

**Vyšší odborná škola zdravotnická, managementu
a veřejnosprávních studií, s.r.o.
Ledecká 35, Plzeň**

ABSOLVENTSKÁ PRÁCE

Název tématu: Client/server aplikace pro vzdálené hromadné spouštění servisních úloh na stanicích uživatelů

Autor práce: Jiří Fictum

Vzdělávací program: Systémový administrátor IT

Vedoucí absolventské práce: Ing., Mgr. Filip Vaculík

Skupina: AIT2

V Plzni dne: 30. března 2016

.....
podpis autora práce

Škola: Vyšší odborná škola zdravotnická, managementu a veřejnosprávních studií, s.r.o.

Vzdělávací program: Systémový administrátor IT

ZADÁNÍ ABSOLVENTSKÉ PRÁCE

Pro absolventskou práci jsem si zvolil téma:

Client/server aplikace pro vzdálené hromadné spouštění servisních úloh na stanicích uživatelů

AIT3

.....
skupina

Jiří Fictum

.....
jméno, příjmení a podpis

V Plzni dne: 30. března 2016

Souhlasím se zvolením daného tématu.

PhDr. Jana Ajglová

.....
ředitelka školy

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že jsem absolventskou práci vypracoval samostatně a všechny použité prameny jsem uvedl v seznamu použitých zdrojů.

.....
podpis autora práce

PODĚKOVÁNÍ

Děkuji především vedoucímu mé absolventské práce, Ing. Mgr. Filipu Vaculíkovi, zejména za jeho obětavost, cenné rady, podněty a připomínky při zpracování absolventské práce.

.....
podpis autora práce

OBSAH

Úvod.....	8
1 Teoretický background.....	9
1.1 Průzkum současné situace správy sítí	9
1.1.1 Síťové nástroje firmy Microsoft	9
1.1.2 Příklady dalších systémů pro správu sítě	11
1.2 Výchozí zkušenosti	12
1.2.1 Příkazová řádka Windows	12
1.2.2 Linuxový terminál bash	14
1.2.3 Python	15
1.3 Záměr	17
1.3.1 Přímé ovládání versus naslouchající služba.....	18
1.3.2 Výhody klientské služby	19
2 Příprava vyvíjené aplikace	20
2.1 Případová studie	20
2.1.1 Uživatelské rozhraní	21
2.2 Testovací prostředí	22
2.2.1 Virtuální počítače a samba.....	22
2.2.2 Vybraná síť pro nasazení	23
2.3 Očekávané problémy.....	23
2.3.1 CMD a Bash – úskalí samotného jazyka	24
2.3.2 Python – úskalí začátečníka	24
2.3.3 Uživatelská oprávnění systému Windows	24
2.3.4 Multiplatformní prostředí.....	25
2.4 Postup práce	25
2.4.1 Programování služby	25
2.4.2 Testování služby	26
2.4.3 Testování jádra aplikace	27
3 Konečná podoba.....	29
3.1 Naprogramované funkce	29
3.1.1 Detekce počítačů v síti a v systému správy	30
3.1.2 Historie a logování	31
3.1.3 Skupiny podle operačního systému	32
3.1.4 Uživatelské skupiny	33

3.1.5 Základní sada skriptů	34
3.2 Přístup k síti, konfigurace samby	35
3.3 Podporované platformy	36
3.4 Uživatelské rozhraní	36
3.4.1 Ovládání služby	37
3.4.2 Ovládání aplikace.....	38
3.5 Systém souborů aplikace.....	39
3.6 Instalace	40
3.7 Autorství a licence	41
Závěr	42
Resumé.....	43
Summary.....	44
Seznam použitých zdrojů.....	45

ÚVOD

Pro svoji závěrečnou práci jsem si zvolil téma sítí a tvorby aplikace pro jejich správu záměrně. Především jsem si přál, aby má práce přinesla pokud možno něco nového. Dále zmíněný problém považuji za nestárnoucí.

Uživatelská prostředí se mohou vyvíjet donekonečna a uživatelům můžeme nabídnout jakýkoliv z nepřeberného množství operačních systémů, jejichž nabídka se stále rozrůstá a doslova předhání s funkcemi, které kus železa umí proměnit ve švýcarský nůž moderního člověka v práci, doma i na cestách.

Vždy však budeme potřebovat obětavé lidi, nasazené pro správu těchto zařízení, síťové administrátory.

Bez nástrojů zpříjemňujících a automatizujících práci adminů, by byl jejich život extrémně vyčerpávající a neuspokojivý. Ne že by na poli pomocného softwaru nebylo na výběr, ale hlavně díky finančnímu rozpočtu a přání nadřízených se administrátor ocitá v situaci značně zúžených možností.

Snad se mi tedy podaří svým úsilím přispět do pomyslného „mlýna“.

1 TEORETICKÝ BACKGROUND

Než se pustíme rovnýma nohama do programování, učiníme zhodnocení situace, v jaké se současný síťový administrátor nachází. Prospěje také shrnutí možností dané současně dostupných programovacích jazyků a programových interpretů. Přes vyslovení záměru se následně dostaneme k načrtnutí samotné koncepce, která bude dále vodítkem během celého procesu programování.

1.1 Průzkum současné situace správy sítí

Každý síťový administrátor má většinou své oblíbené mechanismy, kterými dosahuje větší či menší úrovně hromadné správy uživatelských stanic.

Není však snadné je dohledat nebo vytvořit jejich výčet. Především, chceme znát osobní know-how dotyčného správce. O to se však málokterý admin podělí, tím méně, aby ho dobrovolně zveřejnil.

V praxi se ukazuje, že správci bývají značně vytížení v práci, správu sítě mají často jako druhé zaměstnání. Dokumentace čehokoliv tak pochopitelně nepatří k oblíbeným činnostem.

1.1.1 Síťové nástroje firmy Microsoft

Světově známé Redmontské společnosti Microsoft (dále jako MS) se podařil jedinečný strategický tah, v historii nemající obdoby. Firma využila vhodné doby k vhodnému obchodnímu záměru a dosáhla tak prakticky monopolního postavení ve světě uživatelů osobních počítačů. Od začátku se snažila přidat další nástroje, tak aby uživatel měl k dispozici nástroje pro práci na počítači, nejlépe vše od stejné firmy.

MS se snažil proniknout také mezi serverová řešení – vzpomeňme platformu NT a systém Windows 2000. Dnes jmenujme používanější Windows Server 2003, 2008, 2012 a současný 2016. Poslední z uvedených můžeme brát jako paralelní k poslednímu desktopovému systému Windows 10.¹

¹ Anglická Wikipedie, 15. března 2016

Nakolik jsou serverová řešení MS povedená, ponechme nyní stranou. Windows byl však se svým grafickým rozhraním vždy zacílený na uživatele. Tato charakteristika je v tomto případě velmi znát průřezově všemi verzemi OS.

Naproti tomu linuxové servery ovládá operační systém odjakživa vyvinutý přesně pro účely serverů. Není náhodou, že ve světě internetu jich najdeme přes 90%.²

Společnost Microsoft ale neponechala stranou správu svých stanic. Za nápomocné považují následující dva systémy.

1.1.1.1 Domény

Domény jsou původně síťový nástroj, nezávislý na firmě Microsoft. Jde o jednoduchou adresářovou službu, ve které jsou umístěné informace o objektech v počítačové síti – spravovat prostřednictvím domén můžeme především uživatelské účty a oprávnění uživatelů, počítače a tiskárny. Doménové informace jsou uloženy na počítači, který nazýváme doménovým řadičem (PDC, Primary Domain Controller), případně na záložním počítači (BDC, Backup Domain Controller).

Společnost Microsoft doménové řešení adaptovala na své komerční prostředí. Ve Windows 2000 pak firma pozměnila koncepci a klasické domény nahradila službou Active Directory.³

1.1.1.2 Active Directory

Prostřednictvím Active Directory můžeme tedy spravovat podrobně a hromadně oprávnění jednotlivých uživatelů. Nezáleží, na jaké stanici fyzicky se daný uživatel přihlásí.

Active Directory také umožňuje nastavit skript připravený administrátorem a použít jej hromadně v určený čas na klientských stanicích.

Připomeňme, že služba Active Directory je vázána na „klikací řešení“ serverového operačního systému, jehož licence nepatří k nejlevnějším.

² Dojmy z používání obou systémů jsou jednoznačné. Nainstalovaný linuxový server může běžet nepřetržitě třeba půl roku a nebudete o něm ani vědět. Je to stroj, který pracuje přesně tak, jak jste ho nastavili. Zkuste totéž provést s Windows.

³ Anglická Wikipedie, 30. března 2016

1.1.2 Příklady dalších systémů pro správu sítě

Při správě sítě může administrátor sáhnout také po některém z komplexních řešení jiných společností. Pokusím se ukázat několik příkladů z mnoha. Většinou se jedná o komerční software.

1.1.2.1 LanDesk

Systém LanDesk pochází ze Salt Lake City v Utahu ve Spojených státech amerických. Je jedním z nejstarších řešení hromadné správy, vývojáři na něm pracují už od roku 1985. Zaměření LanDesku je multiplatformní, je možné spravovat nejen uživatelské stanice s některou z verzí Windows, ale také například tablety s Androidem a mnoho dalších typů stanic. LanDesk umí poskytnout komplexní management systémů, služeb, procesů i bezpečnosti.⁴

1.1.2.2 Novell Netware

Novell, Inc. je americká firma, která vyvinula síťový operační systém, tehdy určený ke správě počítačů 80286. Před příchodem Win NT server vykazoval Novell Netware 90% serverových OS. Kolem roku 1995 však společnost ztratila většinu trhu. Systém je nadále pro svoji flexibilitu používán některými většími organizacemi.

Od roku 2005 má tento systém své nástupníky: Novell Open Enterprise Server, OES-Linux a OES-Netware.⁵

1.1.2.3 Citrix

Citrix Systems, Inc. je americká společnost se sídlem na Floridě. Spolupracuje s dalšími více než tisíci velkými společnostmi v dalších 100 zemích. Její vizí je práce na PC kdykoliv, odkudkoliv a z jakéhokoliv stroje.

Citrix poskytuje jistá řešení centrální správy, ta jsou však spíše zaměřená na virtualizaci desktop a serverů, síťování, software ve formě služby a cloudové technologie.⁶

⁴ www.landesk.com, 30. března 2016

⁵ www.novell.com, 30. března 2016

en.wikipedia.org/wiki/Novell, 23. března 2016

⁶ www.citrix.com, 30. března 2016,

1.1.2.4 BatchPatch

Americká společnost Cocobolo Software, Inc. se zaměřila na deployment aktualizací v celé síti pro systém Windows. Její řešení není zdaleka tak robustní jako v případě dvou výše zmíněných firem a její internetové stránky obsahují stručné a jasné informace o funkcích, které systém zvládá. BatchPatch je založený na programu Psexec za Sysinternals Suite, kterou popisují v kapitole o příkazovém řádku Windows.

Kromě aktualizací Windows umí systém hromadně instalovat také aplikace třetích stran a jejich aktualizace, příkladem budiž Java od firmy Adobe. Systém nevyžaduje žádnou instalaci na klientských počítačích. Přesto umí ukázat, zda je stanice online či offline, zjistí stručnější informace o počítači a umí také použít službu vzdáleného zapnutí počítače, Wake On LAN.

Ale především, tento systém umí to nejdůležitější: vzdálené ovládání služeb a hromadné použití administrátorova skriptu. Nejen proto se nejvíce blíží hledanému systému správy stanic.

BatchPatch má pro mě jedinou nevýhodu: licenci, která neumožňuje program používat zdarma.⁷

1.2 Výchozí zkušenosti

Důležité pro mne bylo zjištění, které jsem učinil hlavně díky nadšení svého školitele, že téměř všechno na počítači se dá ve výsledku nastavit skriptem, protože všechna nastavení a nainstalované programy jsou vlastně jen soubory a zápisy v registrech. Považuji za klíčové, že to vše opravdu lze. Bez této premisy bychom vlastně předem nedokázali říci, zda hromadná správa počítačů, postavená na pouhých skriptech, má smysl.

1.2.1 Příkazová řádka Windows

V období před Windows 95 byla příkazová řádka (dále jako CMD) hlavním komunikačním prostředkem mezi uživatelem a operačním systémem. Historicky na ní stál celý provoz OS a až do verze Windows 95 tvořilo grafické uživatelské rozhraní pouze jeho nadstavbu. S nástupem Windows NT (počínaje serverovými Windows 2000) se situace mění a CMD se stává spíše jakýmsi doplňkem grafického rozhraní. Jinými

⁷ batchpatch.com, 10. února 2016

slovy: jako uživatelé Windows si bez CMD krásně vystačíme, protože veškeré nastavení lze „naklikat“.

Grafické prostředí hromadnou správu značně komplikuje. Pro příkazový řádek lze také tvořit skripty – prosté textové soubory, kterým změníme koncovku z *.txt* na *.bat*, popř. *.cmd*.⁸

Původně jsem chtěl v celé práci používat jen příkazy Windows. Narazil jsem však na těžkopádnou a ne zcela intuitivní syntaxi. Například při tvorbě cyklů není použití proměnných zcela jednoznačné – např. chybějící mezera nebo mezera navíc může být problém. Skript je rázem nefunkční. Začátečníkům by se ale mohl líbit fakt, že CMD nerozlišuje velká a malá písmena, není tzv. case senzitivní.

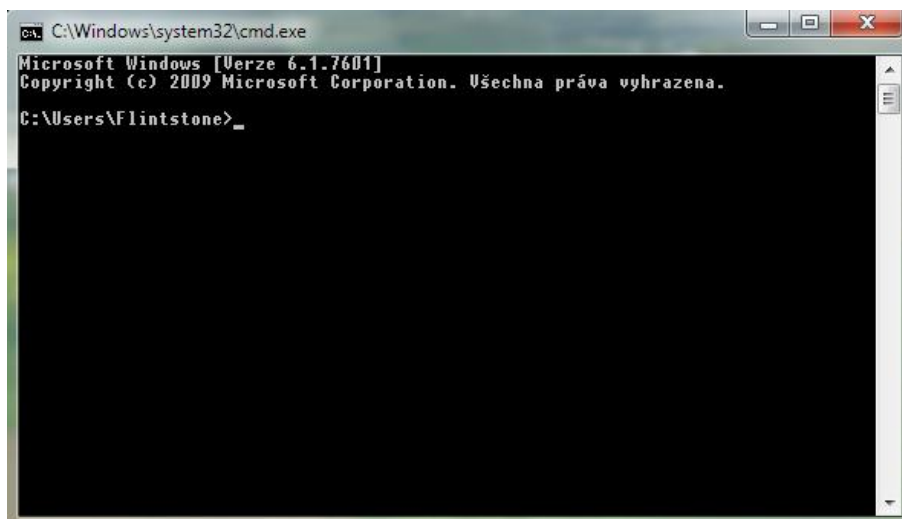
Příkazový řádek ve Windows sám o sobě nemá až tak mnoho možností, ale existuje mnoho rozšíření, které potřebné funkcionality zajistí. Zprv bych chtěl zmínit sadu hotových programů SysinternalsSuite, kterou je možné stáhnout přímo ze stránek Microsoftu⁹. Adresář s programy zkopírujeme přímo na disk C: a cestu C:\SysinternalsSuite\ přidáme do systémové proměnné path. Nyní můžeme jakýkoliv program z této sady použít stejně, jako by byl součástí příkazového řádku odjakživa.

Z této sady bych rád vyzdvihl PsExec, který umožňuje spustit jiný program s nejvyššími oprávněními, a to dokonce vzdáleně. Ostatně, využívá ho komerční systém správy s názvem BatchPatch, ze kterého mimo jiné čerpám inspiraci. Také program Procexp se může velmi hodit – umí zobrazit strom spuštěných procesů, jejich návaznosti na sebe navzájem i na spuštěné služby, vykreslí také využívání hardwarových prostředků. SysinternalsSuite toho umí mnoho. Dříve nebo později rozšířené funkce administrátor ocení.

Další rozšíření poskytuje samotný systém Windows, a to sice příkaz WMIC, kterým se dostaneme v příkazovém řádku do zvláštní konzole. WMIC má své vlastní příkazy a syntaxi hodně podobnou dotazům jazyka SQL. V reinstalačním skriptu jsem jej s výhodou používal třeba pro nastavení hostname počítače.

⁸ Je v tom určitý rozdíl – typ *cmd* je novější a umožňuje použít i nové příkazy. V NT systémech už lze používat oba typy skriptů.

⁹ en.wikipedia.org/wiki/Sysinternals, 10. února 2016 – Microsoft SysinternalsSuite raději odkoupil, aby přibrzdil rozvoj těchto nízkoúrovňových utilit. Tím se vysvětluje zajímavý rozpor: na první pohled totiž Microsoft sám není jejich autorem.



Obrázek 1 - Okno příkazového řádku ve Windows 7.

Systém Windows disponuje také jazykem Visual Basic, který znají pokročilejší uživatelé třeba z tvorby maker v Microsoft Office. Visual Basic je však programovací platforma umožňující práci s celým systémem. Můžeme v něm tvořit také skripty s příponou `.vbs` a tyto skripty spouštět přímo z příkazové řádky příkazem `Cscript`. To se může velmi hodit – třeba při práci uvnitř textových souborů.

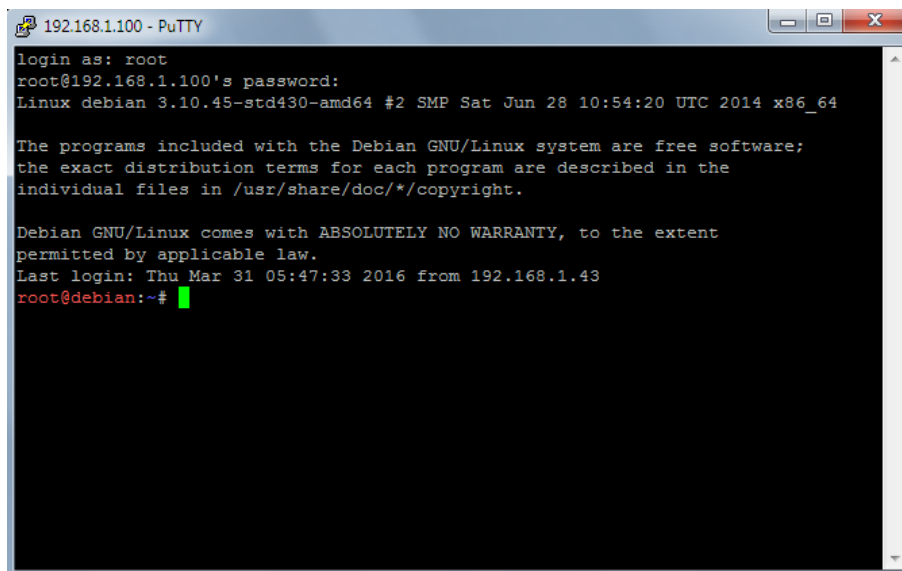
Neměl bych opomenout také PowerShell Windows, do kterého se postupně přesouvá funkcionalita, dříve dostupná pouze z externích aplikací.

Z tréninkových důvodů, ale také z reálné potřeby na mém pracovišti, jsem se rozhodl naprogramovat sadu Windows skriptů, použitelných pro kompletní reinstalaci uživatelského počítače. Skripty provedou instalaci programů, jejich nastavení, ale třeba i instalaci síťových tiskáren. Reinstalační skripty jsou nyní ve stádiu vývoje a probíhá jejich testování. Byly pro mě velmi užitečné – během tvorby nejrůznějších dávek mi poskytly tu nejlepší průpravu pro programování v příkazové řádce Windows.

1.2.2 Linuxový terminál bash

Od druhého ročníku studia jsme byli na škole seznamováni se systémy na bázi Unixu. Abych se lépe naučil v tomto prostředí orientovat, postavil jsem si domácí souborový server s operačním systémem Debian, který naší rodině slouží jako síťové úložiště a umožňuje také zálohu dat pro členy domácnosti. Po zkušenostech s ním a v porovnání s komerčními jednoúčelovými NASy se mi takové řešení velmi zamlouvá.

Bash je jedním z nejrozšířenějších linuxových shellů. Na rozdíl od příkazového řádku Windows je case senzitivní, tj. musíme si dát pozor na použití velkých nebo malých písmen.



```
192.168.1.100 - PuTTY
login as: root
root@192.168.1.100's password:
Linux debian 3.10.45-std430-amd64 #2 SMP Sat Jun 28 10:54:20 UTC 2014 x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Mar 31 05:47:33 2016 from 192.168.1.43
root@debian:~#
```

Obrázek 2 - Linuxový terminál bash po připojení na vzdálený server cestou SSH. Windows a Linux by si spolu nerozuměli, ale program Putty umožňuje správci ovládat Linuxové servery přímo z Windowsového okna.

Linux je na rozdíl od Windows modulární. Správce si tak může vytvořit systém, který obsahuje jen to nejn nutnější.¹⁰

Bash však není jediným interpretem, který lze v linuxu používat. Linuxové jádro podporuje také C, C++, Perl a další. Bashové skripty jsou rozhodně velmi užitečný nástroj. Ale mají i svou slabinu: vyznačují se obdobně těžkopádnou syntaxí jako příkazy Windows. Jakmile potřebujeme využívat cykly, složitější podmínky a rozsáhlejší kód, je dobré sáhnout po jiném, vhodnějším jazyce.

1.2.3 Python

Bash i CMD jsou systémoví interpreti. Python přichází na scénu též jako skriptovací jazyk, zároveň ale velmi vhodný pro tvorbu aplikací, čímž spojuje klíčové výhody nízkourovňových i vysokoúrovňových jazyků. Python v současné době zažívá

¹⁰ Ani se nezdá, jak velká výhoda to může být! Nižšími HW nároky a lepší odezvou systému to jen začíná. Bezpečnější struktura (co není nainstalováno, to nemůže obsahovat bezpečnostní problém) je už výmluvnější. Celková jednoduchost systému ale také není k zahoeení, protože známe fyzikální zákon: čím méně součástí, tím nižší pravděpodobnost poruchy!

vzestup a dalo by se říci, že odsouvá jiné těžkopádnější jazyky poněkud do pozadí. Přináší inovaci v syntaxi: nepoužívá závorky k členění kódu, ale řídí se pouhým odsazením řádek. Skripty lze tvořit v poznámkovém bloku nebo v Notepad++, ale lepší je editor IDLE či PythonWin. Práci může velmi usnadnit některé z robustnějších vývojových prostředí, vytvořených speciálně pro Python¹¹.

Python¹² je objektově orientovaný jazyk, umí tedy využít třídy. S trochou nadsázky lze říci, že vše v pythonu je objekt. Třidu i s jejími metodami si můžeme uložit jako modul a jeho obsah využívat v dalších skriptech. To velmi usnadní práci.

Moduly Pythonu jsou velmi malé – základní instalace Pythonu zabírá na disku něco kolem 30MB. To je v dnešní době nevídaně příjemná velikost – zejména ve srovnání s platformou .Net! Z internetu je možné stáhnout další potřebné moduly. Všechny nekompileované lze také otevřít a kód si projít, což práci s tímto jazykem velmi zpřehledňuje.

Práci mi usnadňuje možnost každou změnu ve skriptu ihned odzkoušet bez nutnosti cokoliv kompilovat. Hardwarové nároky pro python jsou zanedbatelné – vážím si toho tím víc, když zjišťuji, co by mě čekalo s programem pro Android¹³. Příjemná je také možnost předem vyzkoušet chování příkazů v IDLE prostředí mimo hlavní program.

Třešničkou na dortu je následující fakt: příkazová řádka je schopna se odkázat na interpreta Pythonu i na serveru, který teprve překlad skriptu zajistí. V praxi to znamená, že skript poběží bez problému dokonce i na počítači, na kterém žádný Python není nainstalován!

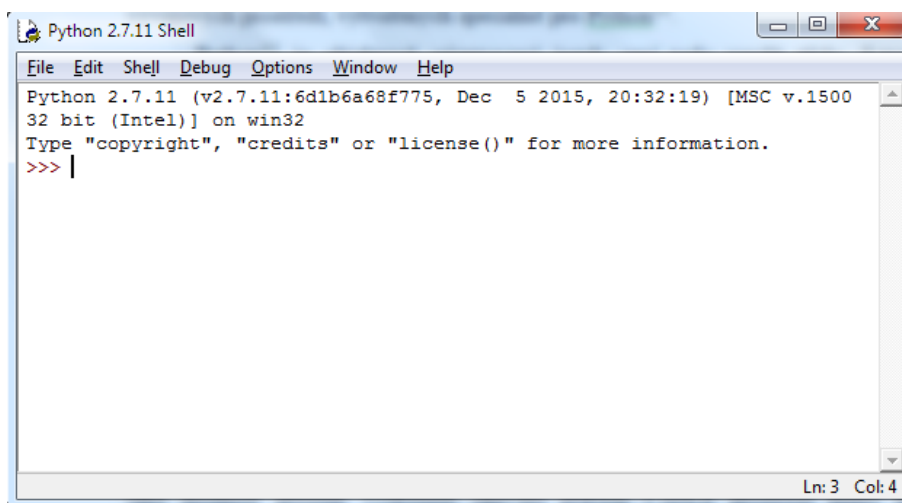
Jazyk Python mi učaroval, i když musím konstatovat, že začátky jsou asi všude obtížné. Přiznávám, že mi zpočátku dělalo dost problém zvyknout si na nové vývojové prostředí – pythonovský program lze tvořit prakticky v čemkoliv, co umí zvýrazňovat syntaxi. Ale ani Notepad++, ani PsPad, ani výchozí IDLE či PythonWin mi neumí pod rukou nabízet použitelné příkazy, třídy a atributy... Takže kód se tvoří spíš po paměti a

¹¹ PyCharm je zdarma, ale lepší Wing IDE bohužel stojí asi 70 USD. Podle všeho i profesionálové na práci s Pythonem skriptují třeba v Notepad++. Python to prostě umožňuje. V takovém C# si to ale moc neumím představit.

¹² Viz zdroje.

¹³ Spolužák se rozhodl programovat pro Android. Bez studia značně náročného na hardware to ovšem nejde.

orientujeme se vlastně jen podle dokumentace a zkušeností. Za tu přenositelnost, nenáročnost, modularitu a rychlost při testování to však rozhodně stojí.



Obrázek 3 - Konzole Pythonu, kde lze zkoušet příkazy interaktivně.

1.3 Záměr

Kdo někdy zkoušel udržovat více počítačů zároveň, ví dobře, že bez systému umožňujícího hromadnou správu a jistý stupeň automatizace, to víceméně nelze. Ale peníze na nákup již hotového komerčního řešení třeba zrovna nejsou nebo nikdy nebudou. Proč také investovat do IT, když výsledek nikdo neuvidí?

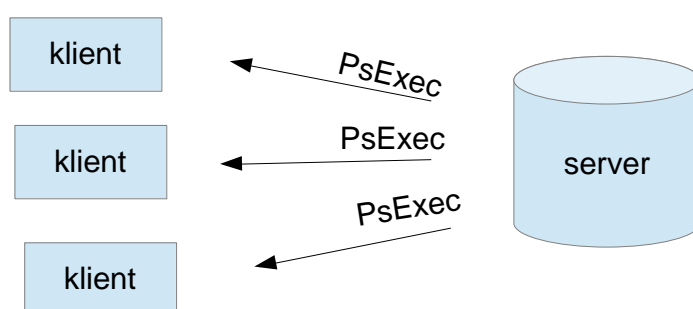
Lék existuje, vyžaduje pouze dávku nadšení a odvahy. Můžeme využít všech vlastních vědomostí k tomu, abychom si vyrobili své vlastní nástroje, které nám kýžené funkcionality automatizované správy sítě zajistí. Nemusíme platit licence. Nemusíme testovat množství programů, abychom z nich vybrali ten, který se nejvíce blíží naším potřebám. Můžeme vyrobit řešení na míru, přesně pro naši síť. Pomineme-li nároky tohoto postupu na zkušenost a nadšení administrátora, je to řešení docela výhodné.

Výsledné skripty (případně aplikace) jsou jeho vlastním dílem, proto si je může stále upravovat ke svému obrazu. Dále přesně ví, co a jak na jeho síti funguje. Při komplikacích v budoucnu mu to značně usnadní hledání příčiny.

Orientační kritérium bych viděl asi toto: pokud řešení, které je třeba zkonstruovat, není svým rozsahem nereálné, pak i při nemalé ceně administrátorových pracovních hodin, ušetříme jistě mnohem větší částky.

1.3.1 Přímé ovládání versus naslouchající služba

Zaprvé, můžeme počítač vzdáleně ze serveru oslovit pomocí hostname, a skript mu zadat k provedení. Musíme však projít bezpečnostní prověrkou – tou jsou zejména přihlašovací údaje administrátorského účtu. Ty je nutné zadat pokaždé. V cestě může navíc stát firewall. Ale i po jeho odstavení skript nemusí být proveden, a to díky rozvrstvení práv ve Windows¹⁴. Nejlépe by se hodilo využít program PsExec, se kterým se mi spuštění jednoduché úlohy na dálku skutečně podařilo. Ale i tak musí být systém dobře nastaven a řešení to není ideální pro nutnost v každém příkazu opakovat přihlašovací údaje.¹⁵



Obrázek 4 - Přímé ovládání stanic serverem.

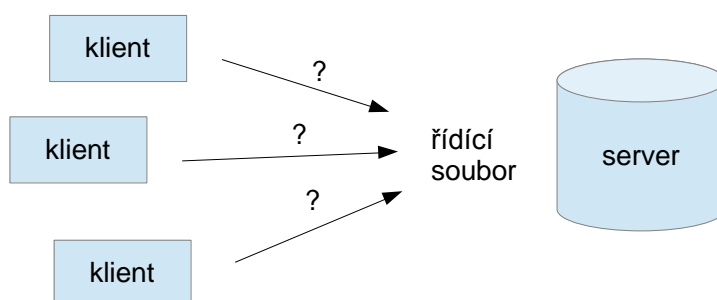
Druhou možností je nastavit všechny počítače v síti tak, aby se samy dotazovaly serveru, zda mají něco provést či nikoli. Je jasné, že v určitém časovém intervalu, třeba po každých deseti sekundách. Dotaz může být realizován docela jednoduše – otevřením řídicího textového souboru, uloženého na serveru nebo síťovém úložišti. Tento textový soubor obsahuje hostname počítačů, které skript ležící vedle mají provést. Analýza souboru, zda v něm je nebo není můj vlastní hostname (z pohledu počítače – klienta), je už velmi jednoduchá.

Druhý postup používá také malware, zneužívající často náš vlastní počítač k vykonávání podezřelých a nežádoucích činností, a to zadávaných serverem hluboko z internetu. Pokud tedy tento princip umožňuje fungovat malware, a to přes všechny

¹⁴ Světe div se, ale „Administrator“ není nejvíce privilegovaný účet ve Windows. Tím je „System“. Toto zmítnutí práv ve Windows poněkud znesnadňuje rychlé pochopení hierarchie práv uživatelů ve Windows. Těch účtů je ve Windows vlastně ještě více a vzdálenou správu to jen komplikuje.

¹⁵ Činnost programu PsExec musí být povolena v registrech systému Windows. Navíc přihlašovací údaje běžící síti nezabezpečené, jako prostý text.

antivirové programy a firewally, proč by nemohl být využit jako spolehlivé řešení k hromadné správě sítě?



Obrázek 5 - Dotazování samotných stanic.

Pro opakované dotazování serveru klientem jsem zkoušel použít Plánovač úloh a skript spouštět po přihlášení uživatele. Ale není to uspokojivé řešení.

Ve Windows můžeme také vytvořit vlastní službu. Stačí k ní skript zkompileovaný do .exe souboru. Bohužel i tak je co nastavovat. Objevil jsem ukázkou čtvrt hodinového videa na youtube, které vytvoření služby demonstruje.¹⁶

S Pythonem lze službu zprovoznit bez nutnosti zapisovat přímo do registrů nebo cokoliv kompilovat. Není třeba nic dalšího nastavovat. I pokud bychom přeci jen chtěli získat zkompileovaný skript, knihovna *py2exe* to pro nás ráda udělá. Takové řešení se mi jeví jako čisté a profesionální.

1.3.2 Výhody klientské služby

K hromadnému provedení úlohy vlastně drahý server od MS, ani žádný jiný, nepotřebujeme. Stačí opravdu jen síťové úložiště, ke kterému mají stanice přístup. Příkaz nemusíme specifikovat pro každé PC zvlášť – stačí jen záznam v jednom řídicím textovém souboru.

Ostatní obstarají běžící služby na klientských počítačích. Z hlediska všech požadavků na síťový server naprosto ideální situace.

Bonus takto řešené správy je nasnadě: editaci řídicích souborů můžeme provádět v zásadě odkudkoliv a jakkoliv. Hlavní program vyvíjené aplikace by nám to měl jen usnadnit.¹⁷

¹⁶ <https://youtu.be/X6o1AvZ06zc>, 10. listopadu 2015

2 PŘÍPRAVA VYVÍJENÉ APLIKACE

V předchozích kapitolách jsem se pokusil zmapovat témata, která mají s budoucím výsledkem práce něco společného.

Aby tvorba aplikace byla dostatečně efektivní, je nutné vytyčit cíle a připravit se na možné překážky. V této fázi by výsledkem práce měla být vhodně zvolená osnova pracovních činností, zohledňující vše z předchozích kapitol.

2.1 Případová studie

Cílem této práce není „release“ ucelené aplikace, ale spíše načrtnutí a zprovoznění komunikačního kanálu mezi ovládacím serverem (Linux) a hromadně ovládanými klientskými počítači (Windows). Aplikace by tedy měla umět:

- vypsat dosažitelná PC na síti a ovládaná PC začleněná do správy, pokud možno s jejich stavem, tj. zda jsou online apod.
- přehledně vypsat základní informace o ovládaném klientském PC jako je zejména verze operačního systému, poslední update, seznam nainstalovaného software, poslední přihlášení uživatele, zda uživatel právě pracuje, případně kdy přestal pracovat (odchod na oběd atd.)
- klientská PC restartovat, vypnout, odhlásit aktuálního uživatele, přihlásit jiného uživatele, s využitím WOL možná PC i zapnout (požadována reakce klientského PC nejlépe do několika sekund)
- spustit jakoukoliv úlohu na klientském počítači s nejvyššími oprávněními (tj. pokud se provede testovací úloha, provedou se pravděpodobně také jakékoliv jiné úlohy)
- oddálit spuštění zadaného skriptu na klientském PC na vhodnou dobu, například vyčkat na odhlášení uživatele a tehdy update Javy provést, naopak jiné úlohy provádět ihned, například restart stanic

¹⁷ Při použití VPN mohu snadno řídicí soubory editovat vzdáleně, třeba z pohodlí svého domova a ve svém oblíbeném editoru.

- shromažďovat přehledně do logu informace a provedených servisních úlohách a zda skončily úspěchem či neúspěchem,
- pokud daná úloha na klientském PC selže, upozornit admina (minimálně zápisem v logu, ale lépe poznámkou v terminálu apod.)
- finálně by aplikace měla obsahovat sadu nejběžnějších a nejpotřebnějších úloh, které bude možné spustit rovnou z příkazové řádky.

Jako výsledná forma aplikace je požadován minimálně skript s parametrem. Další možností je terminálový program (bez grafické nadstavby), terminálový program postavený na nCurses, tj. „pseudografické“ prostředí známé například z Clonezilly, Midnight Commanderu, Htopu či instalátoru linuxové distribuce Debian. Ovládání by bylo výhradně přes klávesnici z jakéhokoli počítače na síti s Windows ze spuštěného programu Putty. Program by tedy běžel na serveru, ovládání by bylo možné i odjinud, bezpečnost pro přihlášení nám řeší samotné přihlášení serverovému terminálu cestou SSH.

Plně grafické nadstavbě programu, postavené například v jazyce Python či C# se věnovat nechci, protože aplikace má běžet v textovém prostředí serveru, nejlépe vzdáleně.

Nabízí se však ještě jedno řešení, a to velmi příhodné. Tímto řešením je webová stránka, která by obsahovala všechny ovládací prvky programu včetně zvoleného grafického stylu programu. Ovládání by tak bylo možné z jakéhokoli počítače v síti, s použitím VPN klidně vzdáleně. Hardwarové nároky na server sice vzrůstají o webové služby, protože by musel běžet alespoň minimalizovaný webový server. Možnosti zabezpečení komunikace – velmi dobré.

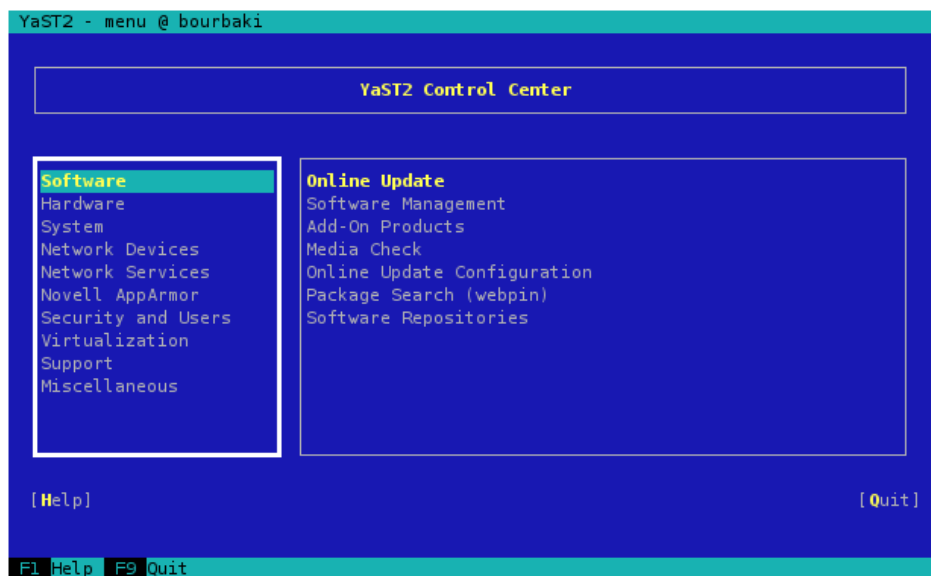
Webová stránka se tedy zdá jako velmi vhodné rozšíření, poskytující dostatečné pohodlí i rychlost při ovládání programu.

2.1.1 Uživatelské rozhraní

Pro výslednou aplikaci je požadováno dosáhnout alespoň formy příkazu s parametry v příkazovém řádku. Využití Pythonu mi umožní program testovat na Windows i v linuxu.

Zároveň bude aplikace napsána tak, aby bylo možné co nejjednodušeji v budoucnu navázat zvoleným grafickým prostředím. Samotné grafické prostředí v této fázi není podstatné, bylo by příliš zavádějící a jeho tvorba by zabrala mnoho času navíc.

Navíc, se změnou zásadních funkcí programu by bylo nutné neustále měnit také grafické prostředí. Proto je tedy nanejvýš rozumné, zaměřit se nejprve na základní funkce programu, aplikační „jádro“.



Obrázek 6 - Ukázka prostředí nCurses.

2.2 Testovací prostředí

Následující kapitoly popisují systém práce, který se mi velmi osvědčil a mohu jej vřele doporučit.

2.2.1 Virtuální počítače a samba

Při tvorbě kódu si plně vystačím se skupinou dvou až tří virtuálních počítačů. Jakmile bude první verze aplikace naprogramována, odladěna a dostatečně otestována ve virtuálním prostředí, bude se hodit zkušební provoz na skutečné síti i s jejími uživateli a její postupné nasazení „naostro“.

Nyní oceňuji svůj domácí server, zprovozněný už na počátku studií. Nyní se mi velmi hodí, protože na něm mohu testovat chování naprogramované služby v naprosto reálném prostředí sítě s plnohodnotným linuxovým serverem a nainstalovanou

sambou.¹⁸ Spolu s VirtualBoxem a dostatečně výkonným osobním počítačem postaveným na platformě AMD mám vše pro dostatečně komfortní práci.

2.2.2 Vybraná síť pro nasazení

Fakt, že již během studia pracuji jako lokální správce sítě v jisté školské organizaci, je mi nyní velmi nápomocný. Protože na tomto pracovišti dosud postrádám chybějící nástroje hromadné správy, rozhodl jsem se postavit vyvíjenou aplikaci tak, aby naplňovala potřeby právě této sítě.

Naše organizace je součástí většího organizačního celku. Z toho vyplývají jistá omezení, která musím bohužel přijmout. Každý přidaný počítač musím nahlásit a na základě jeho fyzické adresy je provedeno přidělení IP adresy z DHCP. Ostatní nenahlášená zařízení na síti vůbec nefungují. Není zde možné zapojit router a vytvořit další síťové segmenty. Dále antivirová ochrana v celé síti je realizována systémem spravovaným centrálně. Do něj jako lokální správce přístup mám. Hromadná antivirová ochrana je velkou výhodou, ale v případě hromadné správy se může stát i komplikací.

Pro serverové služby (souborový server, webový server, poštovní server apod.) síť může provozovat svůj vlastní server. Aby modelový případ byl úplný, volím univerzální a pro mě nejvíce známou distribuci Debian¹⁹.

Ostatní počítače obyčejných uživatelů tvoří desktopy, na kterých poběží systém Windows verze 7 a výše. Jako možný předpokládám také Windows XP, nižší však nikoliv.

2.3 Očekávané problémy

V průběhu práce jsem narazil na komplikace, se kterými jsem počítal v předchozích fázích. Nejvíce času pohltilo nelehké programování služby, kterým jsem bohužel musel téměř bez zkušeností začít. Obavy mám také z různých verzí systému Windows, především z nových „desítek“.

¹⁸ Právě konfigurace samby se mi stala jednou z překážek. Nastavení mé samby tak, aby služba měla povolený zápis, nebylo jednoduché. Nakonec jsem to obešel vytvořením uživatelského účtu Windows se stejnými přihlašovacími údaji, které samba požaduje k přihlášení ke sdílenému disku.

¹⁹ Na bázi debianového jádra jsou postaveny také další velmi rozšířené distribuce Ubuntu a Linux Mint. Obě tyto mohu vřele doporučit i pro domácí uživatele – laiky. Když jsem před lety začínal s linuxem, zvolil jsem si po dostatečném průzkumu na internetu právě Debian. Tato volba se mi časem ukázala jako velmi dobrá.

2.3.1 CMD a Bash – úskalí samotného jazyka

Úskalí těchto dvou jazyků jsem popisoval výše. Jsou jimi především těžkopádná syntaxe a omezená působnost vytvořených skriptů. Překážka byla použitím Pythonu na všechny složitější struktury.

2.3.2 Python – úskalí začátečníka

Na začátku této práce jsem nebyl odborníkem na Python a rozhodně si netroufám tvrdit, že jsem se jím stal v průběhu. Počátky programování v novém jazyce nebyly bez obtíží, ale nyní je považuji za překonané.

2.3.3 Uživatelská oprávnění systému Windows

Dejme tomu, že na daném klientském PC právě pracuje uživatel Karel v uživatelském účtu „Karel“. Přihlásil se svým heslem „vlastovka“ a rozhodně nemá nyní náladu počítač uvolnit pro údržbu. Zároveň však požádal o aktualizaci/instalaci Javy, protože ji potřebuje pro další práci.

Jakmile Karel odejde domů, museli bychom počítač jen pro tento úkon mimo pracovní dobu zapnout. Stanice však nemusí být dost dobře přístupná. (Potřebujeme klíče od kanceláře, fyzicky se musíme na pracoviště dopravit, zakódovaná budova atd.)

Na počítači je přihlášen „Karel“, zatímco naslouchající služba naší aplikace je spuštěná uživatelem „System“. V tomto případě nám situace nahrává, protože zadaný skript se provede bez toho, že by Karel cokoliv postřehl nebo musel na cokoliv klikat. A provede se s nejvyššími právy účtu System. Zbývá tedy skript postavit tak, aby se aktualizace spustila neinteraktivně, protože jakmile by kdesi v účtu System viselo neviditelné okno čekající na odkliknutí, proces by nemohl pokračovat dál a úloha by neskončila úspěšně. To se však dá dobře řešit ve Windows například spuštěním .msi balíčku s příslušným parametrem.

Jiná situace nastává, máme-li Karlovi spustit na dálku například výzvu s informacemi o provedené údržbě nebo jinou aplikaci požadující spolupráci od Karla. Teď naopak potřebujeme viditelné okno spuštěné účtem System, ale zobrazené v přihlášeném účtu Karel. To je docela oříšek, ale ne nepřekonatelný. Musíme na to jen myslet při programování služby a věnovat tomu dostatek pozornosti a testů.

Zbývá rozhodnout, kdy je vhodné zobrazit okno se správčovskými oprávněními uživateli a kdy to naopak vhodné není.

2.3.4 Multiplatformní prostředí

Práce je cílená do struktury sítě v tomto typickém složení: server s linuxovým operačním systémem spolu s klientskými počítači s některou s verzí Windows. Časem bych rád vytvořil také podporu pro linuxové klienty a klienty firmy Apple. Vzhledem k objemu práce však není možné toto docílit už při jejím odevzdání.

I pokud všichni klienti na síti budou počítače s Windows, počítám se situací, kdy některé počítače budou Windows 7, jiné Windows 8 nebo 10, některé dokonce Windows XP. S Windows Vistami a Windows staršími než XP se v této práci zabývat nebudu. V případě Windows Vista proto, že se prakticky vůbec nepoužívají. Nebyl to příliš povedený systém, proto se přecházelo z XP rovnou na Windows 7. V případě systémů starších než Windows XP je vlastně situace stejná (málokde používaný systém, navíc již dosti zastaralý), ačkoliv třeba takové Windows 98 byly ve své době velmi rozšířené.

2.4 Postup práce

Textové soubory ručně upravovat lze, zatímco fungování služby u klienta jde nahradit jen těžko. Tím bylo dané, že jsem do samotného programování vstoupil právě klientskou částí, tj. naslouchající službou.

2.4.1 Programování služby

Brzy jsem zjistil, v čem je programování služby specifické. Chybí totiž zpětná vazba. Při programování služby mohou nastat vlastně jen dva stavy: služba běží tak jak má, nebo její spuštění selže. Veškerá zpětná vazba je dána jediné logováním, které jsme do kódu přidali. Problematické prvky služby tak musíme odladit v samostatném programu a teprve funkční celek do služby vsadit. Prakticky to ovšem i tak znamená sérii neúspěšných pokusů o spuštění služby, úpravy logování (logovat pak musíme prakticky vše), úpravy kódu, zastavení služby, opět spuštění, a tak pořád dokola.

Nápověda se nakonec přece jen našla: a to sice protokoly aplikací správcovské konzole Windows. Výpisy zde nejsou tak podrobné, jako by byly přímo v terminálu, ale číslo řádky a druh chyby zde najdeme.²⁰ Pokud je chyba v syntaxi nebo natolik kritická,

²⁰ Nečekal jsem to, ale při hledání chyby, která je pro běh služby zásadní, se podle aplikačních protokolů dá docela dobře orientovat.

že služba s ní běžet nemůže, protokoly aplikací chybu zaznamenají i s podrobným popisem. Bez tohoto nástroje bych si programování služby pro Windows asi neuměl představit.

Překvapivý byl také fakt, že po úpravě kódu není vůbec nutné celou službu odinstalovat a znovu nainstalovat. Stačí pouhý restart služby. Klávesa F5 pomůže k rychlé aktualizaci stavu služby, myší v okně services.msc provedeme restart. Služba dokonce nemusí být nutně spuštěná z lokálního souboru, není tedy třeba kód po každé úpravě do pokusného virtuálního počítače kopírovat. Stačí jen spustit z příkazového řádku. Upravený kód se načítá se startem služby, a to klidně ze sítě.

Při zásadnějších úpravách kódu běžící, nezastavené služby (častý stav při mém testování), se může stát, že službu nelze zastavit. Respektive systém se o to pokusí, služba ale zůstane viset ve stavu „zastavení“. V takových případech se mi osvědčilo odstranění služby. Pokud to nezabralo, pak pomohlo násilné ukončení procesu pythonservice.exe buď přímo ve správci úloh, nebo z příkazové řádky Windows:

```
sc delete bee41an  
TASKKILL /F /IM pythonservice.exe
```

Tyto dva příkazy lze samozřejmě spojit v dávce, kterou jsem si pracovně nazval *killbee.bat*.

2.4.2 Testování služby

Nejdříve ze všeho jsem měl k dispozici pokusnou dávku uloženou na klientském počítači s Windows 7 ve složce *Po spuštění*. Tímto způsobem jsem testoval základní funkčnost, tj. hromadné ovládání pomocí zápisu v textovém souboru na síťovém úložišti. Dávka zjistila nový úkol vždy při přihlášení uživatele, a pokud byla určená pro daný klientský počítač, úkol se provedl. V této fázi jsem si připravil několik základních skriptů jako samotné úkoly k provedení. Jednalo se o *.bat* a *.cmd* dávky. Nebylo však možné využít jiný čas k vykonání úkolu a bylo nutné zajistit regulérně viditelný soubor proti smazání uživatelem.

Naplánované úlohy na rozdíl od složky „Po spuštění“ slibovaly širší možnosti. Fungování mi však selhávalo. Zadat do systému novou naplánovanou úlohu by pomocí dávky možné bylo (základní podmínka). Provedení zadaných úloh v síti Tandemu však stále selhávalo. Bylo to kvůli antivirovému systému McAfee, který považoval zjišťovací dávku za škodlivou a nepovoloval její spouštění. To jsem ovšem zjistil až později.

Dokončení finální služby v pythonu mi konečně pomohlo rozšířit fungování i během doby práce uživatele. Kromě toho byl systém správy obohacen o využití python skriptů a především o mnohem sofistikovanější řešení zahrnující také logování úspěchu či neúspěchu a vůbec stavu služby. Smazání služby uživatelem už vlastně možné není a její soubory se nepletou nikde mezi uživatelskými.²¹

2.4.3 Testování jádra aplikace

Jak jsem již psal, celý systém správy je schopen fungovat zcela bez serveru. Vlastně je funkční i bez řídicí aplikace, protože řídicí textové soubory lze upravovat také ručně. Jen je to méně pohodlné a ne příliš přehledné. Nicméně pro první testy se službou stačil i pouhý poznámkový blok, případně program Lister, fungující jako integrovaný poznámkový blok v Total Commanderu. Výhoda Listeru je jasná: neptá se na koncovku souboru, ale co lze zobrazit ve formě textových řetězců, to jednoduše zobrazí. Jeho integrace v Total Commanderu z něj dělá editační nástroj textových souborů číslo jedna.

Řekněme, že modelový úkol třeba spočívá v provedení aktualizace poštovního programu Thunderbird. Určitě nechceme, aby se úkol prováděl na jednom klientovi stále dokola. Zároveň ale nejsou zapnuté všechny stanice nebo nemají všechny stanice momentálně možnost úkol provést (například čekáme na polední pauzu pracovníků, ti ale nechodí na oběd ve stejný čas). To tedy znamená, že v řídicím souboru chceme mít úkol nastavený, zároveň ale chceme, aby po jednom provedení služba úkol již dále neprováděla.

S tímto problémem bylo nutné se vypořádat. Úlohám jsem tedy začal přiřazovat čísla, pro pořádek rovnou trojmístná, začínáme tedy na čísle 101.²² Čísla dávají službě možnost základní orientace – podle čísla tedy služba pozná, zda daný úkol již prováděla či nikoli. Číslo úkolu je také základní znamení v logu, ať byl již úkol proveden úspěšně

²¹ Předpokládá se práce uživatele v omezeném účtu. S administrátorskými právy sice lze pomocí správce úloh smazat proces „python-service.exe“, čímž je služba násilně zastavena. To však běžný uživatel udělat nemůže.

Uživatel by neměl mít pro všední práci administrátorská práva už z principu – především kvůli dostatečné antivirové ochraně a kvůli odolnosti systému proti chybám lidského faktoru. Neznamená to nutně restriktivní omezení uživatelů – pokud to je nutné, dočasné přihlášení uživatele s vyššími právy situaci řeší dostatečně.

²² Ještě před úlohou s číslem 1000 předpokládám vyčištění historie, na to je ale program také připraven.

nebo nikoliv. Samozřejmě hlavní program musí s číslováním úkolů také umět pracovat a čísla správně generovat. Nyní může být úloha v řídicím souborů zadaná třeba hodinu, ale provede se na každém klientovi jen jednou.

3 KONEČNÁ PODOBA

Během programování jsem záhy zjistil, že vše značně usnadní, pokud budu dopředu znát jméno vytvořené aplikace. Chtěl jsem, aby jméno odráželo určení aplikace, zároveň aby bylo pokud možno krátké a dalo se dobře použít v programových cestách.

Celý systém zprávy jsem tedy pojmenoval **bee4lan**. Jméno si můžete přeložit jako „být pro síť“ nebo jako „včela pro síť“. Na motivu včelího úlu bych rád, až přijde čas, založil logo programu a grafický návrh pro webové ovládací rozhraní.



Obrázek 7 – Návrh možného loga programu.

3.1 Naprogramované funkce

Podařilo se mi dokončit a odladit tyto funkce:

- vypsat ovládaná klientská PC spolu s jejich základními informacemi – ovládaná PC vypíše hlavní aplikace zadáním parametru `-l`, základní informace o klientských počítačích jsou uvnitř adresáře *clients*
- klientská PC restartovat, vypnout, odhlásit aktuálního uživatele, poslat zprávu – formou vhodného skriptu, který je součástí základní sady
- spustit jakoukoliv úlohu na klientském počítači s nejvyššími oprávněními – splněno spouštěním samotné služby s právy účtu System
- shromažďovat přehledně do logu informace a provedených servisních úlohách včetně informace, zda skončily úspěchem či neúspěchem – splněno logováním na server i lokálně

- třídění klientských stanic do uživatelských skupin – vyřešeno přímou editací textových souborů v adresáři *groups*

Co se mi naopak nepodařilo, ať už z časových důvodů, nebo proto, že to nakonec v této fázi nebylo důležité:

- oddálit spuštění zadaného skriptu na klientském PC na vhodnou dobu – tento úkol je poněkud složitější, vyžaduje časování, zápisy do cronu apod., nechávám na neurčito
- pokud daná úloha na klientském PC selže, upozornit admina – jediné co máme k dispozici jsou logy, upozornění na neúspěšnou úlohu vyřešeno změnou názvu log souboru
- sada nejběžnějších úloh k dispozici přímo z hlavního programu – k dispozici je zatím jen ve formě skriptu, samotná aplikace takové funkce zatím nemá přímo integrované.

3.1.1 Detekce počítačů v síti a v systému správy

Bylo by pěkné, aby se aplikace uměla rozhlédnout po celé místní síti a vypsat nebo zeleně označit zařízení, která jsou právě online, naopak červeně označit zařízení, která jsou momentálně offline, a třeba šedivě zařízení, která nejsou do systému správy vůbec zapojena, ale aplikace je na síti našla.

Obávám se, že jen pouhé zjištění dostupných zařízení na síti LAN, pokud toto má být spolehlivé a rychlé, není vlastně vůbec jednoduchý úkol. Na druhou stranu – začlenění počítače do systému správy budeme vždy provádět lokálně instalací pythonu a bee4lan služby. Požadavek automatického zjištění je tedy pěkný, ale funkčně jej dále nijak nevyužijeme.

Pro spolehlivější práci s klientskými počítači se bude hodit znát nejen jejich IP adresu a hostname, ale také třeba fyzickou adresu a další jejich údaje. Po testech jsem nakonec vytvořil skript *hostinfo.py*, který tyto informace nechá vypsat do textového souboru. Ten se následně uloží přímo na server. Tím tedy činnost nově nainstalované služby musí začít. Uložením se prověří spojení na server spolu s možností zápisu, zároveň na serveru získáme přehledně a jednoduše vše, co pro další správu tohoto klienta potřebujeme. Přitom platí, že základní poznávací znamení stanice je hostname.

Samozřejmě, může se změnit IP adresa, fyzická adresa nebo jiné údaje klientské stanice. V takovém případě je na administrátorovi, aby úlohu *hostinfo.py* opět zadal. Informace pro server tak budou přepsány novými.²³

Program je na těchto informačních souborech postaven – při spuštění si je vždy znovu načte a teprve s těmito informacemi pracuje dál. Smazáním infosouboru tedy docílíme jednoduché vyřazení počítače ze správy, naopak ručním uložením infosouboru docílíme zařazení fiktivního počítače do naší správy (lze s výhodou využít při testování). Opět zde vidíme sílu textových souborů a jejich naprostou transparentnost.

3.1.2 Historie a logování

Určitě potřebujeme, aby služba byla schopná zaznamenat, když dojde k jejímu spuštění, zastavení, restartu, provedení úlohy apod. Šikovně postavené logování by mělo co možná nepřesněji definovat problém, například při výpadku sítě nebo neúspěšnosti prováděné úlohy. Logování očekáváme zejména od klientské služby²⁴. V našem případě se bude hodit uchovávat číslo úlohy, pro jaké klienty byla úloha určena, kdy byla zadána, kdy ukončena a jaké bylo její znění.

Služba by měla určitě logovat centrálně na server, protože tyto soubory mám nejvíce pod kontrolou. Toto logování je prozatím hromadné, tzn. v jednom souboru se mísí záznamy od všech klientů. Při menším počtu stanic ve fázi testování je to určitě přehlednější. Pokud to v praxi na rozlehlejších sítích bude vadit, lze to poupravit v další verzi programu.

Může nastat výpadek síťového spojení. Než se ke klientské stanici dostaneme, dávno už nebudeme vědět, co se vlastně dělo se službou. Objevuje se tedy také potřeba logovat lokálně (samozřejmě v adresáři programu). S tím, že každý druh logování má svůj typický účel, do jisté míry nenahraditelný.

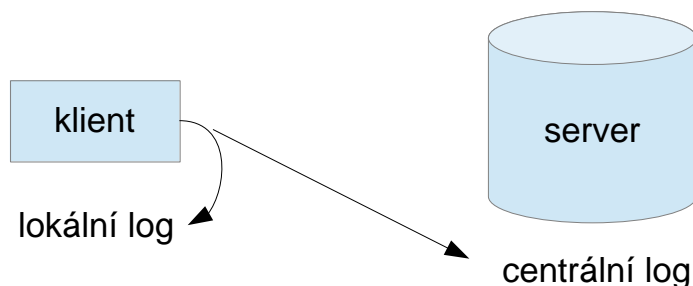
Každá dokončená úloha vytvoří svůj vlastní textový soubor s informací o průběhu. V adresáři *history* se tedy hromadí soubory – ale na první pohled je vidět, jaká úloha selhala, protože v názvu má přidané „_FAIL“. Otevřením souboru zjistíme znění

²³ Možná se časem ukáže jako vhodné refresh infosouborů vynucovat pravidelně. To však není žádný problém.

²⁴ Logy jsou v našem případě jediným prostředkem, kterým může služba dát najevo, co a jak probíhá.

prováděné úlohy jak na klientovi, tak na serveru. Čas zápisu není třeba logovat, protože tím už je čas uložení souboru.²⁵

Program načítá při svém spuštění hlavičky uložených souborů v historii. Tím je řešeno, aby program i po svém vypnutí nebo restartu stále správně věděl, jaké číslo úlohy bylo poslední. Totéž se týká služby na klientech.



Obrázek 8 - Schéma logování klientské služby.

Problémem může být situace, kdy se administrátor rozhodne čistit logy a provede vyčištění pouze na serveru. Potom se žádná úloha nebude chtít provést, protože její číslo bude vždy nižší než čísla provedených úloh ve složce historie u klienta. Na to však lze myslet. Služba se připomene také dostatečně návodnou chybovou hláškou ve svém logu.

3.1.3 Skupiny podle operačního systému

Program při každém svém spuštění načte informace o klientech ze souborů z adresáře *hosts*, a ihned je roztřídí do skupin podle operačního systému. Tyto skupiny lze potom využít při zadávání úkolů – například chceme provést instalaci programu Notepad++ na všech počítačích s Win7. Provedeme to takto:

```
core.py -x install_note.bat -t win7
```

Tato funkce by měla usnadnit tvorbu skriptů. V nich dále není nutné ošetřovat jednotlivé verze cílového systému. Odladěný skript na Windows 7 tak lze rovnou

²⁵ Samozřejmě to nemusí být ideální. Čas z metadat se s kopírováním souborů může přepsat současným, pro jeho zobrazení musíme použít Total Commander nebo zapnout zobrazování podrobností v průzkumníku atd. Čas může také být na různých stanicích nastaven různě. Na druhou stranu – zapisovat čas logování do právě vytvářeného souboru mi přijde jaksi zbytečné.

použít. Prove se pouze na těch počítačích, které spadají pod stejnou verzi operačního systému.²⁶

Podporované operační systémy klientů jsou v této verzi jen Windows XP, 7 a 10. S ostatními neumí kód programu zacházet. Tato možnost by měla přibýt teprve v budoucnu.

Protože tyto skupiny nejsou udělané pro výpis, uvádím dále surový výpis klientských počítačů se zobrazením jejich čísel – právě tato čísla můžeme použít pro zadání jednotlivého cílového počítače bez ohledu na jakékoliv skupiny.

```
YOUR HOSTS:
Number - name - groups:
-----
0 - DESKTOP-FJICMVS - win10, g1
1 - FLINT-XP1       - winXP, kanc1
2 - VIRTUAL         - win7, all
3 - WIN7_NEW        - win7, kanc1
-----
Number of hosts currently: 4
```

3.1.4 Uživatelské skupiny

Lokální síť je zpravidla charakteristická také jistým členěním geografickým. V praxi není výjimkou, když v jedné učebně máme sice stejné počítače, ale rozdílné s jinou učebnou, která byla zařízena až později. Především při správcovských operacích souvisejících s hardwarem (např. reinstalace ovladačů) můžeme ocenit hromadnou práci s klienty v jedné místnosti. Nebo ještě lépe, administrátor má počítače z hlediska hardwaru rozmístěné různě, ale rád by je spravoval po skupinách podle hardwaru. To vše umožňují skupiny definované uživatelem.

Jejich nadefinování je prosté – v adresáři *groups* založíme například soubor *ucebna2.txt*, do něj vložíme hostname všech počítačů, které bychom v této skupině chtěli mít, jedno jméno na řádku. Skupin můžeme založit, kolik potřebujeme. Po spuštění aplikace můžeme pro kontrolu vyvolat jejich zobrazení příkazem:

²⁶ Samozřejmě, nejlepším řešením je mít již odladěný skript a připravený na všechny verze systému, třeba alespoň všechny verze Windows. Umím si však představit, že pokud budeme muset použít příliš rozdílný kód na různé verze OS, bude snazší vytvořit samostatné skripty a použít tuto funkci.

```
core.py -g
```

Editovat můžeme uživatelské skupiny kdykoliv. Změny se projeví, jakmile program znovu spustíme. Smazání obsahu složky *groups* znamená ztrátu uživatelských skupin a rozpuštění jejich členů do hlavní skupiny *all*. Do skupiny *all* spadá každý klient, který nemá definovanou svou vlastní. V této verzi programu je možné přiřadit klienta pouze do jedné skupiny. Prolínání skupin je zamýšleno až pro budoucí verzi.

Ve výpisu vidíme 3 klientské počítače rozdělené do pokusných skupin *g1*, *g2* a *kanc1*. Klienti nezařazení do uživatelských skupin ve výpisu nejsou, jejich celkový počet tedy může být vyšší:

```
USER DEFINED GROUPS:
----- g1 (1 hosts) -----
DESKTOP-FJICMVS

----- g2 (1 hosts) -----
----- kanc1 (2 hosts) -----
FLINT-XP1
WIN7_NEW
```

3.1.5 Základní sada skriptů

Nejlépe bude ukázat přímo výstup z programu po vypsání skriptů po zadání parametru *-s*. Skripty jsou načteny při startu programu. Ve výpisu je jejich řazení provedeno abecedně, v každé skupině zvlášť. V každé skupině skriptů (podle typu souboru) získáváme také informaci o jejich počtu.

Složitější názvy skriptu si nemusíme pamatovat celé – nápovědný tabulátor nám je při psaní příkazu doplní sám.

```
YOUR SCRIPTS:
----- .bat (8 scripts) -----
UTAXwin10x64_setup.bat
down.bat
firefox_konec.bat
killbee.bat
logout.bat
message.bat
reboot.bat
up.bat
----- .cmd (0 scripts) -----
```

```
----- .py (4 scripts) -----  
calc_notepad.py  
disk_usage.py  
hostinfo.py  
user.py
```

3.2 Přístup k síti, konfigurace samby

Sdílený diskový prostor na souborovém serveru je nutnou podmínkou pro fungování aplikace.

Služba na všech klientech potřebuje ke své činnosti právo zápisu do společného logovacího souboru *clients_MAIN.log* v adresáři *logs* na serveru. Adresář je umístěn v kořenovém adresáři programu bee4lan a služba na všech klientech se do něj pokouší zapisovat. Kdyby to nešlo, služba nebude moci fungovat a dojde k jejímu samovolnému zastavení.

Pokud chceme, aby centrální logovací adresář byl umístěn úplně jinde (těm důvodům rozumím), musíme cestu k němu a připojení patřičného sdílení nastavit v kódu služby před tím, než ji budeme distribuovat na jednotlivé klienty k instalaci.

K řídicím souborům *control.cfg* a *todo.bat* v kořenovém adresáři programu službě bohatě dostačuje přístup read-only. Skripty v kořenovém adresáři bee4lan na serveru bude služba chtít nejen číst, ale i spouštět.²⁷

Původním záměrem bylo vyhnout se připojování sdílených disků. V zásadě je možné dosáhnout na sdílené soubory samby bez přímého přihlašování – pokud však soubory jádra aplikace nechceme vystavit jako public, pak musíme nějaké přihlášení provést. Jak to provede služba pod účtem System? Možností přihlášení je v tomto případě méně. Ukazuje se tedy, že připojit disková sdílení přímo službou, je prozatím nejlepší.

Samostatné sdílení vyčleněné jen pro řízení sítě a pro účely bee4lan může být jen dobrý nápad. Cesty k tomuto sdílení je nutné zeditovat ve skriptu služby ještě před instalací na jednotlivé klienty!²⁸

²⁷ V případě potřeby může skript obsahovat instrukci ke zkopírování sama sebe do dočasné složky na lokál, kde nebude problém ani se zápisem. Taková situace ale spíše nenastane. Služba samotná toto nedělá.

3.3 Podporované platformy

Podporované operační systémy klientů jsem již rozepisoval v kapitole „Různé verze operačního systému“. Plánované tedy byly Windows XP, Windows 7, Windows 8 a Windows 10. Zamýšlená práce tak měla pokrývat nejčastěji používané uživatelské operační systémy ve firmách a organizacích.

Na konci práce vypouštím Windows 8, protože zbývající trojice podle mého názoru dostatečně pokrývají nejčastější uživatelské systémy v organizacích. Windows 8 je prostředí zamýšlené pro dotykové obrazovky a tablety. Na desktopech bych se mu však raději vyhnul.

Linuxové operační systémy na klientech budou potřebovat vlastní službu a plně multiplatformní skripty. Windowsové dávky bude tedy nutné přeprogramovat do jazyka python. Očekávám přitom jisté nesnáze vyžadující velké množství času a pozornosti. Zařazení těchto zařízení prozatím není podporováno.

3.4 Uživatelské rozhraní

V konečné fázi program pracuje ve formě příkazu v terminálu, řízeného pomocí parametrů. Spustí se tedy jednorázově, provede přípravu nového úkolu nebo jakoukoliv jinou funkci a můžeme program znovu použít jinak nebo nepoužít vůbec. Je jedno, zda si program připravíme na serveru s linuxem nebo na jednom ze stanic určených pro práci administrátora, u sebe doma, či na síťovém úložišti, jehož systém nás nezajímá. Jediné, co vlastně potřebujeme, je počítač, python a funkční síť.

Finální aplikace umí jen málo věcí, ale i tak by měla být značně přínosná pro zrychlení práce. Umí vypsát soubory zařazené do správy, vypsát skripty, které máme k dispozici, umí rozčlenit počítače do skupin podle operačního systému, vyzná se ve skupinách definovaných uživatelem a hlavně, umí za administrátora správně připravit kontrolní soubor a soubor s hlavním skriptem, který klientské služby provádí. Zadávat lze klienty číslem ze seznamu (to je určitě rychlejší a přesnější než vypisovat hostname) a toto určení kombinovat se jménem uživatelské skupiny, například:

```
core.py -x ff_konec.bat -t ucebna2, win7,5,4
```

²⁸ Do další verze plánuji .ini soubor, shrnující jednotlivá nastavení. Konfigurace šitá pro potřeby každého administrátora by tak měla být mnohem snazší a transparentnější.

Aplikace si vše přebere a seznam hostname vypíše do kontrolního souboru bez duplikací a seřezané podle abecedy.

Pro použití skriptu by se v příkazové řádce hodilo také dokončování pomocí tabulátoru, neboť to zrychlí a zpřesní zadaný příkaz. Pro všechny připravené skripty se krásně nabízí použít programovou složku *scripts*. Tady však tabulátor nefunguje, protože skripty jsou na jiné cestě, než na jaké je uložen hlavní program. Skripty tedy musí být umístěné v kořeni programu, což není příliš přehledné. Ale je to naprosto funkční a tabulátor lze s výhodou používat.

3.4.1 Ovládání služby

Pro ovládání služby jsou předpřipravené skripty *up.bat*, *down.bat*, *bee4lan.py*. Pythonovský skript je služba, kterou chceme do systému nainstalovat. Dávka *up.bat* provede jeho instalaci i následné spuštění. Dávka *down.bat* naopak službu zastaví a odinstaluje. Obsah instalačních miniskriptů můžeme rozepsat takto:

```
python bee4lan.py install
python bee4lan.py start
python bee4lan.py stop
python bee4lan.py remove
```

Díky návaznosti pythonovského kódu na API rozhraní Windows lze nainstalovanou službu ovládat také standardními příkazy pro správu služeb:

```
sc start bee4lan
sc stop bee4lan
sc delete bee4lan
sc query bee4lan
```

Poslední z vyjmenovaných příkazů ukáže informace a stav dané služby. Pokud však máme přístup ke klientskému počítači (v okamžiku instalace to předpokládám), je nejlepší vyvolat správce služeb, v něm službu spustit, zastavit nebo restartovat myší:

```
services.msc
```

Hodit se také bude správce úloh a v případě řešení problémů protokoly aplikací ve Správci událostí:

```
taskmgr.exe
eventvwr.exe
```

Můžeme také chtít zobrazit lokální log služby. Ten je řešen jedním souborem s názvem *hostname_svc.log* v klientském adresáři služby. Pokud se chceme podívat na historii provedených úkolů, stačí si projít adresář

```
C:\Program Files\bee4lan\history
```

Tuto složku je samozřejmě žádoucí nastavit pro uživatele jako read only nebo nejlépe jako skrytou.

3.4.2 Ovládání aplikace

Ovládání aplikace předpokládá práci v příkazové řádce windows nebo v terminálu linux, tím pádem také minimální zkušenosti s tímto prostředím. Koneckonců, aplikace je určená administrátorům.

Spuštění aplikace bez jakýchkoliv parametrů, případně s parametrem *-h*, vyvolá nápovědu:

```
core.py
```

```
core.py -h
```

Zobrazení dostupných skriptů provedeme takto:

```
core.py -s
```

Zobrazení hostů připojených do správy bee4lan získáme takto:

```
core.py -l
```

Zobrazení uživatelských skupin s výpisem jejich členů docílíme takto:

```
core.py -g
```

A konečně provedení skriptu na klientech (lze kombinovat se skupinami) přikážeme takto:

```
core.py -x ff_konec.bat -t ucebna2,win7,5,4
```

```
core.py -x hostinfo.py -t all
```

Příkaz k provedení skriptu si před provedením ještě zobrazí vygenerovaný úkol a vyžádá si potvrzení. Upozorňuji zejména na syntaxi posledního příkazu. Není možné zadat úkol a neupřesnit, na kterých klientech se má provést. Stejně jako není možné zadat, na jakých klientech se má provést a současně nezadat, o jaký úkol půjde.

Ale nyní už výpis, zobrazený po zadání posledního příkazu – než program přepíše kontrolní soubory, žádá potvrzení od administrátora:

YOUR TASK (111):

```
python.exe R:\bee4lan\hostinfo.py
```

TO HOSTS:

```
DESKTOP-FJICMVS  
FLINT-XP1  
VIRTUAL  
WIN7_NEW
```

```
Control.cfg and todo.bat file will be overwritten.  
Do you want to continue [y/n]?:
```

3.5 Systém souborů aplikace

Systém souborů je to hlavní, co bee4lan má. Už ovládací aplikace je v jistém smyslu doplněk, pouze usnadňující práci. Bee4lan má po instalaci tuto strukturu:

Adresář *groups* – obsahuje definice jednotlivých uživatelských skupin. Obsahem jsou textové soubory a jsou určeny k přímé editaci.

Adresář *history* – obsahuje seznam minulých úloh. Obsahem jsou opět textové soubory. Názvem je číslo úlohy, uvnitř najdeme klienty, kterým byla určena i znění úkolu.

Adresář *hosts* – obsahuje seznam klientů zapojených do hromadné správy. Jde opět o textové soubory. Ty však nejsou určeny k editaci, jen pro čtení. Instalace nového klienta do systému bee4lan není úspěšná, dokud se v tomto adresáři neobjeví jeho infosoubor.

Adresář *logs* – obsahuje především hlavním log soubor společný pro všechny klienty. Jde o textový soubor a je určen k přímému přístupu, ne však editaci. Ale je možné jej kdykoliv smazat, vytvoří se nový.

Adresář *install* – obsahuje soubory vhodné k instalaci bee4lan služby na nově přidávaném počítači do systému hromadné správy.

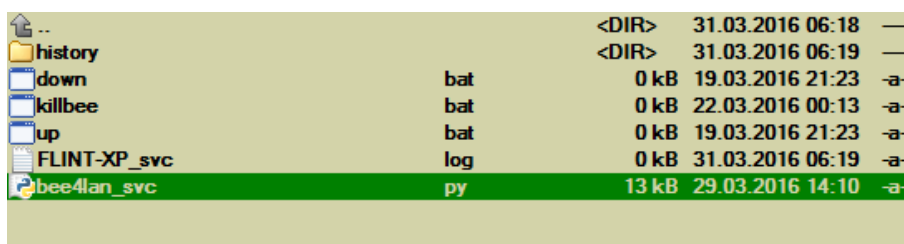
Kořenový adresář *bee4lan* – obsahuje především hlavní aplikaci, kontrolní soubor *control.cfg* a *todo.bat*. Odladěné a používané skripty jsou bohužel z výše uvedených důvodů uloženy také přímo v kořenovém adresáři programu.

..	<DIR>	29.03.2016 04:40	—
groups	<DIR>	10.03.2016 14:28	—
history	<DIR>	29.03.2016 14:01	—
hosts	<DIR>	08.03.2016 01:44	—
install	<DIR>	29.03.2016 04:54	—
logs	<DIR>	29.03.2016 01:42	—
scripts_concepts	<DIR>	22.03.2016 02:24	—
down	bat	0 kB	19.03.2016 21:23 -a-
firefox_konec	bat	0 kB	08.03.2016 23:12 -a-
killbee	bat	0 kB	22.03.2016 00:13 -a-
logout	bat	0 kB	29.03.2016 14:03 -a-
message	bat	0 kB	22.03.2016 00:49 -a-
reboot	bat	0 kB	17.03.2016 04:23 -a-
todo	bat	0 kB	29.03.2016 14:01 -a-
up	bat	0 kB	19.03.2016 21:23 -a-
UTAXwin10x64_setup	bat	0 kB	22.03.2016 12:54 -a-
control	cfg	0 kB	29.03.2016 14:01 -a-
calc_notepad	py	0 kB	21.03.2016 10:04 -a-
core	py	15 kB	27.03.2016 15:20 -a-
disk_usage	py	0 kB	24.02.2016 15:42 -a-
hostinfo	py	1 kB	07.03.2016 23:22 -a-
user	py	0 kB	22.03.2016 00:45 -a-

Obrázek 9 - Ukázka souborového systému programu bee4lan v podání Total Commanderu. Jde o serverou část. Zeleně označený je řídicí soubor.

3.6 Instalace

Bee4lan – instalace služby. Pro program jsem vytvořil instalátor pro Windows prostředí. Podporované systémy na straně klientů jsou zatím pouze operační systémy Windows XP, Windows 7, Windows 10. Tedy, ty nejběžněji spravované. Jeho spuštěním bude nainstalován Python verze 2.7.11 (pokud již není), upravena systémová path pro spouštění python skriptů, a bude také provedena registrace pythonservice.exe. Dále musí být nainstalován skript *bee4lan.py* mezi služby Windows prováděné s právy účtu System, a této službě musí být nastavena vlastnost „spouštět automaticky“. Bez této úpravy služba po restartu počítače sama nenastartuje. Instalace služby zahrnuje také vytvoření jednoduché souborové struktury v cestě bee4lan.



--	<DIR>	31.03.2016 06:18	—
history	<DIR>	31.03.2016 06:19	—
down	bat	0 kB	19.03.2016 21:23 -a
killbee	bat	0 kB	22.03.2016 00:13 -a
up	bat	0 kB	19.03.2016 21:23 -a
FLINT-XP_svc	log	0 kB	31.03.2016 06:19 -a
bee4lan_svc	py	13 kB	29.03.2016 14:10 -a

Obrázek 10 - Ukázka programového systému souboru u klienta při testování. Opět v podání Total Commanderu.

Do té je zkopírován hlavní skript služby²⁹ a je vytvořen podadresář *history* pro provedené úkoly.

Bee4lan – instalace ovládacího programu. Instalátor jsem vytvořil pro prostředí Windows i Linux. Jeho spuštěním bude nainstalován Python verze 2.7.11 (pokud již není nainstalován) a upravena systémová proměnná path pro spouštění python skriptů. Bude též vytvořena souborová struktura bee4lan na vybraném místě síťového úložiště. Adresářová struktura musí být pro funkčnost programu zachována.³⁰

3.7 Autorství a licence

V současném světě byznysu a konzumu bych tuto aplikaci rád věnoval komunitě open-source s tím účelem, aby sloužila dobrým lidem jakéhokoliv zaměření, ať už v soukromí, organizacích či menších firmách.

Od licence GNU-GPL si také slibuji podporu komunity a její další vývoj, který by případně nemusel záviset už jen na mě samotném.

Tak tedy: „Žij a buď užitečný!“

²⁹ Teoreticky by bylo možné spouštět službu na všech klientech tak, aby její skript zůstal uložený jenom jednou, a to na serveru. Výhodu bychom měli při ladění služby, avšak žádnou jinou. Především, při ztrátě síťového spojení by služba nebyla schopná provést vůbec nic, tedy ani zápis do lokálního logu.

³⁰ Samozřejmě licence programu dovoluje upravit si jej ke svému obrazu. Bez adekvátního zásahu v kódu by ale program s upravenou strukturou složek nefungoval.

ZÁVĚR

Jak jsem zprvu očekával, zadání bylo velmi komplexní a náročné svou šíří. Prošel jsem teoretickým plánováním a sbíráním zkušeností, prvními pokusy se vzdálenou správou, programováním, odlaďováním, popisováním. Jsem velmi rád, že jsem došel až sem.

Splnil jsem zadání, ale svůj výsledek zdaleka nepovažuji za konečný. Ke skutečně praktickému používání schází dokončit některé funkce, které jsem si předsevzal a také je považuji za dobré, na které však už nebyl prostor. Program jako každý jiný samozřejmě potřebuje projít náročnou fází dozrávání, kdy se projeví nejen silné stránky, ale také slabá místa a kusy kódu, požadující přepracování do lepší a efektivnější podoby.

Aplikaci bych rád obohatil o plánované grafické rozhraní, realizované formou webové stránky. To však bude možné tvořit, až bude jádro programu usazené, dlouhodobě funkční a celkově vyhovující. Pevně věřím, že mi k tomu pomohou další lidé svými komentáři a nápady.

Nakonec bych rád podotkl, že jsem velmi rád, že mi škola dala příležitost zkusit vytvořit něco kreativního. Přestože překážky na cestě k cíli byly pro mě místy velmi tvrdé a houževnaté, práce mě po celou dobu velmi těšila.

RESUMÉ

Program *bee4lan* je zamýšlen jako správcovský pomocník, umožňující hromadnou správu klientských počítačů s operačním systémem Windows pomocí vlastních administrátorových skriptů. Pro své fungování je pouze síťové úložiště, nastavenou sambu a jeden počítač, na kterém správce pracuje – ideálně linuxový server. Ten ale není pro aplikaci nutností.

Aplikace má serverovou a klientskou část. Díky Pythonu jsou klientská služba i hlavní program náročné na hardware opravdu minimálně.

Cílem vývoje jsou administrátoři, na něž jsou kladené klasické požadavky: spravovat síť co nejlépe, současně s minimálními náklady, tím co je po ruce. Když odmyslíme komplexní řešení za drahé licence a příliš robustní, mnoho možností nezbyvá. Snažil jsem se tedy jednu další přinést.

Řešení hromadné správy tohoto druhu vyžaduje od administrátora jistou dávku nadšení a zkušenosti se skriptovacími jazyky. Na oplátku však může sloužit takovou úrovní variability a transparentnosti, jako málokterý jiný, obzvlášť komerční, program.

SUMMARY

Bee4lan application is intended as a tool for network administrators, who manage LANs with client computers running Windows.

An application needs only Python, samba network share and a computer. This computer could be a server running Linux (best option), a client windows machine, or admin's own computer at home.

This way of network administration provides many possibilities for variability. It can greatly overcome security of Windows operating system and use it for benefit of simplicity and reliability of whole solution. Due to Python language, application and service parts require really very small amount of hardware and software resources.

The program is intended for administrators, who stand for really not easy task: to manage all network desktop stations as good as possible, however, for minimum costs. Therefore, this application is aimed for schools, nonprofit organizations and other smaller companies.

Solution of network administration like this requires an administrator basically skilled in programming with scripting languages. As advantage, the application can offer so high level of variability and transparency, like any other commercial solution ever could.

SEZNAM POUŽITÝCH ZDROJŮ

1. HARMS, Daryl D a Kenneth MCDONALD. Začínáme programovat v jazyce Python. 2., opr. vyd. Brno: Computer Press, 2008, xvi, 456 s. ISBN 978-80-251-2161-0.
2. *Ponořme se do Pythonu 3* [online]. Mark Pilgrim, 2011 [cit. 2015-12-12]. Dostupné z: <http://diveintopython3.py.cz>
3. *Učíme se programovat v jazyce Python 3: How to think like a computer scientist - Learning with Python* [online]. by Jeffrey Elkner, Allen B. Downey and Chris Meyers, (M.K. Bradshaw, Peter Wentworth), přeložil Jaroslav Kubias, 2014 [cit. 2015-12-12]. Dostupné z: <http://howto.py.cz/index.htm>
4. Debian. *Debian -- Univerzální operační systém* [online]. Software in the Public Interest, Inc., 1997, 2015-06-01 [cit. 2015-12-12]. Dostupné z: <https://www.debian.org/index.cs.html>
5. Microsoft Windows XP. *Windows XP Professional Product Documentation* [online]. Microsoft Corporation, 2015 [cit. 2015-12-12]. Dostupné z: <http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/ntcmds.mspx?mfr=true>
6. *Windows Sysinternals: Windows Sysinternals: Documentation, downloads and additional resources* [online]. Microsoft, 2015 [cit. 2015-12-12]. Dostupné z: <https://technet.microsoft.com/en-us/sysinternals>
7. *Diplomová práce - Filip Vaculík 2007* [online]. Filip Vaculík [cit. 2015-12-12]. Dostupné z: <http://mintaka.eu/fav/kiv/dip2007/>
8. *Python 2.7.11 documentation* [online]. [cit. 2016-03-30]. Dostupné z: <https://docs.python.org/2.7/>
9. *BatchPatch: The Ultimate Windows Update Tool* [online]. [cit. 2016-03-30]. Dostupné z: <https://batchpatch.com/>
10. Windows Server: Sc create. *Microsoft Windows* [online]. [cit. 2016-03-30]. Dostupné z: <https://technet.microsoft.com/cs-cz/library/>

PŘEHLED ZKRATEK A POJMŮ

API – Application Programming Interface, rozhraní programování aplikací jako součást operačního systému, zde systému Windows

aplikace – aplikační software neboli počítačový program, který používá uživatel

Cron – softwarový démon, který v Unixu slouží jako plánovač úloh

dávka – skript příkazového řádku Windows

diskové sdílení – network share, prostor na síťovém úložišti, který můžeme připojit jako diskový oddíl, ve windows jako například Z:\

distribuce – úplný operační systém, snadno použitelný uživatelem, odvozený od linuxového jádra

fyzická adresa – MAC adresa, identifikátor síťové karty, zapisovaná nejčastěji ve tvaru šestice dvojíých hexadecimálních čísel, například 01:23:45:67:89:ab

GNU – General Public License

host – v mém programu označení pro počítač připojený do místní sítě LAN, sloužící některému z uživatelů

hostname – jméno počítače v síti, jedinečný identifikátor, například „Jarda-PC3“

IDE – vývojové prostředí pro konkrétní programovací jazyk

kernel – jádro operačního systému

klient – stejně jako host, označení pro počítač v síti LAN sloužící některému z uživatelů

kořenový adresář – v případě pevného disku základní adresář, tj. například C:\, v případě programu jeho základní adresář, kde má uložené všechny soubory, zde například C:\Program Files\bee4lan\

LAN – Local Area Network, místní síť, počítačová síť menšího geografického rozsahu, například tvořená jednou školou, jednou menší firmou, jednou domácností apod.

logování – jakmile program vykoná určitou akci, zapíše výsledek nebo jednoduchou informaci o tom, že akce proběhla, zpravidla s časovým údajem

malware – Malicious Software, škodlivé programy, které mají tendenci proniknout do našeho počítače a jakýmkoliv způsobem u nás parazitovat

metadata – data poskytující informaci o datech uložených v souboru, nejčastěji nás zajímá název souboru, typ, čas vytvoření, čas změny, čas posledního otevření apod.

multiplatformní – mající schopnost fungovat na více platformách či operačních systémech (zde především Windows, Linux, Apple)

NAS - Network Attached Storage, síťové zařízení sloužící jako datové úložiště

open-source – zkrácenina Open-Source Software, tedy počítačový program s otevřeným zdrojovým kódem, tedy přístupným komukoli

OS – operační systém počítače

privilegovaný účet – uživatelský účet se zvýšenými oprávněními, v Linuxu root, ve Windows Administrator nebo System

read-only – přístupný pouze ke čtení, ne k zápisu

refresh – oživení, znovunačtení aktuálních dat

release – vydání programu v nové, ucelené verzi

Samba – svobodná implementace síťového protokolu SMB, šířená pod licencí GNU-GPL, používaná zejména pro vzdálený přístup k souborům pro systémy Windows, uloženým na linuxovém souborovém serveru,

server – počítač poskytující nějaké služby, typicky souborový server, webový server, poštovní server apod.

skript – spustitelný program obsahující instrukce, pomocí kterých počítač provede určitou činnost

služba – ve Windows počítačový program, který se provádí na pozadí, obdobně jako linuxový démon, jádrem služby je nekonečná smyčka a návaznost na Windows API, tím se její kód liší od běžných programů

snapshot – snímek stavu souborového systému, v případě virtuálního počítače tedy stavu celého virtualizovaného stroje

SSH – Secure Shell, síťový protokol pro zabezpečení síťové komunikace a vzdálené provádění příkazů

stanice – uživatelský počítač v síti

System účet – účet Windows mající vyšší oprávnění než jakýkoli uživatelský účet

systémová proměnná path – soubor adresářových cest, ve kterých operační systém automaticky hledá daný soubor nejdříve, spustitelný program zde uložený můžeme spustit z jakéhokoliv umístění bez zadání cesty

PŘÍLOHY

Příloha 1:

Ukázka výpisu nápovědy realizované prostřednictvím knihovny argparse.

```
R:\bee4lan>core -h
usage: core.py [-h] [-x EXECUTE] [-t TARGETS] [-d DELAY] [-l] [-s] [-g]

Bee4lan main controlling program for windows hosts at LAN. You can do it via
your own scripts, which you just prepared to deploy on hosts and placed into
scripts folder. Be well, John Spartan!

optional arguments:
  -h, --help            show this help message and exit
  -x EXECUTE, --execute EXECUTE
                        which script you want to execute
  -t TARGETS, --targets TARGETS
                        Target hosts to launch your script. You can give
                        numbers as follows: 1,2,3,8,9,10,12. You can also give
                        names of your groups and you - of course - can mix
                        both. If you dont use this option, the script will be
                        launched on all your computers. CAUTION: If you use
                        this option, control.cfg and todo.bat files will be
                        overwritten with new generated data!
  -d DELAY, --delay DELAY
                        Delay before the script will be launched on your
                        computers specified in option -t.
  -l, --listhosts        Lists all available hosts which you can control. If you
                        want to edit them manually, go to the 'hosts' folder.
  -s, --showscripts      Show list of your available scripts in 'scripts'
                        folder.
  -g, --showusergroups   Show list of user defined groups. You can create and
                        edit groups in 'groups' folder. Each .txt file is your
                        group. It enables you to handle hosts your way.
```