletBinding(DefaultParameterSetName='Source')]Param(
        [parameter(Position=1,Mandatory=$true,ValueFromPipeline=$true, ParameterSetName='Source')]$Source,
        [parameter(Position=2)][string]$Path = "$env:temp\$((Get-Date).ToString('yyyyMMdd-HH:mm:ss.ffff')).iso",
        [ValidateScript({Test-Path -LiteralPath $_ -PathType Leaf})][string]$BootFile = $null,

[ValidateSet('CDR','CDRW','DVD-RAM','DVD-PLUS','DVD-PLUSRW','DVD-PLUSRW_DUAL-LAYER','DVD-DASH-R','DVD-DASH-
RW','DVD-DASH-R_DUAL-LAYER','DISK','DVD-PLUSRW_DUAL-LAYER','BDR','BD-RE')][string] $Media =
'DVD-PLUSRW_DUAL-LAYER',

        [string]$Title = (Get-Date).ToString("yyyyMMdd-HH:mm:ss.ffff"),
        [switch]$Force,
        [parameter(ParameterSetName='Clipboard')][switch]$FromClipboard
    )

    Begin {
        ($cp = new-object System.CodeDom.Compiler.CompilerParameters).CompilerOptions = '/unsafe'
        if (!('ISOFile' -as [type])) {
            Add-Type -CompilerParameters $cp -TypeDefinition @"

public class ISOFile
{
    public unsafe static void Create(string Path, object Stream, int BlockSize, int TotalBlocks)
    {
        int bytes = 0;
        byte[] buf = new byte[BlockSize];
        var ptr = (System.IntPtr)(&bytes);
        var o = System.IO.File.OpenWrite(Path);
        var i = Stream as System.Runtime.InteropServices.ComTypes.IStream;
```

```

if (o != null) {
    while (TotalBlocks-- > 0) {
        i.Read(buf, BlockSize, ptr); o.Write(buf, 0, bytes);
    }
    o.Flush(); o.Close();
}
}
}
'@
}

if ($BootFile) {
    if('BDR','BDRE' -contains $Media) { Write-Warning "Bootable image doesn't seem to work with media type
$Media" }

    ($Stream = New-Object -ComObject ADODB.Stream -Property @{Type=1}).Open() # adFileTypeBinary
    $Stream.LoadFromFile((Get-Item -LiteralPath $BootFile).FullName)
    ($Boot = New-Object -ComObject IMAPI2FS.BootOptions).AssignBootImage($Stream)
}

$MediaType =
@('UNKNOWN','CDROM','CDR','CDRW','DVDROM','DVDRAM','DVDPLUSR','DVDPLUSRW','DVDPLUSR_DUALAYER','
DVDDASHR','DVDDASHRW','DVDDASHR_DUALAYER','DISK','DVDPLUSRW_DUALAYER','HDDVDROM','HDDVDR','
HDDVDROM','BDROM','BDR','BDRE')

Write-Verbose -Message "Selected media type is $Media with value $($MediaType.IndexOf($Media))"
($Image = New-Object -com IMAPI2FS.MsftFileSystemImage -Property
@{VolumeName=$Title}).ChooseImageDefaultsForMediaType($MediaType.IndexOf($Media))

if (!$Target = New-Item -Path $Path -ItemType File -Force:$Force -ErrorAction SilentlyContinue) { Write-Error
-Message "Cannot create file $Path. Use -Force parameter to overwrite if the target file already exists."; break }
}

Process {
    if($FromClipboard) {
        if($PSVersionTable.PSVersion.Major -lt 5) { Write-Error -Message 'The -FromClipboard parameter is only
supported on PowerShell v5 or higher'; break }
        $Source = Get-Clipboard -Format FileDropList
    }
}

```

```

foreach($item in $Source) {
    if($item -isnot [System.IO.FileInfo] -and $item -isnot [System.IO.DirectoryInfo]) {
        $item = Get-Item -LiteralPath $item
    }

    if($item) {
        Write-Verbose -Message "Adding item to the target image: $($item.FullName)"
        try { $Image.Root.AddTree($item.FullName, $true) } catch { Write-Error -Message
($_.Exception.Message.Trim() + ' Try a different media type.') }
    }
}

End {
    if ($Boot) { $Image.BootImageOptions=$Boot }
    $Result = $Image.CreateResultImage()
    [ISOFile]::Create($Target.FullName,$Result.ImageStream,$Result.BlockSize,$Result.TotalBlocks)
    Write-Verbose -Message "Target image ($($Target.FullName)) has been created"
    $Target
}
}

$source_dir = "C:\temp\isodir" # change this
get-childitem "$source_dir" | New-ISOFile -path C:\temp\isodir.iso

```