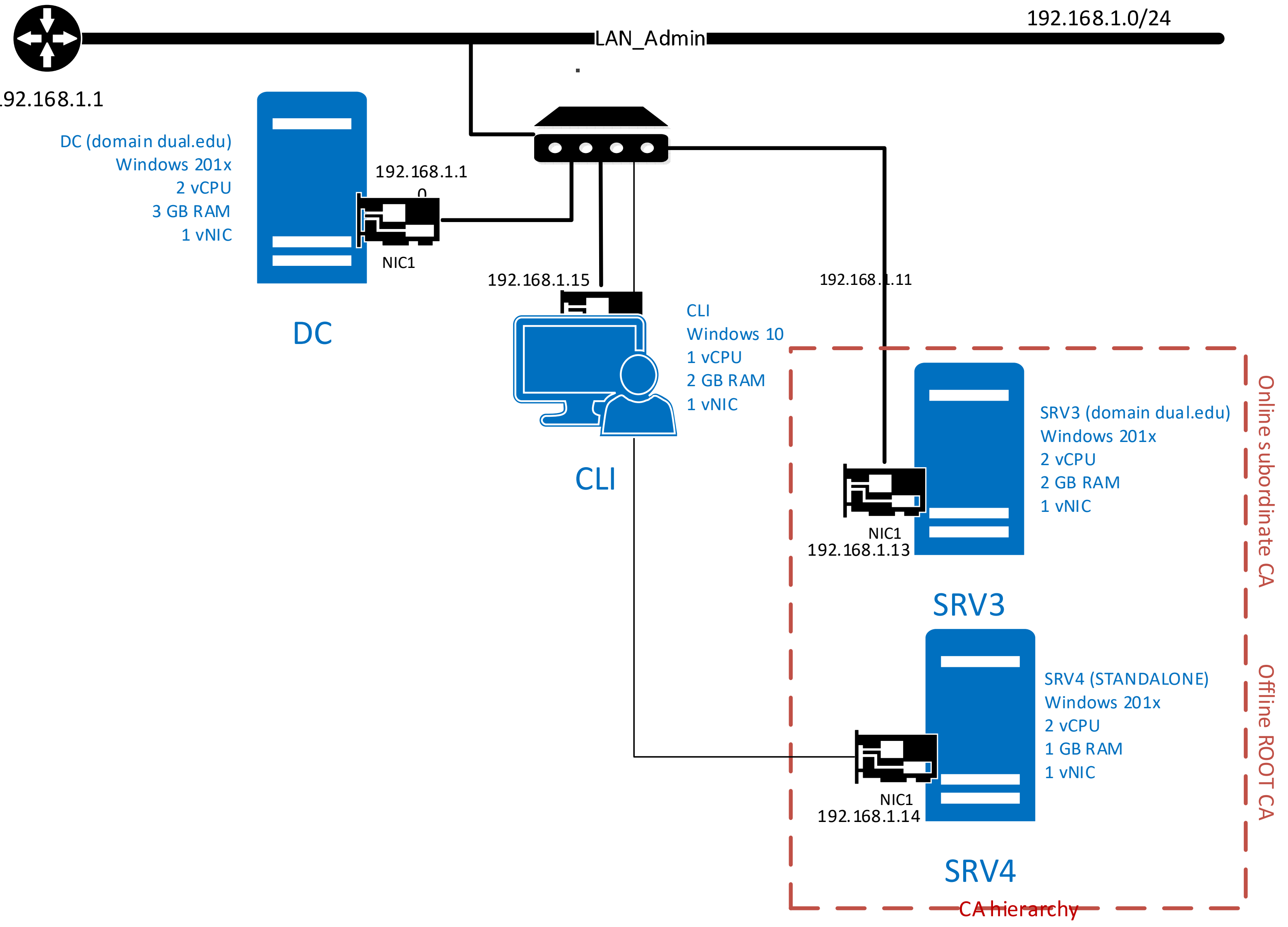
PowerShell pt2

Špecializované IKT systémy Windows

Ing. Stanislav Lukac, PhD.

LAB: CA installation



Agenda

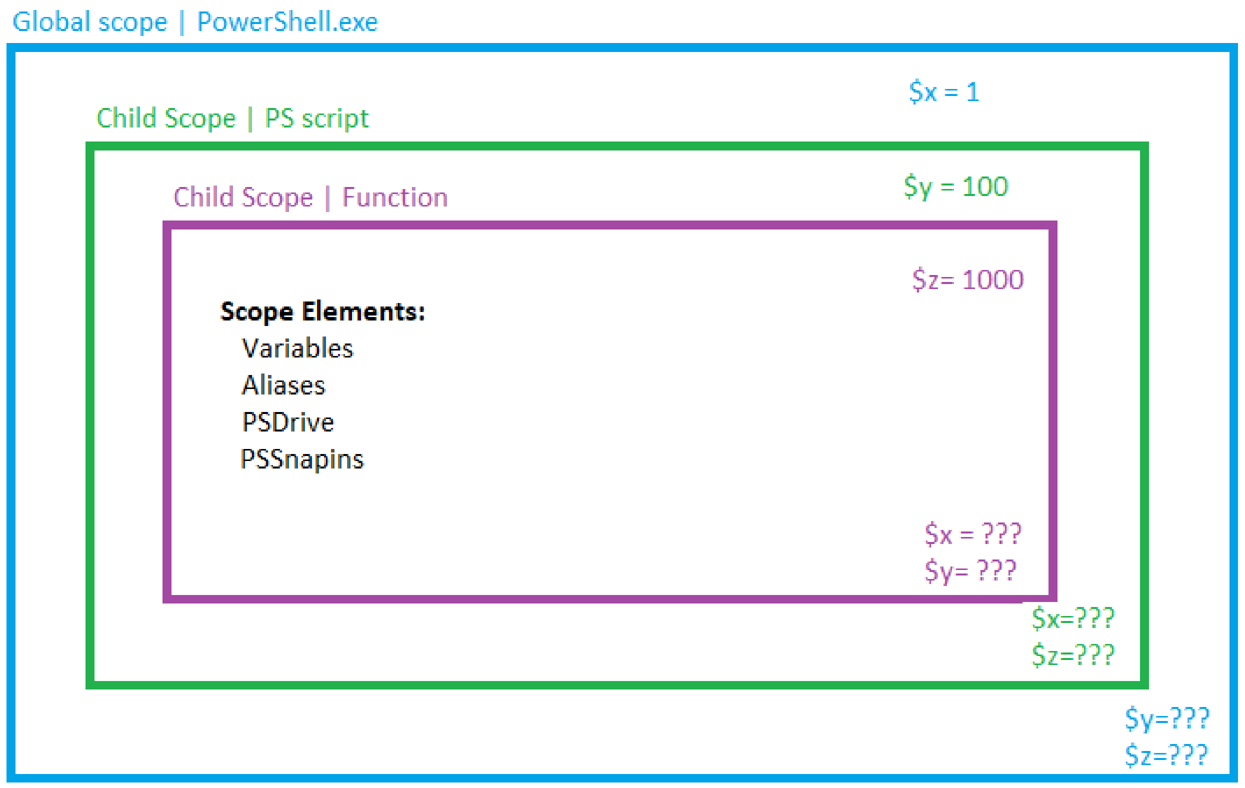
1. PowerShell Scopes
2. PowerShell Pipeline
3. PowerShell Formating
4. PowerShell Output
5. PowerShell Loops

PS Scope

Špecializované IKT systémy Windows

Ing. Stanislav Lukac, PhD.

M2 | SCOPEs



Scope Elements:
- Variables
- Aliases
- PSDrive
- PSSnapins

M2 | SCOPEs

- everything in PS is done in SCOPE
- scope elements are:
  - Variables
  - Aliases
  - PSDrives
  - PSSnapins
- scope identifiers by variable are:
  - Global:
  - Script:
  - Private:

M2 | SCOPE Rules

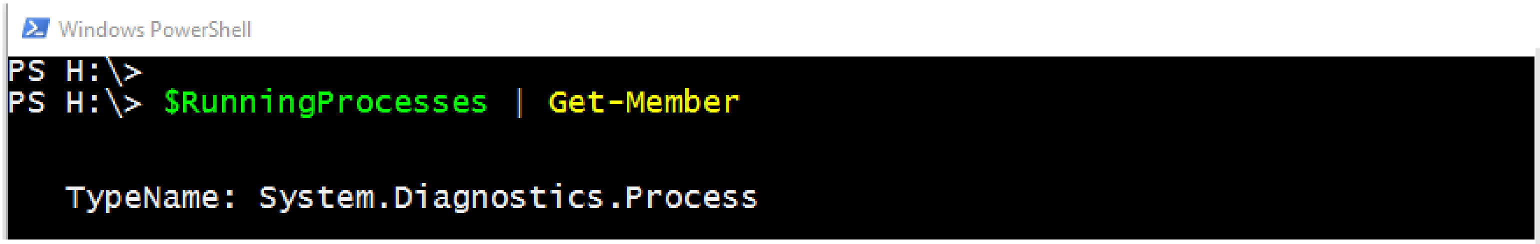
I. Child scope inherit settings from parent scope
II. When scope ends scope elements are gone
III. PowerhSell windows is global scope
IV. Parent not see to child scope

PS Object

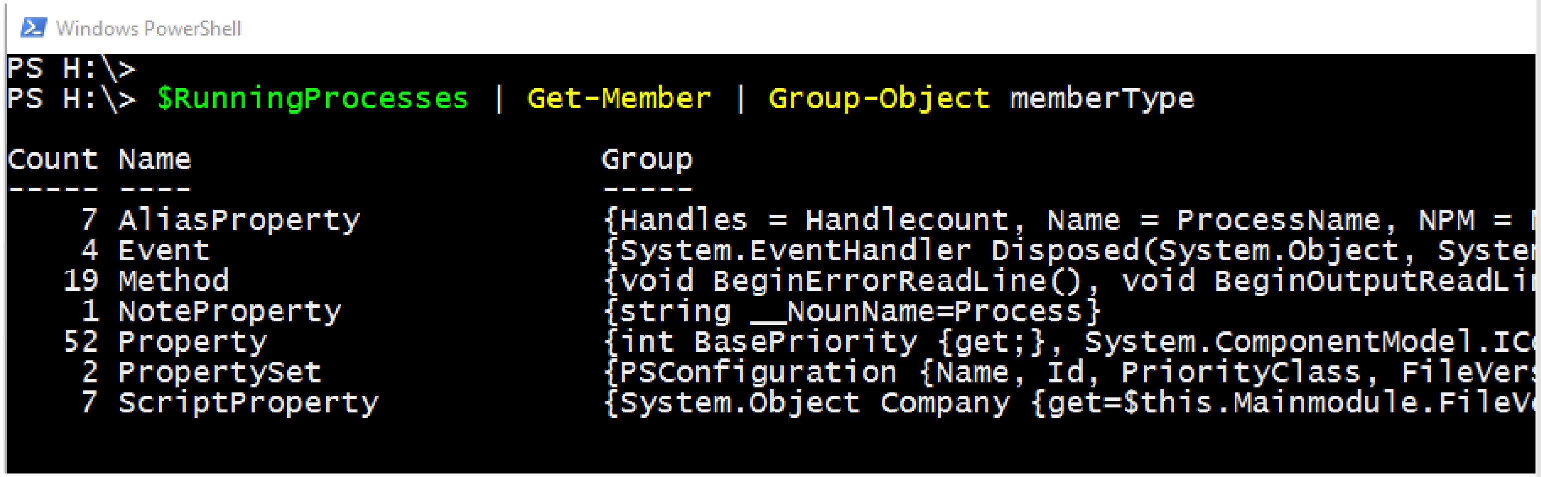
Špecializované IKT systémy Windows

Ing. Stanislav Lukac, PhD.

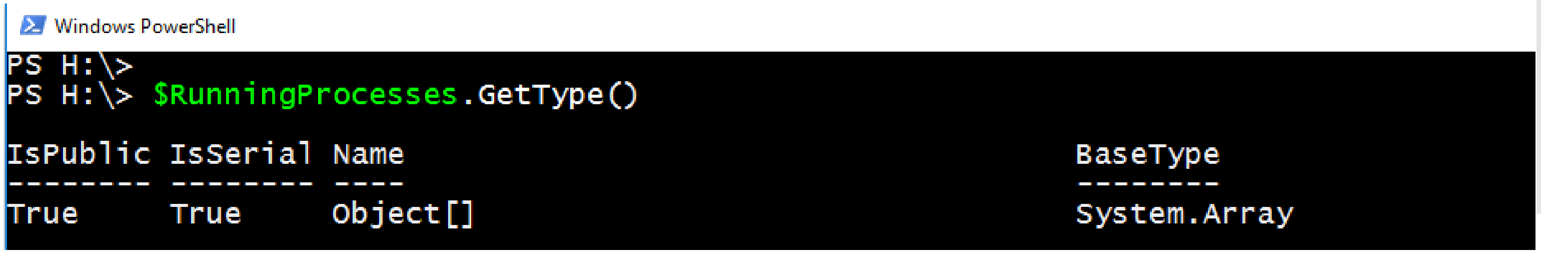
Get-Member


Windows PowerShell

```
PS H:\>
PS H:\> $RunningProcesses | Get-Member


   TypeName: System.Diagnostics.Process
```

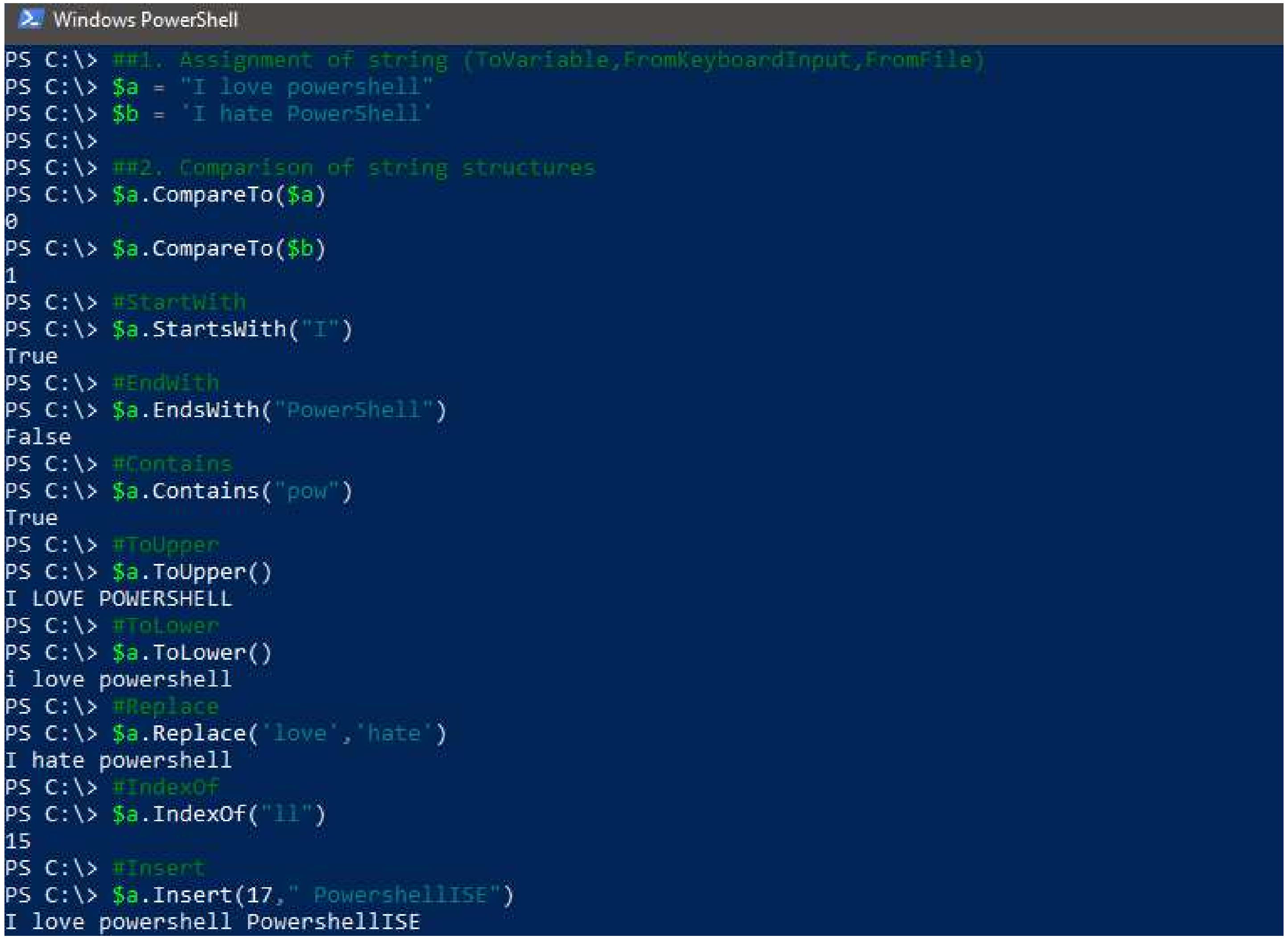

Windows PowerShell

```
PS H:\>
PS H:\> $RunningProcesses | Get-Member | Group-Object memberType

Count Name                      Group
----- ----                      -----
    7 AliasProperty             {Handles = Handlecount, Name = ProcessName, NPM = 
    4 Event                     {System.EventHandler Disposed(System.Object, Syste
   19 Method                    {void BeginErrorReadLine(), void BeginOutputReadLi
    1 NoteProperty              {string __NounName=Process}
   52 Property                  {int BasePriority {get;}, System.ComponentModel.IC
    2 PropertySet               {PSConfiguration {Name, Id, PriorityClass, FileVer
    7 ScriptProperty            {System.Object Company {get=$this.Mainmodule.FileV
```


Windows PowerShell

```
PS H:\>
PS H:\> $RunningProcesses.GetType()

IsPublic IsSerial Name                                     BaseType
-------- -------- ----                                     --------
True     True     Object[]                                 System.Array
```

DEMO::



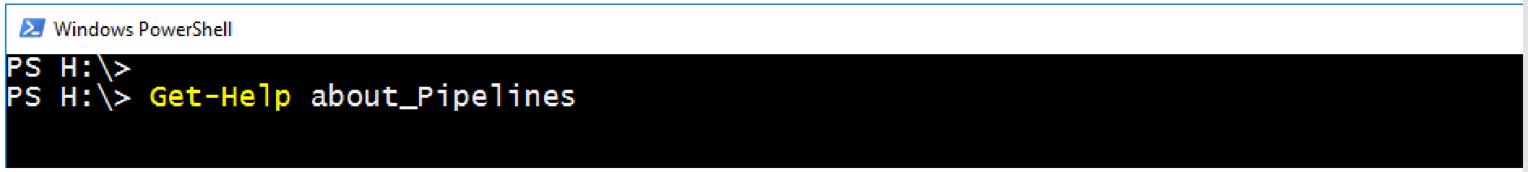
```
PS C:\> ##1. Assignment of string (ToVariable,FromKeyboardInput,FromFile)
PS C:\> $a = "I love powershell"
PS C:\> $b = 'I hate PowerShell'
PS C:\>
PS C:\> ##2. Comparison of string structures
PS C:\> $a.CompareTo($a)
0
PS C:\> $a.CompareTo($b)
1
PS C:\> #StartWith
PS C:\> $a.StartsWith("I")
True
PS C:\> #EndWith
PS C:\> $a.EndsWith("PowerShell")
False
PS C:\> #Contains
PS C:\> $a.Contains("pow")
True
PS C:\> #ToUpper
PS C:\> $a.ToUpper()
I LOVE POWERSHELL
PS C:\> #ToLower
PS C:\> $a.ToLower()
i love powershell
PS C:\> #Replace
PS C:\> $a.Replace('love','hate')
I hate powershell
PS C:\> #IndexOf
PS C:\> $a.IndexOf("ll")
15
PS C:\> #Insert
PS C:\> $a.Insert(17," PowershellISE")
I love powershell PowershellISE
```

PS Pipeline

Špecializované IKT systémy Windows

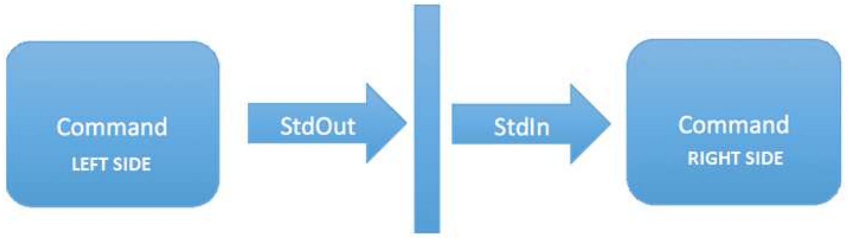
Ing. Stanislav Lukac, PhD.

| Pipeline


Windows PowerShell

```
PS H:\>
PS H:\> Get-Help about_Pipelines
```

Output from one element (left side)is input for another element (right side)



Cmdlet, functions must be designed to work with pipeline

- Param must accept input from pipeline

Everything in PS is handling via PIPELINE

Pipeline
How it works

- Commandlet, function must accept value from pipeline Cmdlet
- PSObject is generic object, if such cmdlet accept PSObject from pipeline
  - byValue - is first option, which PS try to do
  - byPropertyname - is second option

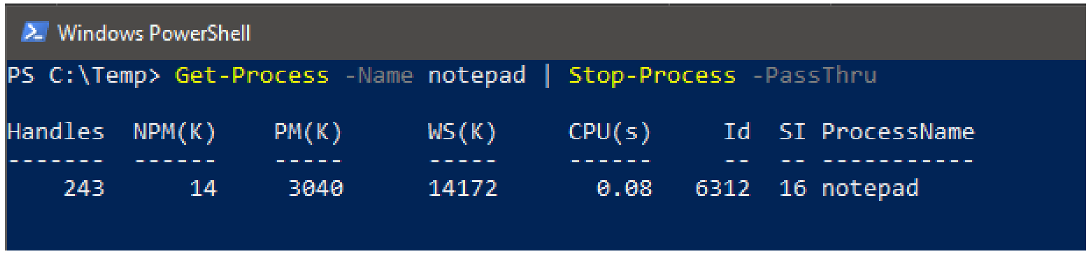
Get-Process notepad | Stop-Process

Get-service | Stop-Process

Pipeline
How it works

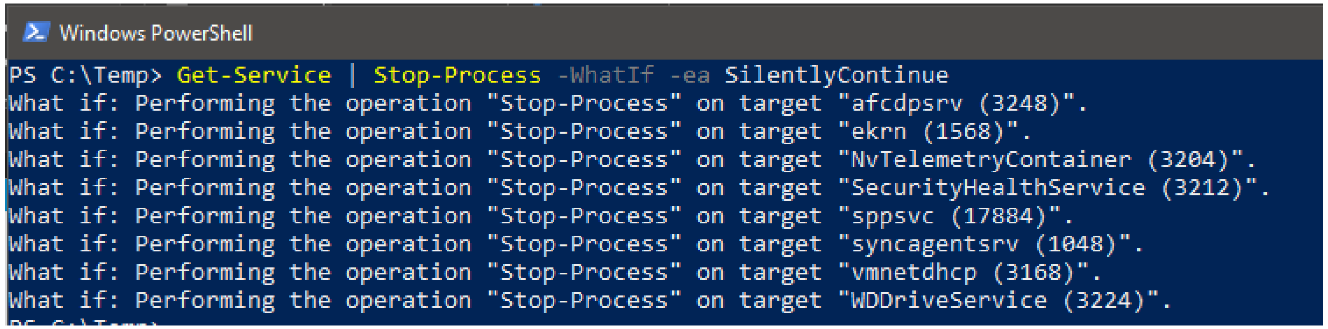
Get-Process notepad | Stop-Process

1. Check if cmdlet on RIGHT side have PARAM which accept input from PIPELINE
2. If exist, such PARAM, check which value is accepting
3. Check which object is generated by cmdlet on LEFT side



Windows PowerShell

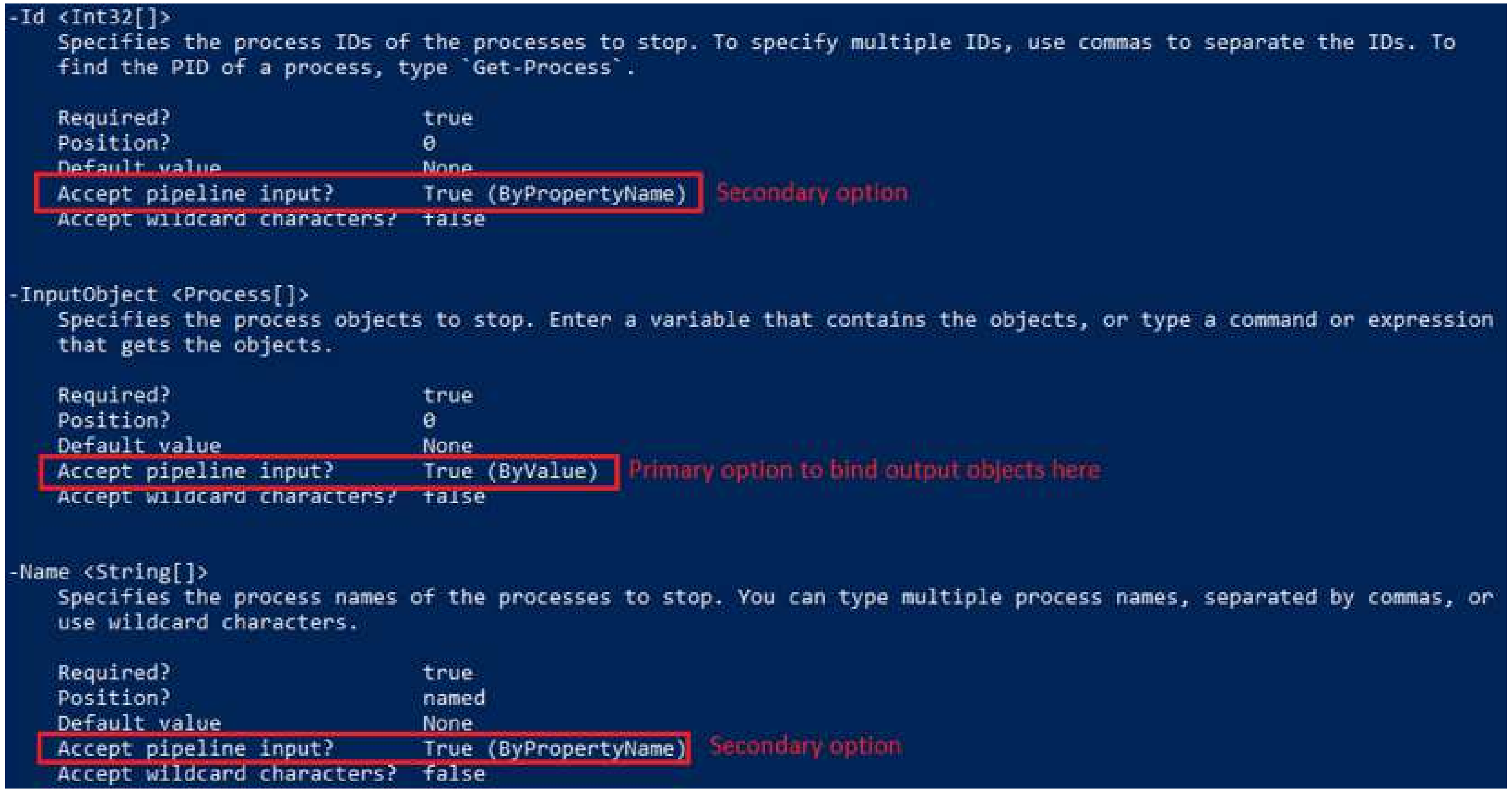
```
PS C:\Temp> Get-Process -Name notepad | Stop-Process -PassThru

Handles  NPM(K)    PM(K)      WS(K)     CPU(s)     Id  SI ProcessName
-------  ------    -----      -----     ------     --  -- -----------
    243      14     3040      14172       0.08   6312  16 notepad
```



Windows PowerShell

```
PS C:\Temp> Get-Service | Stop-Process -WhatIf -ea SilentlyContinue
What if: Performing the operation "Stop-Process" on target "afcdpsrv (3248)".
What if: Performing the operation "Stop-Process" on target "ekrn (1568)".
What if: Performing the operation "Stop-Process" on target "NvTelemetryContainer (3204)".
What if: Performing the operation "Stop-Process" on target "SecurityHealthService (3212)".
What if: Performing the operation "Stop-Process" on target "sppsvc (17884)".
What if: Performing the operation "Stop-Process" on target "syncagentsrv (1048)".
What if: Performing the operation "Stop-Process" on target "vmnetdhcp (3168)".
What if: Performing the operation "Stop-Process" on target "WDDriveService (3224)".
```

DEMO::



```
-Id <Int32[]>
    Specifies the process IDs of the processes to stop. To specify multiple IDs, use commas to separate the IDs. To
    find the PID of a process, type `Get-Process`.

    Required?                    true
    Position?                    0
    Default value                None
    Accept pipeline input?       True (ByPropertyName)
    Accept wildcard characters?  false


-InputObject <Process[]>
    Specifies the process objects to stop. Enter a variable that contains the objects, or type a command or expression
    that gets the objects.

    Required?                    true
    Position?                    0
    Default value                None
    Accept pipeline input?       True (ByValue)
    Accept wildcard characters?  false


-Name <String[]>
    Specifies the process names of the processes to stop. You can type multiple process names, separated by commas, or
    use wildcard characters.

    Required?                    true
    Position?                    named
    Default value                None
    Accept pipeline input?       True (ByPropertyName)
    Accept wildcard characters?  false
```

Object manipulation

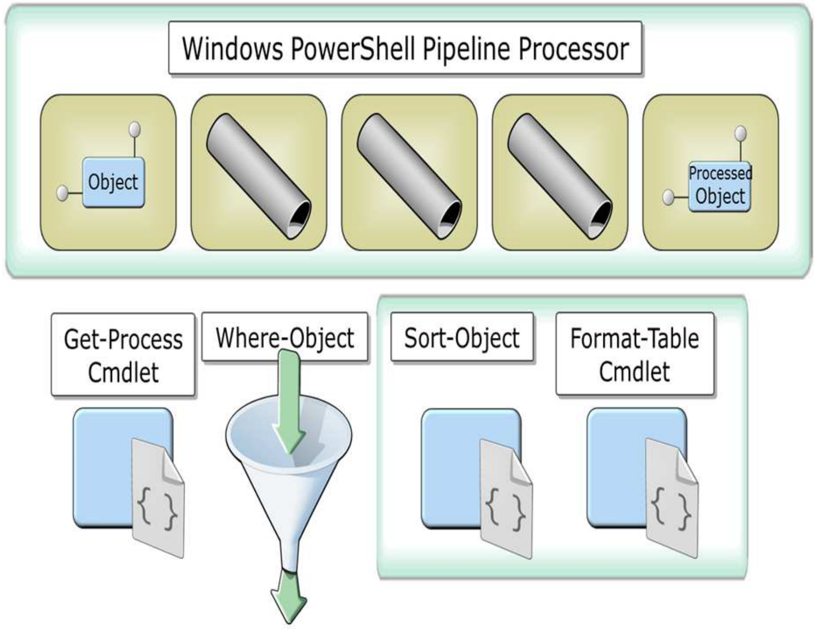
Špecializované IKT systémy Windows

Ing. Stanislav Lukac, PhD.

Working with objects

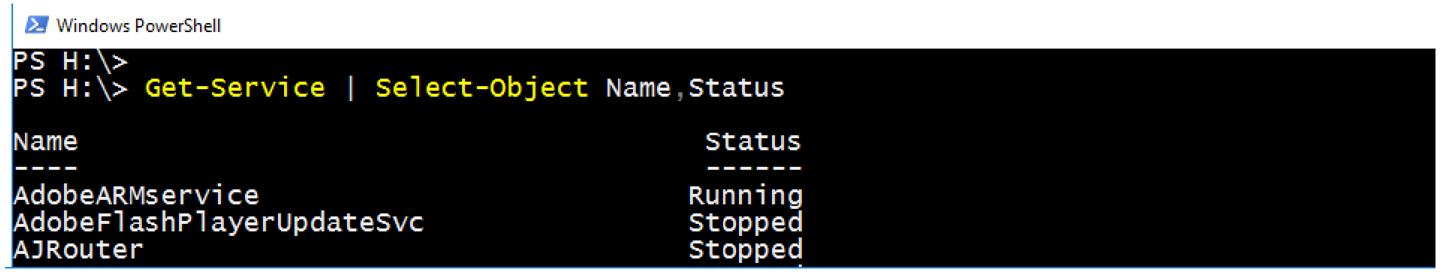
1. Filtering objects
2. Sorting objects
3. Formating

- Grouping objects
- Comparing objects
- Measure objects

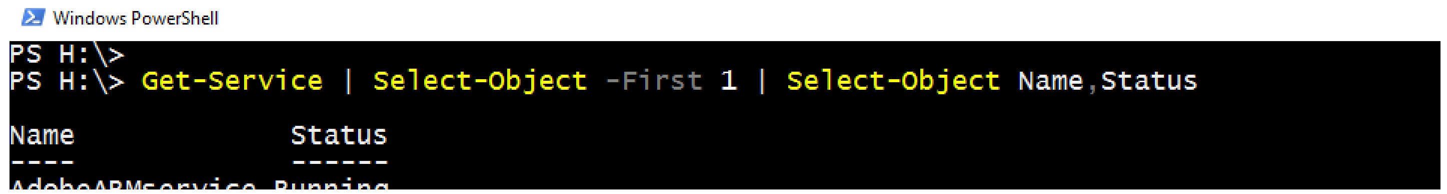


Filtering Object

Select-Object

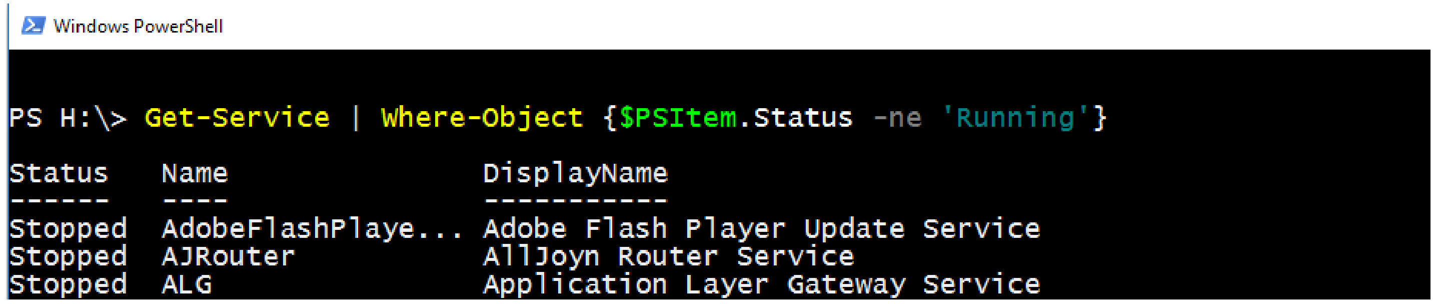

Windows PowerShell

```
PS H:\>
PS H:\> Get-Service | Select-Object Name,Status

Name                                        Status
----                                        ------
AdobeARMservice                            Running
AdobeFlashPlayerUpdateSvc                  Stopped
AJRouter                                   Stopped
```

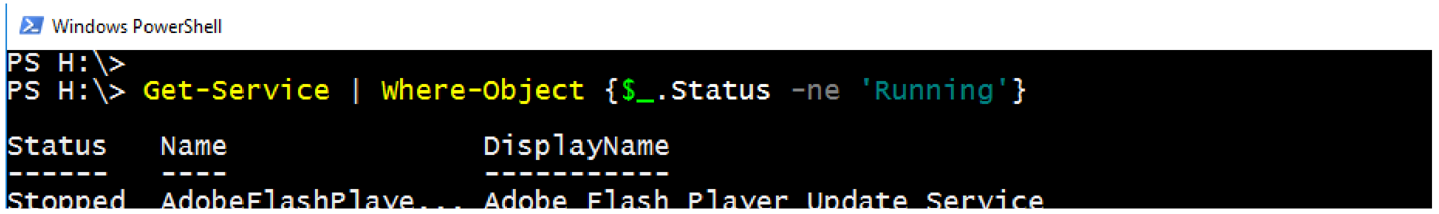

Windows PowerShell

```
PS H:\>
PS H:\> Get-Service | Select-Object -First 1 | Select-Object Name,Status

Name            Status
----            ------
AdobeARMservice Running
```

Where-Object

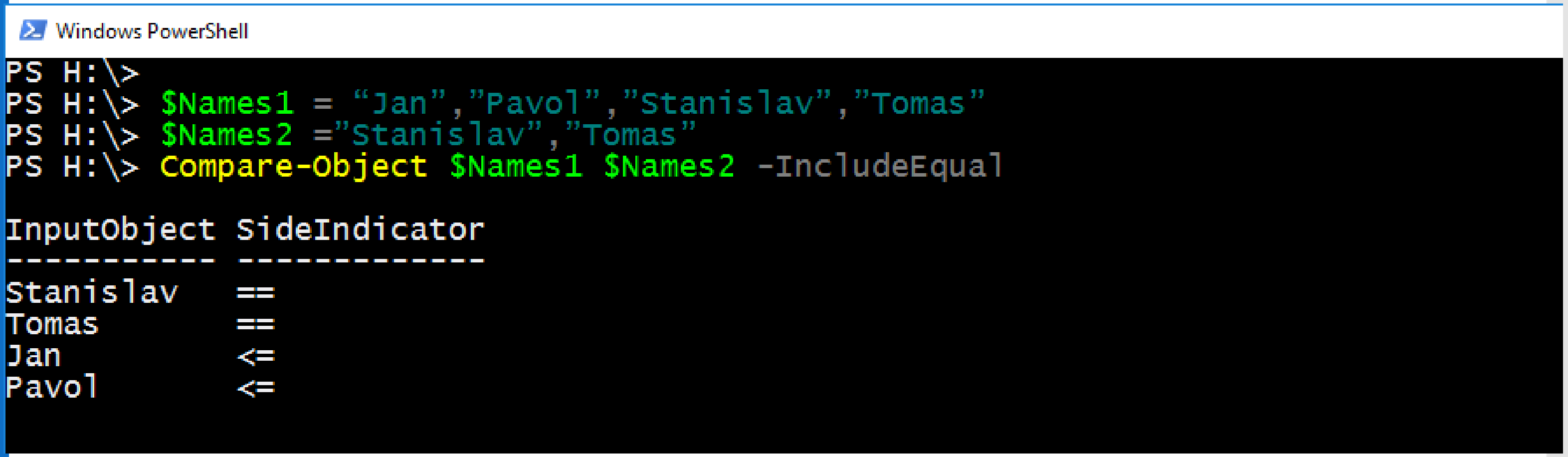

Windows PowerShell

```


PS H:\> Get-Service | Where-Object {$PSItem.Status -ne 'Running'}

Status   Name               DisplayName
------   ----               -----------
Stopped  AdobeFlashPlaye... Adobe Flash Player Update Service
Stopped  AJRouter           AllJoyn Router Service
Stopped  ALG                Application Layer Gateway Service
```


Windows PowerShell

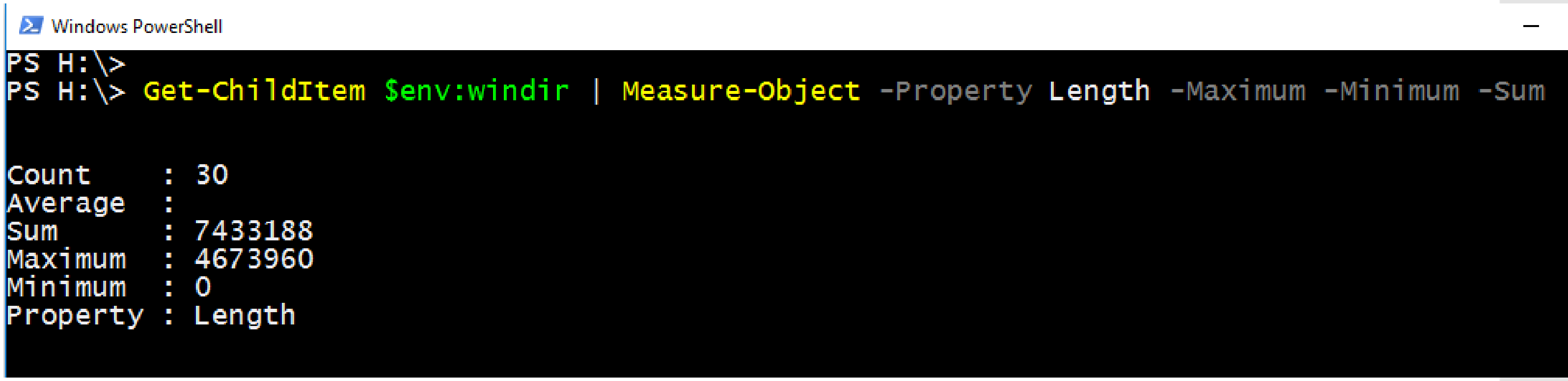
```
PS H:\>
PS H:\> Get-Service | Where-Object {$_.Status -ne 'Running'}

Status   Name               DisplayName
------   ----               -----------
Stopped  AdobeFlashPlaye... Adobe Flash Player Update Service
```

Comparing Objects



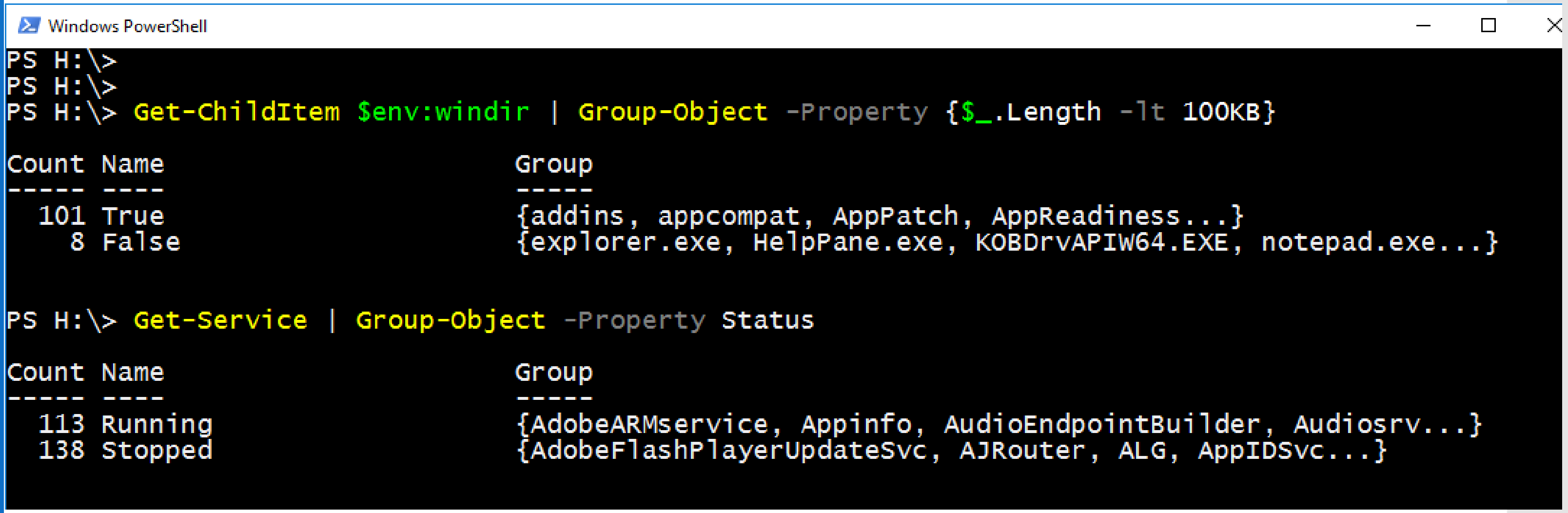
```
PS H:\>
PS H:\> $Names1 = “Jan”,”Pavol”,”Stanislav”,”Tomas”
PS H:\> $Names2 =”Stanislav”,”Tomas”
PS H:\> Compare-Object $Names1 $Names2 -IncludeEqual

InputObject SideIndicator
----------- -------------
Stanislav   ==
Tomas       ==
Jan         <=
Pavol       <=
```

Measure Object



```
PS H:\>
PS H:\> Get-ChildItem $env:windir | Measure-Object -Property Length -Maximum -Minimum -Sum


Count    : 30
Average  :
Sum      : 7433188
Maximum  : 4673960
Minimum  : 0
Property : Length
```

Grouping Object


Windows PowerShell

```
PS H:\>
PS H:\>
PS H:\> Get-ChildItem $env:windir | Group-Object -Property {$_.Length -lt 100KB}

Count Name                      Group
----- ----                      -----
  101 True                      {addins, appcompat, AppPatch, AppReadiness...}
    8 False                     {explorer.exe, HelpPane.exe, KOBDrvAPIW64.EXE, notepad.exe...}


PS H:\> Get-Service | Group-Object -Property Status

Count Name                      Group
----- ----                      -----
  113 Running                   {AdobeARMservice, Appinfo, AudioEndpointBuilder, Audiosrv...}
  138 Stopped                   {AdobeFlashPlayerUpdateSvc, AJRouter, ALG, AppIDSvc...}
```

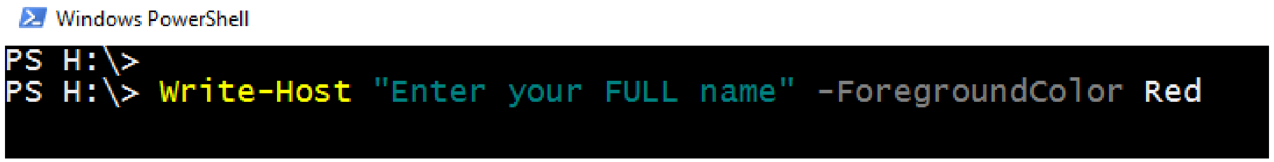
PS Output

Špecializované IKT systémy Windows

Console Input / Output

Write-Host

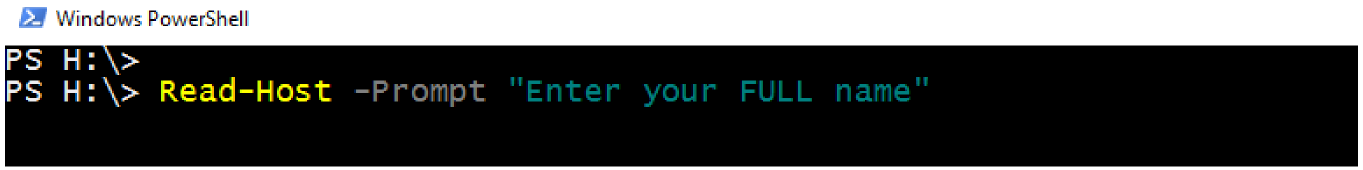
Out-Host



Windows PowerShell

```
PS H:\>
PS H:\> Write-Host "Enter your FULL name" -ForegroundColor Red
```

Read-Host



Windows PowerShell

```
PS H:\>
PS H:\> Read-Host -Prompt "Enter your FULL name"
```

Formating Output

You have 4 cmdlet for formatting output

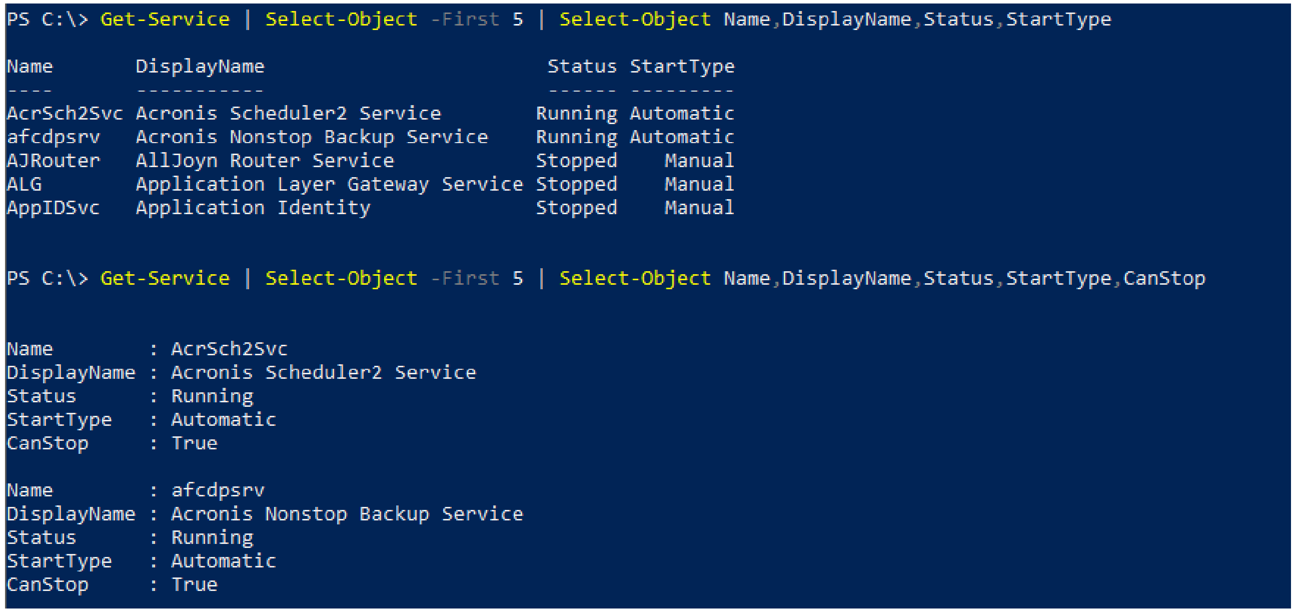
- Format-Table
- Format-List
- Format-Wide
- Format-Custom

USE THEM ALWAYS AT THE END AND ONLY FOR CONSOLE

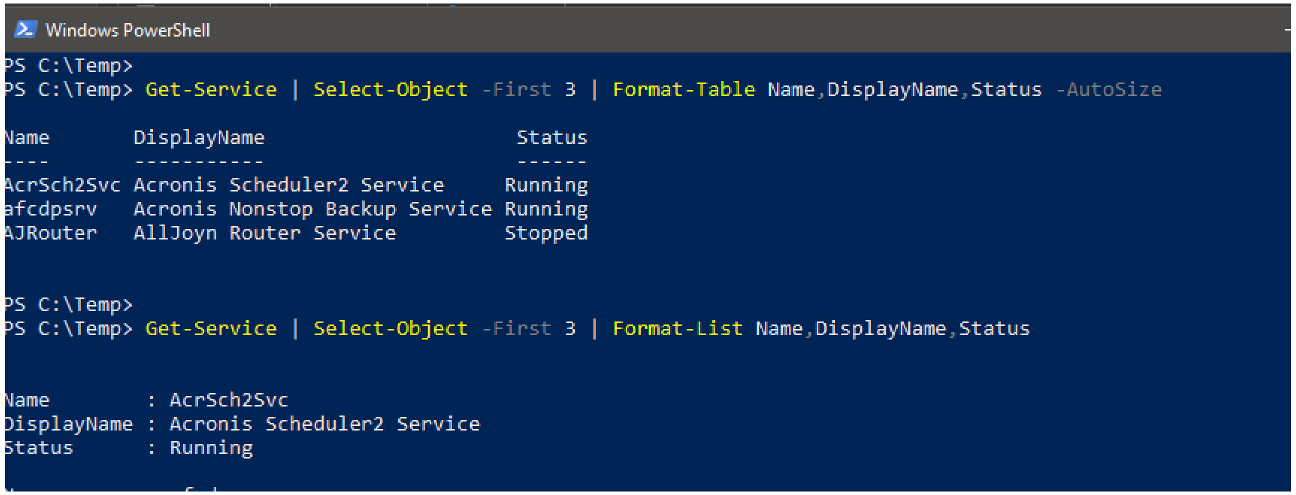
Default formatting rules:

- Displayed property <4 ....format table
- Displayed property <4 ...... format list

DEMO::



```
PS C:\> Get-Service | Select-Object -First 5 | Select-Object Name,DisplayName,Status,StartType

Name        DisplayName                        Status StartType
----        -----------                        ------ ---------
AcrSch2Svc  Acronis Scheduler2 Service        Running Automatic
afcdpsrv    Acronis Nonstop Backup Service    Running Automatic
AJRouter    AllJoyn Router Service            Stopped    Manual
ALG         Application Layer Gateway Service Stopped    Manual
AppIDSvc    Application Identity              Stopped    Manual


PS C:\> Get-Service | Select-Object -First 5 | Select-Object Name,DisplayName,Status,StartType,CanStop


Name        : AcrSch2Svc
DisplayName : Acronis Scheduler2 Service
Status      : Running
StartType   : Automatic
CanStop     : True

Name        : afcdpsrv
DisplayName : Acronis Nonstop Backup Service
Status      : Running
StartType   : Automatic
CanStop     : True
```

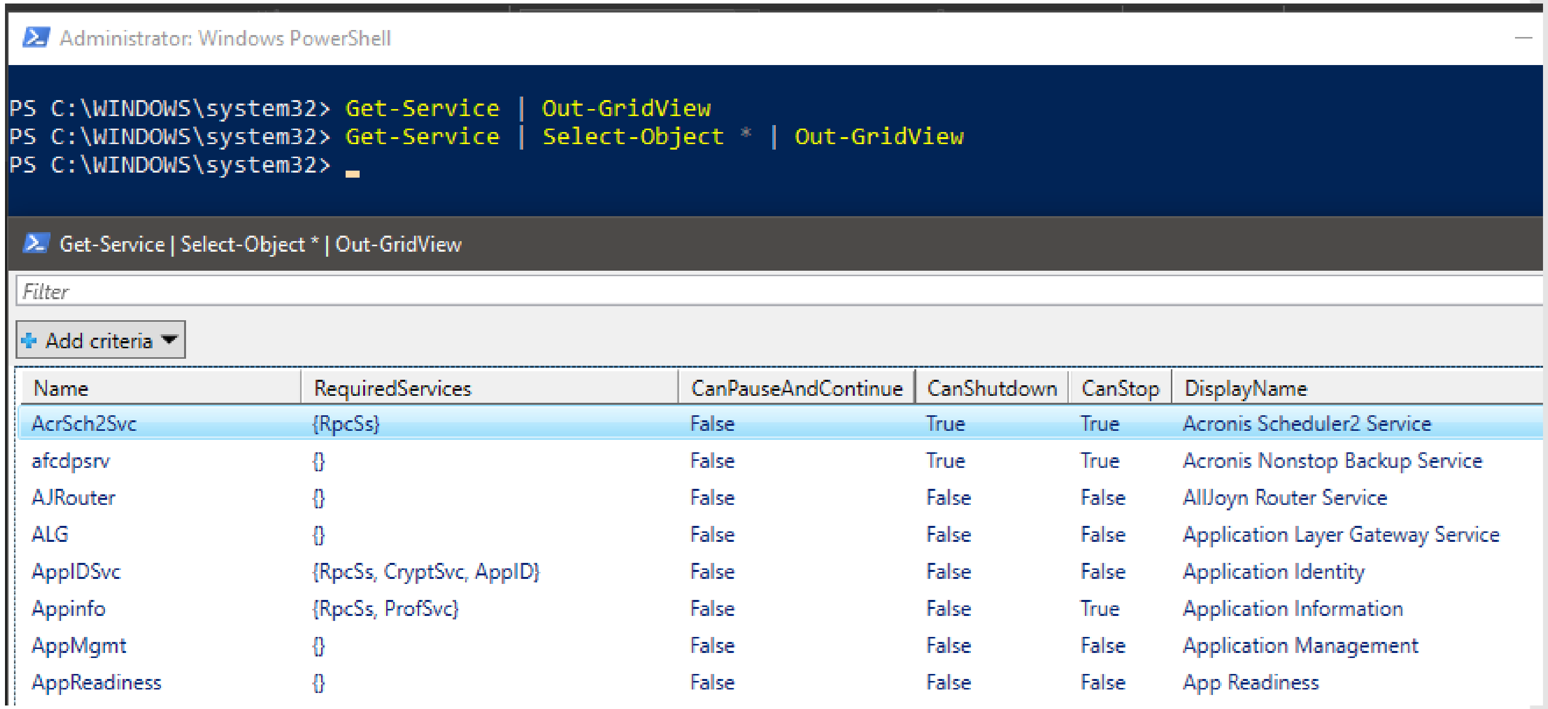


```
Windows PowerShell
PS C:\Temp>
PS C:\Temp> Get-Service | Select-Object -First 3 | Format-Table Name,DisplayName,Status -AutoSize

Name        DisplayName                     Status
----        -----------                     ------
AcrSch2Svc  Acronis Scheduler2 Service     Running
afcdpsrv    Acronis Nonstop Backup Service Running
AJRouter    AllJoyn Router Service         Stopped


PS C:\Temp>
PS C:\Temp> Get-Service | Select-Object -First 3 | Format-List Name,DisplayName,Status


Name        : AcrSch2Svc
DisplayName : Acronis Scheduler2 Service
Status      : Running
```

DEMO::

Graphical table output with formatting, sorting options

Required installed PowerShellISE



Administrator: Windows PowerShell

```
PS C:\WINDOWS\system32> Get-Service | Out-GridView
PS C:\WINDOWS\system32> Get-Service | Select-Object * | Out-GridView
PS C:\WINDOWS\system32> _
```

Get-Service | Select-Object * | Out-GridView
Filter
Add criteria

Name	RequiredServices	CanPauseAndContinue	CanShutdown	CanStop	DisplayName
AcrSch2Svc	{RpcSs}	False	True	True	Acronis Scheduler2 Service
afcdpsrv	{}	False	True	True	Acronis Nonstop Backup Service
AJRouter	{}	False	False	False	AllJoyn Router Service
ALG	{}	False	False	False	Application Layer Gateway Service
AppIDSvc	{RpcSs, CryptSvc, AppID}	False	False	False	Application Identity
Appinfo	{RpcSs, ProfSvc}	False	False	True	Application Information
AppMgmt	{}	False	False	False	Application Management
AppReadiness	{}	False	False	False	App Readiness

Exporting Output

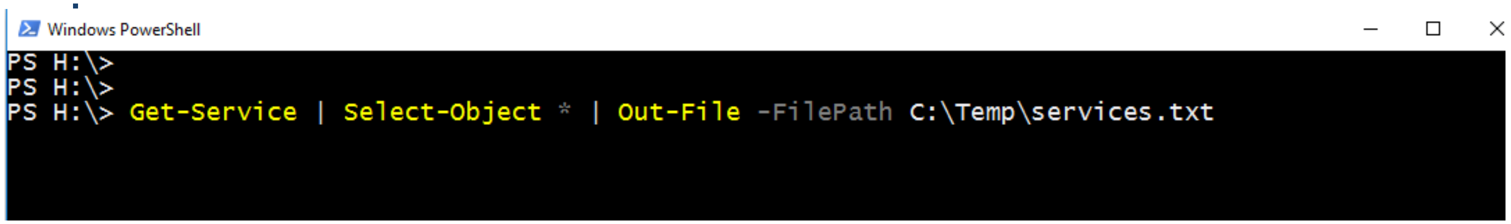
You have several cmdlet for output into …

- Out-File
- Out-Host

- Export-CSV
- Export-CLIXML

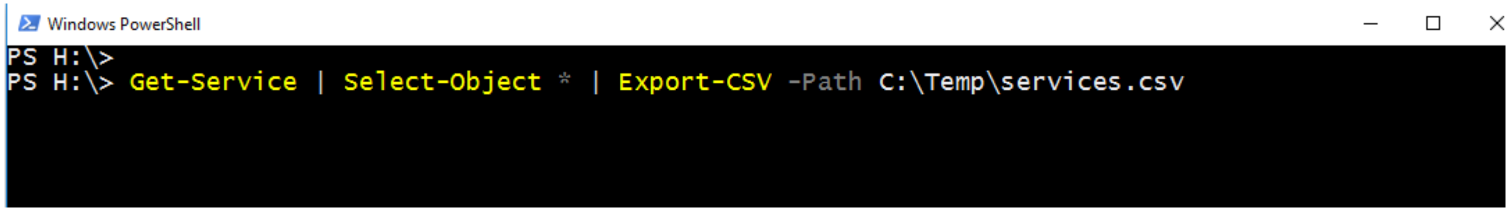
- ConverTo-HTML

DEMO::



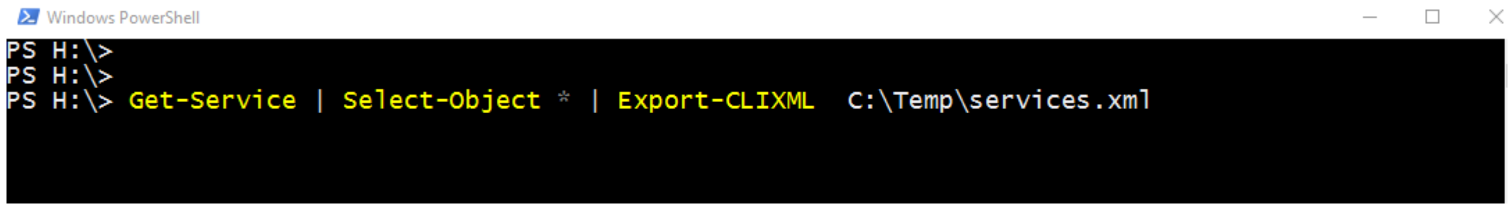
Windows PowerShell

```
PS H:\>
PS H:\>
PS H:\> Get-Service | Select-Object * | Out-File -FilePath C:\Temp\services.txt
```



Windows PowerShell

```
PS H:\>
PS H:\> Get-Service | Select-Object * | Export-CSV -Path C:\Temp\services.csv
```



Windows PowerShell

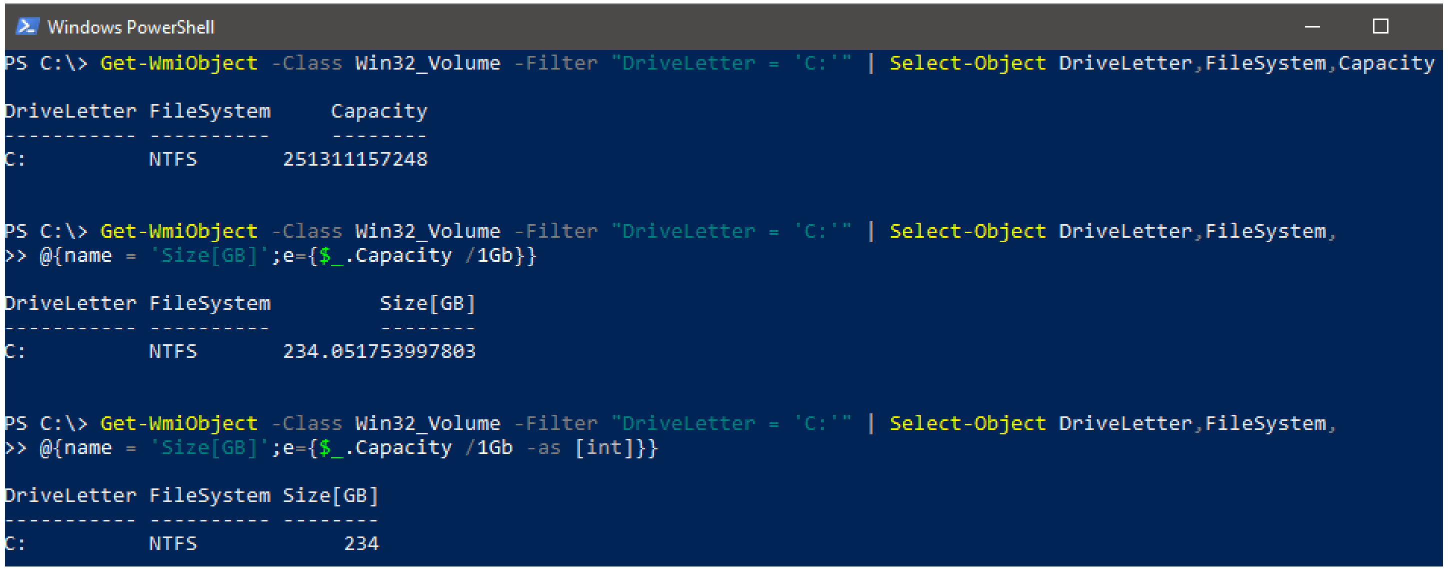
```
PS H:\>
PS H:\>
PS H:\> Get-Service | Select-Object * | Export-CLIXML  C:\Temp\services.xml
```

Creating Own propertie “light”

Own Formated view

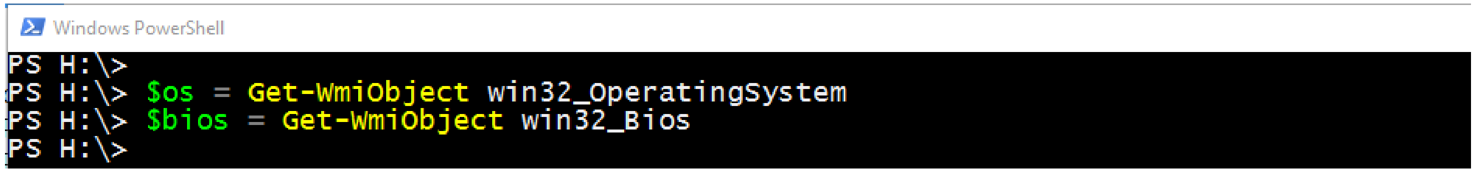
- By using hash table

  @{ n=OwnPropertyName; e=CODE}



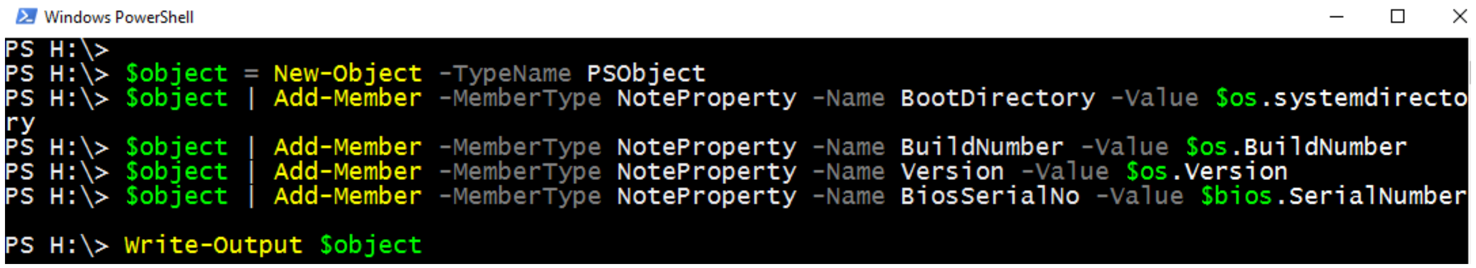
```
PS C:\> Get-WmiObject -Class Win32_Volume -Filter "DriveLetter = 'C:'" | Select-Object DriveLetter,FileSystem,Capacity

DriveLetter FileSystem     Capacity
----------- ----------     --------
C:          NTFS       251311157248


PS C:\> Get-WmiObject -Class Win32_Volume -Filter "DriveLetter = 'C:'" | Select-Object DriveLetter,FileSystem,
>> @{name = 'Size[GB]';e={$_.Capacity /1Gb}}

DriveLetter FileSystem         Size[GB]
----------- ----------         --------
C:          NTFS       234.051753997803


PS C:\> Get-WmiObject -Class Win32_Volume -Filter "DriveLetter = 'C:'" | Select-Object DriveLetter,FileSystem,
>> @{name = 'Size[GB]';e={$_.Capacity /1Gb -as [int]}}

DriveLetter FileSystem Size[GB]
----------- ---------- --------
C:          NTFS            234
```

Creating Own Object



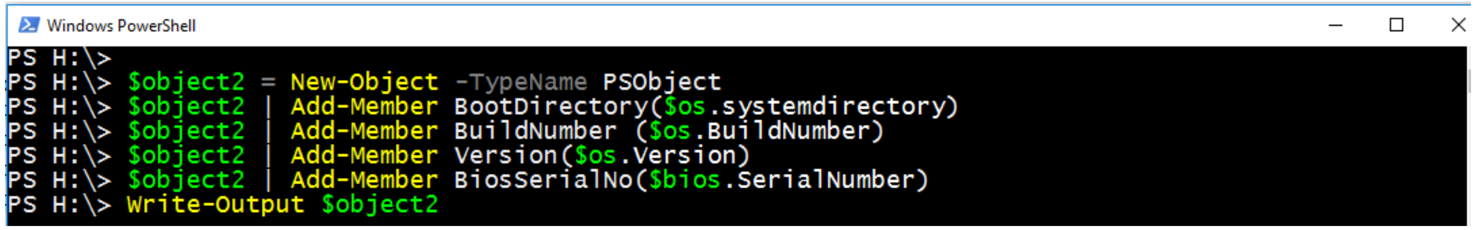
```
PS H:\>
PS H:\> $os = Get-WmiObject win32_OperatingSystem
PS H:\> $bios = Get-WmiObject win32_Bios
PS H:\>
```

Method1



```
PS H:\>
PS H:\> $object = New-Object -TypeName PSObject
PS H:\> $object | Add-Member -MemberType NoteProperty -Name BootDirectory -Value $os.systemdirecto
ry
PS H:\> $object | Add-Member -MemberType NoteProperty -Name BuildNumber -Value $os.BuildNumber
PS H:\> $object | Add-Member -MemberType NoteProperty -Name Version -Value $os.Version
PS H:\> $object | Add-Member -MemberType NoteProperty -Name BiosSerialNo -Value $bios.SerialNumber

PS H:\> Write-Output $object
```

Method2

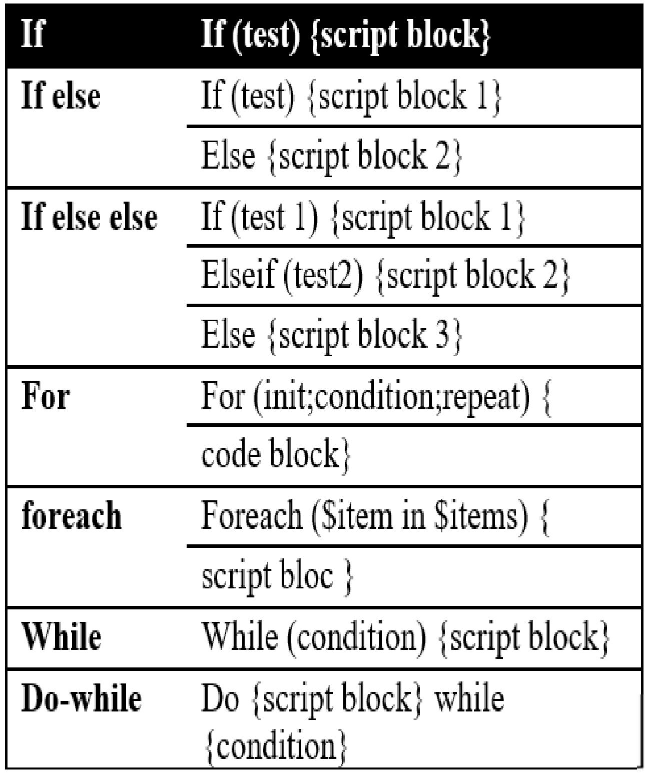


```
PS H:\>
PS H:\> $object2 = New-Object -TypeName PSObject
PS H:\> $object2 | Add-Member BootDirectory($os.systemdirectory)
PS H:\> $object2 | Add-Member BuildNumber ($os.BuildNumber)
PS H:\> $object2 | Add-Member Version($os.Version)
PS H:\> $object2 | Add-Member BiosSerialNo($bios.SerialNumber)
PS H:\> Write-Output $object2
```

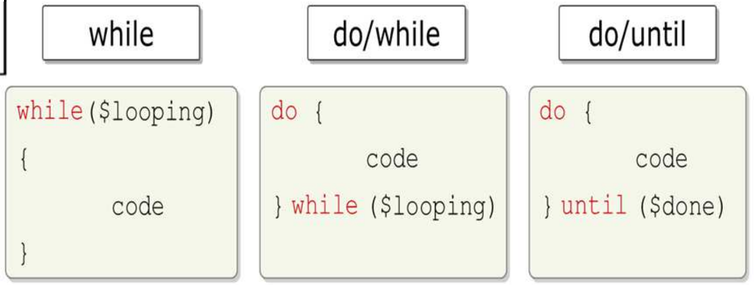
PS Loops

Špecializované IKT systémy Windows

Loops



If	If (test) {script block}
If else	If (test) {script block 1}
	Else {script block 2}
If else else	If (test 1) {script block 1}
	Elseif (test2) {script block 2}
	Else {script block 3}
For	For (init;condition;repeat) {
	code block}
foreach	Foreach ($item in $items) {
	script bloc }
While	While (condition) {script block}
Do-while	Do {script block} while {condition}



while

```
while($looping)
{
        code
}
```

do/while

```
do {
        code
} while ($looping)
```

do/until

```
do {
        code
} until ($done)
```

Switch

- By default without options, switch performs case-sesnsitive match
- Syntax

```
switch ($var) {
  varvalue1 {script block}
  varvalue2 {script block}
  varvalue3 {script block}
  default {script block}
}
```

NEXT

::Free book

https://books.goalkicker.com/PowerShellBook/

::PowerShell cook book free

http://www.powertheshell.com/cookbooks/