ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA ELEKTROTECHNICKÁ

KATEDRA ELEKTROMECHANIKY A VÝKONOVÉ ELEKTRONIKY

BAKALÁŘSKÁ PRÁCE

Správa Windows v prostředí počítačové sítě

Lukáš Hofman **2013**

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta elektrotechnická
Akademický rok: **2012/2013**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Lukáš HOFMAN**
Osobní číslo:	**E10B0020K**
Studijní program:	**B2644 Aplikovaná elektrotechnika**
Studijní obor:	**Aplikovaná elektrotechnika**
Název tématu:	**Správa Windows v prostředí počítačové sítě**
Zadávající katedra:	**Katedra elektromechaniky a výkonové elektroniky**

Zásady pro vypracování:

1. Možnosti správy a konfigurace v prostředí Windows.
2. Automatizace nastavení a ovládání. Ukázky skriptů a zdrojových kódů. Rozdíly mezi verzemi Windows.
3. Uveďte možnosti skriptování pro adresářovou službu Active Directory.
4. Srovnání se správou OS Linux.

Rozsah grafických prací: **podle doporučení vedoucího**

Rozsah pracovní zprávy: **20 - 30 stran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

Student si vhodnou literaturu vyhledá v dostupných pramenech podle doporučení vedoucího práce.

Vedoucí bakalářské práce: **Ing. Jiří Basl, Ph.D.**
Katedra aplikované elektroniky a telekomunikací

Datum zadání bakalářské práce: **15. října 2012**

Termín odevzdání bakalářské práce: **7. června 2013**

Doc. Ing. Jiří Hammerbauer, Ph.D.
děkan

L.S.

Prof. Ing. Václav Kůs, CSc.
vedoucí katedry

V Plzni dne 15. října 2012

Abstrakt

Tato bakalářská práce se zabývá možnostmi administrace, automatizace a vzdálené správy operačního systému Windows a adresářové služby Active Directory. Využívají se převážně prostředky, které jsou přímo součástí Windows. Jsou zde popsány rozdíly mezi různými verzemi Windows. Dále je porovnána správa systému Windows a Linux. V práci jsou obsaženy ukázky příkazů, dávkových souborů a skriptů.

Klíčová slova

Operační systém, Microsoft Windows, administrace, automatizace, vzdálená správa, dávkový soubor, skript, příkazový řádek, WSH, PowerShell, COM, WMI, registr, Active Directory, ADSI, Linux.

Abstract

Managing Windows in a Network

This thesis deals with the possibilities of administration, automation and remote managing of Windows and Active Directory service. There are used mainly tools those are part of Windows. Differences between versions of Windows are mentioned. Also administration of Windows and Linux is compared. Thesis contains examples of commands, batch files and scripts.

Key words

Operating system, Microsoft Windows, administration, automation, remote administration, batch file, script, command line, WSH, PowerShell, COM, WMI, registry, Active Directory, ADSI, Linux.

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně, s použitím odborné literatury a pramenů uvedených v seznamu, který je součástí této bakalářské práce.

Dále prohlašuji, že veškerý software, použitý při řešení této bakalářské práce, je legální.

.....

podpis

V Plzni dne 27.5.2013 Lukáš Hofman

Obsah

Seznam zkratek

AD ..... Active Directory

ADSI..... Active Directory Service Interfaces

COM ..... Component Object Model

LDAP..... Lightweight Directory Access Protocol

PC ..... počítač

RSAT..... Remote Server Administration Tools

WMI ..... Windows Management Instrumentation

WSH ..... Windows Script Host

W2008 ..... Windows Server 2008

W7 ..... Windows 7

XP ..... Windows XP

<*text*>..... popisný text v ostrých závorkách zastupuje nějaký konkrétní název, např. zkratka <*PC*> nahrazuje jméno určitého počítače; obsahuje-li konkrétní název mezery, je nutné jej uzavřít do uvozovek

Úvod

Většina dnešních operačních systémů se ovládá primárně prostřednictvím grafického uživatelského rozhraní. Nejinak je tomu u operačního systému Microsoft Windows. Jedná se totiž o pohodlný způsob, nevyžadující hlubší znalosti. Není to však příliš vhodná a efektivní metoda pro konfigurování systému či provádění opakovaných činností. Obzvlášť při správě většího počtu počítačů je vhodné používat takové prostředky, které potřebné úkony automatizují. V prostředí počítačové sítě je dále výhodné aplikovat postupy, umožňující vzdálenou správu počítačů. Větší sítě na bázi Windows často využívají adresářovou službu Active Directory, kterou je žádoucí také ovládat automatizovaně. Windows naštěstí poskytuje dostatek možností, jak toho všeho dosáhnout.

Cílem této práce je ukázat možnosti administrace, automatizace a vzdálené správy operačního systému Windows a adresářové služby Active Directory. Využijí se nástroje, které jsou přímo součástí Windows, případně se dají zdarma doinstalovat. Zmiňovány jsou rozdíly a odlišnosti mezi různými verzemi Windows, zejména mezi nejrozšířenějšími systémy XP a W7. Fakta uváděná o W7 platí většinou i pro verzi Vista. Na závěr je porovnána správa systému Windows a Linux.

V této práci jsou obsaženy četné ukázky příkazů, dávkových souborů a skriptů. Pro fungování většiny z nich je nutné splnit několik podmínek: Windows v edici Professional nebo vyšší, uživatel s administrátorským oprávněním na lokálním resp. vzdáleném počítači, pro Windows ve verzi Vista a vyšší vypnuté řízení uživatelských účtů (User Account Control) případně spouštět příkazy se zvýšeným oprávněním volbou „Spustit jako správce“.

1 Příkazový řádek Windows

Příkazový řádek (Command Prompt) je tradiční součástí Windows. Ve starých verzích Windows (95, 98, Me) byl tvořen 16bitovým systémem MS-DOS a spouštěl se programem *command.com*. V novějších verzích se již jedná o běžnou konzolovou aplikaci (nemá grafické rozhraní, pracuje s textovým výstupem) tvořenou souborem *cmd.exe*.

Nejrychlejší cestou ke spuštění příkazového řádku je otevřít dialogové okno „Spustit" klávesovou zkratkou Win+R a zadat *cmd*. Další možnost je v Průzkumníku Windows kliknout na zvolenou složku pravým tlačítkem myši při současném stisknutí klávesy Shift a poté v kontextovém menu vybrat položku „Zde otevřít příkazové okno", čímž se otevře příkazový řádek a přednastaví se v něm zvolená složka jako pracovní. V XP tato volba v menu není, ale lze ji přidat příkazem `reg add "HKCR\Folder\shell\Zde otevřít příkazové okno\command" /ve /d "cmd /k cd %1"`.

Příkazový řádek zobrazuje tzv. výzvu (prompt) a čeká na zadání příkazu. Výzva se standardně skládá z názvu aktuální pracovní složky a znaku „>". Při běžném spuštění příkazového řádku je jako aktuální složka nastavena složka profilu uživatele. Výzvu příkazového řádku je možné změnit příkazem *prompt* následovaným textem, který se má zobrazovat. V textu lze použít speciální kombinace kódů. Vhodné je k standardní výzvě PG přidat kódy $+$M příkazem `prompt $+$M$P$G`. Kód $+ zajistí zobrazení znaků „+" v případě použití příkazu *pushd* (viz kapitola níže) a kód $M zobrazí u namapované síťové jednotky skutečnou síťovou cestu. Pro trvalé uložení výzvy je nutné nastavit systémovou proměnnou PROMPT, ve W7 to lze provést třeba příkazem `setx PROMPT $+$M$P$G /M`.

1.1 Práce s cestami

Při práci se soubory a složkami je možné zadávat úplné (absolutní) cesty, ale jednodušší je pracovat v rámci aktuální složky a uvádět relativní cesty. Ty nepopisují celou cestu od kořenové složky jednotky, ale od té aktuální. V cestách lze používat zástupné znaky (***** = libovolný počet znaků, **?** = jeden znak) a speciální označení složek (**.** = aktuální, **..** = nadřazená, \ = kořenová).

Pro změnu aktuální pracovní složky se používá většinou příkaz *cd* (např. `cd C:\Windows\System32`). Změna aktuální jednotky se provádí zadáním písmene jednotky následované dvojtečkou (např. `D:`). Současná změna složky i jednotky se docílí příkazem *cd* s přepínačem /d (např. `cd /d D:\Data`). Dále je k dispozici dvojice příkazů *pushd* a *popd*.

Příkaz *pushd* následovaný cestou nejen změní aktuální složku a jednotku, ale zároveň si zapamatuje předchozí. Po zadání `popd` dojde k návratu zpět do předešlého umístění. Příkaz *pushd* také umožňuje nastavit jako aktuální síťovou složku, což provádí namapováním této složky jako síťové jednotky, tj. obdobně jako příkaz `net use * \\PC\sdilena_slozka`.

1.2 Příkazy

Seznam základních příkazů příkazové řádky lze vypsat pomocí `help`. Nápověda ke konkrétnímu příkazu se získá zadáním tohoto příkazu s přepínačem /? (např. `dir /?`). To většinou platí i pro utility získané od třetích stran, ale setkat se lze také s přepínači -?, -h nebo --help. V případě nástroje *net* (slouží zejména pro práci se sítí) obdržíme podrobnou nápovědu k dané operaci zadáním `net help <operace>` (např. `net help use`).

Příkazový řádek rozlišuje dva typy příkazů, a to interní a externí. Interní příkazy (např. dir, copy) jsou integrovány přímo v příkazovém řádku (programu *cmd.exe*), jde je používat pouze v něm nebo v dávkových souborech. Naopak externí příkazy (např. ipconfig, xcopy) jsou samostatné konzolové aplikace (programy).

Při každém zadání příkazu se vyhodnotí, o jaký typ příkazu se jedná. Pokud jde o interní příkaz, je ihned vykonán. V opačném případě bude zadaný text považován za název programu. Program je nejprve hledán v aktuální složce. Není-li nalezen, jsou prohledány další složky uvedené v systémové proměnné PATH. Tato proměnná obsahuje zejména složku „C:\Windows\System32“, ve které je uložena většina externích programů příkazového řádku. Pokud nebyla zadána přípona programu, jsou při prohledávání složek postupně dosazovány přípony dle proměnné PATHEXT. Ta ve výchozím nastavení obsahuje ve W7 řetězec „.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC“, v XP přípona MSC chybí. Obsah proměnných lze vypsat příkazem `echo %<proměnná>%`.

W7 obsahuje oproti XP řadu nových externích příkazů, např.: bcdedit, choice, clip, cmdkey, forfiles, icacls, mklink, ocsetup, pnpunattend, pnputil, query, quser, robocopy, rpcping, setx, takeown, timeout, waitfor, wbadmin, wevtutil, where, whoami. Naopak některé příkazy byly zrušeny, jako třeba: eventtriggers, net send, net print, netsh diag, ntbackup.

1.3 Přesměrování

Výstup příkazů standardně probíhá na obrazovku (konzolu) a vstup je přijímán z klávesnice, avšak je možné to změnit pomocí přesměrování. Výstup se často přesměrovává do souboru, ale použít lze i jiné výstupní zařízení, např. nulové zařízení (NUL) či tiskárnu.

K přesměrování výstupu se používá symbol „>“ a pro připojení výstupu na konec existujícího souboru je třeba použít „>>“. Vedle standardního výstupu (stdout) odesílají mnohé příkazy různá chybová hlášení na chybový výstup (errout), který je standardně vypisován také na obrazovku. Pro přesměrování chybového výstupu se před symbol přesměrování přidává číslo chybového popisovače „2“. Také standardní vstup z klávesnice může být přesměrován a nahrazen daty ze souboru. K přesměrování vstupu slouží symbol „<“.

Ukázky použití přesměrování (příkaz *dir* vypisuje obsah složky, příkaz *sort* seřazuje vstupní text):

- přesměrování standardního výstupu do souboru „vypis.txt“ a chybového výstupu (např. zprávy „Soubor nebyl nalezen.“) do souboru „chyby.txt“

```
dir *.bat > vypis.txt 2> chyby.txt
```

- přesměrování a připojení standardního i chybového výstupu do souboru „vypis.txt“

```
dir *.cmd >> vypis.txt 2>&1
```

- přesměrování chybového výstupu do virtuálního nulového zařízení (do ztracena), standardní výstup se zobrazí normálně na obrazovce

```
dir C:\neexistuje 2> NUL
```

- přesměrování vstupu (načtení textu) ze souboru „vypis.txt“ vytvořeného výše a následné přesměrování výstupu (seřazených řádek dle časového údaje) do souboru „serazeno.txt“

```
sort /+13 < vypis.txt > serazeno.txt
```

1.4 Roura a spojování příkazů

Výstup z jednoho programu lze předat na vstup dalšího (pokud to podporuje), stačí je spojit symbolem „|“ nazývaným roura (pipe). Takto lze řetězit za sebe libovolný počet příkazů. Ukázky použití roury:

- výpis běžících procesů (příkaz tasklist) se seřadí (příkaz sort) a v případě dlouhého výpisu se bude stránkovat (příkaz more)

```
tasklist /nh | sort | more
```

- zobrazení počtu aktuálně spuštěných služeb

```
net start | find /c "   "
```

- vypsání aktuálních síťových připojení typu „ftp“ a „microsoft-ds“

```
netstat | findstr ":ftp :microsoft-ds"
```

Na jeden řádek je možné zadat více příkazů pomocí spojovacího symbolu „&“. Také lze nastavit závislost následujícího příkazu na návratovém kódu (errorlevel) předchozího. Symbol „&&“ zajistí spuštění dalšího příkazu pouze v případě úspěchu (návratovém kódu 0) předcházejícího příkazu. Přesně obrácenou funkci má symbol „||“.

Ukázka spojování příkazů (při dostupnosti počítače PC1 se vypíše text „Dostupny“, v opačném případě se vypíše „Nedostupny“ a dojde k ukončení dávkového souboru):

```
ping -n 1 PC1 >NUL && echo Dostupny || (echo Nedostupny & exit /b)
```

1.5 Dávkové soubory

Dávkové soubory (batch files) jsou soubory s příponou BAT nebo CMD, které obsahující posloupnost příkazů pro příkazový řádek. Jde v nich používat i jednoduché programové konstrukce (větvení, cyklus), ovšem nedá se hovořit o nějakém pokročilém skriptovacím jazyce.

Dávkový soubor je možné spustit s parametry, stejně jako jiné programy a příkazy. Parametry se zapisují za název souboru a oddělují se mezerami (<*dávkový soubor*> <*1.parametr*> <*2.parametr*>…). K jednotlivým předaným parametrům se poté přistupuje prostřednictvím speciálních proměnných %1 až %9. Všechny parametry dohromady lze získat pomocí %*. Existuje také proměnná %0, která obsahuje samotný název dávkového souboru. Je-li parametrem název souboru, je možné využít modifikátory parametrů pomocí konstrukce %~<*modifikátory*><*číslo parametru*>. Nejčastěji používané modifikátory jsou: **d** = písmeno jednotky, **p** = cesta, **n** = jméno souboru, **x** = přípona, **z** = velikost. Např. proměnná %~dp0 obsahuje plnou cestu k umístění samotného dávkového souboru, %~nx1 znamená jméno souboru (včetně přípony) předaného v prvním parametru a %~z2 vrátí velikost souboru uvedeného v druhém parametru.

Následující ukázka dávkového souboru vypíše jméno uživatele, přihlášeného na vzdáleném počítači. Jméno počítače lze předat prostřednictvím parametru nebo jej zadat až po spuštění dávkového souboru. Jméno přihlášeného uživatele se zjišťuje z registru vzdáleného počítače. Šlo by to samozřejmě realizovat i jednodušeji pomocí rozhraní WMI příkazem `wmic /node:%PC% ComputerSystem get UserName`, ale cílem ukázky je hlavně demonstrovat možnosti dávkových souborů.

```
:: vypnutí výpisu příkazů na obrazovku
@echo off

:: pokud nebylo předáno jméno počítače prostřednictvím parametru,
 zobrazí se výzva k jeho zadání
if "%1"=="" (set /P PC=Zadejte PC: || exit /b) else (set PC=%1)

:: pokud je PC nedostupný, vypíše se zpráva na chybový výstup a
 skočí se na návěští KONEC
ping -n 1 -w 1 %PC% >NUL || (echo PC nedostupny. >&2 & goto KONEC)

:: spuštění služby "Vzdálený registr" na vzdáleném PC
sc \\%PC% start RemoteRegistry >NUL

:: načtení jména přihlášeného uživatele ze vzdáleného registru
for /F %%A in ('reg query \\%PC%\HKU ^| findstr /R /B
 /C:"HKEY_USERS\\S-1-5-[0-9][0-9]-[0-9-]*$"') do (
  if exist \\%PC%\C$\Users (
    :: pro vzdálený W7
    for /F "tokens=2*" %%B in ('reg query "\\%PC%\%%~A\Volatile
 Environment" /V UserName') do  set USER=%%C
  ) else (
    :: pro vzdálený XP
    for /F "tokens=3 delims=\" %%B in ('reg query
 "\\%PC%\%%A\Volatile Environment"') do  set USER=%%B
  )
)

if defined USER (
  echo Je prihlasen uzivatel "%USER%".
) else (
  echo Nikdo neni prihlasen.
)

:KONEC

:: pokud byl dávkový soubor spuštěn přímo z prostředí Windows,
 pozastaví se okno s výpisem
echo %CMDCMDLINE% | find "%~0" >NUL  && pause
```

2 Windows Script Host (WSH)

WSH je všestranné skriptovací prostředí. Nabízí skriptovací jazyk *VBScript* (Microsoft Visual Basic Scripting Edition) a *JScript* (implementace jazyka JavaScript), ale podporuje i další doinstalované jazyky jako např. Perl či Python. Soubory skriptů v jazyce VBScript mají příponu VBS a pro JScript je to přípona JS. Ukázky skriptů v jazyce VBScript jsou použity v celé této práci.

Skripty jsou dobrým kompromisem mezi komplexností plnohodnotného programovacího jazyka, který se může zkompilovat do spustitelné podoby, a snadným používáním dávkových souborů. Stručně řečeno, skripty jsou na půl cesty mezi dávkovými soubory a plnohodnotnými programovacími jazyky. Oproti dávkovým souborům mají podstatnou výhodu v tom, že se v nich můžou používat objekty. [1]

2.1 Objekty

Platforma WSH je založená na objektech. Hlavní objekt *WScript* je přístupný přímo. Obsahuje např. metodu *Echo* pro zobrazení textu na obrazovce nebo vlastnost *Arguments* pro získání parametrů z příkazové řádky. Další objekty se musí nejprve vytvořit (přiřadit nějaké proměnné), ve VBScriptu příkazem `Set proměnná = CreateObject("objekt")` a v jazyce JScript pomocí příkazu `proměnná = WScript.CreateObject("objekt")` případně `proměnná = new ActiveXObject("objekt")`. Přímo součástí WSH jsou objekty *WScript.Network* (pracuje se zdroji v síti), *WScript.Shell* (slouží např. pro práci s registrem) a *Scripting.FileSystemObject* (ovládá souborový systém). Prostřednictvím rozhraní COM (viz kapitola 4) lze využít libovolné další objekty přítomné ve Windows nebo poskytnuté některými aplikacemi.

Ukázka využití objektu *WScript.Network* v jazyce VBScript (trvalé namapování síťové složky jako jednotky „N:“ a připojení síťové tiskárny):

```
Set WshNetwork = CreateObject("WScript.Network")
WshNetwork.MapNetworkDrive "N:", "\\PC\sdilena_slozka", True
WshNetwork.AddWindowsPrinterConnection "\\PC\tiskarna"
```

2.2 Interprety skriptů

Pro spouštění skriptů jsou k dispozici dva interprety. Výchozí *WScript* běží v grafickém uživatelském prostředí, zatímco *CScript* pracuje v příkazovém řádku. Oba mají shodné přepínače, jejich seznam se zobrazí příkazem `wscript /?` resp. `cscript /?`.

Interpret CScript je nutné použít zejména při rozsáhlejších výstupech, realizovaných příkazem *WScript.Echo*, protože jinak by se jednotlivé řádky výpisu zobrazovaly postupně prostřednictvím grafických oken. Při použití interpretu CScript se standardně vypisuje logo, obsahující informaci o verzi WSH a autorská práva Microsoftu, což často znepřehledňuje žádoucí výstup skriptu. Tomuto jde zamezit přidáním přepínače /Nologo (`cscript` **`/Nologo`** *<soubor skriptu>*). Trvalé potlačení zobrazování loga lze nastavit příkazem `cscript /Nologo /S` (platí pouze pro aktuálně přihlášeného uživatele) nebo `reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows Script Host\Settings" /v DisplayLogo /d 0 /f`.

2.3 Šifrování skriptů

Zajímavá možnost je skripty zašifrovat a zabránit tak jejich prohlížení zvídavými uživateli. K tomuto účelu může posloužit následující skript v jazyce VBScript, využívající objekt *Scripting.Encoder*.

```
If WScript.Arguments.Count = 0 Then WScript.Quit

SourcePath = WScript.Arguments(0)

Set FSO = CreateObject("Scripting.FileSystemObject")

SourceExt = "." & FSO.GetExtensionName(SourcePath)

Select Case LCase(SourceExt)
  Case ".vbs": DestExt = ".vbe"
  Case ".js":  DestExt = ".jse"
  Case Else
    WScript.Echo "Zdrojový soubor není podporovaného typu."
    WScript.Quit
End Select

Text = FSO.OpenTextFile(SourcePath).ReadAll

Set objEncoder = CreateObject("Scripting.Encoder")
EncodedText = objEncoder.EncodeScriptFile(SourceExt, Text, 0, "")

DestPath = Replace(SourcePath, SourceExt, DestExt)

Const ForWriting = 2
FSO.OpenTextFile(DestPath, ForWriting, True).Write EncodedText
```

Uložením výše uvedeného do souboru „ScriptEncoder.vbs“ a spuštěním příkazu `ScriptEncoder.vbs <soubor skriptu>` dojde k vytvoření zašifrovaného skriptu ve složce s původním skriptem. Obsah skriptu je změněn na nečitelnou změť znaků při zachování jeho funkčnosti. Zašifrované soubory v jazyce VBScript mají příponu VBE a v případě JScriptu příponu JSE. Je však důležité upozornit, že lze najít i nástroje pro dešifrování skriptů, takže je nutné brát toto zabezpečení s rezervou a neuvádět ve skriptech citlivé údaje jako třeba přístupová hesla. Citujme slova jednoho z největších znalců skriptování ve Windows, Lee Holmese: „Uvažte, proč chcete skript šifrovat. Máte v něm heslo? To je špatně, zeptejte se na něj v průběhu skriptu v dialogu. Nechcete interakci, ale úplnou automatizaci? Použijte Windows Scheduled Tasks, která poběží pod potřebným vyšším účtem. Nejde to ani takhle? Pak to máte celé navržené špatně, a znovu si ten návrh skriptu promyslete!“

3 PowerShell

Jedná se o poměrně nový příkazový interpret pro Windows, jehož první verze byla vydána v roce 2006. Přináší vynikající skriptovací jazyk s řadou propracovaných konstrukcí. PowerShell je postaven na platformě *Microsoft .NET Framework*, jejíž rozsáhlé možnosti plně využívá. Umožňuje též jednoduchý přístup k rozhraní COM a WMI. Příkazy PowerShellu předávají výstupní data ve formě objektů. S tím je také spojeno používání objektové roury místo standardní textové, kterou využívá příkazový řádek Windows či systémy Linux. Prostředí PowerShellu je tvořeno spustitelným souborem *powershell.exe*, umístěným ve složce „C:\Windows\System32\WindowsPowerShell\v1.0“.

V současné době existuje PowerShell již ve verzi 3.0, která je integrována do Windows 8 a Windows Server 2012. Součástí W7 a W2008 R2 je verze 2.0. W2008 obsahuje první verzi tohoto interpretu. Starší systémy Windows (XP, 2003, Vista) standardně PowerShell neobsahují, ale dá se stáhnout a doinstalovat jako balíček *Windows Management Framework Core*, který zároveň zahrnuje službu WinRM pro vzdálené připojení. Úspěšná instalace do XP je podmíněna přítomností balíku *Service Pack 3* a platformy *.NET Framework* minimálně ve verzi 2.0.

PowerShell se používá nejen pro správu operačního systému Windows, ale s využitím doplňkových modulů také pro další produkty společnosti Microsoft (Active Directory, Exchange, Internet Information Services, SQL Server, SharePoint, Hyper-V, System Center). Důležitou roli hraje při správě systému Windows Server v odlehčené verzi *Core*, která postrádá grafické konfigurační rozhraní.

Součástí PowerShellu je podrobná nápověda. Základní údaje o určitém příkazu lze získat zadáním `help <příkaz>`, přidáním přepínače `-Full` se vypíšou všechny dostupné informace včetně příkladů použití. Kompletní přehled všech dostupných příkazů, aliasů a souborů nápovědy lze vypsat prostřednictvím `help *`. Jelikož jsou výstupy příkazů PowerShellu ve formě objektů, vzniká často potřeba prozkoumat dostupné vlastnosti a metody daného objektu. K tomuto účelu se používá příkaz *Get-Member*, jehož použití je následující: `<příkaz> | Get-Member`.

3.1 Příkazy

Interní příkaz PowerShellu se nazývá *cmdlet*. Běží přímo v rámci samotného PowerShellu, není kvůli němu spouštěn žádný nový proces. Název cmdletu je složen ze

slovesa a podstatného jména (samozřejmě v angličtině), což umožňuje výstižné pojmenování příkazů srozumitelné pro člověka. Alternativně lze využít tzv. *alias*, což je jiný název pro daný příkaz. Např. místo cmdletu *Get-ChildItem* jde použít aliasy *dir*, *gci* nebo *ls* (zjištěno příkazem `Get-Alias | Where-Object {$_.Definition -eq "Get-ChildItem"} | Format-Table Name`).

Většině cmdletů lze předat určité přepínače a parametry, použití je potom následující: *cmdlet -přepínač parametr,parametr...* Následující ukázka vypíše běžící procesy začínající řetězcem „svc“ nebo končící na „ss“: `Get-Process` **`-Name svc*,*ss`**. V tomto případě však není uvedení přepínače –Name povinné, neboť se jedná o „pozicí určený vstupní parametr“, takže příkaz může vypadat i takto: `Get-Process` **`svc*,*ss`**.

Přestože PowerShell ve verzi 2 disponuje velkým množstvím vlastních příkazů, v některých oblastech (např. sdílení zdrojů v síti) je stále nutné využívat jak objekty navržené pro WSH, tak třídy rozhraní WMI. Následuje několik příkladů:

- využití objektu rozhraní WSH pro trvalé namapování síťové složky jako jednotky „N:“ a pro připojení síťové tiskárny

```
$WshNetwork = New-Object -COM WScript.Network
$WshNetwork.MapNetworkDrive("N:", "\\PC\sdilena_slozka", $true)
$WshNetwork.AddWindowsPrinterConnection("\\PC\tiskarna")
```

- využití rozhraní WMI pro nasdílení složky a vypsání přehledu sdílených prostředků

```
([wmiclass] "Win32_Share").Create("C:\Data", "sdilena_slozka", 0)
Get-WmiObject Win32_Share
```

Z prostředí PowerShellu je možné spouštět také externí programy a skripty, stejně jako v klasickém příkazovém řádku.

3.2 Skripty

Skripty PowerShellu jsou textové soubory s příponou PS1, které obsahující patřičné příkazy a povely pro tento interpret. Jejich spouštění je však z bezpečnostních důvodů ve výchozím nastavení zakázáno. Pro povolení spuštění skriptu jde aplikovat příkaz `powershell` **`-ExecutionPolicy Bypass`** `-File <soubor skriptu>`. Trvalého povolení lze docílit příkazem `powershell Set-ExecutionPolicy Bypass`. Soubory skriptů nejsou s PowerShellem asociovány přímo, ale pro jejich spuštění je třeba vybrat z kontextového menu souboru volbu „Run with PowerShell“, případně použít příkaz

`powershell -File <soubor skriptu>`. Skript je samozřejmě možné spustit také přímo z příkazového prostředí PowerShellu zadáním jména souboru skriptu.

Pro vytváření, úpravy a ladění skriptů je výhodné používat prostředí *Windows PowerShell ISE*, které je součástí PowerShellu od verze 2. Jinou dobrou volbou je editor, který je obsažen v rozšíření *PowerGui* od firmy Quest.

3.3 Moduly

Moduly přidávají do PowerShellu od verze 2 další příkazy (cmdlety) a rozšiřují tak jeho funkcionalitu. Jsou uloženy ve složkách, jejichž výčet je definován v proměnné prostředí „$env:PSModulePath“. Ve W7 a W2008 R2 je již několik modulů obsaženo, např. *AppLocker* pro správu zásad řízení aplikací či *BitsTransfer* pro inteligentní přenos souborů na pozadí. V kapitole 8.5 je popsán modul *ActiveDirectory* pro správu AD. Další moduly lze najít na internetu, případně mohou být součástí některých aplikací. Před každým používáním cmdletů z modulu je nejprve nutné modul naimportovat. To se provádí příkazem `Import-Module <název modulu>`, případně lze zavést všechny moduly hromadně spuštěním PowerShellu s přepínačem –ImportSystemModule. Aktuálně načtené moduly je možné vypsat pomocí `Get-Module`. Pro zobrazení seznamu dostupných cmdletů z určitého modulu se používá příkaz `Get-Command -Module <název modulu>`.

Moduly mohou být napsány i jako prosté skripty v jazyce PowerShell, díky čemuž lze jednoduše vytvářet vlastní moduly. Stačí vytvořit textový soubor, který bude obsahovat vlastní funkce, jež budeme chtít prostřednictvím modulu využívat. Poté se musí založit ve složce určené pro moduly podsložka s názvem <*nazev_modulu*>, do níž se nakopíruje skript s funkcemi jakožto soubor „<*nazev_modulu*>.psm1“. K modulu lze také připojit popisné informace (metadata) a nápovědu ve stylu PowerShellu. Takovýto modul je možné velmi jednoduše distribuovat na další počítače prostým překopírováním.

3.4 Vzdálené připojení

Připojení ke vzdáleným počítačům zajišťuje služba *Windows Remote Management* (WinRM), na kterou v tomto PowerShell zcela spoléhá. Tato služba a PowerShell verze 2 musejí být nainstalovány na obou stranách spojení. Služba WinRM zajišťuje propojení mezi ovládajícím a ovládaným počítačem prostřednictvím protokolu HTTP, což dovoluje překonat překážky v podobě firewallů či filtrování síťového provozu. Pracuje v podstatě jako webový server a klient. [6]

Službu WinRM je nutné na vzdáleném počítači před prvním použitím nakonfigurovat, protože je standardně vypnutá a výchozí nastavení připojení vylučuje. To lze provést příkazem `powershell Enable-PSRemoting -Force`.

Interaktivní připojení ke vzdálené konzole PowerShellu se provede zadáním příkazu `powershell -NoProfile -NoExit Enter-PSSession <PC>`. Vzdálené připojení je indikováno jménem ovládaného počítače ve výzvě příkazového řádku PowerShellu. Veškeré zadané a spouštěné příkazy jsou prováděny na vzdáleném stroji, jde tedy o obdobu spojení typu Telnet.

Druhým způsobem volání příkazů na vzdáleném počítači je použití konstrukce `Invoke-Command <PC> {<příkaz>}`. Výstupní data se poté mohou zpracovat pomocí roury a kombinovat s místními zdroji.

4 Component Object Model (COM)

Technologie COM byla vyvinuta společností Microsoft. Jedná se o objektově orientovaný systém softwarových komponent, které jsou schopné vzájemné spolupráce. Je to univerzální způsob, jak spolu mohou aplikace komunikovat a sdílet informace. Objekty COM jsou součástí Windows a některých aplikací. Lze je využít ve WSH i PowerShellu.

Tab. 4.1: Přehled často používaných objektů COM

Název objektu	**Použití**
ADODB.Connection ADODB.Recordset	práce s databázemi a datovými zdroji; využito v kapitole 8.2 k vyhledávání v AD (databáze typu LDAP)
CDO.Message	automatické odesílání e-mailů
Msxml2.XmlHttp	odesílání http požadavků a načítání dat z webových stránek
Shell.Application	ovládání uživatelského rozhraní Windows
WScript.Network WScript.Shell Scripting.FileSystemObject	objekty skriptovacího prostředí WSH (viz kapitola 2.1)
InternetExplorer.Application	ovládání aplikace Internet Explorer
Word.Application Excel.Application PowerPoint.Application Outlook.Application	ovládání aplikací kancelářského balíku Microsoft Office

Ukázka použití objektu *CDO.Message* v jazyce VBScript (odeslání e-mailové zprávy v lokální síti s SMTP serverem, autentizace proběhne na základě přihlášení do Windows):

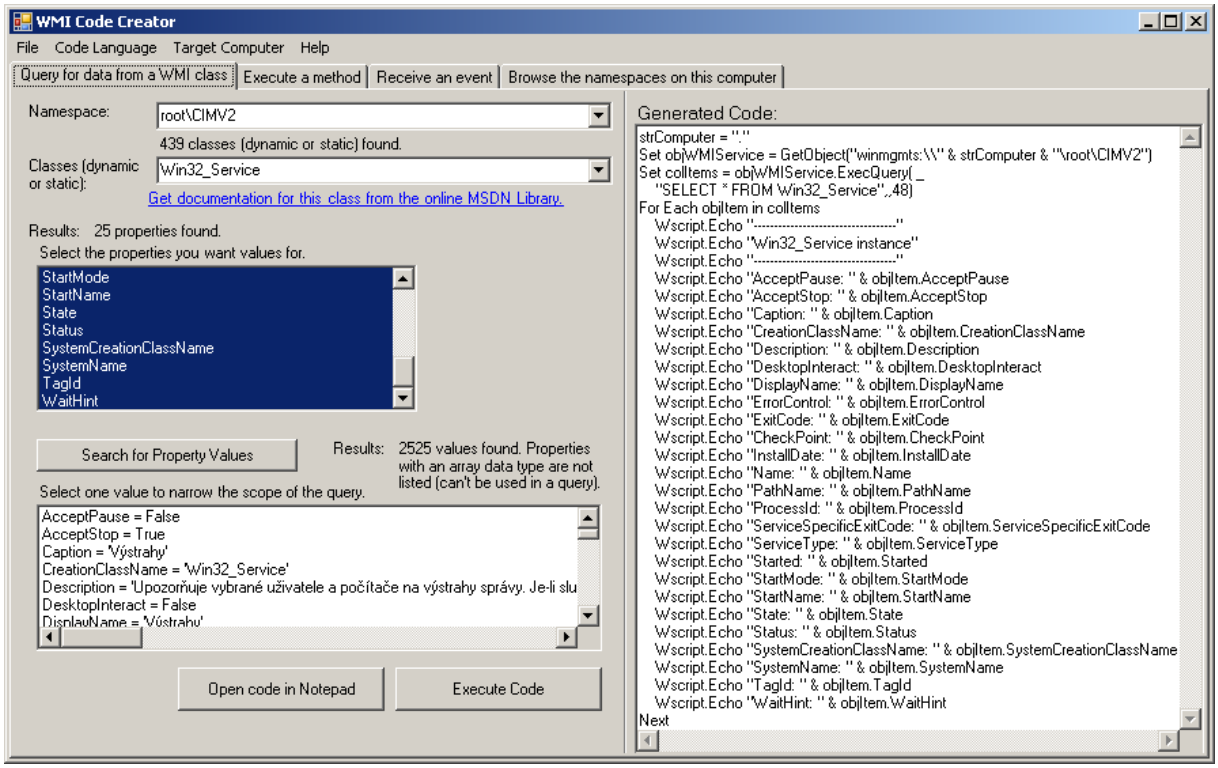
```
Set objMessage = CreateObject("CDO.Message")
With objMessage
  .From = "<e-mail odesílatele>"
  .To = "<e-mail příjemce>"
  .Subject = "Předmět"
  .TextBody = "Text zprávy."
  .AddAttachment "D:\Data\priloha.txt"
  ConfigPath = "http://schemas.microsoft.com/cdo/configuration/"
  With .Configuration
    .Fields(ConfigPath & "smtpserver") = "<SMTP server>"
    .Fields(ConfigPath & "sendusing") = 2
    .Fields(ConfigPath & "smtpauthenticate") = 2 'NTLM autentizace
    .Fields.Update
  End With
  .Send
End With
```

5 Windows Management Instrumentation (WMI)

Rozhraní WMI je univerzálním prostředkem pro pokročilou administraci Windows a patří k základním mechanismům správy tohoto operačního systému. Využívají jej i pokročilé nástroje jako např. *System Center Configuration Manager*. WMI je výbornou platformou především pro sběr systémových informací a hromadné dotazování. Využít jde také pro vykonání nejrůznějších akcí prostřednictvím volání metod, dostupných u řady tříd WMI. Nabízí též možnosti sledování a zpracování událostí, jako jsou vytvoření/změna/smazání souboru, spuštění či ukončení procesu nebo služby apod. Umožňuje skriptování registru (viz kapitola 6.3). Velkou výhodou WMI je snadný a přirozený přístup ke vzdáleným počítačům.

WMI je tvořeno tzv. jmennými prostory (namespace), které obsahují třídy (classes). Výchozí a nejčastěji využívaný namespace je „root\CIMV2“. Pro práci s WMI se používá dotazovací jazyk WQL (WMI Query Language), který vychází z jazyka SQL (Structured Query Language).

K usnadnění tvorby skriptů nebo zdrojových kódů využívajících WMI existuje řada nástrojů. Za všechny jmenujme *WMI Code Creator* a *Scriptomatic* od Microsoftu.



Obr. 5.1: Okno programu WMI Code Creator

5.1 Využití WMI prostřednictvím WSH

Skriptovací prostředí WSH používá pro přístup k rozhraní WMI princip vytváření objektových proměnných, které poté slouží k dalším operacím.

Ukázka využití WMI pomocí WSH v jazyce VBScript (vypsání spuštěných služeb, nastavení automatického spouštění služby „Vzdálený registr“ a její okamžité spuštění):

```
PC = "."
Set objWMIService = GetObject("winmgmts:\\" & PC & "\root\CIMV2")

Set colItems = objWMIService.ExecQuery("SELECT * FROM Win32_Service")
For Each objItem in colItems
  WScript.Echo objItem.Name & ";" & objItem.DisplayName
Next

Set colItems = objWMIService.ExecQuery("SELECT * FROM Win32_Service
 WHERE Name='RemoteRegistry'")
For Each objItem in colItems
  WScript.Echo "ReturnValue: " & objItem.ChangeStartMode("Automatic")
  WScript.Echo "ReturnValue: " & objItem.StartService
Next
```

Pro vykonání na vzdáleném počítači stačí u proměnné *PC* v ukázce zadat místo tečky jeho jméno.

5.2 WMI v PowerShellu

PowerShell pracuje s WMI velmi přirozeně. Díky objektově orientovanému uspořádání WMI lze výstupní data jednoduše zpracovávat pomocí objektové roury. Pro zpřístupnění rozhraní WMI slouží cmdlet *Get-WmiObject*, za nějž se jako parametr přidává požadovaná třída WMI. K získání seznamu dostupných vlastností a metod dané třídy lze využít příkaz `Get-WmiObject <WMI třída> | Get-Member`.

Ukázka využití WMI pomocí PowerShellu (provádí totéž co ukázka v kapitole 5.1):

```
Get-WmiObject Win32_Service -Filter "State='Running'" | select Name,
 DisplayName
Get-WmiObject Win32_Service -Filter "Name='RemoteRegistry'" |
 ForEach {$_.ChangeStartMode("Automatic"); $_.StartService()}
```

Pro vykonání na vzdálených počítačích je možné přidat přepínač –ComputerName *<PC1>,<PC2>…*

5.3 Nástroj WMIC

WMIC (WMI Console) je nástroj příkazového řádku rozhraní WMI. Na rozdíl od WSH a PowerShellu zcela izoluje uživatele od objektové podstaty WMI. Nástroj je tvořen spustitelným souborem *wmic.exe* ve složce „C:\Windows\System32\wbem“. Ovládání probíhá prostřednictvím příkazu *wmic* s kolekcí přepínačů a parametrů, jejichž spojováním je možné sestavit i poměrně komplikované úlohy. Nápovědu získáme přidáním přepínače /?.

Pro často používané třídy WMI lze využívat zjednodušené názvy (aliasy), např. pro Win32_Service alias Service nebo pro Win32_NetworkAdapter alias NIC. V opačném případě je nutné aplikovat přepínač *path* a zadat plné jméno třídy, např. `wmic` **`path`** `Win32_Service`. Seznam dostupných aliasů a odpovídajících tříd se vypíše příkazem `wmic alias get friendlyname,target`.

Ukázka využití WMI pomocí WMIC (provádí totéž co ukázka v kapitole 5.1):

```
wmic service where State="Running" get Name,DisplayName
wmic service where Name="RemoteRegistry" call ChangeStartMode
 "Automatic"
wmic service where Name="RemoteRegistry" call StartService
```

Pro vykonání na vzdálených počítačích lze použít přepínač /node:<*PC1*>,<*PC2*>… Předat se může také soubor se seznamem počítačů pomocí /node:@<*soubor se seznamem*>. Vhodné je přidat přepínač /failfast:1, který odstraní prodlevy při nedostupnosti některých počítačů.

Nástroj WMIC nabízí bohaté možnosti formátování výstupu pomocí transformačních šablon. K tomuto účelu se používá přepínač /format:<*název šablony*>. Takto lze získat výstup vhodný např. pro zpracování tabulkovým procesorem (šablona *csv*) nebo pro přímé zobrazení tabulky v internetovém prohlížeči (šablona *htable*). Následují ukázky formátování výstupu s přesměrováním do souboru (výstupem jsou informace o operačním systému vzdálených počítačů, jejichž seznam je uveden na jednotlivých řádkách souboru „PC.txt“ v aktuální složce).

```
wmic /node:@PC.txt /failfast:1 os get /format:csv > os.csv
wmic /node:@PC.txt /failfast:1 os get /format:htable > os.htm
```

6 Registr

Nedílnou součástí Windows je registr. Je to jakási databáze sloužící k ukládání údajů a nastavení systému, aplikací a uživatelů. Při každé úpravě konfigurace systému, instalaci nové aplikace či hardwaru se veškerá nastavení zapisují právě do registru. Systém Windows neposkytuje konfigurační rozhraní pro všechny existující možnosti nastavení, proto je někdy nezbytné provést úpravy přímo v registru. Často je také přímá úprava registru snadnější a rychlejší, než procházení konfiguračními okny pro provedení určitého nastavení. Je však důležité upozornit, že neuvážený zásah do registru může způsobit problémy nebo v krajním případě nefunkčnost Windows. Aby se po přímé úpravě registru projevily změny ovlivňující chování systému, je často nutné restartovat počítač, někdy se stačí pouze odhlásit a znovu přihlásit. Hlavním nástrojem pro přímou práci s registrem je *Editor registru* (regedit.exe).

Pro práci s registrem na vzdáleném počítači na něm musí běžet služba *Vzdálený registr* (RemoteRegistry). V XP se tato služba spouští automaticky. Ve W7 je nutné automatické spouštění nastavit, případně před připojením k vzdálenému registru službu ručně nastartovat např. příkazem `sc \\<PC> start RemoteRegistry`. Oprávnění pro vzdálený přístup lze ovlivnit změnou oprávnění na klíč „HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurePipeServers\winreg“ na cílovém počítači.

6.1 Struktura registru

Prvky registru připomínají stromovou strukturu a lze je rozdělit na klíče, podklíče a jednotlivé hodnoty.

Kořenové klíče registru:

- HKEY_CLASSES_ROOT (HKCR)

uchovává údaje o přidružení (asociaci) programů k příponám souborů, je tvořen sloučením klíčů HKEY_LOCAL_MACHINE\SOFTWARE\Classes a HKEY_USERS\<*SID*>_Classes (*SID* je identifikátor zabezpečení aktuálně přihlášeného uživatele)

- HKEY_CURRENT_USER (HKCU)

obsahuje nastavení pro aktuálně přihlášeného uživatele, je to zástupce na klíč HKEY_USERS\<*SID*>

- HKEY_LOCAL_MACHINE (HKLM)

sdružuje data o konfiguraci počítače

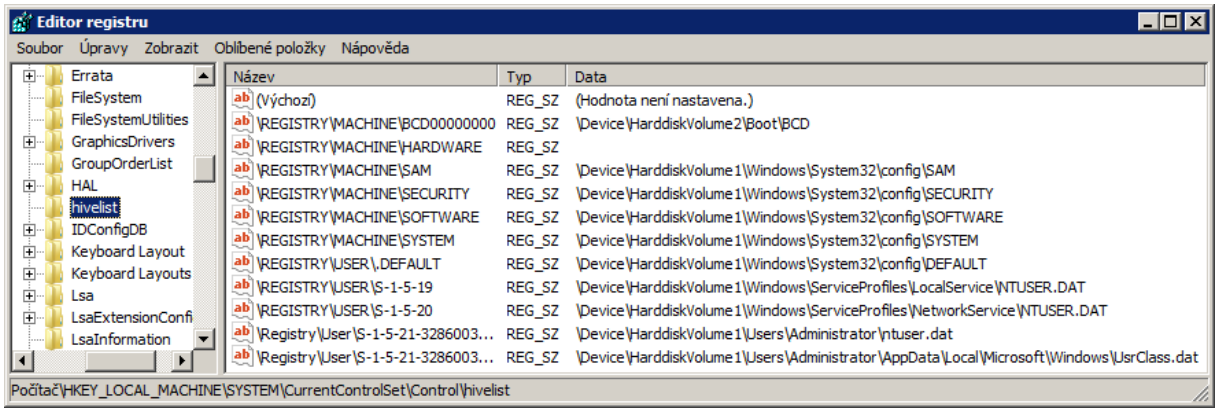
- HKEY_USERS (HKU)

obsahuje podregistry aktuálně připojených uživatelů

- HKEY_CURRENT_CONFIG (HKCC)

obsahuje údaje o právě používaném hardwarovém profilu, je to zástupce na klíč „HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Hardware Profiles\Current“ (W7 již hardwarové profily nepoužívá)

Registr je uložen na disku jako několik samostatných souborů (podregistrů). Příslušné soubory se načtou do paměti při spuštění operačního systému nebo přihlášení uživatele a přiřadí se do registru. Fyzické umístění podregistrů lze vyčíst z hodnot v klíči HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\hivelist (viz obrázek 6.1). V cestách k souborům nejsou uvedena písmena jednotek, protože ty se vytvářejí až po sestavení registru. K podregistru „HARDWARE“ není přidružen žádný soubor, tento klíč se vytváří dynamicky při každém spuštění systému.



Obr. 6.1: Seznam podregistrů a jejich umístění v souborech

6.2 Registrační soubory

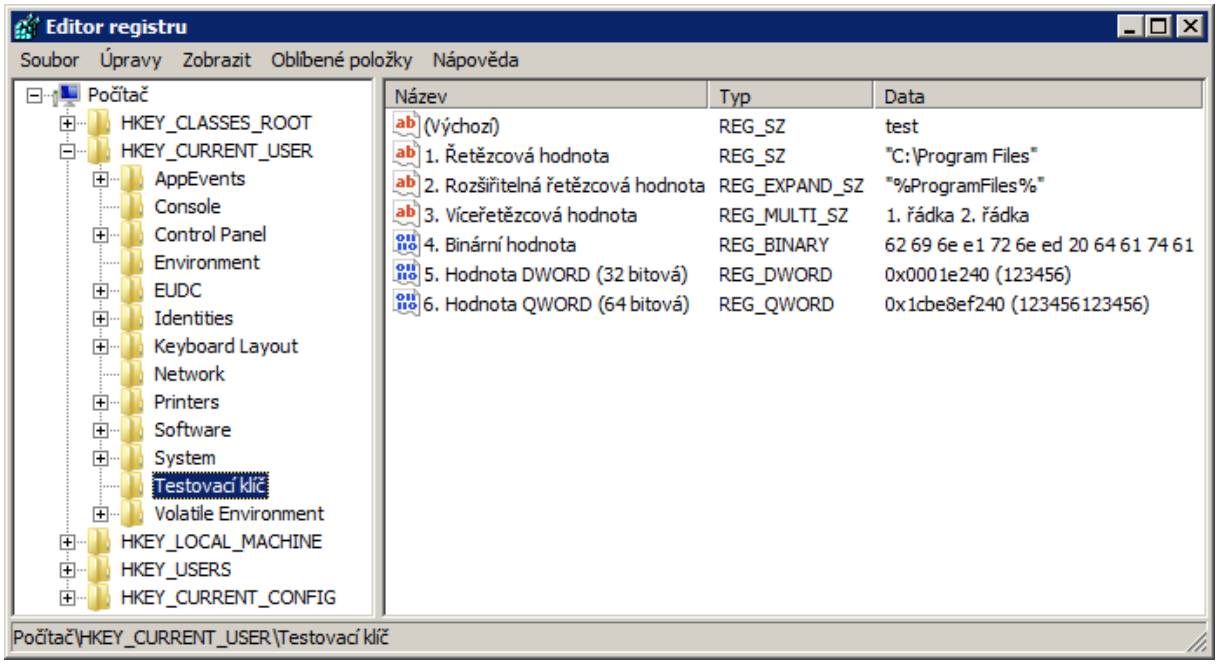
Jedná se o textové soubory s příponou REG, které lze využít k importu nastavení do registru případně k jeho záloze. Od Windows 2000 jsou registrační soubory standardně v textovém formátu UNICODE, ovšem používat lze i starší verze v prostém textu (mají v hlavičce uvedeno „REGEDIT4“), určené pro Windows 9x a NT. Registrační soubor lze importovat do registru jeho spuštěním a potvrzením výzvy, případně bezobslužně příkazem `regedit /s <registrační soubor>` nebo `reg import <registrační soubor>`.

Ukázka registračního souboru a zobrazení obsahu registru po jeho importu:

```
Windows Registry Editor Version 5.00

; případné odstranění existujícího klíče včetně jeho obsahu
[-HKEY_CURRENT_USER\Testovací klíč]

; vytvoření klíče
[HKEY_CURRENT_USER\Testovací klíč]
; zápis hodnot a dat
@="test" ;tato speciální hodnota se nazývá "výchozí"
"1. Řetězcová hodnota"="\"C:\\Program Files\""
"2. Rozšiřitelná řetězcová hodnota"=hex(2):22,00,25,00,50,00,72,00,\
  6f,00,67,00,72,00,61,00,6d,00,46,00,69,00,6c,00,65,00,73,00,25,\
  00,22,00,00,00
"3. Víceřetězcová hodnota"=hex(7):31,00,2e,00,20,00,59,01,e1,00,64,\
  00,6b,00,61,00,00,00,32,00,2e,00,20,00,59,01,e1,00,64,00,6b,00,\
  61,00,00,00,00,00
"4. Binární hodnota"=hex:62,69,6e,e1,72,6e,ed,20,64,61,74,61
"5. Hodnota DWORD (32 bitová)"=dword:0001e240
"6. Hodnota QWORD (64 bitová)"=hex(b):40,f2,8e,be,1c,00,00,00
"dočasné"="abc"
; odstranění hodnoty
"dočasné"=-
```



Obr. 6.2: Obsah registru po importu ukázkového registračního souboru

6.3 Skriptování registru

Pro práci s registrem je možné využít skriptovací prostředí WSH, konkrétně objekt *WScript.Shell*. Ten poskytuje potřebné metody RegRead, RegWrite a RegDelete. Níže uvedený skript v jazyce VBScript umožňuje prostřednictvím registru vytvořit či upravit přidružení (asociaci) programu k zadané příponě souboru.

```
Title = "Asociace přípony souboru"

strPripona = InputBox("Zadejte příponu souboru", Title)
If strPripona = "" Then WScript.Quit

' vytvoření objektu pro práci s registrem
Set WshShell = CreateObject("WScript.Shell")

On Error Resume Next
' načtení typu souboru
strTypSouboru = WshShell.RegRead("HKCR\." & strPripona & "\")
If Err Then
  blnNovyTypSouboru = True
  strTypSouboru = strPripona & "file"
Else
  ' načtení příkazu pro otevření souboru
  strOtevritSoubor = WshShell.RegRead("HKCR\" & strTypSouboru &
 "\shell\open\command\")
End If
On Error Goto 0

strOtevritSoubor = InputBox("Nastavení příkazu pro otevření přípony
 " & UCase(strPripona), Title, strOtevritSoubor)

If strOtevritSoubor <> "" Then
  ' zapsání nového typu souboru
  If blnNovyTypSouboru Then WshShell.RegWrite "HKCR\." & strPripona
 & "\", strTypSouboru

  ' zapsání příkazu pro otevření souboru
  WshShell.RegWrite "HKCR\" & strTypSouboru &
 "\shell\open\command\", strOtevritSoubor, "REG_EXPAND_SZ"

  MsgBox "Nastaveno.", vbInformation, Title
End If
```

Dále lze pro práci s registrem použít rozhraní WMI, které v porovnání s WSH poskytuje navíc metody k prohledávání klíčů a hodnot (EnumKey, EnumValues) a umožňuje pracovat s registrem na vzdáleném počítači. Následující ukázka v jazyce VBScript vypíše obsah klíče s „položkami po spuštění“ vzdáleného počítače.

```
PC = "<PC>"
Set objReg = GetObject("winmgmts:\\" & PC &
 "\root\default:StdRegProv")

Const HKEY_LOCAL_MACHINE = &H80000002
strKey = "Software\Microsoft\Windows\CurrentVersion\Run"

objReg.EnumValues HKEY_LOCAL_MACHINE, strKey, arrValues

If Not IsNull(arrValues) Then
  For Each strValue In arrValues
    objReg.GetStringValue HKEY_LOCAL_MACHINE, strKey, strValue,
 strData
    WScript.Echo strValue & "=" & strData
  Next
End If
```

PowerShell zpřístupňuje registr prostřednictvím své koncepce *PSDrive*. S registrem je zacházeno jako se stromovou adresářovou strukturou a jde jej ovládat stejnými prostředky jako souborový systém. Mapovány jsou dvě části registru, klíč HKEY_CURRENT_USER jako jednotka „HKCU:“ a klíč HKEY_LOCAL_MACHINE jako „HKLM:“. Pro přístup do jiných klíčů se používá příkaz `Set-Location Registry::<klíč>`.

Druhou možností ovládání registru z PowerShellu je využití třídy *.NET Frameworku*, která navíc umožňuje připojení ke vzdálenému registru. Následující ukázka zobrazí název operačního systému vzdáleného počítače.

```
$PC = "<PC>"
$vetev = [Microsoft.Win32.RegistryHive]::LocalMachine
$klic = "SOFTWARE\Microsoft\Windows NT\CurrentVersion"
$cil = [Microsoft.Win32.RegistryKey]::OpenRemoteBaseKey($vetev, $PC)
$cil.OpenSubKey($klic).GetValue("ProductName")
```

Ke správě registru slouží rovněž řádkový příkaz *reg*, který lze využít zejména v dávkových souborech. Kromě běžných operací (čtení, vytváření, mazání klíčů a hodnot) umožňuje též import/export částí registru či načtení/uvolnění (load/unload) podregistrů.

7 Automatické spouštění programů

7.1 Plánovač úloh

Součástí Windows je nástroj, který umožňuje provádění určitých akcí na základě časového plánu nebo nějaké aktivační události. Akce znamená spuštění programu a ve W7 také zobrazení zprávy nebo odeslání e-mailu. Aktivační událostí může být spuštění počítače či přihlášení nebo nečinnost uživatele. Vlastní spouštění naplánovaných úloh zajišťuje služba *Plánovač úloh* (Schedule). V XP se plánovač nachází v ovládacích panelech pod položkou *Naplánované úlohy*. Ve W7 je tento nástroj přepracován a nyní se jedná o modul snap-in *Plánovač úloh* pro konzolu MMC (Microsoft Management Console). Spustit jej lze např. příkazem `taskschd.msc`. W7 již standardně obsahuje řadu vlastních systémových naplánovaných úloh, např. každou středu v 1:00 provádí úloha s názvem „ScheduledDefrag" defragmentaci disků. Je-li počítač v této době vypnutý, defragmentace se po jeho zapnutí spustí při první době nečinnosti.

V XP jsou naplánované úlohy uloženy ve složce „C:\Windows\Tasks" ve formě binárních souborů, ve W7 jsou umístěné v „C:\Windows\System32\Tasks" ve formátu XML (Extensible Markup Language). Informace o činnosti plánovače úloh se zapisují v XP do souboru „C:\Windows\SchedLgU.txt" a ve W7 do protokolu událostí, který se může otevřít např. příkazem `eventvwr /c:Microsoft-Windows-TaskScheduler/Operational`.

Pro interaktivní správu naplánovaných úloh na vzdáleném počítači lze využít v XP příkaz `explorer \\<PC>\::{D6277990-4C6A-11CF-8D87-00AA0060F5BF}` a ve W7 `compmgmt.msc /computer=<PC>`. Je důležité upozornit, že při připojení z XP na W7 se zobrazují pouze úlohy vytvořené příkazem *schtasks* s přepínačem /v1 nebo pomocí *at*.

Ke správě naplánovaných úloh je možné kromě grafického rozhraní využít také řádkové příkazy *schtasks* a *at*. Příkaz *schtasks* umožňuje plnohodnotnou správu naplánovaných úloh. Pro práci se vzdálenými počítači lze použít přepínač /s <*PC*>. Příkaz *at* má omezené možnosti, ale dobře poslouží zejména pro naplánování jednorázových akcí. Např. příkaz `at \\<PC> 23:00 shutdown /s` naplánuje na vzdáleném počítači jeho vypnutí v 23:00 téhož dne. V XP jsou takovéto jednorázové úlohy po proběhnutí automaticky odstraněny, na W7 však zůstávají. Odstranit je lze případně dodatečně třeba příkazem `cmd /C at || at /delete /yes`. Úlohy vytvořené pomocí *at* dostávají automatický název „At<*pořadové číslo úlohy*>". Jsou spouštěny skrytě (neinteraktivně) v kontextu uživatele „NT AUTHORITY\SYSTEM".

Vytvořit lze také naplánovanou úlohu, která bude spuštěna při zápisu určité události do protokolu událostí (event log). V XP se k tomuto účelu používá výhradně příkaz *eventtriggers*. Ve W7 je možné využít příkaz `schtasks /Create /SC ONEVENT …` nebo při ručním vytváření naplánované úlohy vybrat aktivační událost „Při události“.

7.2 Položky po spuštění

Pro zajištění automatického spouštění programu při každém přihlášení uživatele se využívá několik možností. V prvé řadě to může být záznam v registru. Např. pro nastavení automatického spouštění aplikace „Microsoft Outlook 2010“ lze aplikovat příkaz `reg add HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v Outlook /d "\"C:\Program Files\Microsoft Office\Office14\OUTLOOK.EXE\""`. Aby zmíněné platilo pouze pro aktuálně přihlášeného uživatele, je nutné zaměnit kořenový klíč HKLM za HKCU.

Další možností je vytvoření zástupce programu ve složce *Po spuštění*. Složka společná pro všechny uživatele je ve W7 „%ProgramData%\Microsoft\Windows\Start Menu\Programs\Startup“ a složka platná pouze pro aktuálně přihlášeného uživatele je „%AppData%\Microsoft\Windows\Start Menu\Programs\Startup“. V XP jsou to složky „%AllUsersProfile%\Nabídka Start\Programy\Po spuštění“ a „%UserProfile%\Nabídka Start\Programy\Po spuštění“.

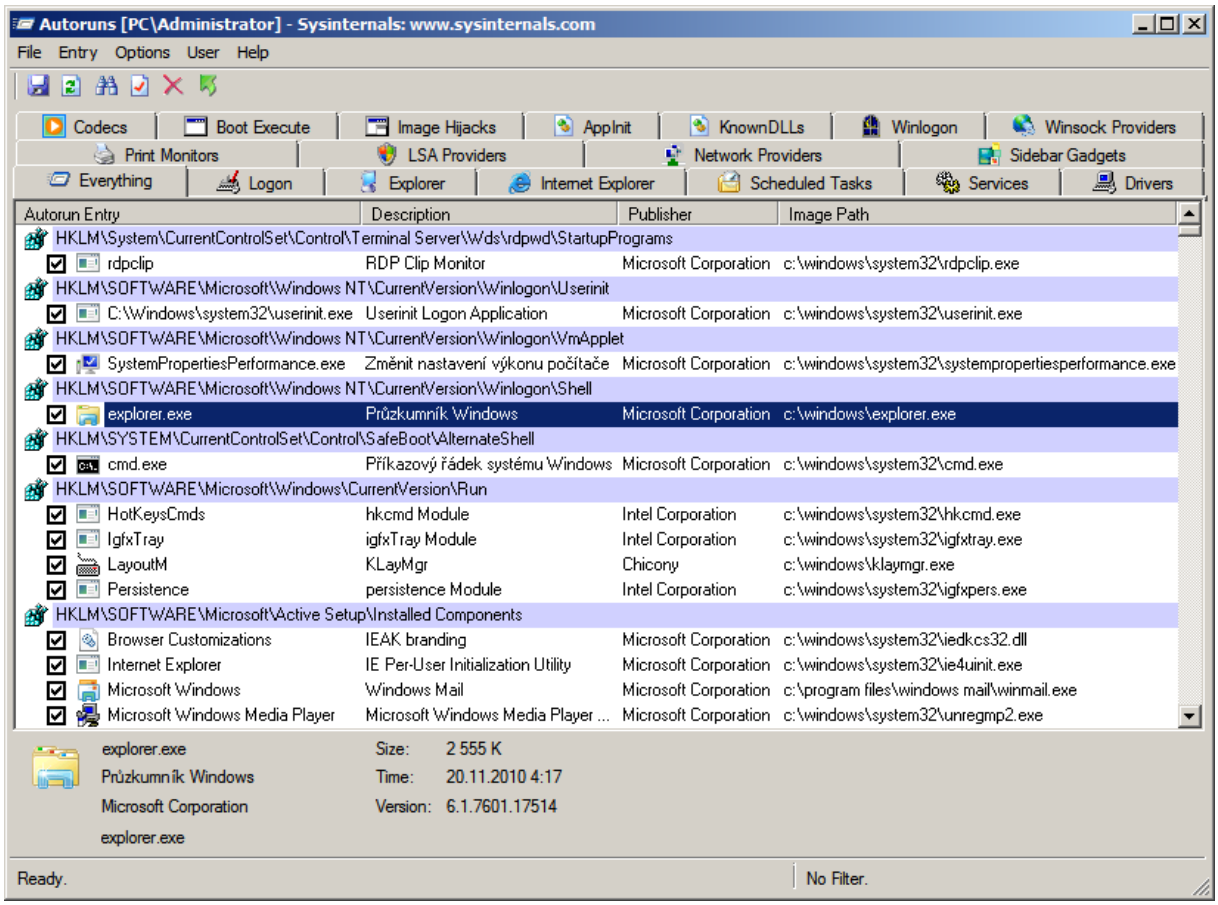
Položky po spuštění lze prohlížet či zakazovat pomocí nástroje *Konfigurace systému* (msconfig.exe) na záložce „Po spuštění“.

7.3 Skripty zásad skupin

Pro nastavení spouštěcích/ukončovacích skriptů počítače nebo skriptů pro přihlášení/odhlášení uživatele se používá *Editor místních zásad skupiny* (gpedit.msc). K připojení na vzdálený počítač je možné použít `gpedit.msc /gpcomputer:"<PC>"`. Položky pro zadání cest ke skriptům se nacházejí v sekcích „Konfigurace počítače\Nastavení systému Windows\Skripty (spouštěcí nebo ukončovací)“ resp. „Konfigurace uživatele\Nastavení systému Windows\Skripty (pro přihlášení nebo odhlášení)“. V počítačové síti se službou Active Directory lze spouštění skriptů nastavit globálně pomocí doménových zásad skupin (Domain Group Policy) a využít jdou také přihlašovací skripty (Logon Scripts) pro uživatele.

7.4 Program Autoruns

Pro prohlížení a úpravu spouštěcích položek se může využít program *Autoruns*, který je dostupný na adrese http://technet.microsoft.com/sysinternals/bb963902 nebo jej lze spustit přímo pomocí odkazu \\live.sysinternals.com\tools\autoruns.exe. Kromě běžných „položek po spuštění“ zobrazuje též naplánované úlohy, služby, ovladače a další.



Obr. 7.1: Okno programu Autoruns

Součástí programu je také řádková utilita *autorunsc.exe*. Může být užitečná např. k pravidelnému monitorování změn ve spouštěcích položkách pomocí následujícího dávkového souboru, umístěného do složky s utilitou, který vypíše případné rozdíly od jeho posledního spuštění.

```
@echo off
cd /d %~dp0
move NEW.txt OLD.txt
autorunsc.exe -AcceptEula -a -c > NEW.txt
fc /u NEW.txt OLD.txt || pause
```

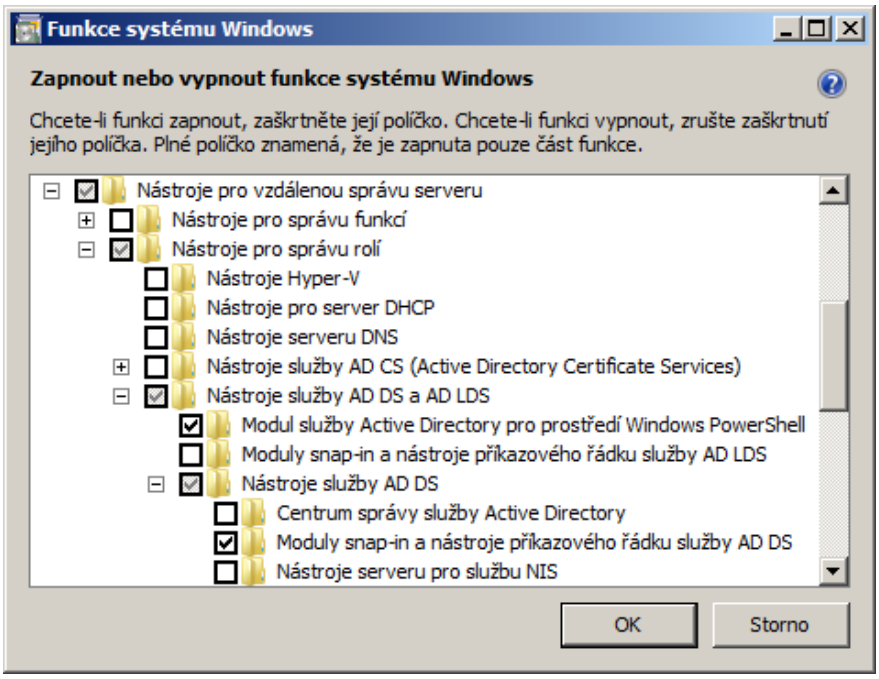
8 Skriptování adresářové služby Active Directory (AD)

AD patří k pilířům správy větších sítí na bázi Windows. Je to centrální úložiště informací o objektech (uživatelích, počítačích) v počítačové síti a zajišťuje také jejich bezpečnou autentizaci. Pro ukládání a přístup k datům používá standardizovaný komunikační protokol LDAP (Lightweight Directory Access Protocol). Data jsou uložena ve stromové struktuře pomocí záznamů. Každý záznam obsahuje sadu atributů v závislosti na jeho třídě.

Pro jednoznačnou identifikaci záznamů se používá rozlišovací jméno (Distinguished Name, DN), které popisuje úplnou cestu k dané položce. DN se skládá ze jména objektu (Common Name, CN), jmen jednotlivých kontejnerů (Organization Unit, OU) a domén (Domain Component, DC), např. „CN=novak,OU=uzivatele,DC=firma,DC=cz“.

8.1 Remote Server Administration Tools (RSAT)

Pro W7 je na adrese http://www.microsoft.com/cs-cz/download/details.aspx?id=7887 k dispozici balík *Nástroje pro vzdálenou správu serveru* (Remote Server Administration Tools), který obsahuje mimo jiné také nástroje pro správu AD. Pro některé postupy, popsané v kapitolách 8.3 a 8.5, je nutné tento balík nainstalovat. Poté je potřeba otevřít konfigurační okno *Zapnout nebo vypnout funkce systému Windows* (např. příkazem `OptionalFeatures`) a zaškrtnout funkce „Modul služby Active Directory pro prostředí Windows PowerShell“ a „Moduly snap-in a nástroje příkazového řádku služby AD DS“.



Obr. 8.1: Zapnutí funkcí pro správu AD z balíku RSAT

Výše popsanou instalaci a zapnutí funkcí lze provést také bezobslužně pomocí následujícího dávkového souboru:

```
:: instalace balíku RSAT
pushd <složka s instalačním souborem>
wusa Windows6.1-KB958830-x86-RefreshPkg.msu /quiet

:: zapnutí potřebných funkcí
ocsetup ^
RemoteServerAdministrationTools-Roles;^
RemoteServerAdministrationTools-Roles-AD;^
RemoteServerAdministrationTools-Roles-AD-Powershell;^
RemoteServerAdministrationTools-Roles-AD-DS;^
RemoteServerAdministrationTools-Roles-AD-DS-SnapIns
```

Ve W2008 R2 je již balík RSAT nainstalován, stačí pouze potřebné funkce zapnout. To jde realizovat stejným způsobem uvedeným výše nebo také s využitím PowerShellu:

```
Import-Module ServerManager
Add-WindowsFeature RSAT-AD-PowerShell,RSAT-ADDS-Tools
```

8.2 Vyhledávání v AD

Vyhledávání údajů v AD je jednou z nejčastějších úloh v tomto prostředí. Používá se k tomu vyhledávací filtr LDAP, jehož úkolem je vymezit, které objekty budou vraceny. Filtr je založen na atributech objektů a jejich hodnotách. Skládá se z libovolného počtu podmínek, uzavřených v kulatých závorkách: (*atribut=hodnota*). Jednotlivé podmínky stojí vedle sebe a vztah mezi nimi určuje logický operátor (**&** = AND, **|** = OR, **!** = NOT), který je umístěn před nimi: (*operátor*(*1. podmínka*)(*2. podmínka*)…). V hodnotách podmínek lze používat zástupný znak „*“. Filtry jde použít ve skriptech, řádkových utilitách (`dsquery * -filter "<filtr>"`) i PowerShellu (`Get-ADObject -LDAPFilter "<filtr>"`).

Ukázky vyhledávacích filtrů:

- vyhledání počítačů, jejichž jména začínají řetězcem „pc“ nebo „nb“

(&(objectCategory=computer)(|(name=pc*)(name=nb*)))

- vyhledání všech nezakázaných uživatelských účtů; *userAccountControl* je speciální binární atribut, složený z řady příznaků (flagů), příznak **2** značí zakázaný účet

(&(objectCategory=user)(!userAccountControl:1.2.840.113556.1.4.803:=**2**))

Standardně se prohledávají od základního kořene rekurzivně všechny podřízené kontejnery (subtree). Rozsah vyhledávání je případně možné omezit určením kořene pro vyhledávání (search base) a zvolením rozsahu hledání (scope).

Ukázka vyhledávání v AD prostřednictvím jazyka VBScript a komponenty ADO (ActiveX Data Objects):

```
' vytvoření ADO objektů pro přístup do AD
Set adoConnection = CreateObject("ADODB.Connection")
Set adoCommand = CreateObject("ADODB.Command")
adoConnection.Provider = "ADsDSOObject"
adoConnection.Open
adoCommand.ActiveConnection = adoConnection

Filtr = "<filtr>"
Atributy = "distinguishedName,displayName"

' omezení rozsahu vyhledávání na konkrétní OU bez podkontejnerů
Base = "LDAP://OU=oddeleni,DC=firma,DC=cz"
Scope = "OneLevel"

' sestavení a spuštění dotazu
Dotaz = "<" & Base & ">;" & Filtr & ";" & Atributy & ";" & Scope
adoCommand.CommandText = Dotaz
Set adoRecordset = adoCommand.Execute

' procházení a vypisování načtených záznamů
Do Until adoRecordset.EOF
  WScript.Echo adoRecordset.Fields("distinguishedName").Value
  WScript.Echo adoRecordset.Fields("displayName").Value
  adoRecordset.MoveNext
Loop
```

Ukázka vyhledávání v AD pomocí PowerShellu a třídy .NET Frameworku:

```
$Searcher = New-Object DirectoryServices.DirectorySearcher

$Searcher.Filter = "<filtr>"

# omezení rozsahu vyhledávání na konkrétní OU bez podkontejnerů
$Searcher.SearchRoot = "LDAP://OU=oddeleni,DC=firma,DC=cz"
$Searcher.SearchScope = "OneLevel"
```

```
$Searcher.FindAll() | ForEach {
  Write-Host $_.GetDirectoryEntry().distinguishedName
  Write-Host $_.GetDirectoryEntry().displayName
}
```

8.3 Nástroje příkazového řádku pro správu AD

Pro správu uživatelů a skupin v AD lze použít příkaz *net* a jeho operace *user* a *group*. Nutné je přidat přepínač /DOMAIN, jinak by se operace prováděly na lokálním počítači. Následující ukázka vytvoří nového uživatele, přidá jej do skupiny a vypíše členy této skupiny.

```
net user novak /add /fullname:"Jan Novák" /DOMAIN
net group Skupina1 novak /add /DOMAIN
net group Skupina1 /DOMAIN
```

Pokročilejší schopnosti nabízí rodina nástrojů *ds*, které je však nutné nejprve nainstalovat. Pro W7 jsou dostupné v balíku *RSAT* pod názvem „Moduly snap-in a nástroje příkazového řádku služby AD DS“ (viz kapitola 8.1). Pro XP jsou obsaženy v balíku *Sada nástrojů pro správu systému Windows Server* (Windows Server Administration Tools Pack) na adrese http://www.microsoft.com/cs-cz/download/details.aspx?id=6315. V systému Windows Server 2003 jsou tyto nástroje přítomny.

Nástroje *ds* se skládají z těchto utilit: dsadd, dsget, dsmod, dsmove, dsquery, dsrm. Nápověda k jednotlivým příkazům se vypíše přidáním přepínače /?. Jednotlivé utility se vzájemně doplňují a využívají spojování pomocí roury. Příkaz *dsquery* vyhledává požadované záznamy a předává jejich rozlišovací jména (DN) přes rouru dalšímu příkazu ke zpracování.

Ukázky použití nástrojů *ds*:

- nastavení popisu (description) počítače

```
dsquery computer -samid PC1 | dsmod computer -desc "Můj počítač"
```

- vypsání jmen počítačů a jejich popisu ze zadané OU

```
dsquery computer OU=pc,DC=firma,DC=cz | dsget computer -samid -desc
```

- vypsání rekurzivně rozbaleného seznam skupin, do kterých uživatel patří

```
dsget user CN=novak,OU=uzivatele,DC=firma,DC=cz -memberof -expand
```

- vypsání zobrazovaných jmen členů zadané skupiny

```
dsquery group -samid Skupina1 | dsget group -members | dsget user
 -display
```

8.4 Active Directory Service Interfaces (ADSI)

Jedná se o sadu rozhraní COM pro správu adresářové služby. Vyniká snadnou použitelností a jednoduchou syntaxí. ADSI lze využít ve WSH i PowerShellu. Pro usnadnění tvorby skriptů v jazyce VBScript se může použít nástroj *ADSI Scriptomatic* od Microsoftu.

Ukázka využití ADSI pomocí WSH v jazyce VBScript (vytvoření nového uživatele v zadané OU a vypsání všech uživatelů z této OU včetně jejich zobrazovaných jmen):

```
Set OU = GetObject("LDAP://OU=uzivatele,DC=firma,DC=cz")

Set NewUser = OU.Create("User", "CN=novak")
NewUser.Put "displayName", "Jan Novák"
NewUser.SetInfo

For Each User In OU
  WScript.Echo User.cn & vbTab & User.displayName
Next
```

Ukázka využití ADSI pomocí PowerShellu (provádí totéž co ukázka výše):

```
$OU = [ADSI] "LDAP://OU=uzivatele,DC=firma,DC=cz"

$NewUser = $OU.Create("User", "CN=novak")
$NewUser.Put("displayName", "Jan Novák")
$NewUser.SetInfo()

$OU.Children | select cn, displayName
```

Rozhraní ADSI nabízí vedle správy AD ještě další možnosti. Umožňuje ovládání lokálních skupin, uživatelů, služeb a sdílení. Následující ukázka v jazyce VBScript přidá uživatele „novak“ do skupiny Administrators na vzdáleném počítači PC1.

```
Set objGroup = GetObject("WinNT://PC1/Administrators,group")
objGroup.Add("WinNT://novak")
```

8.5 Využití PowerShellu pro správu AD

Práce se záznamy (objekty) AD je v objektově orientovaném PowerShellu poměrně přirozená a intuitivní. Použít lze jak rozhraní ADSI, tak třídy .NET Frameworku. Vlastní cmdlety (příkazy) však pro tuto oblast standardně neobsahuje, ale pro W7 a PowerShell od verze 2 je k dispozici patřičné rozšíření v balíku *RSAT* pod názvem „Modul služby Active Directory pro prostředí Windows PowerShell“ (viz kapitola 8.1). Nainstalováním získáme

rozsáhlou kolekci cmdletů, které pokrývají hlavní potřeby práce s AD. Před každým jejich použitím je nutné přidaný modul nejprve naimportovat příkazem `Import-Module ActiveDirectory`. Seznam dostupných cmdletů lze získat pomocí příkazu `Get-Command -Module ActiveDirectory`. Modul zahrnuje také koncepci *PSDrive*, kdy je služba AD mapována jako virtuální jednotka „AD:“, s kterou je možné pracovat podobně jako se souborovým systémem.

Ukázky použití cmdletů z modulu ActiveDirectory pro PowerShell:

- nastavení popisu počítače

```
Set-ADComputer PC1 -Description "Můj počítač"
```

- vypsání jmen počítačů a jejich popisu ze zadané OU s využitím jednotky PSDrive

```
Set-Location "AD:\OU=pc,DC=firma,DC=cz"
Get-ADComputer -Filter * -Properties cn,description | select cn,
 description
```

- vypsání zobrazovaných jmen členů zadané skupiny

```
Get-ADGroupMember Skupina1 | Get-ADUser -Properties displayName |
 select displayName
```

- vyhledání všech nezakázaných uživatelů, jejichž jména obsahují řetězec „nov“

  - pomocí vlastního filtrování PowerShellu

```
Get-ADUser -Filter {cn -like "*nov*" -and enabled -eq $true}
```

  - pomocí filtru LDAP

```
Get-ADUser -LDAPFilter
 "(&(cn=*nov*)(!userAccountControl:1.2.840.113556.1.4.803:=2))"
```

Pro správu AD z PowerShellu lze doporučit také balík *ActiveRoles Management Shell for Active Directory* od firmy Quest. Jedná se o hojně používaný doplněk, obsahující kolekci cmdletů. Vznikl dlouho před příchodem výše uvedeného modulu z dílny Microsoftu. Pracuje i v PowerShellu verze 1.0 a v systémech starších než W7.

9 Porovnání správy systému Windows a Linux

Operační systémy Linux (resp. systémy na bázi Unix) mají velmi propracovaný příkazový řádek (shell), který nabízí velké množství příkazů a vynikající skriptovací jazyk. Těžiště pokročilejší správy Linuxu spočívá hlavně v práci v příkazovém řádku. Je možné si vybrat z více shellů, z nichž nejpoužívanější je *bash* (Bourne Again Shell). Svými schopnostmi mnohonásobně předčí příkazový řádek Windows. Avšak s příchodem PowerShellu, který se v mnohém inspiroval systémy Unix, se situace vyrovnala.

Tab. 9.1: Porovnání názvů často používaných příkazů

Účel příkazu	**Linux**	**Windows**	
		Příkazový řádek	**PowerShell**
nápověda k příkazu	man, -?, --help	help, /?	Get-Help
zobrazení zprávy	echo	echo	Write-Output
vymazání obrazovky	clear	cls	Clear-Host
změna aktuální složky	cd	cd	Set-Location
zobrazení jména aktuální složky	pwd	cd	Get-Location
vypsání obsahu složky	ls	dir	Get-ChildItem
zobrazení obsahu souboru	cat	type	Get-Content
vyhledávání textových řetězců	grep	find, findstr	Select-String
kopírování souboru	cp	copy	Copy-Item
vymazání souboru	rm	del	Remove-Item
vypsání spuštěných procesů	ps	tasklist	Get-Process
konfigurace síťového rozhraní	ifconfig	ipconfig, netsh	*až ve verzi 3.0*

V Linuxu se hojně využívá propojování příkazů prostřednictvím roury, zatímco v příkazovém řádku Windows ji lze použít jen u několika příkazů (more, sort, find, findstr). Objektově orientovaný PowerShell využívá v široké míře rouru objektovou.

Diskové uspořádání v Linuxu je stejně jako ve Windows realizováno prostřednictvím adresářů (složek) a souborů. Linux ale používá k oddělení částí cest normální lomítko (/), kdežto Windows běžně využívá zpětné lomítko (\). V Linuxu také neexistují žádná písmena jednotek, ale diskové oddíly a mechaniky jsou připojeny (namontovány) do určitých podadresářů jediného disku.

Je důležité upozornit, že Linux na rozdíl od Windows rozlišuje v cestách, příkazech a přepínačích velikost písmen (case sensitive). Tudíž v jednom adresáři může klidně existovat více souborů se stejným jménem, lišícím se pouze velikostí písmen (např. test, Test a TEST). Příkazy je nutné psát malými písmeny a je třeba dodržovat správnou velikost přepínačů. Např. příkaz `ls -a` provádí trochu něco jiného než `ls -A`.

Konfigurace systému se v Linuxu provádí pod uživatelem *root*, který disponuje nejvyšším oprávněním. Je to obdoba uživatele Administrator ve Windows.

Windows ukládá většinu nastavení do registru (binární databáze), kterou lze spravovat pouze k tomu určenými prostředky. Linux však uchovává nastavení ve formě textových konfiguračních souborů, které jde číst a upravovat pomocí obyčejného textového editoru. Většina konfiguračních souborů je uložena v adresáři „/etc“ nebo ve formě skrytých souborů v domovském adresáři uživatele.

Plánované spouštění příkazů či skriptů zajišťuje v Linuxu služba (démon) *cron*, což je obdoba plánovače úloh ve Windows. Nastavení se provádí úpravou speciálního konfiguračního souboru „/var/spool/cron/crontabs/<*jméno uživatele*>“ pomocí příkazu `crontab -e`. Jednorázové akce lze naplánovat příkazem *at*, vzniklé konfigurační soubory jsou potom uloženy v adresáři „/var/spool/cron/atjobs“.

Závěr

Cílem této práce bylo ukázat možnosti administrace, automatizace a vzdálené správy operačního systému Windows a adresářové služby Active Directory. Dále zjistit rozdíly mezi verzemi Windows a nakonec porovnat správu systému Windows a Linux.

Byly využity převážně prostředky, které jsou přímo součástí Windows. Pro některé účely je však vhodné použít další nástroje: program *Autoruns* pro prohlížení a úpravu spouštěcích položek, některé funkce z balíku *RSAT* pro správu AD, *PowerShell* pro systémy starší než W7.

Prostředkem pro administraci a automatizaci Windows je v první řadě *příkazový řádek*. Umožňuje spouštění příkazů a prostřednictvím dávkových souborů podporuje i jednoduché programové konstrukce. Další osvědčený způsob je skriptovací prostředí *WSH*. To podporuje několik skriptovacích jazyků a umožňuje přístup k rozhraní COM a WMI. Novější možnost je použít příkazový interpret *PowerShell*, který nabízí vynikající skriptovací jazyk a umožňuje využít .NET Framework, COM i WMI.

Při správě Windows je často nutné pracovat s registrem. K tomu lze využít *registrační soubory* a také všechny výše zmíněné prostředky.

Důležitou součástí automatizace je také automatické spouštění programů. Plánované spouštění zajišťuje *plánovač úloh*. Spouštění programů při přihlášení uživatele obstarají např. *položky po spuštění* a při startu počítače *skripty zásad skupin*. Pro prohlížení a úpravu spouštěcích položek se může využít program *Autoruns*.

Bylo ukázáno, že v prostředí počítačové sítě je možné většinu popsaných nástrojů použít také pro vzdálenou správu počítačů.

Pro skriptování AD lze využít *příkazový řádek*, *WSH* i *PowerShell*. Pokročilejší funkce jsou k dispozici v balíku *RSAT*.

Byly zjištěny a popsány četné rozdíly mezi verzemi Windows. Proto je nutné zejména při přechodu z XP na vyšší verze prověřit funkčnost původních dávkových souborů a skriptů.

Porovnaná byla také správa systému Windows a Linux. Hlavním prostředkem pro správu Linuxu je příkazový řádek (shell). Ten svými schopnostmi mnohonásobně předčí příkazový řádek Windows, avšak je srovnatelný s PowerShellem. Linux uchovává nastavení ve formě textových konfiguračních souborů, kdežto Windows ukládá většinu nastavení do registru. Plánované spouštění příkazů a skriptů zajišťuje v Linuxu služba (démon) *cron*.

Seznam literatury a informačních zdrojů

[1] MUELLER, John. *Příkazový řádek Windows pro Windows Vista, 2003, XP a 2000*. Brno: Computer Press, 2008. 656 s. ISBN 978-80-251-1961-7.

[2] BITTO, Ondřej. *Příkazový řádek Windows 7*. Brno: Computer Press, 2011. 231 s. ISBN 978-80-251-3506-8.

[3] POLITZER, Michal. *Windows XP: tajemství a kouzla Příkazového řádku*. Praha: Computer Press, 2003. 83 s. ISBN 80-722-6874-0.

[4] BOTT, Ed, Carl SIECHERT a Craig STINSON. *Mistrovství v Microsoft Windows 7*. Brno: Computer Press, 2010. 936 s. ISBN 978-80-251-2817-6.

[5] MALINA, Patrik. *Powershell: podrobný průvodce skriptováním*. Brno: Computer Press, 2007. 340 s. ISBN 978-80-251-1816-0.

[6] MALINA, Patrik. *Jak vyzrát na Windows PowerShell 2.0*. Brno: Computer Press, 2010. 464 s. ISBN 978-80-251-2732-2.

[7] HOLMES, Lee. *Windows PowerShell Cookbook: The Complete Guide to Scripting Microsoft's New Command Shell*. 2nd ed. Farnham: O'Reilly, 2010. 882 s. ISBN 978-0-596-80150-2. Dostupné také z: http://it-ebooks.info/book/131/

[8] MISANI, Mark a Dan YORK. *Linux pro administrátory Windows*. Brno: Computer Press, 2004. 504 s. ISBN 80-251-0317-X.