

# PowerShell

Ročníková práce, Ondřej Švorc I3B



# Obsah

1. Pokročilé techniky a postupy
2. Import uživatelů a jejich správa v AD
3. Práce se souborovým systémem, zabezpečení
4. Novinky v PowerShell 7.2



# Pokročilé techniky a postupy

1. Funkce
2. Pipelines (Roury)
3. Vícevláknové zpracování (Multithreading)
4. PowerShell Remoting
5. TaskScheduler



# Funkce

## Pokročilé techniky a postupy

### Definice

- ☐ část kódu, která je zaobalena do pojmenovaného bloku – vykoná se zavoláním tohoto bloku
- ☐ před svým použitím musí být definována – funkce jsou psány **nad** kód, který je volá
- ☐ **duplicitní** kód lze přidat do funkce, a tím jeho opakovanost **minimalizovat** či úplně **odstranit**
- ☐ může mít **návratovou hodnotu** – nejčastěji objekt nebo kolekci objektů
- ☐ zlepšuje **konzistenci** kódu a celkovou **čitelnost** skriptu – docíleno korektními návrhovými návyky (správné názvosloví, dodržování konvencí jazyka atp.)
- ☐ dokáže přejímat **parametry**, s jimiž může dále při zavolání pracovat
- ☐ je schopna napodobit chování **cmdletu**
- ☐ podporuje zakomponování do **pipeline**



# Funkce

Pokročilé techniky a postupy

## Funkce vs Cmdlet

- ☐ cmdlet je prakticky vzato **funkce**, ale **pokročilá**
- ☐ způsob volání se liší, resp. různý je styl předávání parametrů
- ☐ cmdlet nabízí oproti funkci **validaci** parametrů při jeho volání
- ☐ u klasických funkcí se validace musí implementovat **manuálně**
- ☐ styl převzetí parametrů lze u cmdletu upravovat, u funkce nikoliv
- ☐ stylem převzetí parametrů rozumíme atribut **ValueFromPipeline**
- ☐ ValueFromPipeline - parametr musí být předán pomocí pipeline
- ☐ pokud jej neuvedeme, parametr bude muset být předán manuálně



# Funkce

Pokročilé techniky a postupy

## Funkce vs Cmdlet



```
1 function GetUserPassword ([string] $userName) {  
2     # Klasická funkce  
3 }  
4  
5 GetUserPassword("jan")
```



```
1 function Get-UserPassword {  
2     [CmdletBinding()]  
3     param(  
4         [Parameter(Mandatory)]  
5         [string] $UserName  
6     )  
7  
8     # Cmdlet  
9 }  
10  
11 Get-UserPassword -UserName "jan"
```

# Funkce

Pokročilé techniky a postupy

## Praktický příklad

Cílem je vytvořit funkci, která bude na základě uživatelského jména automaticky generovat uživatellovo heslo.

## Ukázkové vstupy a výstupy

*Vstup:* jan

*Výstup:* Jj123456

*Vstup:* petr

*Výstup:* Pp123456



# Funkce

Pokročilé techniky a postupy

## 1. Definice funkce

```
function Get-UserPassword {  
  
}
```

### Syntaxe

- ❑ **function** – klíčové slovo označující funkci
- ❑ **Get-UserPassword** - název funkce dle oficiálních standardů (tzv. PascalCase)
- ❑ **{ }** – složené závorky vymezující oblast funkce – kód v ní napsaný náleží k funkci





# Funkce

## Pokročilé techniky a postupy

### 2. Přejímání parametrů funkcí

- ❑ při volání funkce můžeme předávat **parametry**
- ❑ parametrem rozumíme cokoliv reprezentující hodnotu – proměnná, objekt, funkce s návratovou hodnotou
- ❑ v našem případě chceme funkci předat **uživatelské jméno** – bez něho by funkce nemohla vygenerovat heslo

```
function Get-UserPassword {  
    [CmdletBinding()]  
    param(  
        [Parameter(Mandatory)]  
        [string] $UserName  
    )  
}
```

„Funkce, chovej se jako cmdlet.“

„Funkce, tady jsou tvé parametry.“

„Funkce, tohle je tvůj parametr a je povinný.“



# Funkce

## Pokročilé techniky a postupy

### 3. Vygenerování hesla funkcí

```
return ($UserName.Substring(0, 1).ToUpper() + $UserName.Substring(0, 1).ToLower() + $passwordEnding)
```

1. `$UserName.Substring(0, 1)` → vrátí nový řetězec, neovlivní ten původní
2. první parametr funkce `Substring` udává, kolik znaků má přeskočit → v našem případě žádné
3. druhý parametr udává, kolik znaků má vybrat → v našem případě pouze jeden, a to ten první
4. vestavěná funkce `ToUpper()` změní všechny znaky řetězce z malých písmen na velká → „J“
5. vestavěná funkce `ToLower()` změní všechny znaky řetězce z velkých písmen na malá → „j“
6. na konec příkazu přidáme kýžené číslice, a celý řetězec nyní tvoří **návratovou hodnotu** – vygenerované heslo



# Funkce

Pokročilé techniky a postupy



```
1 [string] $passwordEnding = "123456"
2
3 # Funkce, která dokáže vytvořit heslo na základě uživatelského jména
4 function Get-UserPassword {
5     [CmdletBinding()]
6     param(
7         [Parameter(Mandatory)]
8         [string] $UserName
9     )
10
11     return ($UserName.Substring(0, 1).ToUpper() + $UserName.Substring(0, 1).ToLower() + $passwordEnding)
12 }
13
14 Get-UserPassword -UserName "jan"
```

# Pipelines (Roury)

Pokročilé techniky a postupy

## Princip

- ☐ předávání **výstupu** z jednoho příkazu na **vstup** druhého příkazu
  - \*výstupem rozumíme jeden nebo více **objektů**
  - \*příkaz se v PowerShellu nazývá **cmdlet**
- ☐ předávaný prvek si lze představit jako štafetový kolík, který si běžci při běhu postupně předávají
- ☐ díky rourám je kód kratší a často přehlednější
- ☐ roury podporují **vícevláknové** zpracování

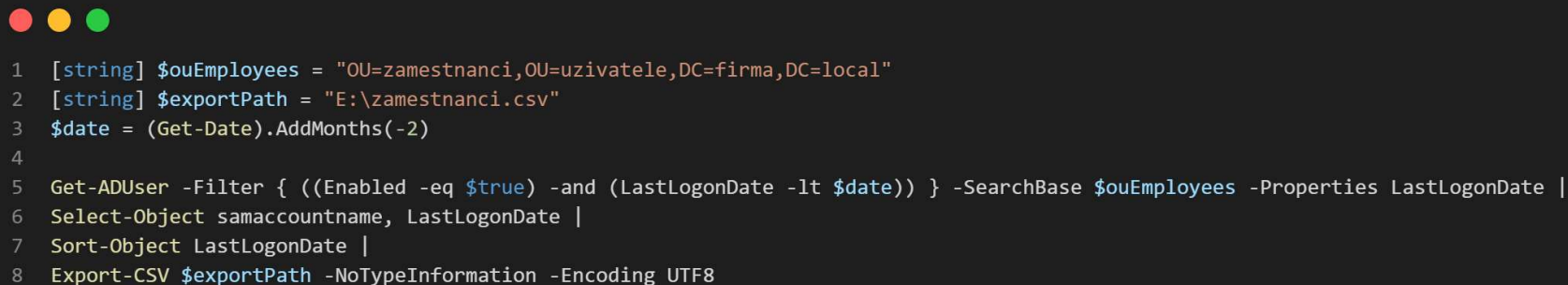


# Pipelines (Roury)

Pokročilé techniky a postupy

## Příklad

Účty bývalých zaměstnanců rušíme zpravidla po 2 měsících neaktivity. Pro statistické účely si chceme seznam neaktivních uživatelů uložit jako CSV.

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. It contains a PowerShell script with 8 lines of code, numbered 1 through 8. The script defines variables for employee OU, export path, and date, then uses a pipeline of cmdlets to find and export inactive users.

```
1 [string] $ouEmployees = "OU=zamestnanci,OU=uzivatele,DC=firma,DC=local"
2 [string] $exportPath = "E:\zamestnanci.csv"
3 $date = (Get-Date).AddMonths(-2)
4
5 Get-ADUser -Filter { ((Enabled -eq $true) -and (LastLogonDate -lt $date)) } -SearchBase $ouEmployees -Properties LastLogonDate |
6 Select-Object samaccountname, LastLogonDate |
7 Sort-Object LastLogonDate |
8 Export-CSV $exportPath -NoTypeInfoation -Encoding UTF8
```

# Vícevláknové zpracování

Pokročilé techniky a postupy

## Princip

- ☐ spíše známé jako **multithreading** - využití více vláken a jader najednou (**asynchronní** zpracování kódu)
- ☐ možnost vykonat další příkaz **bez čekání** na dokončení příkazu **předešlého**
- ☐ příkazy se zpracovávají **naráz**, každý z nich v **jiném** vlákne
- ☐ vhodné pro **obrovské** množství repetitivních příkazů, jejichž synchronní provedení je příliš pomalé
- ☐ jsou-li příkazy **nenáročné**, synchronní přístup bývá **rychlejší** – nedochází k předávání dat mezi vlákny
- ☐ základem paralelního zpracování jsou tzv. **Úlohy** (Jobs)
- ☐ dostupné od verze **PowerShell Core 3**
- ☐ modernější konstrukce je `ForEach-Object -Parallel`
- ☐ parametr **-Parallel** dostupný od **PowerShell Core 7.0**



# Vícevláknové zpracování

Pokročilé techniky a postupy

## Úlohy a jejich nejběžnější stavy

Stav	Popis
Completed	Úloha byla dokončena a výstupní data lze načíst nebo úlohu odstranit.
Running	Úloha právě probíhá a nelze ji odstranit bez vynuceného zastavení.
Blocked	Úloha stále běží, ale hostitel je před pokračováním vyzván k provedení nějaké akce.
Failed	Během provádění úlohy došlo k chybě.



# Vícevláknové zpracování

Pokročilé techniky a postupy

## Vysvětlení

- ☐ moderní procesory mají **4 - 16 jader**
- ☐ 1 jádro může pracovat na 1 vlákne **současně**
- ☐ 1 vlákno může být vykonáváno vždy jen 1 jádrem
- ☐ ve výchozím stavu běží proces v jednom tzv. **hlavním vlákne**
- ☐ rozdělíme-li program na **více vláken**, umožníme využití **více jader**
- ☐ pokud proces rozdělíme na více vláken než je jader, jádra budou neustále **využívána** na **maximum**
- ☐ **X vláken = X využívaných jader** (vztah platí za předpokladu, že počet vláken je menší rovno počtu jader)
- ☐ k zrychlení dojde právě tehdy, když rozdělení procesu na více vláken a následné shromažďování informací z těchto vláken **netrvá déle**, než běžné, synchronní provedení kódu



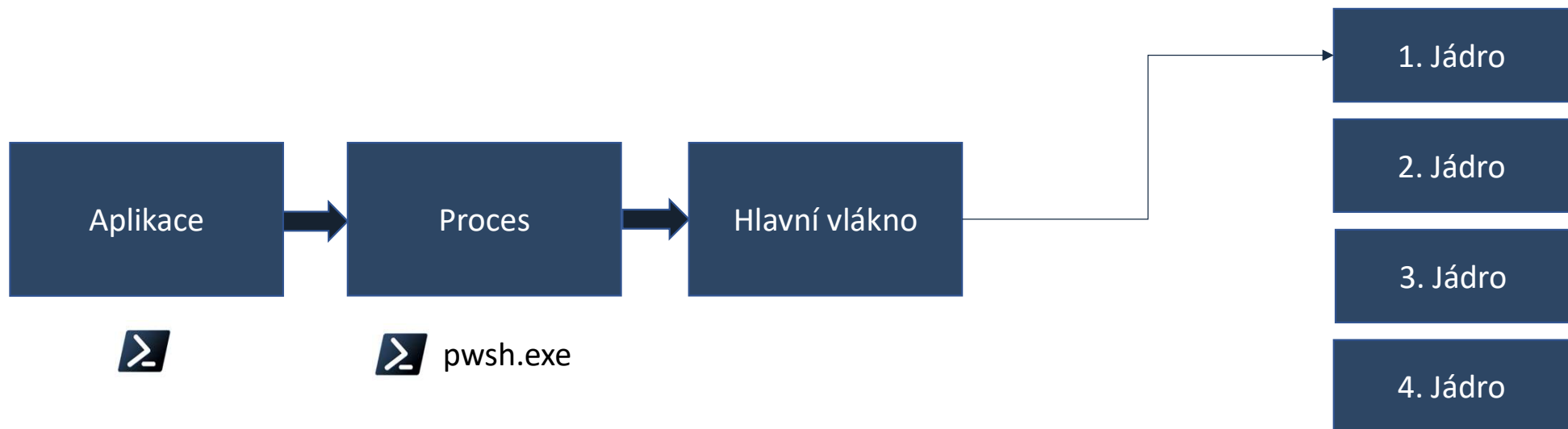


# Vícevláknové zpracování

Pokročilé techniky a postupy

## Synchronní provedení se 4 jádry k dispozici – schematicky

Pozn.: To, jakému jádru bude vlákno přiřazeno určuje OS. Vlákno může být přiřazeno jakémukoliv jádru.

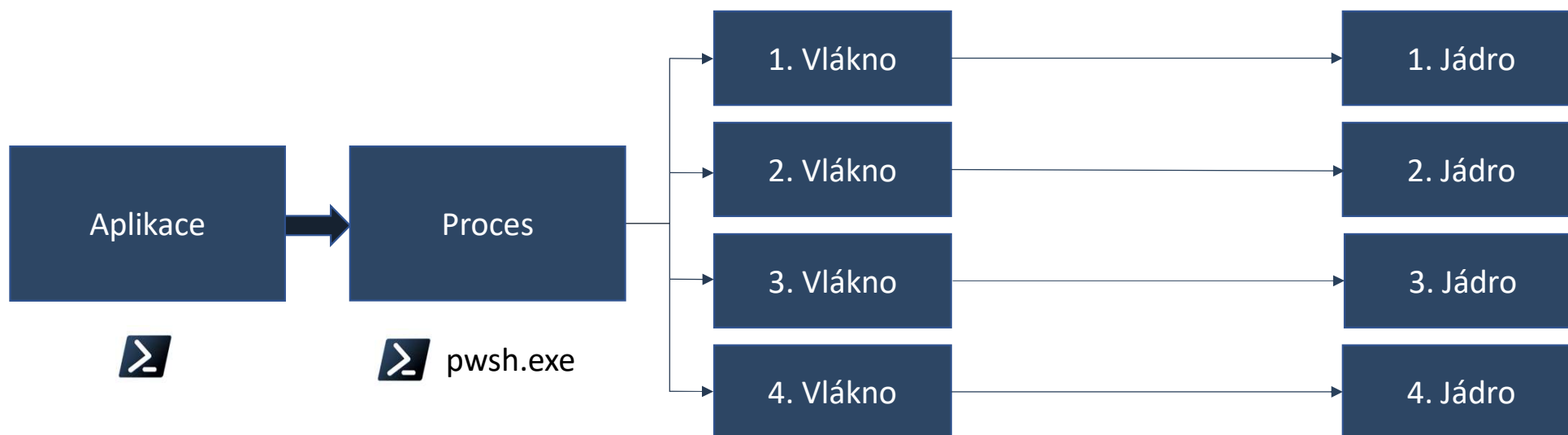


# Vícevláknové zpracování

Pokročilé techniky a postupy

## Asynchronní provedení se 4 jádry a při rozdělení na 4 vlákna – schematicky

Pozn.: To, jakému jádru bude vlákno přiřazeno určuje OS. Vlákno může být přiřazeno jakémukoliv jádru.



# Vícevláknové zpracování

Pokročilé techniky a postupy

## Vysvětlení příkazů – praktický příklad

### ☐ Measure-Command { KÓD }

- ☐ změří rychlost zaobaleného kódu na daném PC

### ☐ %

- ☐ zkrácený zápis cmdletu ForEach-Object

### ☐ \$\_

- ☐ symbolizuje momentální objekt/hodnotu přicházející z pipeline



# Vícevláknové zpracování

Pokročilé techniky a postupy

## Vysvětlení příkazů – praktický příklad

### ☐ **Start-Job, Remove-Job**

- ☐ Start-Job vytvoří na pozadí úlohu pod novou instancí PowerShellu (nový proces)
- ☐ Remove-Job úlohu na pozadí odstraní

### ☐ **\$using:názevProměnné**

- ☐ odkazuje na proměnnou definovanou globálně ve skriptu (v hlavním vlákne)

### ☐ **args[číslo]**

- ☐ pole argumentů – číslo udává index argumentu, počínaje nulou, tj. nultý argument



# Praktický příklad – Synchronní přístup

Vícevláknové zpracování

## Vytvoření 100 skupin

```
1 #Doména
2 [string] $domainName = "firma"
3 [string] $domainEnding = "local"
4
5 #Hlavní organizační jednotka
6 [string] $root = "skola"
7
8 Measure-Command {
9     1..100 | % {
10         New-ADGroup -Name $_ -GroupScope 1 -Path "OU=skupiny,OU=$root,DC=$domainName,DC=$domainEnding"
11     }
12 }
```

```
Days           : 0
Hours           : 0
Minutes         : 0
Seconds         : 6
Milliseconds    : 158
Ticks           : 61582693
TotalDays       : 7,12762650462963E-05
TotalHours      : 0,00171063036111111
TotalMinutes    : 0,102637821666667
TotalSeconds    : 6,1582693
TotalMilliseconds : 6158,2693
```

# Praktický příklad – Asynchronní přístup

Vícevláknové zpracování

## Vytvoření 100 skupin

```
1  #Doména
2  [string] $domainName = "firma"
3  [string] $domainEnding = "local"
4
5  #Hlavní organizační jednotka
6  [string] $root = "skola"
7
8  Measure-Command {
9      1..100 | % {
10         Start-Job -Name $_ -ScriptBlock {
11             New-ADGroup -Name $args[0] -GroupScope 1 -Path "OU=skupiny,OU=$using:root,DC=$using:domainName,DC=$using:domainEnding"
12         } -ArgumentList $_
13
14         Remove-Job -State Completed
15     }
16 }
```

```
Days           : 0
Hours          : 0
Minutes        : 1
Seconds        : 35
Milliseconds    : 569
Ticks          : 955695982
TotalDays      : 0,0011061296087963
TotalHours     : 0,0265471106111111
TotalMinutes   : 1,59282663666667
TotalSeconds   : 95,5695982
TotalMilliseconds : 95569,5982
```

# Vícevláknové zpracování

Pokročilé techniky a postupy

## Získání dat z Jobu



```
1 Start-Job -Name "job" -ScriptBlock {  
2     [int] $a = 5  
3     [int] $b = 10  
4     [int] $c = 15  
5  
6     return @($a, $b, $c);  
7 }  
8  
9 $result = Get-Job -Name "job" | Receive-Job  
10 Remove-Job -State Completed
```



# PowerShell Remoting

Pokročilé techniky a postupy

## Úvod

- ❑ funkce, která umožňuje spouštět PS příkazy nebo skripty na **vzdáleném** počítači
- ❑ možnost navázání spojení **není omezena** pouze na Windows OS – multiplatformní funkce
- ❑ příkaz spustí službu **WinRM**, vytvoří nové pravidlo v bráně firewall, a tím povolí příchozí spojení
- ❑ na **serverových** distribucích je automaticky **povolena** (od verze Windows Server 2012 R2)
- ❑ na **klientských** distribucích musí být manuálně **zapnuta** (**Enable-PSRemoting –Force**)
- ❑ WinRM = Windows Remote management – implementace protokolu Ws-Management
- ❑ Ws-Management = spouští příkazy PowerShellu na vzdálených zařízeních s Windows OS
- ❑ tj. připojení z Linux/Windows na Windows, ale nikoliv z Windows na Linux – **jiný přístup**
- ❑ příkazy pro vzdálené připojení požadují přihlašovací údaje administrátora - **Credential**





# PowerShell Remoting

Pokročilé techniky a postupy

## Vzdálené připojení z Windows na Linux

- ❑ služba WinRM dokáže zprostředkovat spojení pouze se zařízením s Windows OS
- ❑ používá se protokol **SSH** - PowerShell Core 7.0, **Remoting over SSH**
- ❑ jak na Linuxu, tak i na Windows musí být nainstalován **PowerShell Core 7.0+**
- ❑ struktura cmdletů pro vzdálené připojení zůstává prakticky stejná
- ❑ jediný rozdíl – u cmdletu **New-PSSession** musíme přidat vlastnost **–SSHTransport**
- ❑ ta říká, že pro vzdálené spojení bude **využívat** protokolu **SSH**, nikoliv služby WinRM
- ❑ zprostředkovatel spojení – **OpenSSH** klient + server (klient se připojuje na server)
- ❑ OpenSSH klient inicializuje SSH spojení, OpenSSH server čeká na příchozí SSH spojení
- ❑ autentizace – přihlašovací údaje uživatele na Linuxu; SSH klíč



# PowerShell Remoting

Pokročilé techniky a postupy

## Ukládání Credential šifrovaně do 2 separátních souborů



```
1 # Ziskani Credentials od uzivatele
2 $credential = Get-Credential
3
4 # Ulozeni hesla do textových souborů
5 $credential.UserName | ConvertFrom-SecureString | Set-Content "G:\username.txt"
6 $credential.Password | ConvertFrom-SecureString | Set-Content "G:\password.txt"
```

# PowerShell Remoting

Pokročilé techniky a postupy

## Získání Credential ze 2 separátních souborů



```
1 # Ziskani hesla z textoveho souboru
2 $adminName = Get-Content "G:\username.txt" | ConvertTo-SecureString
3 $adminPassword = Get-Content "G:\password.txt" | ConvertTo-SecureString
4
5 # Vytvoreni Credential objektu
6 $adminCredential = New-Object System.Management.Automation.PsCredential -ArgumentList $adminName, $adminPassword
```

# PowerShell Remoting

Pokročilé techniky a postupy

## Ukládání a načítání Credential šifrovaně do/z XML souboru



```
1 # Ziskani Credentials objektu a zasifrovani objektu do XML souboru
2 $credential = Get-Credential
3 $credential | Export-CliXml -Path 'G:\credentials.xml'
4
5 # Ziskani Credentials ze souboru
6 $credential = Import-CliXml -Path 'G:\credentials.xml'
```

# PowerShell Remoting

Pokročilé techniky a postupy

## Vysvětlení příkazů – praktický příklad

### ☐ **Test-WSMan** názevZařízení

- ☐ zjistí, zdali je služba PS Remoting na daném PC zapnuta (pokud není PC zapnutý, vrátí vždy false)

### ☐ **New-PSSession**

- ☐ naváže vzdálené spojení s PC

### ☐ **Invoke-Command**

- ☐ vykoná příkaz na PC, se kterým je navázáno spojení



# PowerShell Remoting

Pokročilé techniky a postupy

## Vysvětlení příkazů – praktický příklad

### ☐ Disconnect-PSSession

- ☐ přeruší vzdálené připojení – spojení je ve stavu „**disconnected**“

### ☐ Remove-PSSession

- ☐ odstraní přerušené spojení a uvolní veškeré prostředky, které využívalo

### ☐ Stop-Computer

- ☐ vypne počítač



# PowerShell Remoting

Pokročilé techniky a postupy

## Praktický příklad

V počítačové učebně U1 se nachází 15 počítačů. Potřebujeme se ujistit, že jsou po 17. hodině všechny počítače vypnuty. Domů ale odcházíme už v 15:00. Jak můžeme co nejefektivněji k problému pomocí PowerShellu přistoupit?

## Možnosti řešení (seřazeny od nejhorších, po nejlepší)

1. Půjdeme zpět do školy – manuálně zkontrolujeme, že jsou počítače vypnuty.
2. Půjdeme zpět do školy – spustíme skript, který zkontroluje, zdali jsou vypnuty.
3. Půjdeme zpět do školy – spustíme skript, který zkontroluje, zdali jsou vypnuty, a pokud ne, vypne je.
4. Připojíme se vzdáleně a uděláme to, co v předchozím kroku.
5. Uděláme skript, který se každý den automaticky spustí, provede kontrolu a případně vypne počítače.



# PowerShell Remoting

Pokročilé techniky a postupy



```
1 $domainCredential = Import-CliXml -Path 'E:\credentials.xml'
2
3 [string[]] $computers = Get-ADComputer -Filter * -SearchBase "OU=U1,OU=pocitace,OU=skola,DC=firma,DC=local" | Select-Object -ExpandProperty Name
4
5 $computers | ForEach-Object {
6     if (Test-WSMan $_) {
7         $session = New-PSSession -ComputerName $_ -Credential $domainCredential
8         $computer = $_
9         Invoke-Command -Session $session -ScriptBlock { Stop-Computer -ComputerName $using:computer -Force }
10        $session | Disconnect-PSSession | Remove-PSSession
11    }
12 }
```



# TaskScheduler

Pokročilé techniky a postupy

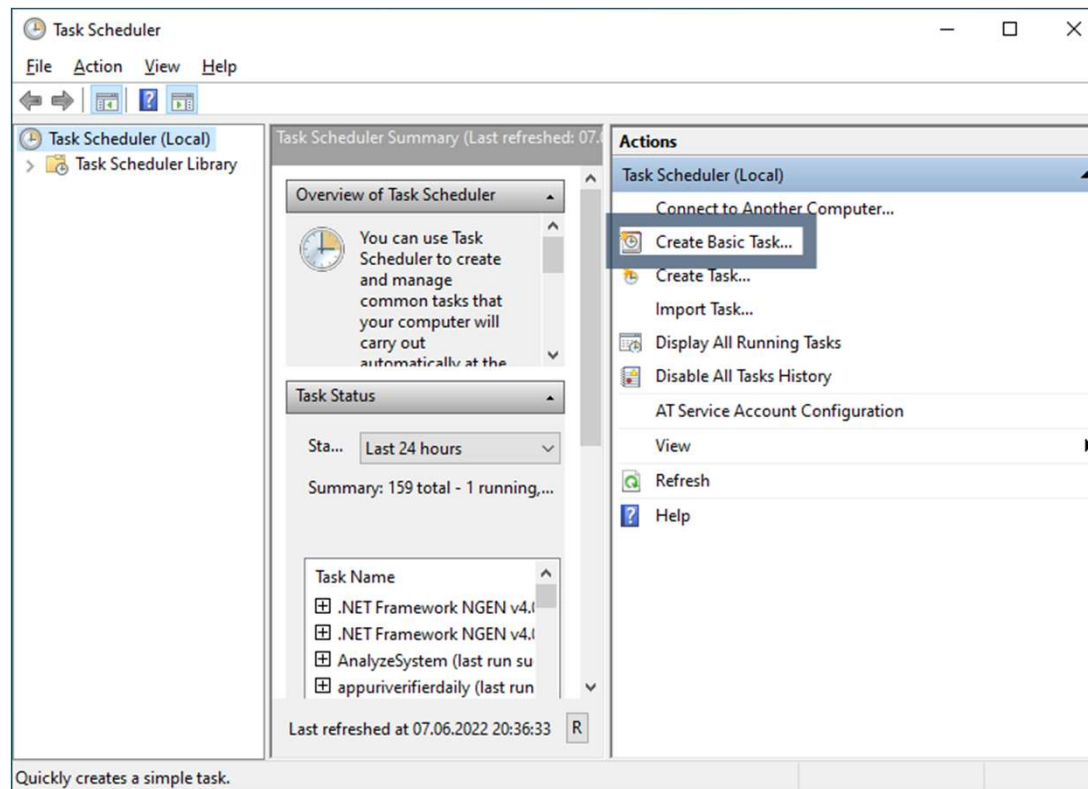
## Úvod

- ☐ v českém jazyce **plánovač úloh**
- ☐ využíván k plánování **automatizovaných** úloh
- ☐ v daný čas nebo při dané události provede příslušnou akci (spustí skript apod.)
- ☐ pro naše potřeby naplánujeme úlohu dle zadání – spustit skript každý den v 17:00
- ☐ celkem 2 možnosti – **manuální** konfigurace anebo **PowerShell**
- ☐ program pro spouštění skriptu – **powershell.exe** nebo **pwsh.exe** (PowerShell Core)



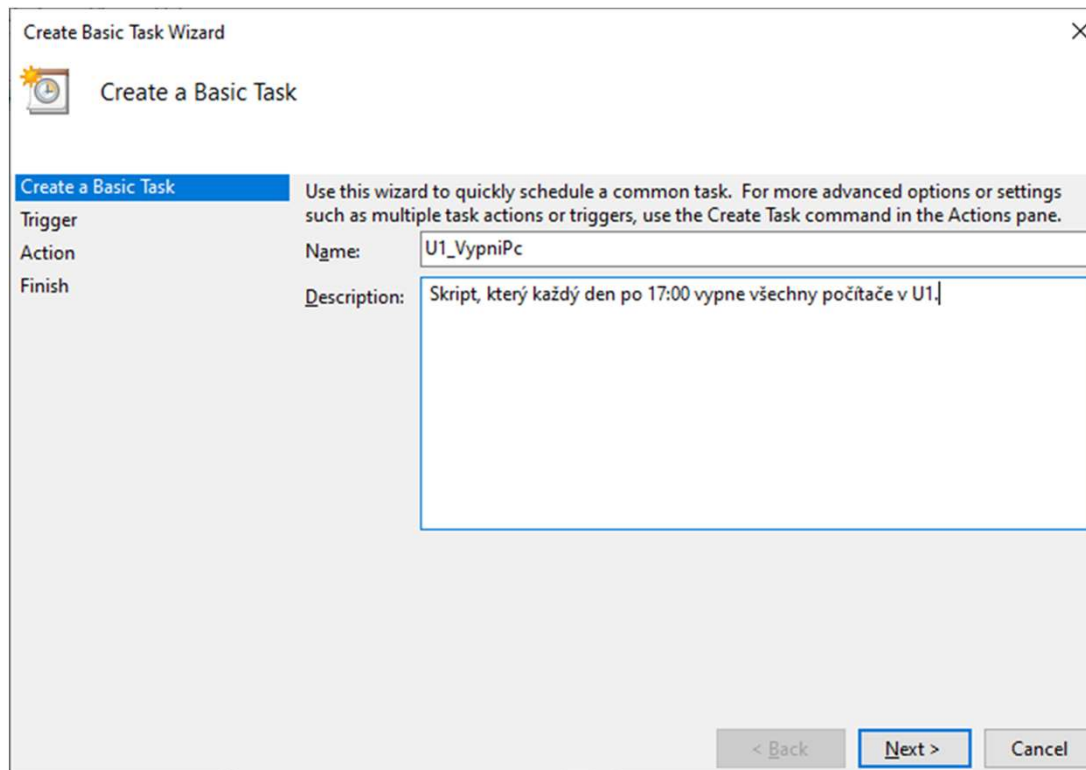
# TaskScheduler

Pokročilé techniky a postupy



# TaskScheduler

Pokročilé techniky a postupy



Create Basic Task Wizard

Create a Basic Task

Use this wizard to quickly schedule a common task. For more advanced options or settings such as multiple task actions or triggers, use the Create Task command in the Actions pane.

Name: U1\_VypniPc

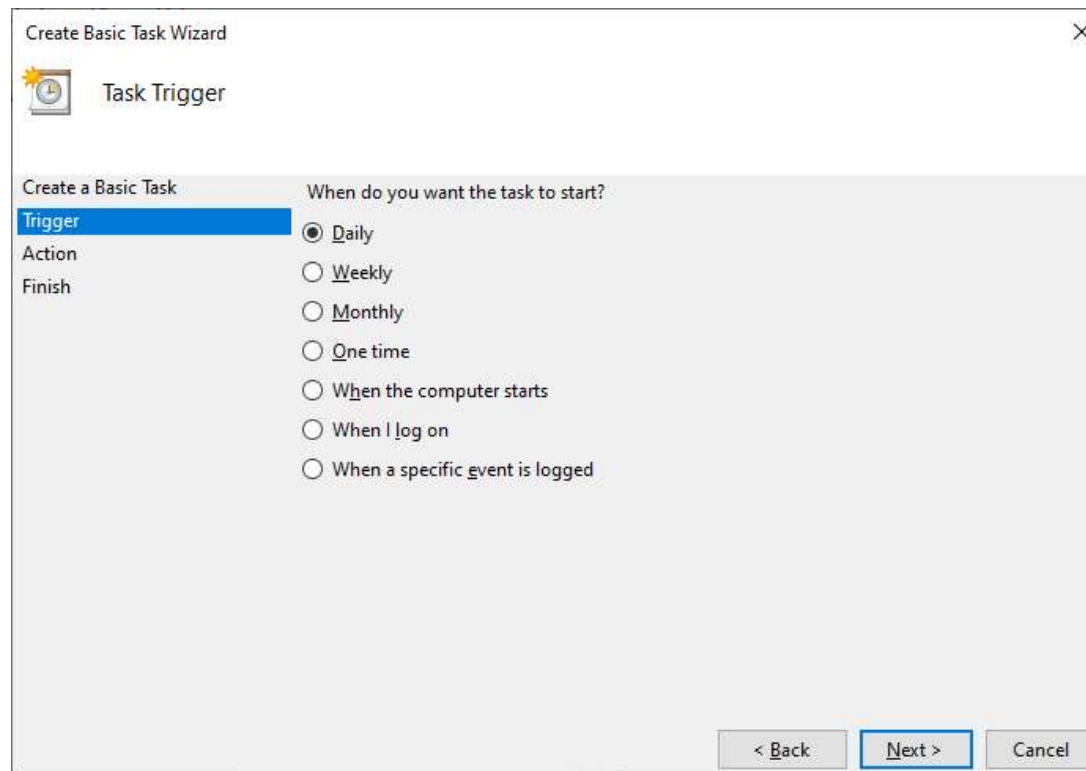
Description: Skript, který každý den po 17:00 vypne všechny počítače v U1.

< Back Next > Cancel



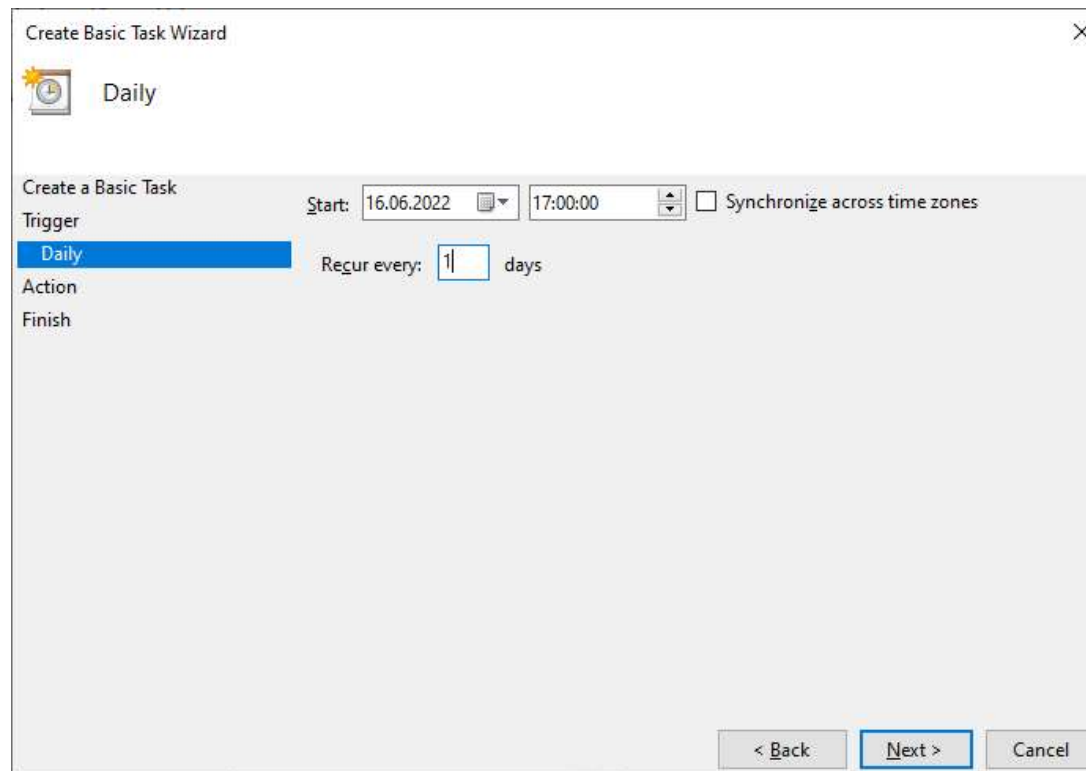
# TaskScheduler

Pokročilé techniky a postupy



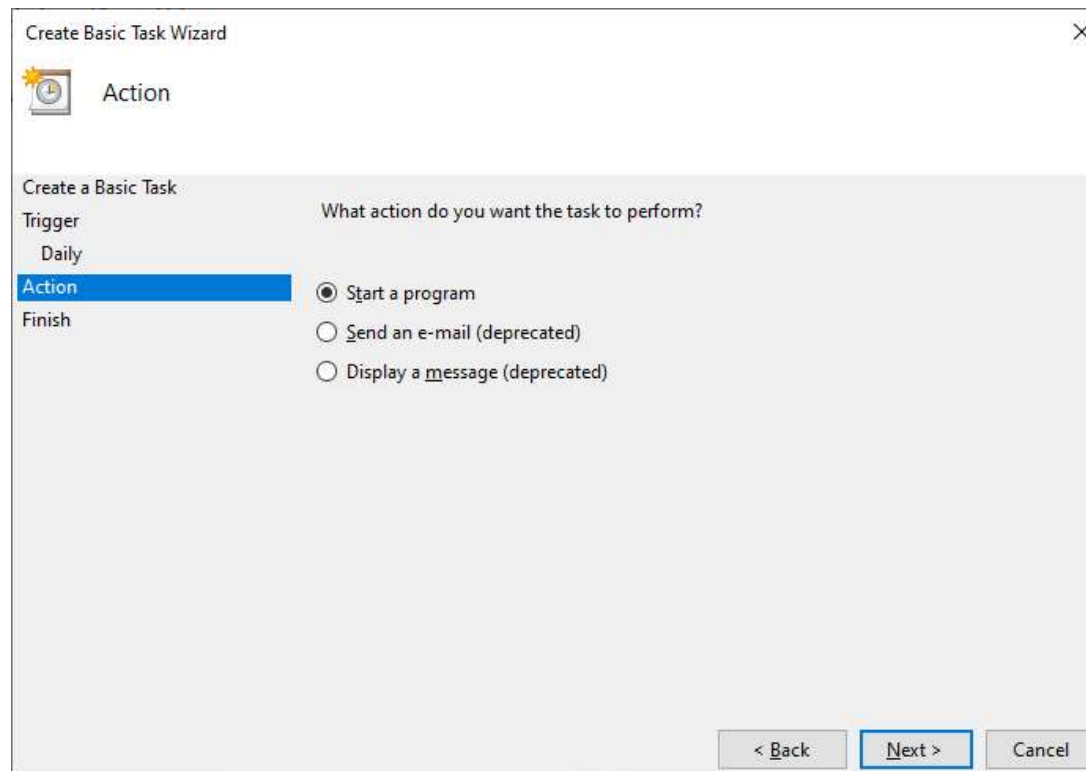
# TaskScheduler

Pokročilé techniky a postupy



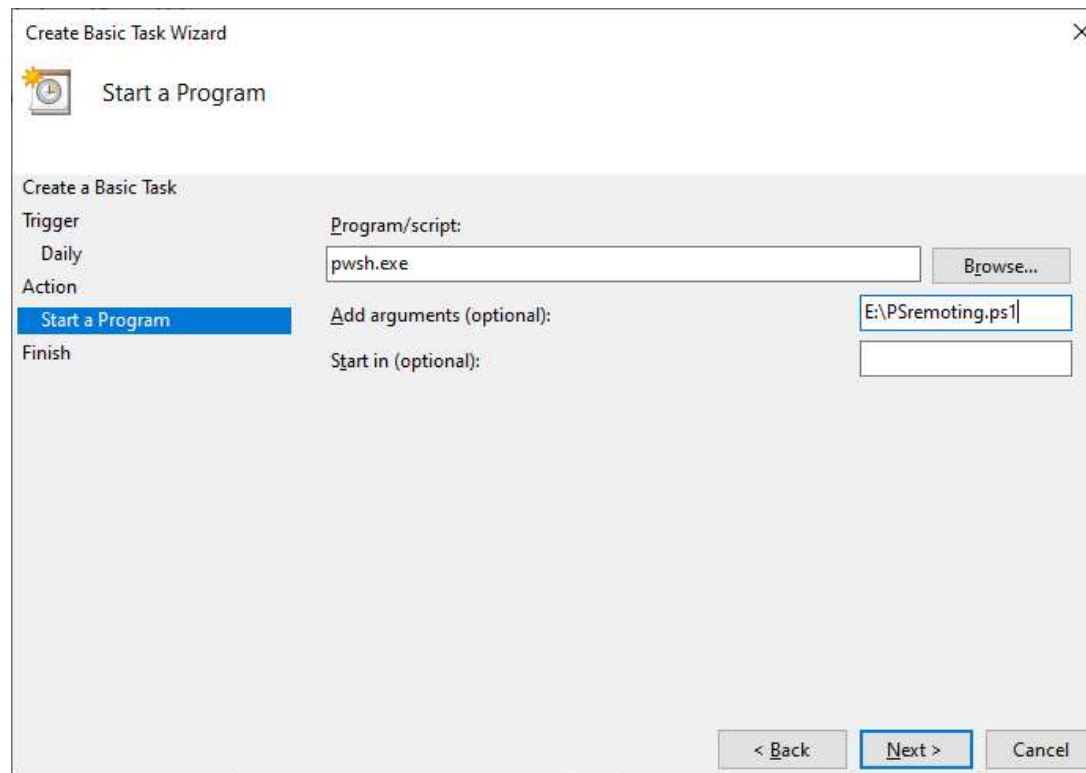
# TaskScheduler

Pokročilé techniky a postupy



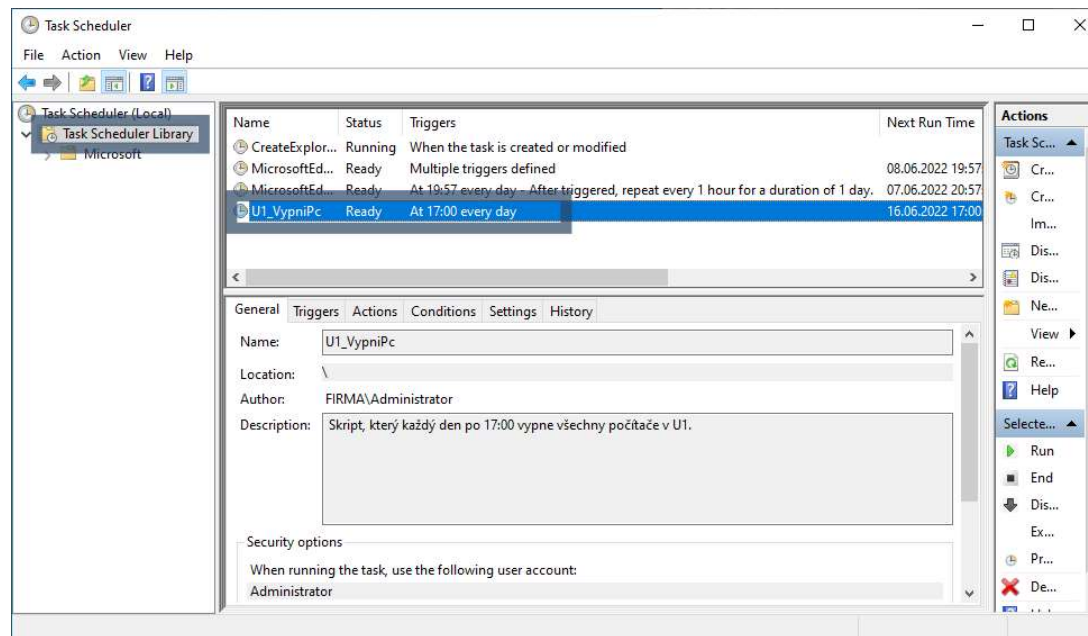
# TaskScheduler

Pokročilé techniky a postupy



# TaskScheduler

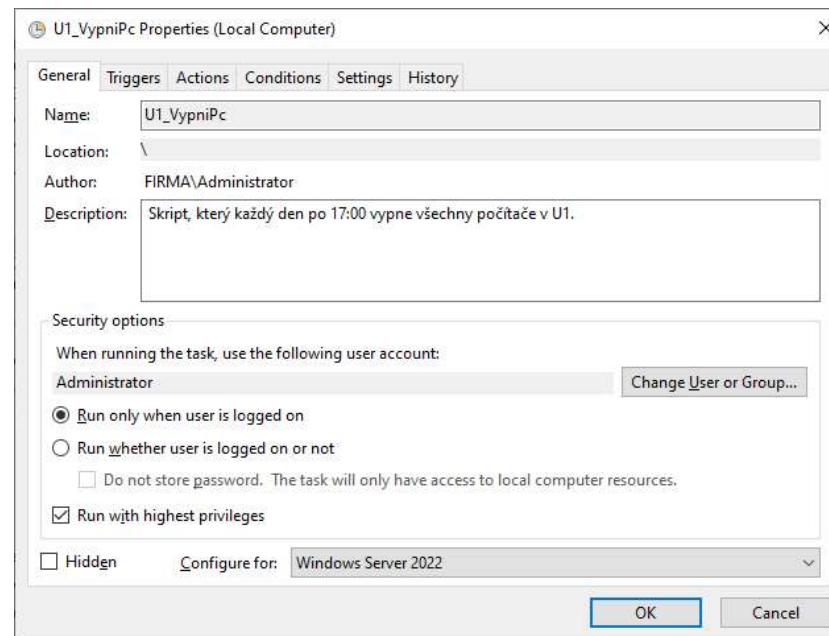
Pokročilé techniky a postupy





# TaskScheduler

Pokročilé techniky a postupy



# TaskScheduler

Pokročilé techniky a postupy

## Vysvětlení příkazů – PowerShell

### ☐ **New-ScheduledTaskTrigger**

- ☐ vytvoří trigger objekt – tzn. za jakých okolností se má naplánovaná úloha vykonat

### ☐ **New-ScheduledTaskAction**

- ☐ vytvoří objekt naplánované úlohy

### ☐ **Get-ScheduledTask**

- ☐ vrátí objekt naplánované úlohy (pokud existuje)



# TaskScheduler

Pokročilé techniky a postupy

## Vysvětlení příkazů – PowerShell



- ☐ alias cmdletu Where-Object

### ☐ **Register-ScheduledTask**

- ☐ zapíše novou úlohu do Plánovače úloh

### ☐ **Unregister-ScheduledTask**

- ☐ vymaže úlohu z Plánovače úloh



# TaskScheduler

Pokročilé techniky a postupy



```
1 [string] $taskName = "U1_VypniPc"
2 [string] $time = "5:00 PM"
3 [string] $executor = "pwsh.exe"
4 [string] $path = "E:\PSremoting.ps1"
5 [string] $description = "Skript, který každý den po 17:00 vypne všechny počítače v U1."
6
7 $taskExists = Get-ScheduledTask | ? { $_.TaskName -like $taskName }
8
9 if (!$taskExists) {
10     $taskTrigger = New-ScheduledTaskTrigger -Daily -At $time
11     $taskAction = New-ScheduledTaskAction -Execute $executor -Argument $path
12     Register-ScheduledTask -TaskName $taskName -Description $description -Trigger $taskTrigger -Action $taskAction -RunLevel Highest -Force
13 }
```

# TaskScheduler

Pokročilé techniky a postupy



```
1 $taskExists = Get-ScheduledTask | ? { $_.TaskName -like "U1_VypniPc" }
2
3 if ($taskExists) {
4     $newTrigger = New-ScheduledTaskTrigger -Weekly -At "5:30 PM"
5     Set-ScheduledTask -TaskName $taskName -Trigger $newTrigger
6 }
```

# TaskScheduler

Pokročilé techniky a postupy



```
1 $taskExists = Get-ScheduledTask | ? { $_.TaskName -like "U1_VypniPc" }  
2  
3 if ($taskExists) {  
4     Unregister-ScheduledTask -TaskName $taskName -Confirm:$false  
5 }
```

# Obsah

Import uživatelů a jejich správa v AD

1. Importování dat
2. Praktický příklad – zadání
3. Praktický příklad – synchronní přístup
4. Praktický příklad – asynchronní přístup



# Importování dat

Import uživatelů a jejich správa v AD

## Princip

- ❑ data nejsou definována staticky ve skriptu – jsou **dynamicky** přejímána z externího souboru
- ❑ nejčastější formát **CSV** (možnost exportu .xlsx do .csv)
- ❑ např.: tabulka nově nastupujících žáků, zaměstnanců apod.
- ❑ data lze přečíst a tzv. **deserializovat** do kolekce objektů, s nimiž lze dále interně pracovat
- ❑ funkce s parametry jsou v tomto případě zcela ideální – dokáží reagovat na **různorodost** dat

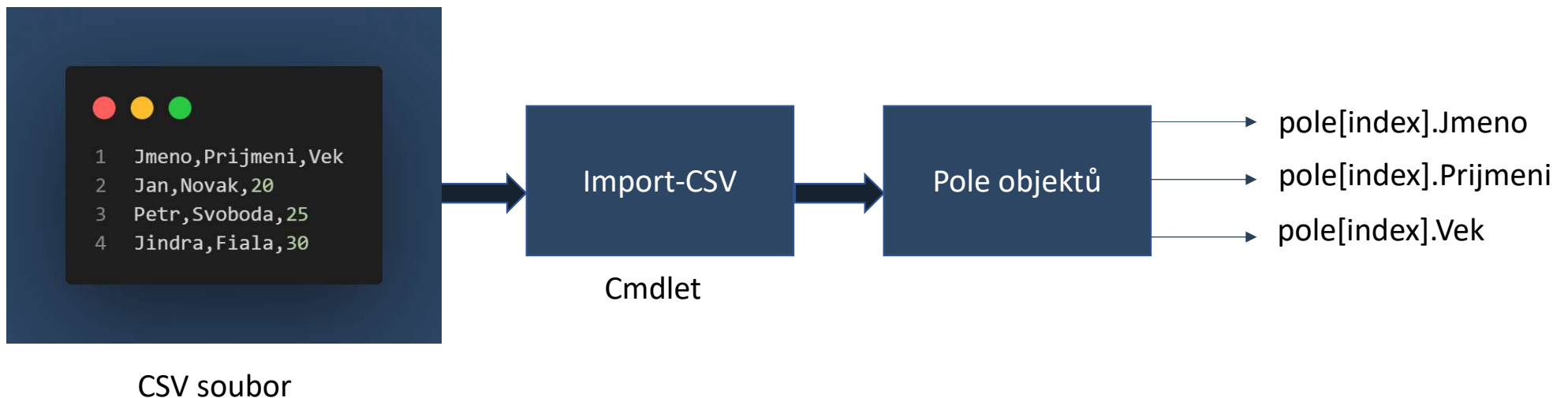




# Importování dat

Import uživatelů a jejich správa v AD

## Proces zpracování CSV souboru - schematicky



# Importování dat

Import uživatelů a jejich správa v AD

## Proces zpracování CSV souboru - kód



```
1 $students = Import-Csv -Path "E:\students.csv"
2
3 $students | % {
4     Write-Host $_.Jmeno
5     Write-Host $_.Prijmeni
6     Write-Host $_.Vek
7 }
```

# Importování dat

Import uživatelů a jejich správa v AD

## Proces zpracování CSV souboru bez hlaviček

```
1 [string[]] $headers = @("Jmeno", "Prijmeni", "Vek")
2
3 $students = Import-Csv -Path "E:\studentsNoHeaders.csv" -Header $headers | Select-Object -Unique $headers
4
5 $students | % {
6     Write-Host $_.Jmeno
7     Write-Host $_.Prijmeni
8     Write-Host $_.Vek
9 }
```

# Praktický příklad

Import uživatelů a jejich správa v AD

## Zadání

Prestižní univerzita má přijmout 500 nových studentů. Tabulku s jejich celými jmény vždy dostáváme ve formátu CSV. Naším prozatímním úkolem bude vytvořit každému z nich co nejefektivněji doménový účet s heslem.

## Možnosti řešení

1. Synchronní přístup
2. Asynchronní přístup



# Praktický příklad – synchronní přístup

Import uživatelů a jejich správa v AD

```
1 #Doména
2 [string] $domainName = "firma"
3 [string] $domainEnding = "local"
4
5 #Hlavní organizační jednotka ROOT
6 [string] $root = "skola"
7
8 #Studenti a jejich umístění
9 [string] $studentsPath = "OU=studenti,OU=uzivatele,OU=$root,DC=$domainName,DC=$domainEnding"
10
11 $students = Import-Csv -Path "E:\studentsList500.csv"
12
13 #Synchronní přístup
14 foreach ($student in $students) {
15     Write-Host "Vytvarim studenta:" $student.Name
16     $generatedPassword = $student.Name.Substring(0, 1).ToUpper() + $student.Name.Substring(0, 1).ToLower() + "123456@"
17     New-ADUser -Name $student.Name -AccountPassword (ConvertTo-SecureString -AsPlainText $generatedPassword -Force) -CannotChangePassword 0
18     -ChangePasswordAtLogon 0 -Enabled 1 -Path $studentsPath
19     Add-ADGroupMember -Identity "studenti" -Members $student.Name
20 }
```

```
Days           : 0
Hours           : 0
Minutes         : 0
Seconds         : 52
Milliseconds    : 840
Ticks           : 528405331
TotalDays       : 0,000611580244212963
TotalHours      : 0,0146779258611111
TotalMinutes    : 0,880675551666667
TotalSeconds    : 52,8405331
TotalMilliseconds : 52840,5331
```

# Praktický příklad – asynchronní přístup

Import uživatelů a jejich správa v AD



```
1 #Doména
2 [string] $domainName = "firma"
3 [string] $domainEnding = "local"
4
5 #Hlavní organizační jednotka ROOT
6 [string] $root = "skola"
7
8 #Studenti a jejich umístění
9 [string] $studentsPath = "OU=studenti,OU=uzivatele,OU=$root,DC=$domainName,DC=$domainEnding"
10
11 $students = Import-Csv -Path "E:\studentsList500.csv"
12
13 #Asynchronní přístup
14 $students | ForEach-Object -Parallel {
15     Write-Host "Vytvarim studenta: " + $_.Name
16     $generatedPassword = $_.Name.Substring(0, 1).ToUpper() + $_.Name.Substring(0, 1).ToLower() + "123456@"
17     New-ADUser -Name $_.Name -AccountPassword (ConvertTo-SecureString -AsPlainText $generatedPassword -Force) -CannotChangePassword 0
18     -ChangePasswordAtLogon 0 -Enabled 1 -Path $using:studentsPath
19     Add-ADGroupMember -Identity "studenti" -Members $_.Name
20 } -ThrottleLimit 2
```

```
Days           : 0
Hours          : 0
Minutes        : 0
Seconds        : 32
Milliseconds    : 314
Ticks          : 323148526
TotalDays      : 0,000374014497685185
TotalHours     : 0,008976347944444444
TotalMinutes   : 0,5385808766666667
TotalSeconds   : 32,3148526
TotalMilliseconds : 32314,8526
```

# Zabezpečení souborového systému

## Základní pojmy

### ☐ **ACL** (Access Control List)

- ☐ seznam uživatelských oprávnění pro soubor, složku nebo jiný objekt
- ☐ definuje, kteří uživatelé a skupiny mohou k objektu přistupovat a jaké operace mohou provádět

### ☐ **ACE** (Access Control Entry)

- ☐ položka seznamu ACL přímo definující oprávnění pro daného uživatele, skupinu či jiný objekt
- ☐ mezi tyto oprávnění obvykle patří **čtení**, **zápis** a **vykonání**



# Zabezpečení souborového systému

Číslo parametru	Typ parametru	Poznámka k parametru
1	IdentityReference/String	Název objektu (uživatel, skupina, počítač, ...), na který se ACE vztahuje
2	FileSystemRights	Název oprávnění (FullControl, Modify, ...), které má ACE aplikovat
3	InheritanceFlags	Příznaky dědičnosti (None, ObjectInherit, ...) určují, jak se z objektu dědí
4	PropagationFlags	Určuje, jak se ACE šíří do podřízených objektů (None, InheritOnly, ...)
5	AccessControlType	Typ ACE (Allow, Deny) - povolující, nebo zakazující



```
1 $ace = New-Object System.Security.AccessControl.FileSystemAccessRule(1. parametr, 2. parametr, 3. parametr, 4. parametr, 5. parametr)
```



# Zabezpečení souborového systému



```
1 # Zobrazení všech možných oprávnění (FullControl, Modify, ReadAndExecute, ...)
2 [enum]::GetValues('System.Security.AccessControl.FileSystemRights')
3
4 # Zobrazení typů dědičnosti (None, ContainerInherit, ObjectInherit)
5 [enum]::GetValues('System.Security.AccessControl.InheritanceFlags')
6
7 # Zobrazení všech flagů (None, NoPropagateInherit, InheritOnly, ...)
8 [enum]::GetValues('System.Security.AccessControl.PropagationFlags')
9
10 # Zobrazení typu oprávnění ACE (Allow, Deny)
11 [enum]::GetValues('System.Security.AccessControl.AccessControlType')
```

# Zabezpečení souborového systému

FileSystemRights	Oprávnění (CZ)	Oprávnění (EN)
FullControl	Úplné řízení	Full Control
ExecuteFile	Procházet složkou / Spouštět soubory	Traverse Folder / Execute File
ReadData	Zobrazovat obsah složky / Číst data	List Folder / Read Data
ReadAttributes	Číst atributy	Read Attributes
ReadExtendedAttributes	Číst rozšířené atributy	Read Extended Attributes
CreateFiles	Vytvářet soubory / Zapisovat data	Create Files / Write Data
AppendData	Vytvářet složky / Připojovat data	Create Folders / Append data
WriteAttributes	Zapisovat atributy	Write Attributes
WriteExtendedAttributes	Zapisovat rozšířené atributy	Write Extended Attributes
DeleteSubdirectoriesAndFiles	Odstraňovat podložky a soubory	Delete Subfolders and Files
Delete	Odstraňovat	Delete
ReadPermissions	Číst oprávnění	Read Permissions

# Zabezpečení souborového systému

InheritanceFlags	Číselné vyjádření	Poznámka
None	0	Objekty toto ACE dědit nebudou
ContainerInherit	1	Pouze kontejnerové objekty (složka) budou dědit toto ACE
ObjectInherit	2	Pouze nekontejnerové objekty (soubory) budou dědit toto ACE

# Zabezpečení souborového systému


PropagationFlags	Číselné vyjádření	Poznámka
None	0	Určuje, že nejsou nastaveny žádné příznaky dědičnosti
NoPropagateInherit	1	Určuje, že ACE není rozšířen na podřízené objekty
InheritOnly	2	Určuje, že se ACE rozšíří pouze do podřízených objektů

# Zabezpečení souborového systému

AccessControlType	Číselné vyjádření	Poznámka
Allow	0	<b>Povolení</b> přístupu k zabezpečenému objektu
Deny	1	<b>Odepření</b> přístupu k zabezpečenému objektu

# Zabezpečení souborového systému

## Zabezpečení disku



```
1 $acl = Get-Acl "W:/"
2
3 $ace = New-Object System.Security.AccessControl.FileSystemAccessRule("BUILTIN\Users","ReadAndExecute","None","None","Allow")
4 $acl.SetAccessRule($ace)
5
6 $ace = Get-Acl $DiskPath | Select-Object -ExpandProperty Access | Where-Object IdentityReference -eq "EVERYONE"
7 $acl.RemoveAccessRule($ace)
8
9 $ace = Get-Acl $DiskPath | Select-Object -ExpandProperty Access | Where-Object IdentityReference -eq "CREATOR OWNER"
10 $acl.RemoveAccessRule($ace)
11
12 $acl | Set-Acl
```

# Zabezpečení souborového systému

## Vyprázdnění ACL



```
1 $acl = Get-Acl "W:/"
2 $acl.Access | % { $acl.RemoveAccessRule($_) }
```

# Zabezpečení souborového systému

## Přerušení dědičnosti



```
1 $acl = Get-ACL "W:/"
2 $acl.SetAccessRuleProtection($True, $True)
3 $acl | Set-Acl
```



# Novinky v PowerShell 7.2

- ❑ verze dlouhodobé podpory (LTS) postavená na .NET 6.0 (každá sudá verze má LTS)
- ❑ prediktivní IntelliSense dle historie příkazů – modul PSReadLine
- ❑ Set-PSReadLineOption -**PredictionSource History**
- ❑ vylepšení podpory ANSI escape kódů – standard pro formátování textu v terminálech
- ❑ přidání objektu **\$PSStyle** pro přístup k atributům pro zmíněné formátování



# Zdroje

Your Getting Started Guide to Powershell Functions. ATA Learning - High quality, in-depth, no-fluff how-to articles on IT, sysadmin, cloud and DevOps. [online]. Copyright © ATA Learning [cit. 23.02.2022]. Dostupné z: <https://adamtheautomator.com/powershell-functions/>

What's New in PowerShell 7.2 - PowerShell | Microsoft Docs. [online]. Copyright © Microsoft 2022 [cit. 06.03.2022]. Dostupné z: <https://docs.microsoft.com/en-us/powershell/scripting/whats-new/what-s-new-in-powershell-72?view=powershell-7.2>

File:OneDrive Folder Icon.svg - Wikimedia Commons. [online]. Dostupné z: [https://commons.wikimedia.org/wiki/File:OneDrive\\_Folder\\_Icon.svg](https://commons.wikimedia.org/wiki/File:OneDrive_Folder_Icon.svg)

How to Manage File System ACLs with PowerShell Scripts. [online]. Copyright @ Netwrix Corporation 2022 <https://blog.netwrix.com/2018/04/18/how-to-manage-file-system-acls-with-powershell-scripts/>

Invoke-Command: The Best Way to Run Remote Code. *ATA Learning - High quality, in-depth, no-fluff how-to articles on IT, sysadmin, cloud and DevOps.* [online]. Copyright © ATA Learning [cit. 10.05.2022]. Dostupné z: <https://adamtheautomator.com/invoke-command/>

Why is multithreading faster? - Quora. *Quora - A place to share knowledge and better understand the world* [online]. Dostupné z: <https://www.quora.com/Why-is-multithreading-faster>

Run Commands in Parallel in PowerShell | Delft Stack. *Best Tutorial About Python, Javascript, C++, GIT, and more – Delft Stack* [online]. Copyright © 2020. All right reserved [cit. 13.06.2022]. Dostupné z: <https://www.delftstack.com/howto/powershell/run-commands-in-parallel-in-powershell/>

Definice a význam ACL (Access Control List) - SharTec. *SharTec - Slovníček technických pojmů* [online]. Copyright © 2022 SharTec [cit. 14.06.2022]. Dostupné z: <https://cz.shartec.eu/acl/>

PropagationFlags Výčet (System.Security.AccessControl) | Microsoft Docs. [online]. Copyright © Microsoft 2022 [cit. 14.06.2022]. Dostupné z: <https://docs.microsoft.com/cs-cz/dotnet/api/system.security.accesscontrol.propagationflags?view=net-6.0>

PowerShell Setting advanced NTFS permissions - Stack Overflow. *Stack Overflow - Where Developers Learn, Share, & Build Careers* [online]. Dostupné z: <https://stackoverflow.com/questions/26543127/powershell-setting-advanced-ntfs-permissions>

powershell - save PSCredential in the file - Stack Overflow. *Stack Overflow - Where Developers Learn, Share, & Build Careers* [online]. Dostupné z: <https://stackoverflow.com/questions/40029235/save-pscredential-in-the-file>

# Zdroje

How To Execute Parallel Script Blocks in PowerShell -- Redmondmag.com. [online]. Dostupné z: <https://redmondmag.com/articles/2020/08/20/parallel-script-blocks-in-powershell.aspx>

PowerShell ForEach-Object Parallel Feature - PowerShell Team. *DevBlogs - Microsoft Developer Blogs* [online]. Dostupné z: <https://devblogs.microsoft.com/powershell/powershell-foreach-object-parallel-feature/>

Důležité informace o zabezpečení vzdálené komunikace PowerShellu s využitím WinRM - PowerShell | Microsoft Docs. [online]. Copyright © Microsoft 2022 [cit. 13.06.2022]. Dostupné z: <https://docs.microsoft.com/cs-cz/powershell/scripting/learn/remoting/winrmsecurity?view=powershell-7.2>

How to Automate PowerShell Scripts with Task Scheduler. *Netwrix Blog | Insights for Cybersecurity and IT Pros* [online]. Copyright © 2022 Netwrix Corporation. All rights reserved. [cit. 13.06.2022]. Dostupné z: <https://blog.netwrix.com/2018/07/03/how-to-automate-powershell-scripts-with-task-scheduler/>

Test-WSMan (Microsoft.WSMan.Management) - PowerShell | Microsoft Docs. [online]. Copyright © Microsoft 2022 [cit. 07.06.2022]. Dostupné z: <https://docs.microsoft.com/en-us/powershell/module/microsoft.wsman.management/test-wsman?view=powershell-7.2>

Get-Credential (Microsoft.PowerShell.Security) - PowerShell | Microsoft Docs. [online]. Copyright © Microsoft 2022 [cit. 07.06.2022]. Dostupné z: <https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.security/get-credential?view=powershell-7.2>

How to Run PowerShell Commands on Remote Computers . *How-To Geek - We Explain Technology* [online]. Copyright © 2022 LifeSavvy Media. All Rights Reserved [cit. 07.06.2022]. Dostupné z: <https://www.howtogeek.com/117192/how-to-run-powershell-commands-on-remote-computers/>

Remove-Job (Microsoft.PowerShell.Core) - PowerShell | Microsoft Docs. [online]. Copyright © Microsoft 2022 [cit. 15.06.2022]. Dostupné z: <https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/remove-job?view=powershell-7.2>

Disconnect-PSSession (Microsoft.PowerShell.Core) - PowerShell | Microsoft Docs. [online]. Copyright © Microsoft 2022 [cit. 15.06.2022]. Dostupné z: <https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/disconnect-pssession?view=powershell-7.2&viewFallbackFrom=powershell-6>

How to Connect Windows to Linux and Linux to Windows using PowerShell 7 SSH Remoting ( PS Remoting Over SSH) | DevOps Automateinfra Learning. *DevOps Automateinfra Learning | Most Trending and Quality DevOps Cloud Automation tutorials, guides and learning* [online]. Dostupné z: <https://automateinfra.com/2021/04/12/how-to-connect-windows-to-linux-and-linux-to-windows-using-ssh-powershell-remoting/>

# Zdroje

Set-ScheduledTask (ScheduledTasks) | Microsoft Docs. [online]. Copyright © Microsoft 2022 [cit. 20.06.2022]. Dostupné z: <https://docs.microsoft.com/en-us/powershell/module/scheduledtasks/set-scheduledtask?view=windowsserver2022-ps>

PowerShell: Import-CSV with no headers and remove partial duplicate lines - Stack Overflow. *Stack Overflow - Where Developers Learn, Share, & Build Careers* [online]. Dostupné z: <https://stackoverflow.com/questions/20526095/powershell-import-csv-with-no-headers-and-remove-partial-duplicate-lines>

How to Connect Windows to Linux and Linux to Windows using PowerShell 7 SSH Remoting ( PS Remoting Over SSH) | DevOps Automateinfra Learning. *DevOps Automateinfra Learning | Most Trending and Quality DevOps Cloud Automation tutorials, guides and learning* [online]. Dostupné z: <https://automateinfra.com/2021/04/12/how-to-connect-windows-to-linux-and-linux-to-windows-using-ssh-powershell-remoting/>

Introducing PSRemoting Linux and Windows : How to Guide. *ATA Learning - High quality, in-depth, no-fluff how-to articles on IT, sysadmin, cloud and DevOps.* [online]. Copyright © ATA Learning [cit. 20.06.2022]. Dostupné z: <https://adamtheautomator.com/psremoting-linux/>

Vzdálená komunikace PowerShellu přes SSH - PowerShell | Microsoft Docs. [online]. Copyright © Microsoft 2022 [cit. 20.06.2022]. Dostupné z: <https://docs.microsoft.com/cs-cz/powershell/scripting/learn/remoting/ssh-remoting-in-powershell-core?view=powershell-7.2>

How to SSH into Linux Machine using Windows 10 PowerShell - YouTube. *YouTube* [online]. Copyright © 2022 Google LLC [cit. 20.06.2022]. Dostupné z: [https://www.youtube.com/watch?v=ql3nGTjyqGI&ab\\_channel=CodeSOS](https://www.youtube.com/watch?v=ql3nGTjyqGI&ab_channel=CodeSOS)

PowerShell Remoting from Linux to Windows - Quickbreach Blog. *Home - Quickbreach Blog* [online]. Copyright © All rights reserved. [cit. 20.06.2022]. Dostupné z: <https://blog.quickbreach.io/blog/powershell-remoting-from-linux-to-windows/>

# Děkuji za pozornost

Jsou-li nějaké dotazy, ptejte se! 😊