Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra informatiky a výpočetní techniky

Bakalářská práce

Propagace a distribuce softwarových produktů

Plzeň, 2012 Jiří Beneš

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 8. května 2012 Jiří Beneš

Abstract

The objective of this bachelor's work is point to the promotion possibilities and software products distribution. In the part dedicated to promotion I try to focus on the matter how to create the web site more beneficial and competitive. There have been ranking and SEO methods discussed in detail. I mention everything with in detail explained practical examples. The part referred to distribution has been concentrated on the distribution packages forms. The distribution packages problems within the Linux and Microsoft Windows operational systems have been in detail discussed. At the first mentioned system I concentrate on DEB and RPM packages, which are both theoretically and practically assessed. At the second system I focus on the MSI packages. The vast tools range has been tested for both these systems and the scripts have been created in the end to ensure an easy creation of the above mentioned distribution packages.

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Kamilu Ježkovi za cenné rady a připomínky.

Obsah

1 Úvod 7
2 Internetová propagace a ranking 8
2.1 Propagace produktu formou reklamy 9
2.1.1 Propagace softwarových produktů v konkrétní společnosti 9
2.2 Algoritmus PageRank 10
2.2.1 Vzorec PageRanku 10
2.2.2 Funkce PageRanku 11
2.2.3 Software pro výpočet PageRanku 11
2.2.4 Regulace hodnot PageRanku 12
2.3 Životní cyklus rankingu webové stránky 14
3 Propagace za použití SEO metod 15
3.1 Optimální tvar URL adresy 16
3.1.1 Vytváření optimální URL adresy pomocí souboru .htaccess 17
3.1.2 Vytváření optimální URL adresy pomocí jazyku PHP 18
3.1.3 Vytváření optimální URL adresy pomocí systému WordPress 18
3.2 Změna URL adresy – přesměrování stránky 19
3.2.1 Přesměrování webové stránky pomocí souboru .htaccess 19
3.2.2 Přesměrování webové stránky pomocí jazyku PHP 20
3.2.3 Přesměrování statické webové stránky 20
3.3 Budování zpětných odkazů 21
3.3.1 Blokování nechtěných zpětných odkazů 21
3.3.2 Volba správného anchor textu 22
3.3.3 Vyhnutí se zápornému BadRanku 22
3.3.4 Výběr správného cíle odkazu 23
3.3.5 Kontrola pravosti PageRanku 23
3.4 Volba klíčových slov 24
3.4.1 Klíčová slova typu „long tail“ 25
3.5 Volba tagů 26
4 Distribuční balíčky pro OS Linux 28
4.1 Balíčkovací systém APT 29
4.1.1 Konstrukce DEB balíčku 29

4.1.2 Nástroje pro vytváření DEB balíčků ... 33
4.1.3 Ovládání programu Deb-creator ... 34
4.2 Balíčkovací systém RPM ... 36
4.2.1 Konstrukce RPM balíčku ... 36
4.2.2 Nástroje pro vytváření RPM balíčků ... 39
5 Distribuční balíčky pro OS MS Windows ... 42
5.1 Nástroje pro vytváření MSI balíčků ... 43
5.1.1 Exe to MSI Converter ... 43
5.1.2 Clickteam Install Creator ... 44
5.1.3 Bytessence Install Maker ... 45
5.1.4 War Setup ... 46
5.1.4 Windows Installer XML toolset (WiX) ... 47
5.1.5 MakeMsi ... 48
5.1.6 Shrnutí testovaných nástrojů ... 49
6 Realizace distribuční formy ... 50
6.1 Nástroj DPKG-DEB (ALIEN) – realizace skriptů pro vytváření DEB/RPM balíčků ... 51
6.1.1 Zadání úlohy ... 51
6.1.2 Analýza úlohy ... 51
6.1.3 Popis implementace ... 52
6.1.4 Uživatelská příručka ... 53
6.1.5 Závěr ... 56
6.2 Nástroj MakeMsi – realizace skriptů pro vytváření MSI balíčků ... 57
6.2.1 Zadání úlohy ... 57
6.2.2 Analýza úlohy ... 57
6.2.3 Popis implementace ... 59
6.2.4 Uživatelská příručka ... 61
6.2.5 Závěr ... 64
7 Celkový závěr ... 65

1 Úvod

Bakalářská práce zahrnuje dvě velké kapitoly – propagaci a distribuci softwarových produktů. Vzhledem k tomu jsem si dovolil rozdělit tyto kapitoly jak v celkovém obsahu bakalářské práce, tak i nyní, v úvodu.

Propagace softwarových produktů má za úkol seznámit uživatele s problematikou rankingu a tím spojených SEO metodách. V práci detailně popisuji, jak zvýšit návštěvnost webových stránek a tím tak zajistit kvalitní propagaci pro daný softwarový produkt. Lze se zde také dočíst, jak propagují softwarové produkty velké korporace. Kapitola o propagaci je spíše teoretická, ale obsahuje praktické příklady.

Distribuce softwarových produktů má pak uživatele seznámit s tím, jak zajistit uživatelsky přívětivou distribuci v operačních systémech Microsoft Windows a Linux. Velká pozornost je tedy věnována distribučním balíčkům, které jsou ve formě instalátorů a umožňují tak snadnou instalaci daného softwarového produktu. Tato kapitola o distribuci je rozdělena do dvou podkapitol a to právě podle použitého operačního systému (viz výše). U každé podkapitoly je pak uvedena teoretická část, zabývající se vlastnostmi jednotlivých balíčků a realizační, kde je uveden popis použitých nástrojů a vytvořených skriptů.

2 Internetová propagace a ranking

Propagace by měla cílovému uživateli přinést určitý zisk. Já se v této práci budu věnovat spíše internetové propagaci, čili propagaci, která by měla uživateli zajistit více návštěvníků jeho webových stránek. Předpokládejme tedy, že na webových stránkách je představován nějaký produkt, který chceme zpeněžit. Pokud získáme více návštěvníků, kteří se o našem produktu dozvědí, bude pak i vyšší pravděpodobnost, že se tento produkt prodá. S navýšením počtu návštěvníků pak úzce souvisí ranking.

Obecně ranking je docela různorodý pojem. V našem případě se bude konkrétně jednat o jakési číselné ohodnocení, které je nezávislé na hledaném dotazu. Toto číselné ohodnocení je pak použito k řazení webových stránek a to tak, že čím vyšší číslo, tím vyšší pozice v internetovém vyhledávači. K tomu, aby bylo možno vypočítat číselné ohodnocení (rank) se pak používají různé algoritmy. Každý vyhledávač má svůj specifický algoritmus, například velmi známý Google používá PageRank, z českého prostředí je pak asi nejznámější Seznam, který používá S-rank.

Jelikož vyhledávač Google je velmi známý široké veřejnosti a jedná se i o nejslavnější vyhledávač, rozhodl jsem se, že se zaměřím právě na něj. Blíže se tedy podívám na algoritmus PageRank.

2.1 Propagace produktu formou reklamy

Jelikož je záměrem této práce hlavně propagace softwarových produktů, je zaměřena na webové stránky. Ovšem webové stránky a celkově elektronická forma propagace není vše. Pokud chceme propagovat software, je dobré mít i tištěnou formu. Může se tak v podstatě jednat o reklamu na náš produkt. Taková reklama se pak hodí, když chceme náš produkt prezentovat zákazníkovi, aby měl stručné informace o tom, co může případně koupit.

2.1.1 Propagace softwarových produktů v konkrétní společnosti

Pro správné propagování našeho softwarového produktu je třeba více faktorů. Jeden z těchto faktorů je také vhodná reklama. Pokud budeme mít „perfektní“ webové stránky, ale obsahově nebudeme mít dostatečně dobře vysvětlené informace o produktu, zřejmě se neprodá. Abych zjistil, jak funguje reálná propagace softwarových produktů, požádal jsem společnost Konica Minolta, zdali by mi mohla poskytnout materiály, které k propagaci svých produktů používají. Nadnárodní společnost Konica Minolta se ve zkratce zabývá vším, co se týká tisku či skenování. Vzhledem k profilu této společnosti si myslím, že informace, které níže uvedu, budou použitelné pro jakoukoliv propagaci.

Důležitá věc pro představení našeho produktu jsou takzvané reklamní brožury – ať už v elektronické podobě, či v tištěné. Tyto brožury by měly obsahovat stručný popis produktu. Je zde vhodné vynikající grafické zpracování – potencionální zákazník by měl získat dobrý dojem. Brožury by pak měly být k dispozici v následujících formách:

- elektronická brožura pro představení produktu – umístěná na našem webu
- tištěná (elektronická) brožura určená pro přímý prodej

Oba tyto typy brožur pro produkty lze vidět v elektronické podobě na přiloženém DVD. Jedná se brožury představující nástroje ABBYY FineReader a SafeQ. Příklad jednoho, ze správných webových řešení, jak propagovat softwarové produkty lze vidět níže:

http://www.konicaminolta.cz/business-solutions/products/applications.html

Tato podkapitola trochu odbočuje od jádra této práce, avšak myslím si, že je vhodné jí tu uvést. Je totiž důležité nesoustředit se pouze na jednu oblast – například ranking a zanedbat tak vlastní představení cílového produktu. Navíc elektronické brožury jsou lehce přenositelné a můžeme je tak snadno rozšířit (např. do e-mail schránek). Každá brožura obsahuje odkazy na webové stránky s produktem a to opět pomáhá ke zviditelnění našeho webu – což souvisí s rankingem. Takže jak píši výše, vše je provázané a je třeba se nesoustředit jen na určitou oblast.

2.2 Algoritmus PageRank

Algoritmus PageRank vypočte číslo, které si vyhledávač Google přiřazuje ke každé webové stránce. Vyjadřuje něco jako věrohodnost, či důležitost. Jednoduše se dá říct, že čím vyšší číslo, tím vyšší pozice v internetovém vyhledávači (viz ranking). Algoritmus PageRank může mít více variant, já jsem použil tu nejpoužívanější (z nalezených zdrojů, ze kterých jsem čerpal, měl nejvyšší četnost).

2.2.1 Vzorec PageRanku

PageRank normálně dosahuje hodnot v rozmezí nuly až jedné. Co se týče následujícího vzorce, tak zjednodušeně řečeno, funguje tak, že vypočte rank podle toho, kolik a jak důležitých webových stránek na danou webovou stránku odkazuje (viz obrázek č. 2.1).

$$PR(A) = \frac{1-d}{m} + d\cdot\left[\frac{PR(T1)}{C(T1)}+\frac{PR(T2)}{C(T2)}+\frac{PR(T3)}{C(T3)}+\ldots+\frac{PR(Tn)}{C(Tn)}\right]$$

Obr. 2.1 – Vzorec PageRanku

PR(A)	PageRank webové stránky A
d	dampening faktor (nastavený pravděpodobně na hodnotu 0,85)
m	celkový počet oindexovaných webových stránek
$T_1 \div T_n$	webové stránky, odkazující na webové stránky A
C(T)	počet odkazů vedoucích z webové stránky T

! pozn.: výše uvedený vzorec není originální od Google, jedná se pouze o jeho nejpravděpodobnější řešení (jedná se o nejpoužívanější vzorec – existuje více variant).

Vstupní hodnoty pro $PR(T_1 \div T_n)$ jsou vypočteny z předchozích iterací výpočtu ranku. Po několika iteracích pak vzoreček dobře konverguje. Platí, že čím nižší je dampening faktor, tím lepší je konvergence vzorečku. V praxi se pak jednotlivé hodnoty PageRanku většiny webových stránek drží těsně nad nulou [1] [11].

2.2.2 Funkce PageRanku

Když se podíváme na vzorec PageRanku, tak se dá říct, že webová stránka v podstatě předává část svého PageRanku webovým stránkám, na které odkazuje. Čím více obsahuje odkazů, tím méně každé webové stránce předá. Platí pak, že čím méně má webová stránka odkazů, tím více PageRanku se každým odkazem přeposílá. Vše logicky vyplývá z výše uvedeného vzorce.

Dále je pak důležité, že PageRank nijak nezávisí na hledaném slově, jedná se o skalární veličinu. PageRank se přiřazuje každé webové stránce, nikoliv celému webu. Co se pak týče řazení webových stránek v prohlížeči, tak PageRank je hlavní kritérium, ale ne jediné. K vyššímu zařazení v internetových vyhledávačích je pak dobré využít některou ze SEO metod (viz samostatná kapitola)[1].

2.2.3 Software pro výpočet PageRanku

Pro zjištění PageRanku konkrétních webových stránek existuje plno tzv. PageRank checkerů (někdy jen zkráceně PR checker). Jedná se o software, který má v sobě implementován algoritmus PageRanku a dokáže tak vypočítat hodnotu PageRanku pro většinu webových stránek.

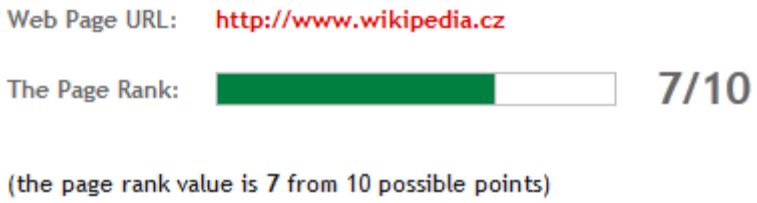
Tento software je distribuován buď jako různé addony, které rozšíří webové prohlížeče o možnost zjištění PageRanku, nebo je plno webů, kde stačí do příslušného okénka napsat URL adresu požadované webové stránky a obratem dostaneme ohodnocení PageRankem (u těchto webů je zpravidla uveden nejen samotný PageRank, ale i konkurenční Jyxorank, S-Rank, apod.)

Je důležité uvědomit si, že všechen tento software vypočítá hodnoty PageRanku pouze přibližně. Problém je v tom, že originální vzorec pro výpočet PageRanku je znám pouze Googlu, z čehož logicky plyne, že jakýkoliv software, který umí hodnotu PageRanku vypočítat, je pouze přibližným řešením.

PageRank checkery zobrazují hodnotu od nuly do deseti. Čím větší číslo, tím vyšší ohodnocení webové stránky (z toho plyne i vyšší pozice v internetovém vyhledávači). Jak vypadá takový PageRank checker lze vidět na obr. č. 2.2. Testovanou webovou stránkou byla

www.wikipedia.cz

a po provedení výpočtu dostala ohodnocení 7 bodů z 10 možných.



Obr. 2.2 – PageRank checker

2.2.4 Regulace hodnot PageRanku

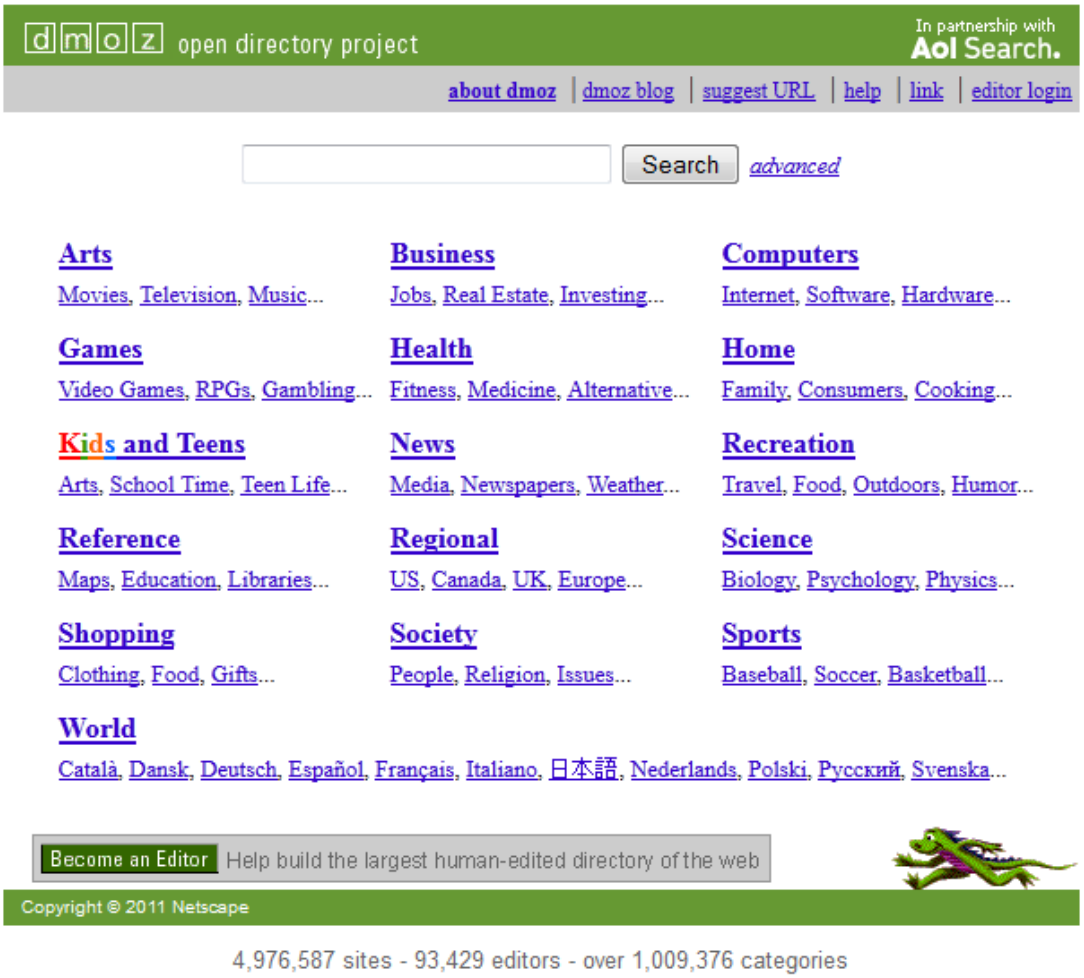
Regulace hodnot PageRanku je pro problém zviditelnění webových stránek asi ta nejdůležitější věc. Čím vyšší hodnota PageRanku, tím lepší pozice v internetovém vyhledávači. Když se podíváme, z čeho je složen vzorec PageRanku, tak je jasné, že se jeho výsledná hodnota dá uměle zvýšit, či snížit.

Vzhledem k tomu, že snižování hodnoty PageRanku je pro řešení daného problému negativním jevem, tak se budu věnovat pouze umělému zvyšování. Hlavní myšlenkou, jak zvýšit hodnotu PageRanku webové stránky je získat co nejvíce odkazů ze stránek, které už mají hodnotu PageRanku vysokou. Odkazy můžeme rozdělit na zpětné a dopředné. Zpětné odkazy jsou takové, které směřují na naší webovou stránku. Dopředné odkazy jsou pro změnu ty, které jsou na naší webové stránce a směřují pryč. Pro vyšší hodnotu PageRanku potřebujeme co nejvíce zpětných odkazů.

Zpětné odkazy můžeme získat buď pomocí internetových katalogů a nebo prolinkováním spřátelených webů, či nasazením různých spam-robotů na internetová fóra (tato metoda je ale neetická).

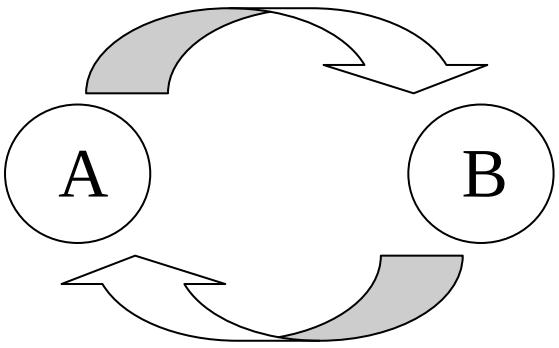
Internetový katalog je na internetu seznam odkazů na webové stránky, které jsou setříděny do stromu kategorií a podkategorií. Webové stránky do internetového katalogu většinou navrhují majitelé těchto stránek. Po přidání webové stránky do internetového katalogu se nejdříve zkontroluje obsah a poté je webová stránka zařazena do určité kategorie (práce editorů). Je důležité, neplést si internetový katalog s Internetovým vyhledávačem. Funkce internetového vyhledávače je z větší části automatizovaná, naproti tomu údržba internetového katalogu je náročnější na lidské zdroje. Na obr. č. 2.3 je možné vidět jeden takový internetový katalog, který nalezneme na:

http://dmoz.org



Obr. 2.3 – Internetový katalog

Prolinkováním spřátelených webů je myšlena vzájemná výměna odkazů. Pokud například na webové stránce A vytvoříme odkaz na webovou stránku B (pro webovou stránku B se jedná o zpětný odkaz) a na webové stránce B vytvoříme odkaz na webovou stránku A (pro webovou stránku B se jedná o zpětný odkaz), získáme tím zpětné odkazy jak na A, tak B. Pro obě webové stránky to tedy představuje lepší hodnocení PageRankem. Výše popsané prolinkování webových stránek A a B jednoduše ukazuje obr. č. 2.4 [1].



Obr. 2.4 – prolinkování webů

2.3 Životní cyklus rankingu webové stránky

Každá webová stránka má nějaký životní cyklus jejího rankingu. Takový cyklus popisuje ve formě grafu obr. č. 2.5. Když se vytvoří nová webová stránka, tak její ohodnocení PageRankem bude nula. Je to proto, že webová stránka je mladá a vyhledávač jí nestačí spočítat. Po nějakém čase získá pár odkazů a její ohodnocení bude stoupat až do určité meze, kde se ustálí. Ovšem může se stát, že pokud je webová stránka zdokonalována (třeba SEO metodami), může její ohodnocení nadále stoupat. V opačném případě ale i klesat (konkurenční obsah). U obr. č. 2.5 je uveden příznivý průběh – *A* a nepříznivý – *B* [1].

Obr. 2.5 – životní cyklus rankingu webové stránky

3 Propagace za použití SEO metod

Search Engine Optimization (zkráceně SEO) je metoda konstruování a modifikování webových stránek tak, aby byly ve výsledku v co nejlepší formě pro zpracování internetových vyhledávačů. Cílem těchto metod je pak vyšší pozice v internetových vyhledávačích a tím pádem i vyšší návštěvnost webových stránek. SEO metody můžeme rozdělit na etické a neetické.

Etické SEO metody jsou takové metody, které se snaží o vylepšení kvality samotných webových stránek. Naproti tomu neetické SEO metody se nesnaží o vylepšení kvality, ale jejich hlavním a jediným cílem je vyšší pozice v internetových vyhledávačích. Jsou proto použity různé podvodné metody. Mezi takovéto podvodné metody se řadí třeba spamovací roboti, skrytý text nebo i pověry (například pomlouvání konkurenčních webových stránek v internetových fórech). U těchto neetických metod je ale velký problém v tom, že internetové vyhledávače si takové metody kontrolují a brání se jim. Může se tedy stát, že pokud je taková metoda odhalena, tak se daným webovým stránkám razantně sníží ranking a tedy i budou mít nižší pozici ve vyhledávačích. Tyto metody jsou účinné jen krátkodobě. Nadále se tedy budu věnovat pouze etickým SEO metodám [7].

3.1 Optimální tvar URL adresy

Pro optimální tvar URL adresy je dobré, aby obsahovala klíčová slova. Není to však hlavní podmínka, ani zcela důležitá, ovšem internetovým vyhledávačům to zase o něco zjednoduší výpočet a tím je i větší pravděpodobnost, že požadovaná webová stránka získá vyšší pozici. Když už URL adresa obsahuje nějaká klíčová slova, je zase kontraproduktivní přidávat do ní stejná slova. Dále je pak dobré vědět, že nezáleží na tom, jestli je klíčové slovo obsažené v subdoméně nebo v cestě za doménou, vpředu nebo vzadu.

Další dobrou vlastností je udržet URL adresu krátkou. Z předchozího textu by šlo možná mylně říct, že čím více klíčových slov, tím lépe – URL adresa by sice byla dlouhá, ale obsahovala by veškerá klíčová slova. To je ale omyl, protože internetové vyhledávače jsou konstruovány tak, aby takto dlouhé URL adresy nebyly zvýhodňované.

Dále může být problém ohledně duplicity URL adres. V tabulce č. 3.1 jsou uvedeny čtyři různé URL adresy. Tyto URL adresy vidí vyhledávač jakožto čtyři různé webové stránky. Proto je nutné dodržovat při odkazování na webovou stránku pouze jednu verzi. Pokud by se tak nedělo, tak by docházelo k vytváření duplicitních webových stránek a hodnocení PageRanku by se pak zbytečně rozdělovalo. V praxi bychom tedy z níže uvedené tabulky 1 vybrali například možnost 3, která by byla použita jako jediná verze URL adresy.

Možnost:	**URL adresa:**
1.	www.example.com/
2.	www.example.com/index.html
3.	http://example.com
4.	example.com/index.html

Tab. 3.1 – duplicitní URL adresy

K optimálnímu tvaru URL adresy patří i správné oddělování slov. Je totiž možné, že pokud bude klíčové slovo obsažené v URL adrese nesprávně oddělené, tak ho internetové vyhledávače vůbec nenajdou. Například internetový vyhledávač Google, aby rozeznal začátek a konec slova, je třeba jednotlivé slova oddělit buď znakem mínus nebo tečkou. Český internetový vyhledávač Seznam je na tom o něco lépe, protože dokáže rozeznat i podtržítko, jako znak pro oddělení slova. Pro přehlednost je uvedena tabulka č. 3.2, která shrnuje výše popsané oddělování slov v URL adrese.

URL adresa:	Vyhledávač Google:	Vyhledávač Seznam:	Výsledek:
example.com/abcabc	abcabc	abcabc	*
example.com/abc_abc	abc_abc	abc abc	****
example.com/abc-abc	abc abc	abc abc	********

Tab. 3.2 – oddělování slov v URL adrese

Lze vidět, že nejoptimálnější oddělování slov je znakem mínus (nebo tečkou), protože ho dokážou rozpoznat oba uvedené internetové vyhledávače. Méně optimální je pak oddělování pomocí znaku podtržítka. A jako naprosto neoptimální řešení je pak oddělování slov bez znaku.

Výše uvedené byly obecné a známé věci, které se pro optimální tvary URL adres používají. Nyní se ale podíváme na detailnější postupy [1].

3.1.1 Vytváření optimální URL adresy pomocí souboru .htaccess

Soubor .htaccess slouží ve zkratce k tomu, aby autor nějakých webových stránek mohl sám, tedy bez nutnosti správce upravovat určité vlastnosti serveru. Tento soubor může fungovat na serveru Apache a jedná se o skrytý soubor (v OS Linux signalizuje tečka před souborem skrytý soubor). Jedna z vlastností souboru .htaccess je možnost měnit zobrazení URL adresy. Lze tedy pomocí tohoto konfiguračního souboru změnit například tuto adresu:

```
http://www.eshop.cz/index.php?p=123
```

Na tuto:

```
http://www.eshop.cz/zbozi-123
```

Konstrukce, která by se použila v konfiguračním souboru .htaccess by pak vypadala následovně [10]:

```
RewriteCond %{HTTP_HOST} ^www\.eshop\.cz
RewriteRule ^zbozi-(.*)\.html$ http://www.eshop.cz/index.php?p=$1 [L,QSA]
```

3.1.2 Vytváření optimální URL adresy pomocí jazyku PHP

Může se stát že výše uvedený soubor .htaccess nebudeme mít k dispozici. V tomto případě je vhodné poohlídnout se po skriptovacím jazyku PHP. Jazyk PHP umožňuje vytvářet optimální URL adresy a to v závislosti na konkrétním redakčním programu či elektronickém obchodu.

- Pokud máme spíše menší web o pár stránkách, bude pro nás nejlepší použít funkce jako include či require. Tyto funkce umožňují vložit dynamicky vygenerovaný obsah do statické stránky, kterou si libovolně pojmenujeme.

- V případě, že máme větší web, bude třeba vytvořit nějaký PHP skript pro komunikaci s databází. Tyto skripty pak obvykle vytvářejí název webové stránky z titulku článku či názvu propagovaného zboží.

Příklady převodu dynamických adres na statické lze vidět na webových stránkách viz níže [10]:

http://php.vrana.cz/vytvoreni-pratelskeho-url.php

http://php.vrana.cz/pekna-url-bez-mod_rewrite.php

3.1.3 Vytváření optimální URL adresy pomocí systému WordPress

WordPress je redakční publikační systém, který je ve formě open-source. Tento systém se hodně využívá na blogování a může posloužit i jako základ pro firemní stránky (spíše nízkorozpočtové). Výhodou tohoto systému je také široká škála různých pluginů, které mohou systém vylepšit o různá rozšíření či nové funkce.

Systém WordPress lze stáhnout na adrese níže:

http://wordpress.org/

Přičemž návod, jak s tímto systémem pracovat, je opět na adrese níže [10]:

http://www.cwordpress.cz/wordpress-manual-cesky

3.2 Změna URL adresy – přesměrování stránky

Může se stát, že z nějakého důvodu budeme chtít změnit URL adresu u svých webových stránek. V této chvíli si je dobré uvědomit, zdali je to opravdu nutné. Ve světě SEO nerozhoduje to, jak „hezké“ máme názvy URL adres, ale to, jaký máme rank. Tudíž pokud máme název, který je již zaběhlý a je na tuto URL adresu příznivý rank, není jediný důvod ke změně názvu této URL adresy. Naopak změna by znamenala výrazný posun dolů, co se návštěvnosti týče. Ovšem pokud jsou důvody pro změnu URL adresy opravdu podstatné, je třeba zajistit, aby se nám odkazy směrované na původní URL adresu přesměrovaly na novou URL adresu. V následujícím textu tedy uvedu některé z postupů, jak toto přesměrování zajistit.

3.2.1 Přesměrování webové stránky pomocí souboru .htaccess

Zde je nejprve nutné uvědomit si, jak moc chceme URL adresu změnit. Pokud se jedná o přesměrování v rámci jedné domény, jako lze vidět níže:

```
www.web.cz/strankaA.html  →  www.web.cz/strankaB.html
```

Pak zvolíme následující konstrukci:

```
RewriteEngine on
RewriteRule stranka-A\.html /stranka-B.html [R]
```

Uvedu ještě jeden způsob přesměrování a sice:

```
web.cz  →  www.web.cz
```

Konstrukce by pak obsahovala o jeden příkaz více:

```
RewriteEngine on
RewriteCond %{HTTP_HOST} ^web\.cz [nc]
RewriteRule (.*) http://www.web.cz/$1 [R=301,L]
```

Pozornost bych ještě věnoval parametrům použitých ve výše uvedených konstrukcí (jedná se o znaky uvnitř hranatých závorek). Pro tyto parametry jsem vytvořil tabulku č. 3.3, kde jsou uvedeny i se stručným popisem [10].

Parametr:	Popis:
R	Přesměrování s kódem 302
R=301	Přesměrování s http kódem 301
F	Nastavuje kód 403
P	URL adresa neexistuje již dlouhou dobu
NC	Nezáleží na velikosti písmen
OR	Logické OR použité u podmínky
L	Poslední pravidlo – další se již nebudou používat

Tab. 3.3 – Popis jednotlivých parametrů

3.2.2 Přesměrování webové stránky pomocí jazyku PHP

Přesměrování za použití jazyku PHP je nejvhodnější způsob. Je ale nutné, aby byl podporován poskytovatelem webhostingu a aby uměl pracovat s funkcí Header). Pokud je toto splněné, použijeme konstrukci, která je uvedena níže [10].

```
<?php
header("HTTP/1.1 301 Moved Permanently") ;
header("Location: http://www.web.cz/nova-stranka.html") ;
header("Connection: close") ;
?>
```

3.2.3 Přesměrování statické webové stránky

Statická stránka lze přesměrovat dvěma způsoby.

1. Pomocí metaznačky refresh
2. Pomocí jazyku JavaScript

První způsob, tedy pomocí metaznačky refresh, je uveden níže. V podstatě stačí, když uvedenou konstrukci překopírujeme do hlavičky webové stránky, kterou chceme přesměrovat.

```
<meta http-equiv="refresh" content="0;url=http://nova-adresa.cz/nova-stranka.html">
```

Atribut content obsahuje číselnou hodnotu, která udává, po kolika vteřinách se webová stránka přesměruje (zde tedy okamžitě).

Druhý způsob, tedy pomocí jazyku JavaScript, je neúčinný a z hlediska internetových vyhledávačů i nevhodný [10].

3.3 Budování zpětných odkazů

Tato problematika už byla podrobně řešena v předchozí kapitole o rankingu, ale v krátkosti ještě připomenu její princip.

Zpětný odkaz, je takový odkaz, který ukazuje na naší webovou stránku. Čím více těchto zpětných odkazů máme, tím větší pravděpodobnost vyššího ohodnocení PageRankem. A čím vyšší ohodnocení PageRankem, tím vyšší pozice v internetových vyhledávačích a tím i větší návštěvnost webových stránek.

Jelikož, jak píši výše, tato problematiku již byla probírána v předchozí kapitole, tak v následujícím textu budu uvádět spíše praktické rady, které se při budování zpětných odkazů mohou hodit.

3.3.1 Blokování nechtěných zpětných odkazů

Předpokládejme, že máme webové stránky, na kterých provozujeme nějaké diskusní fórum, případně jiný způsob, jak umožnit uživatelům umístit libovolný komentář právě na naše stránky. Může se pak stát, že uživatel napíše komentář, ve kterém bude umístěn i odkaz na jinou webovou stránku (na nějakou pro nás nežádoucí). V tomto případě by se při kliknutí na odkaz naše webová stránka přesměrovala na jinou – cizí. Pro nás by to ale znamenalo, že ranky, které ukazují na naší stránku by se předaly i té cizí, což nechceme. Abychom tomu zabránili, použijeme příkaz nofollow. Níže vidíme, jak vypadá standardní zápis kódu odkazu:

```
<a href="http://www.URL-odkazu.cz">Text-odkazu</a>
```

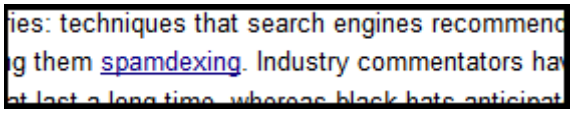
Při použití příkazu nofollow by pak vypadal následovně:

```
<a href="http://www.URL-odkazu.cz" rel="nofollow">Text-odkazu</a>
```

Pokud použijeme konstrukci s příkazem nofollow, tak internetové vyhledávače nebudou předávat ranky, které směřují na naše stránky dál na cizí. Pro nás to tedy bude znamenat, že nepřijdeme o žádné ranky. Navíc některé vyhledávače dokonce webové stránky s příkazem nofollow vůbec nesledují [10].

3.3.2 Volba správného anchor textu

Text nazývaný anchor znamená část hypertextového odkazu. Tento odkaz lze vidět na obrázku č. 3.1, kde se jedná o podtržený text. Po kliknutí bychom pak měli být přesměrováni na stránku, která souvisí s klíčovým slovem obsaženým právě v podtrženém textu.



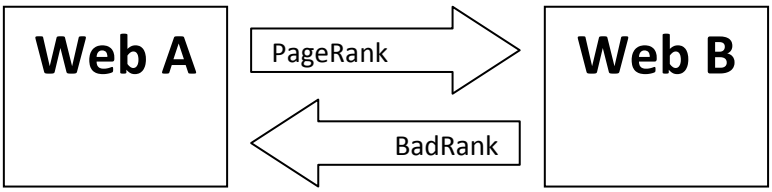
Obr. 3.1 – Ukázka anchor textu

Konstrukce pro vytvoření anchor textu by pak vypadala následovně [10]:

```
Prodej <a href="http://www.web.cz/strana.html" title="ojetých aut">auta</a> lze vidět na uvedené webové stránce.
```

3.3.3 Vyhnutí se zápornému BadRanku

BadRank funguje na podobném principu jako PageRank. Oproti PageRanku ale není založen na kontrole příchozích odkazů, ale odchozích. Jedná se v podstatě o algoritmus, který dohlíží na to, jestli z našeho webu neodkazujeme na „špatné“ stránky. Pokud totiž ano, tak dostaneme vysoký BadRank, a to pro náš web znamená cestu zpět. Je tedy dobré hlídat si, na které webové stránky odkazujeme. Obrázek č. 3.2 zjednodušeně popisuje, jaký je rozdíl mezi BadRankem a PageRankem.



Obr. 3.2 – Rozdíl mezi BadRankem a PageRankem

BadRank je dobrý systém proti odstranění nechtěného spamu, který může být obsažen na webových stránkách. Postupně se tento systém BadRanku začíná implementovat do většiny internetových vyhledávačů [10].

3.3.4 Výběr správného cíle odkazu

Máme dvě možnosti, jak můžeme směrovat odkazy. Buď je směrujeme na hlavní stránku našeho webu, nebo na jednotlivé podstránky. Je jasné, že oboje má své klady i zápory. Buď budeme mít vysoké ranky směrující na hlavní stránku webu, nebo o něco menší směrující na podstránky. V prvém případě, tedy odkazováním na hlavní stránku, získáme tyto výhody:

- Vyšší ranky směrující na naší webovou stránku
- Můžeme posílit vysoce konkurenční klíčová slova

Použití vysoce konkurenčních slov je sice dobré, ale vzhledem k množství webových stránek bude velmi těžké získat pozitivní pořadí v internetových vyhledávačích.

V případě odkazování na jednotlivé podstránky je situace trochu jiná. Webové stránky sice nebudou mít tak velké ranky, ale zato získají tyto výhody:

- Možnost přesměrování přesně tam, kam uživatel chtěl
- Možnost posilování velkého množství klíčových slov
- Možnost zaměřit se na méně konkurenční klíčová slova

Výběr směrování odkazů tedy závisí na tom, co chceme na webových stránkách propagovat. Pokud by se mělo jednat o internetový obchod, kde je vhodné, aby uživatel při dotazu byl přesměrován přímo na požadovaný produkt, bylo by lepší použít směrování na jednotlivé podstránky. V případě, že by se jednalo o propagování jednoho určitého produktu, kde by jednotlivé podstránky obsahovaly ceník a podobně, bylo by lepší použít směrování na hlavní stránku [10].

3.3.5 Kontrola pravosti PageRanku

Může se stát, že narazíme na web, který bude mít vysoký PageRank, ale přitom bude mít málo zpětných odkazů. Takový web má s velkou pravděpodobností falešně navýšenou hodnotu PageRanku. Existují ovšem nástroje, které falešný PageRank odhalí. Jeden takový nástroj je uveden na obrázku č. 3.3. Jedná se o nástroj CHECKPAGERANK.NET [10].

Obr. 3.3 – Ukázka validace

3.4 Volba klíčových slov

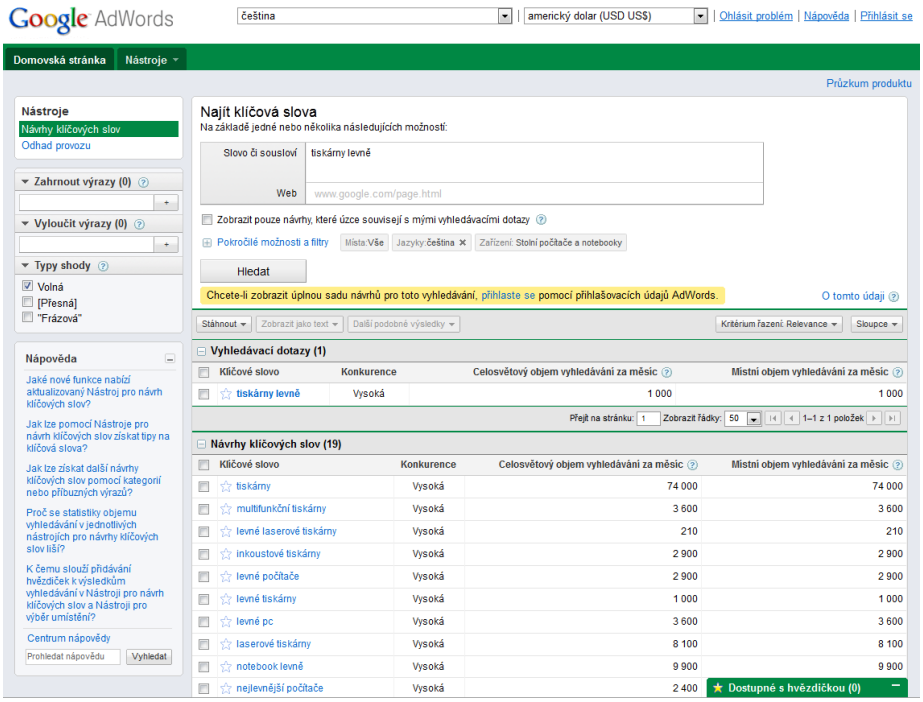
Volba vhodných klíčových slov je jedna z nejdůležitějších věcí, co se týče SEO. Pokud se totiž nezvolí správná klíčová slova, tak se stane, že webové stránky vyhledávače vůbec nenajdou. Takový problém může nastat například jen při špatném zápisu tvaru slova, protože internetové vyhledávače neumí skloňovat.

Pokud vytváříme web, tak je vhodné nejdříve stanovit klíčová slova, která by měla odpovídat obsahu. Dále pak zjistit vyhledávanost těchto slov a duplicitu (kvůli konkurenci – pokud mají klíčová slova perfektní vyhledávanost, ale je u nich velká konkurence, je lepší vybrat jiná klíčová slova, která budou podobná). Programátor těchto webových stránek by si měl odzkoušet i roli uživatele, přesněji vyzkoušet si, jak by uživatel vyhledával jeho webové stránky. Je totiž dost pravděpodobné, že například člověk, který je v informatice více gramotný bude obsah webové stránky vyhledávat odlišnými klíčovými slovy než člověk, které je v informatice méně gramotný.

Co se týče zjištění, zdali námi navrhnutá klíčová slova mají či nemají velkou konkurenci, je dobré použít nějaký software, který by nám s tím pomohl. Třeba internetový vyhledávač Google má v sobě zabudovanou službu Google AdWords, která dokáže výše popsaný problém celkem přehledně řešit.

Když se podíváme na obrázek č. 3.4, tak poznáme, jak taková služba vypadá. Do kolonky *Najít klíčová slova* napíšeme slovo, nebo sousloví, které chceme otestovat. V našem případě jsou to tyto slova: *tiskárny levně*. Pod těmito slovy je uvedena ještě webová stránka, ze které jsou tato slova vyhledávána (www.google.com). V kolonce *Vyhledávací dotazy* pak už je vidět konkurence klíčových slov, celosvětový objem vyhledávání za měsíc a místní objem vyhledávání za měsíc. Na další kolonce *Návrhy klíčových slov* se nám zobrazují jednotlivá alternativní klíčová slova. Opět je u nich uvedena i konkurence, celosvětový objem vyhledávání za měsíc a místní objem vyhledávání za měsíc. Nyní tedy máme srovnání našich klíčových slov vůči alternativním klíčovým slovům. Dalším postupem tedy bude upravit klíčová slova tak, aby byla související a zároveň měla co nejnižší konkurenci (samozřejmě to musí být slova, která budou opět často vyhledávaná, proto je nutné zvolit nějaký poměr mezi těmito kritérii).

Ještě podotknu, že existuje i plno jiných nástrojů, které nám pomohou s optimální volbou klíčových slov. Jako příklad uvedu nástroj Market Samurai [9].



Obr. 3.4 – Google AdWords

3.4.1 Klíčová slova typu „long tail“

Abych vysvětlil pojem „long tail“, bude nejlepší demonstrování na příkladu. Máme nějaké klíčové slovo:

ubytování

„Long tail“ k tomuto slovu pak bude níže uvedené sousloví:

ubytování Šumava

Je tedy vidět, že „long tail“ je v podstatě další klíčové slovo, které upřesňuje to první. Je jasné, že první klíčové slovo povede rychleji k cíli, ale je konkurenční. Pokud je v dané oblasti velká konkurence, je lepší soustředit se právě na „long tail“, protože pomocí něho bude větší šance, že uživatel se prokliká právě k vašemu webu (dle statistik) [10].

3.5 Volba tagů

Nejdůležitější tag na webové stránce je bezpochyby titulek. Titulek má totiž jako jediný velký význam ve většině internetových vyhledávačů. Každá webová stránka by měla mít odlišný titulek a to i v rámci stejného webu. V titulku by měla být implementována klíčová slova, která odpovídají obsahu webové stránky. Je také dobré ke klíčovým slovům přidat i název firmy či serveru. Sice se tím možná o něco sníží pozice v internetových vyhledávačích, ale je nutné si uvědomit, že webové stránky jsou hlavně pro uživatele (a ne pro internetové vyhledávače), takže je také důležité budovat značku. Pro titulek je tedy ideální, aby jako první byl uveden název serveru (či firmy) a za ním pak klíčová slova. Doporučená délka titulku je 70 znaků. Jak takový titulek vypadá, můžeme vidět viz níže.

```
<head>
<title>Titulek webové stránky</title>
</head>
```

Další důležitý tag je popisek (meta-description), protože ho některé internetové vyhledávače zobrazují u popisu webové stránky (u jednotlivých výsledků hledání). Je tedy dobré do tohoto tagu napsat nějaký smysluplný popisek. Doporučená délka popisku je 250 znaků. Opět je uveden viz níže.

```
<meta name=”description” content=”Popisek webové stránky”>
```

Následuje tag nadpisy, tedy H1 až Hn. Když dáme něco do nadpisu, tak by to logicky mělo mít větší váhu. V tomto případě platí, že čím delší je tag H1, tím menší má význam klíčové slovo obsažené v něm. Dále platí, že největší váha je u H1 a čím vyšší číslo (například H6), tím menší váha. Nadpis H1 se na webové stránce může opakovat jen jednou, u ostatních nadpisů i vícekrát. Nadpis H1 je uveden viz níže.

```
<H1>Nadpis webové stránky</H1>
```

Jako poslední důležitý tag uvedu popisek u obrázků (alt). Je dost pravděpodobné, že budeme chtít na naší webovou stránku vložit nějaký obrázek. V tomto případě se hodí vložený obrázek také nějak popsat, k čemuž slouží výše uvedený tag. U každého takto vloženého obrázku by měl být vyplněný atribut alt, který je použit k zastoupení obsahu obrázku. Může se totiž stát, že uživatel, který by chtěl vyhledat určitý obsah na naší webové stránce, by měl vypnuté zobrazování obrázků. Internetový vyhledávač by pak nedokázal určit, co na obrázku je. Ovšem, pokud máme obrázek popsaný, tak atribut alt řekne internetovému vyhledávači, co se na obrázku nachází. Popisek u obrázků zapsaný v kódu opět vidíme, viz níže.

```
<img src=”obrazek.gif” alt=”obsah obrázku” title=”shrnutí obsahu”>
```

Ještě podotknu, že u obrázků o velikosti 1x, které jsou dobré jen ke grafickým účelům, je nutné nechat atribut alt prázdný. Mohlo by se totiž stát, že internetový vyhledávač by takové klíčové slovo bral jako spam [9].

4 Distribuční balíčky pro OS Linux

Distribuční balíčky pro OS Linux jsou, co se týče kompatibility, o něco složitější než distribuční balíčky pro OS MS Windows. Důvodem je nutnost rozlišení Linuxové distribuce. Tyto distribuce mohou být založené na odlišném balíčkovacím systému. Musíme tedy balíčky vytvářet tak, aby byly závislé na konkrétní distribuci Linuxu. V následující tabulce č. 4.1 je uveden přehled možných balíčkovacích systémů a výběr známějších Linuxových distribucí, které daný systém využívají.

Balíčkovací systém:	**Linuxová distribuce:**
APT (*.deb)	Debian
	Gnoppix
	Edubuntu
	Kubuntu
	Ubuntu
	Xubuntu
	Google Chrome OS
RPM (*.rpm)	CentOS
	Fedora
	MandrivaLinux
	Red Hat
	SUSE
	YOPER
Pkgsrc	BlackMouse
Pacman	Arch Linux
Conary	ForesightLinux

Tab. 4.1 – přehled balíčkovacích systémů

Balíčkovací systémy slouží k uchování více informací jako jeden samostatně spustitelný celek. Například pomocí balíčkovacího systému APT můžeme vytvořit soubor, který bude mít příponu **.deb* a dále s ním pracovat jakožto s instalačním prvkem. Bude pak tedy tvořit obdobu například **.msi* souboru, který je hojně využíván v OS MS Windows.

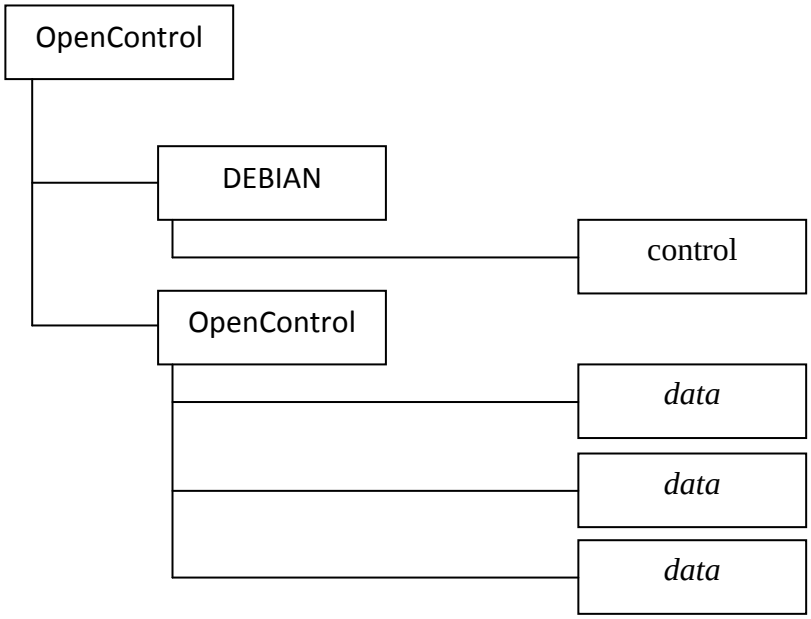
Náš záměr je vytvořit distribuční balíček, tedy balíček, ve kterém bude obsažen nějaký program, který by mělo být snadné nainstalovat na OS Linux (či OS MS Windows) a zajistit tak jeho distribuci. Vzhledem k rozšířenosti jednotlivých distribucí se budeme věnovat hlavně balíčkovacím systémům APT a RPM.

4.1 Balíčkovací systém APT

Balíčkovací systém APT se používá spíše u nekomerčních Linuxových distribucí. Výsledný instalační soubor má koncovku **.deb* a lze jej jednoduše spustit v grafickém (GUI) i v řádkovém (CLI) režimu. Co se týče vytváření těchto balíčků, lze použít nějaký nástroj, který provede potřebné operace nebo balíček vytvořit ručně.

4.1.1 Konstrukce DEB balíčku

Postup, jak vytvořit takový DEB balíček ručně, tedy bez použití jakýchkoli nástrojů, je celkem snadný. Máme nějaký program či jiná data (v našem případě OpenControl), který chceme importovat do jednoho instalačního balíčku. V prvé řadě musíme vytvořit centrální adresář, do kterého se zkopírují další dva adresáře, a sice adresář s vlastním programem a adresář s kontrolním souborem. Tuto situaci přehledně popisuje i stromová struktura na obrázku č. 4.1.



Obr. 4.1 – Stromová struktura DEB balíčku

Když máme takto připravenou stromovou strukturu adresářů, je třeba nastavit kontrolní soubor. Ten se nastaví podle tabulky č. 4.1. V tabulce jsou obsaženy důležité položky, které se musí nastavit přímo v kontrolním souboru. Tyto údaje jsou pak při instalaci hotového balíčku použity ke kontrole závislostí k tomu, aby bylo možno zobrazit informace o instalovaném programu. V podstatě se jedná o předání informací danému instalačnímu nástroji (například Ubuntu Software Center). Po nastavení položek kontrolního souboru a

správném uspořádání stromové struktury budoucího DEB balíčku, je na řadě samotné sestavení balíčku. K tomu je zapotřebí dpkg, což je nástroj na bázi operačního systému Linux Debian určený pro správu balíčků. Používá se především k instalaci, odebírání a poskytování informací o DEB balíčků. Nástroj dpkg je sám o sobě nízko-úrovňový software a ze svých konkurenčních nástrojů jako je aptitude či synaptic je méně používaný, protože nemá tak důmyslný způsob nakládání s balíky a nemá tak přátelské rozhraní. Výhodou ale je jeho jednoduchost.

Položky kontrolního souboru:		**Význam položek:**
Project name		Jméno projektu (programu)
Maintainer		Jméno správce/vlastníka projektu
Version number		Verze projektu
Architecture		Architektura, na které funguje daný projekt (např. i386)
Depends		Balíky potřebné pro instalaci programu
Description	Title	Popis hlavičky – stručný obsah
	Body	Popis těla – obsáhlejší obsah

Tab. 4.1 – Obsah kontrolního souboru DEB balíčku (základní položky)

Použití nástroje dpkg je možné v řádkovém režimu (CLI) a to přímo v terminálu dané Linuxové distribuce. Stačí napsat příkaz dpkg a zvolit správný parametr. Přehled takových základních parametrů, které jsou třeba pro práci s balíky, je možno vidět v tabulce č. 4.2.

Základní příkaz:	**Parametr:**	**Alternativa:**	**Popis:**
dpkg	-b	--build	Sestaví **.deb* balík
	-c	--contents	Zobrazí obsah balíku
	-I	--info	Zobrazí informace o balíku
	-f	--field	Zobrazí ovládací pole balíku
	-e	--control	Extrahuje kontrolní informace z balíku
	-x	--extract	Extrahuje soubory obsažené v balíku
	-X	--vextract	Extrahuje soubory obsažené v balíku a zobrazí jejich názvy

Tab. 4.2 – Přehled základních parametrů při použití nástroje dpkg

Nástroj dpkg není problém používat i na OS Linux ve virtuálním prostředí, či s ním operovat přes vzdálenou plochu. Funkčnost ve virtuálním prostředí byla odzkoušena za pomoci programu VMware player (virtualizační softwarový balík). Vzdálená plocha pak pomocí

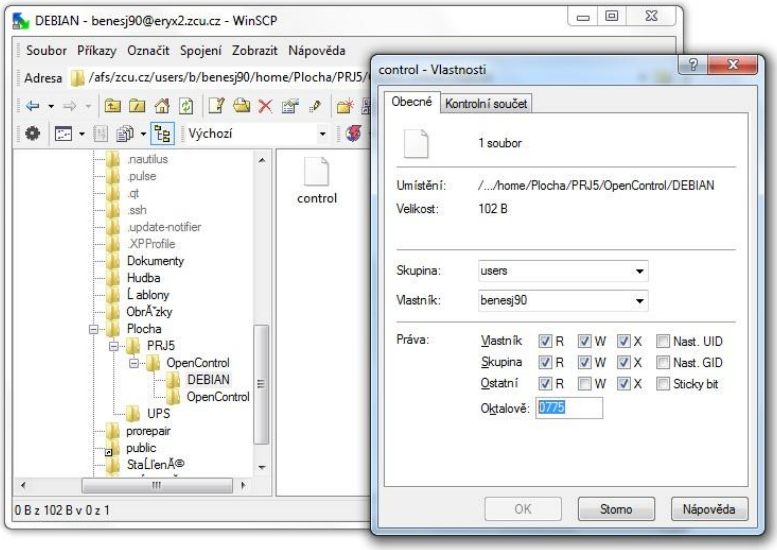
programu PuTTy (klient protokolů SSH a Telnet) a terminálového serveru Eryx, kde byl k dispozici OS Linux Debian.

Před samotným použitím nástroje dpkg je ale ještě nutné správně nastavit přístupová práva kontrolního souboru a adresáře, ve kterém tento soubor leží. K tomu slouží příkaz chmod, který dle tabulky č. 4.3 nastaví práva přístupu k souboru v závislosti na vybrané hodnotě.

Hodnota	Právo přístupu		
	čtení	zápis	spuštění
0	ne	ne	ne
1	ne	ne	ano
2	ne	ano	ne
3	ne	ano	ano
4	ano	ne	ne
5	ano	ne	ano
6	ano	ano	ne
7	ano	ano	ano

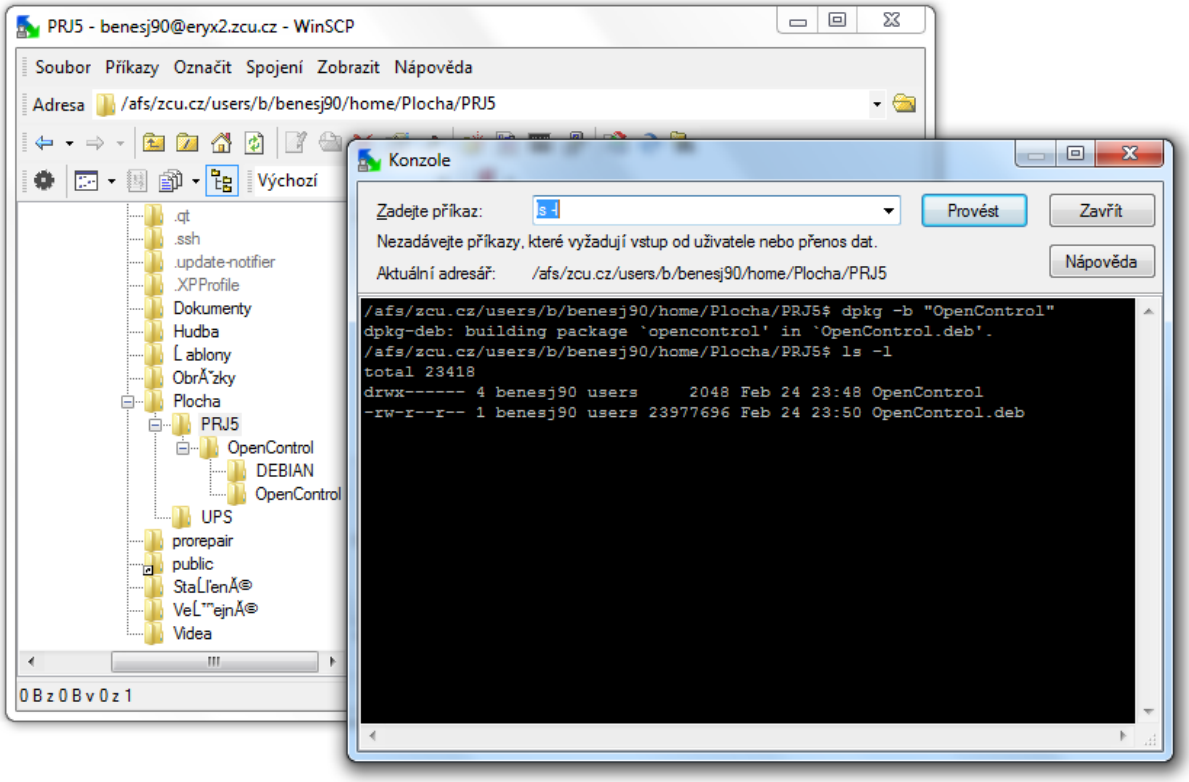
Tab. 4.3 – Orientační rozpis pro nastavení práv příkazem chmod

Tento příkaz se zapisuje do Linuxového terminálu ve formátu "*chmod XXX soubor*". Uvedená *X*-ka představují jednotlivé hodnoty z tabulky č. 4, přičemž první *X*-ko určuje práva vlastníka, druhé určuje práva skupiny a třetí práva k ostatním. Alternativou k zápisu do terminálu může být program WinSCP (SFTP a FTP klient), který lze vidět na obrázku č. 4.1.



Obr. 4.1 – Program WinSCP při nastavování přístupových práv

Po nastavení přístupových práv už nám nezbývá nic jiného, než použít nástroj dpkg s parametrem *–b* a tím vytvořit balíček (viz obrázek č. 4.2). K této operaci byl opět použit program WinSCP, díky němuž bylo možné připojit se k OS Linux Debian, běžícím na terminálovém serveru Eryx a vytvořit konzolu, která by se dala ovládat přes OS MS Windows (tento program byl zvolen vzhledem k jednoduchým operacím typu kopírování a přesouvání, což je při tvoření DEB balíku užitečné) [2].



Obr 4.2 – Vytváření DEB balíčku v konzole programu WinSCP

Detail konzole, při vytváření DEB balíčku:

```
/afs/zcu.cz/users/b/benesj90/home/Plocha/PRJ5$ dpkg –b "OpenControl"
dpkg-deb: building package 'opencontrol' in 'OpenControl.deb'.
/afs/zcu.cz/users/b/benesj90/home/Plocha/PRJ5$ ls –l
total 23418
drwx------      4 benesj90 users       2048   Feb  24 23:48          OpenControl
-rw-r--r--      1 benesj90 users   23977696   Feb  24 23:50          OpenControl.deb
```

4.1.2 Nástroje pro vytváření DEB balíčků

Nástrojů pro vytváření DEB balíčků existuje více. V podstatě se jedná o nástroje, do kterých zadáme požadované údaje, jako jsou vstupní cesta ke stromové struktuře DEB balíčku, výstupní cesta pro výsledný soubor a informace pro kontrolní soubor. Na základě těchto údajů nám pak použitý nástroj vytvoří DEB balíček. Tyto nástroje mohou být buď ve formě programů, které se instalují do určité Linuxové distribuce, či ve formě webových aplikací, které není nutné instalovat (např. http://www.hanovsolutions.com/easydeb/).

Pokud bychom se bavili o programech, které je nutné nainstalovat, je jich k dispozici přiměřeně dost. Ovšem problém je, že ne všechny fungují korektně a bezproblémově na dané Linuxové distribuci (bývá to v důsledku takzvané „domácí“ tvorby těchto programů). Proto je třeba tyto programy nejdříve odzkoušet a vybrat pak ten, který funguje nejlépe. V tabulce č. 4.4 máme takový seznam programů uvedený. Jedná se o programy, které byly testovány v OS Linux Ubuntu (11.10) a na základě složitosti jejich instalace a následného ovládání byl pak stanoven jejich rank, který je zde uveden v procentech (čím vyšší, tím lepší).

Abych své hodnocení programů nějak konkretizoval, uvedu, jak jsem postupoval. V první fázi jsem jednotlivé programy stáhnul a to buď pomocí příkazu z repositáře nebo ručně. S tím souvisí i instalace, jelikož pokud je program k dispozici v repositáři, tak ho lze i snadno nainstalovat. Hodnocení je ale relevantní, repositářů totiž existuje více a já zkoušel pouze ten, který je v distribuci Ubuntu implicitně nastavený. Z těchto důvodů přikládám složitosti instalace pouze 20% z celkového hodnocení. V další fázi jsem všechny programy, uvedené v tabulce č. 4.4 podrobil hlavní operaci – sestavení DEB balíčku. Zde jsem 50% hodnocení přikládal funkcionalitě a zbylých 30% časové náročnosti (vzhledem k pochopení ovládání programu). Dle tohoto výše popsaného systému hodnocení jsem „přibližně“ sestavil i další tabulku v kapitole o distribuci v OS MS Windows a tím spojeným balíčkovacím systémem MSI.

Testované programy	Webové stránky programů	Rank
Deb-creator	http://sourceforge.net/projects/deb-creator/	95 %
Debian Package Maker	http://kangkong-soft.blogspot.com/	70 %
Packin Debian Package Creator	http://sourceforge.net/projects/packin/	65 %
Ubucompilator	http://ubucompilator.wordpress.com/	65 %
Debomatic	http://www.makelinux.net/man/1/D/debomatic	55 %
Deb Creator	http://debcreator.cmsoft.net/	55 %

Tab. 4.4 – Seznam testovaných programů v OS Linux Ubuntu (aktuálnost webových stránek – 25. 2. 2012)

Z tabulky č. 4.4 je patrné, že program Deb-creator obstál s nejvyšším rankem, ukážeme si tedy, jak takový program vypadá a jak se s ním pracuje [8].

4.1.3 Ovládání programu Deb-creator

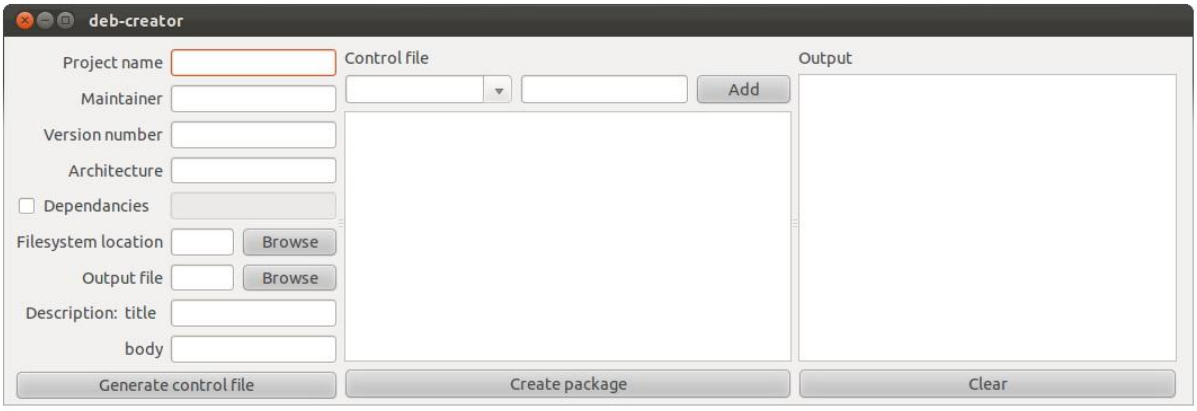
Po nainstalování programu Deb-creator (v prostředí OS Linux Ubuntu by mělo stačit otevření pomocí Ubuntu Software Center a postupování dle instrukcí) je třeba před samotným spuštěním ještě do terminálu napsat níže uvedený příkaz.

```
sudo apt-get install gtk2-engines-pixbuf
```

Jedná se o ošetření, aby se po spuštění programu nevypisovala chybová hlášení (upozornění). Po tomto kroku stačí napsat další příkaz viz níže a tím se provede spuštění.

```
deb-creator
```

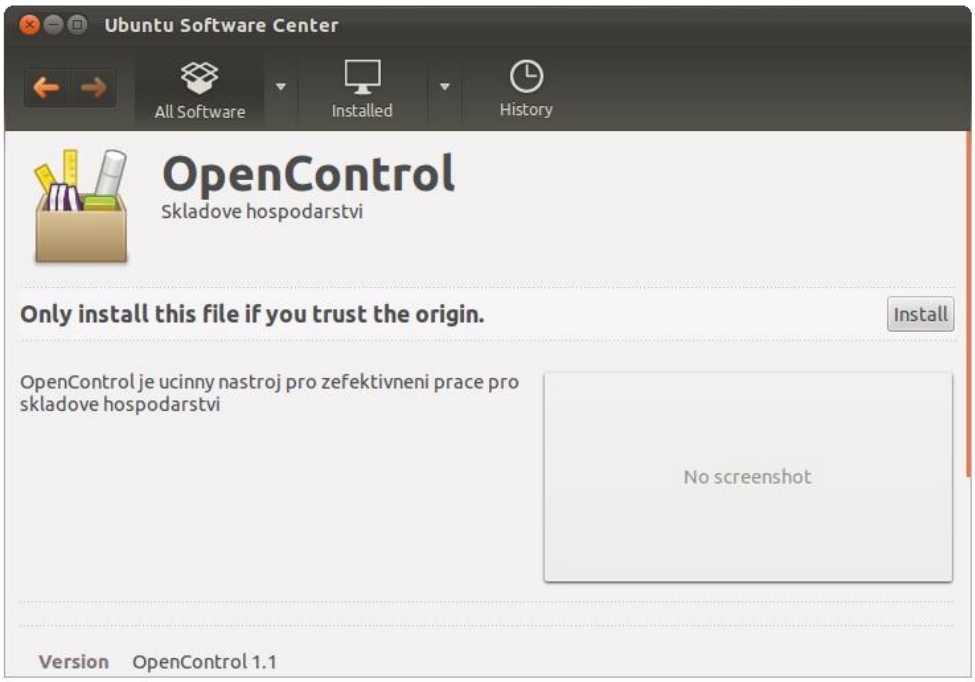
Nyní bychom měli mít spuštěný program Deb-creator, čili měl by se nám naskytnout stejný pohled, jako je na obr. č. 4.2. Je tedy zřejmé, že bude třeba vyplnit všechny položky, které odpovídají položkám uvedených v tabulce č. 4.1. Jedná se tedy o nastavení kontrolního souboru. Jsou zde navíc pouze položky *Filesystem location* a *Output file*. *Filesystem location* slouží k nastavení cesty k datům, z nichž chceme vytvořit DEB balíček. *Output file* pak analogicky k cestě, kam se hotový DEB balíček uloží (výstupní soubor). Pokud máme toto nastavené, stačí potvrdit *Generate control file* a rázem se nám vytvoří kontrolní soubor. Dále pak můžeme kontrolní soubor buď modifikovat (lze zadat více položek, než je těch implicitních, například domovská stránka projektu, velikost souboru, priority a další) nebo potvrdit *Create package* a tím vytvořit již plně funkční DEB balíček. Takto můžeme jednoduše operovat i s tvořením více balíčků najednou.



Obr. 4.2 – Grafické prostředí programu Deb-creator

Výsledný DEB soubor se nachází v adresáři, který jsme zvolili při nastavování programu. Nezbývá, než program zkusit nainstalovat. Pokud jsme správně nastavili všechny potřebné

položky kontrolního souboru a vytvoření DEB balíčku bylo tedy úspěšné, uvidíme po spuštění stejné okno, jako je na obrázku č. 4.3. Jedná se o nástroj Ubuntu Software Center, což je vlastně takový průvodce celou instalací DEB balíčku (funguje i jako správce nainstalovaných DEB balíčků a je součástí OS Linux Ubuntu).



Obr. 4.3 – Průvodce instalací programu *(*.deb* balíčku) OpenControl

4.2 Balíčkovací systém RPM

Balíčkovací systém RPM se používá spíše u komerčních Linuxových distribucí a byl původně vyvinut firmou Red Hat. Jedná se o celkem zastaralý systém, což také potvrzuje množství záplat (nikdo nepočítal s tím, že se tento systém bude používat až do současnosti). Výsledný instalační soubor má koncovku **.rpm* a lze jej jednoduše spustit v grafickém (GUI) i v řádkovém (CLI) režimu. Co se týče vytváření těchto balíčků, lze použít nějaký nástroj, který provede potřebné operace nebo balíček vytvořit ručně [3].

4.2.1 Konstrukce RPM balíčku

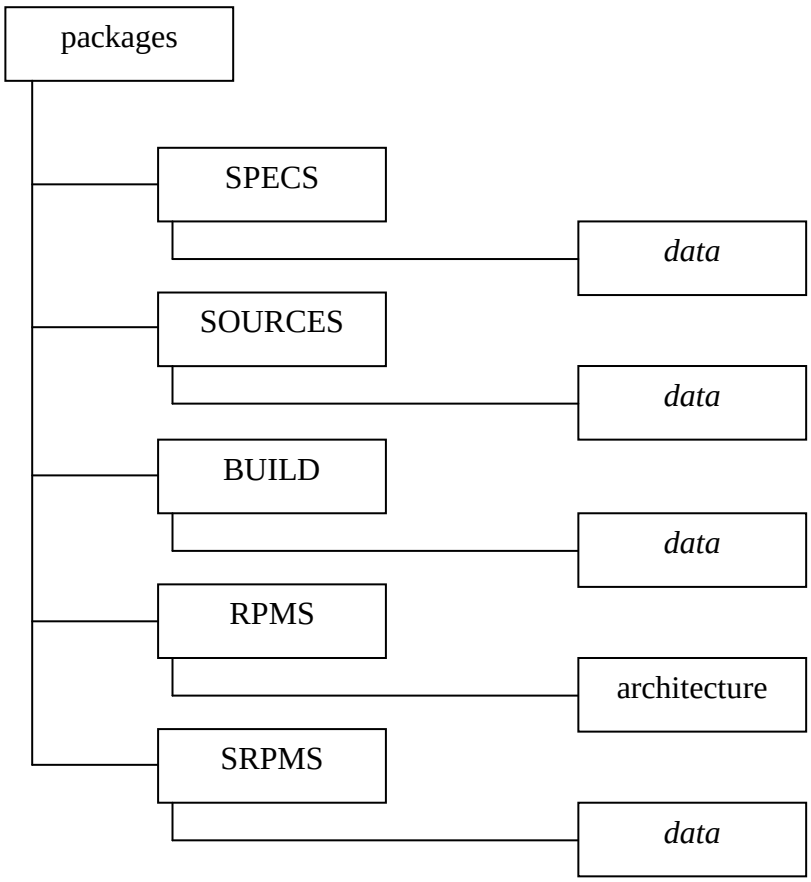
Struktura RPM balíčku je trochu odlišná od DEB balíčku, ovšem má podobnou funkci speciálního souboru (obdoba kontrolního souboru u DEB balíčku). Základním rozdílem je zde ale to, že do RPM balíčku není možné zakomponovat jen samotná audiovizuální data. Vždy se musí jednat o nějaký program, po kterém chceme integrací do RPM balíčku získat možnost instalace do Linuxových distribucí. Pokud bych to shrnul, tak DEB balíček můžeme využít i jako obdobu **.zip* či **tar.gz* balíčku pro jakákoliv data, u **.rpm* to nelze. Z toho také plyne důležitost odlišení pojmů binární a zdrojový balíček. U **.deb* balíčků jsme se těmito pojmy nezabývali, protože celý balíčkovací systém byl popisován globálně, tak, aby byla docílena univerzalita pro jakákoliv data, která bychom případně chtěli do tohoto balíčku přidat. V užším použití, například pro programy, bychom se ale s pojmy binární a zdrojový balíček seznámit museli (informace o těchto pojmech budou probírány v této kapitole a jedná se o analogicky stejnou funkcionalitu, jako v případě použití u **.deb* balíčků).

Binární balíček (binary package) je balíček, který již obsahuje zkompilované programy. Jedná se o balíček, který je pro nás už hotový, tzn. balíček, který je připraven k instalaci. Jejich koncovka je klasicky **.rpm* a jejich formát bývá většinou *architecture.rpm,* kde *architecture* udává architekturu, pro kterou byl daný balíček zkompilován (např. *i386.rpm*). Tento balíček je **základní** - nejpoužívanější a co se funkce týče, srovnatelný s předchozím DEB balíčkem. Tudíž v realizační části se budeme zabývat právě binárním balíčkem.

Zdrojový balíček (source package) je balíček, který obsahuje zdrojový kód programu a další věci, potřebné ke zkonstruování binárního balíčku. Tento balíček není možné rovnou instalovat (respektive možné to je, ale nedojde ke korektní instalaci – nedojde k žádnému konfliktu, ale ani k žádnému výsledku), slouží jen jako předstupeň binárního balíčku. Jejich koncovka je **.src.rpm* a ukládají se do složky, přímo určené ke zdrojovým balíčkům.

Kromě binárního a zdrojového balíčku je nutné objasnit si ještě pojem tarová koule. Tarová koule, neboli tarball je komprimovaný, nebo nekomprimovaný archiv programu *tar* (tape archiver). Nekomprimované archivy jsou ale spíše vzácností. Tyto koule jsou základním prvkem pro archivování dat v OS Linuxu a jsou třeba i pro konstrukci RPM balíčků.

Na obrázku č. 4.4 máme pro představu uvedenou stromovou strukturu adresářů s daty, ze kterých se posléze vytvoří **.rpm* balíček.



Obr. 4.4 – Stromová struktura **.rpm* balíčku

Stromová struktura se na rozdíl od **.deb* balíčků nevytváří uživatelem, ale naopak už je vytvořena operačním systémem (či použitým nástrojem) a uživatel do ní musí nakopírovat potřebná data. Celá tato stromová struktura se nachází na této systémové cestě */usr/src/packages*, kde adresář *packages* může být, v případě OS Linux Red Hat, nahrazen adresářem *redhat* (nebo i jinak, záleží také na nástroji, který provede sestavení). Když bychom se zaměřili na jednotlivé podadresáře *packages*, tak jich máme celkem pět a každý z nich plní jinou funkci. Adresář *SPECS* obsahuje speciální soubory, které fungují jako řídící faktory při sestavování balíčku. V adresáři *SOURCES* jsou zdrojové soubory, na základě nichž se vytvoří binární balíček, tedy hotový **.rpm* balíček. Dále máme adresář *BUILD*, kde jsou holé zdrojové soubory a probíhá zde kompilace. Předposlední adresář *RPMS* obsahuje ještě podadresáře, které mají v názvech typ architektury procesoru (např. i386). Do těchto podadresářů se ukládají výsledné **.rpm* soubory. V posledním adresáři *SRPMS* pak budou výsledné **zdrojové** **.rpm* soubory.

Postup, jak ručně vytvořit **.rpm* balíček, je tedy snadný. Předpokládejme, že existuje soubor *OpenControl_1.1.tar.gz* (prostě program OpenControl, akorát zabalený v tarové kouli). Našim úkolem bude nakopírovat tento soubor do adresáře *SOURCES* (viz stromová struktura na obr. 6). Další věcí bude vytvoření speciálního souboru a následného zkopírování do adresáře *SPECS*. Tento soubor musí mít formát *OpenControl_1.1.spec* (důležitá je přípona souboru a dodržení stejného jména základu, jako má tarová koule). Jak vypadá obsah speciálního souboru u **.rpm* balíčku lze vidět v tabulce č. 4.5.

Položky speciálního souboru:	Význam položek:
Summary	Shrnutí o jaký program se jedná (např. kalkulačka)
Name	Název programu
Version	Verze programu
Source	Webová stránka programu
Packager	Jméno vlastníka programu (může být uveden i mail)
BuildRoot	Systémová cesta k hotovému programu
%description	Popis balíku (krátká informace)
%prep	Příprava na kompilaci (vykonává se jako skript)
%build	Vlastní kompilace (také se vykonává jako skript)
%install	Vlastní instalace souborů v balíčku
%files	Popisuje, které soubory mají být zahrnuty do balíčku

Tab. 4.5 – Obsah speciálního souboru **.rpm* balíčku (základní položky)

Když máme správně nastavený speciální soubor a správně rozmístěná data ve stromové struktuře, nezbývá nám než sestavit samotný **.rpm* balíček. To uděláme tak, že spustíme terminálovou konzolu a přemístíme se do adresáře SPECS. Dále spustíme příkaz viz níže.

```
rpmbuild –ba OpenControl_1.1.spec
```

Důležité je neplést si program *rpm* a *rpmbuild*. Pro sestavování balíčků je totiž nutné použít *rpmbuild*, *rpm* slouží už jen ke správě. Parametr *–ba* znamená, že chceme vytvořit binární i zdrojový balíček. Existuje i plno dalších užitečných parametrů, stačí zadat příkaz viz níže (v manuálu se lze pak dočíst, že podle zvoleného parametru můžeme zkonstruovat buď jen samotný binární balíček, či jen zdrojový).

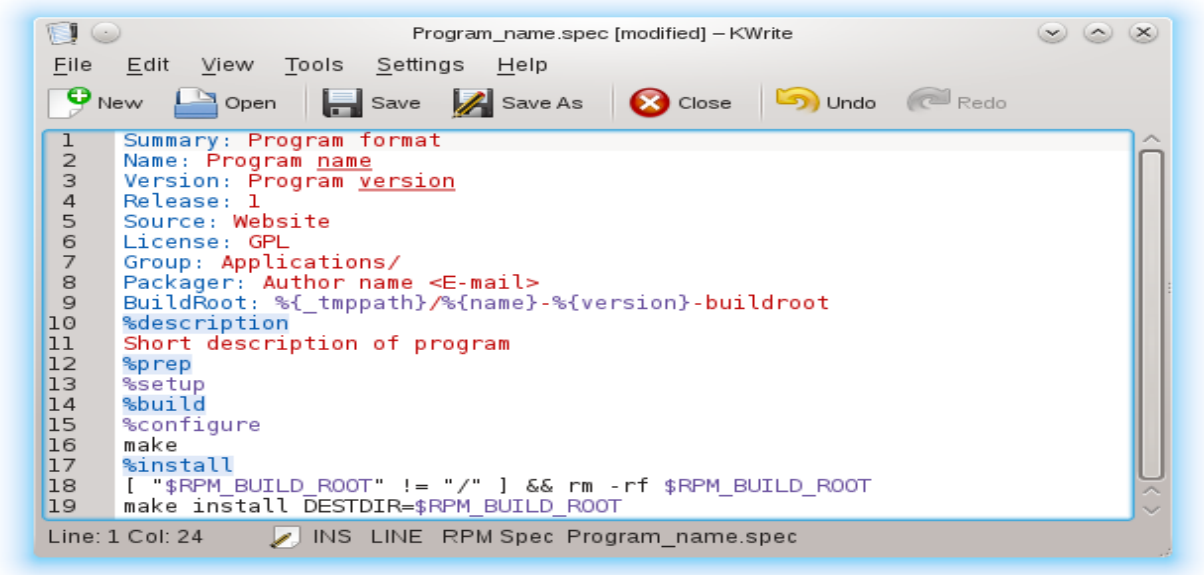
```
man rpmbuild
```

Výsledkem by měly být dva balíčky, jeden zdrojový ve formátu *OpenControl_1.1.src.rpm* (ve složce SOURCES) a druhý binární (ve složce RPMS), určený pro danou architekturu ve formátu například *OpenControl_1.1.i386.rpm* (pro nás ten hlavní).

Je ještě důležité připomenout, že výroba RPM balíčku pro program OpenControl byla provedena na OS Linux za použití distribuce MandrivaLinux (vydání 2010.1) a dále pak na distribuci OpenSUSE (verze 12.1). Postup u každé Linuxové distribuce je mírně odlišný (např. stromová struktura). Níže je uvedena adresářová cesta a struktura pro vytvoření RPM balíčku v OS Linux OpenSUSE (například pro program OpenControl):

```
benny@linux-w6jz:~/rpmbuild> pwd
/home/benny/rpmbuild
benny@linux-w6jz:~/rpmbuild> ls -l
total 24
drwxr-xr-x 3 benny users 4096 May  2 16:21 BUILD
drwxr-xr-x 2 benny users 4096 May  2 14:17 BUILDROOT
drwxr-xr-x 3 benny users 4096 May  2 14:48 RPMS
drwxr-xr-x 3 benny users 4096 May  2 15:57 SOURCES
drwxr-xr-x 2 benny users 4096 May  2 16:39 SPECS
drwxr-xr-x 2 benny users 4096 May  2 18:30 SRPMS
```

Je tedy patrné, že adresářová cesta je v tomto případě trochu odlišná, než je tomu například u OS Linux Red Hat či MandrivaLinux (speciální soubor uveden na obr č. 4.5) [3] [4].



Obr. 4.5 – Příklad speciálního souboru (makra jako *%prep* jsou nastavená implicitně)

4.2.2 Nástroje pro vytváření RPM balíčků

Nástroje pro tvorbu RPM balíčků mají obdobnou ovladatelnost i grafické prostředí, liší se pouze ve své interní činnosti (v tom, jak se chová k datům, jak je zpracovává). Trochu

problémy dělá akorát množství balíčků, které je třeba nainstalovat. Proto je jednodušší vytvářet RPM balíčky ručně nebo použít webové aplikace, které to dokážou, jako je například:

http://www.hanovsolutions.com/easydeb

Jedná se o webový server, který umožňuje vytvořit *.rpm i *.deb balíček bez jakékoliv instalace programů do operačního systému (viz obr. č. 4.6). Uživatelské prostředí je přívětivé, takže není problém s ovladatelností (viz obr. č. 4.7).

Obr. 4.6 – Webová aplikace *Easydeb* pro tvorbu **.deb* a **.rpm* balíčků

Obr. 4.7 – Uživatelské prostředí webové aplikace Easydeb

Když se vrátíme ještě k programům pro tvorbu *.rpm balíčků, které je nutné instalovat do operačního systému, uvedeme si krátký přehled těch, jež byly otestovány. Tento přehled je k dispozici v tabulce č. 4.6. Testování proběhlo na OS Linux MandrivaLinux (vydání 2010.1).

U těchto programů je nutno při případné instalaci počítat s nutností instalace poměrně dosti knihoven (bez některých knihoven se program ani nenainstaluje).

Testované programy:	**Webové stránky programů:**
RPM Package Maker	http://kde-apps.org/content/show.php/RPM+Package+Maker?content=33228
KRPMBuilder	http://krpmbuilder.sourceforge.net/
RPM Creator	http://www.softoxi.com/download-rpm-creator-linux.html
CheckInstall	http://checkinstall.izto.org/download.php

Tab. č. 4.6 – Seznam testovaných programů v OS Linux MandrivaLinux a OS Linux OpenSUSE (aktuálnost webových stránek – 27. 2. 2012)

Lze si všimnout, že u tabulky č. 7 není k dispozici rank, který byl použit u *.deb balíčků. Je to proto, že všechny testované programy, které jsou uvedené v tabulce, byl velký problém nainstalovat a celkově dostat do stavu, kdy byly plně funkční. Proto byly některé knihovny instalovány naráz, tzn. pro ušetření času a zefektivnění práce byl problém řešen globálně. Vzhledem k tomu, ale není možné určit přesný čas strávený nad instalací a následným seznamováním se s uživatelským prostředím daného programu, tím pádem není možné ani určit rank.

Programy měly také problém s jinými Linuxovými distribucemi. Zkoušel jsem instalaci v distribucích MandrivaLinux, Fedora, OpenSUSE a Red Hat. Všechny programy se mi povedlo nainstalovat pouze v distribucích MandrivaLinux a OpenSUSE, proto jsem testy prováděl zde. Ještě bych se zmínil, že při instalování knihoven jsem se několikrát dostal do stavu tzv. dependency hell. Jedná se o stav, kdy jedna knihovna je závislá na druhé, tudíž po nainstalování knihovny instalujete další a další, až se dostanete zase k té první. Takový stav se pak může opakovat do nekonečna. Zde byl nápomocný nástroj urpmi, který dokáže nainstalovat všechny tyto knihovny naráz a vyhnout se tak dependency hell [12].

5 Distribuční balíčky pro OS MS Windows

Operační systém MS Windows obsahuje instalační službu Microsoft Windows Installer (implicitně v produktech MS Windows 2000 a novějších). Tato služba slouží k instalaci a správě instalačních balíčků ve formátu *.msi. V současné době jsou tyto balíčky velmi populární. Dokazuje to i fakt, že většina softwaru je distribuovaná právě pomocí MSI balíčků (např. známý MS Office). Vzhledem k těmto, výše uvedeným faktorům, jsou balíčky MSI nejideálnějším řešením pro problém distribuce v operačních systémech s MS Windows.

Instalační balíček MSI je obvykle distribuován jakožto jeden soubor (*.msi). Může se ale stát, že souborů bude více. V tabulce č. 5.1 lze vidět tyto soubory i se stručným komentářem.

Formát souboru:	Komentář k souboru:
MSI (*.msi)	Instalační balíček
	Umožňuje přehlednou instalaci softwaru
MSP (*.msp)	Opravný balíček
	Slouží pro distribuci oprav již nainstalovaných produktů
MST (*.mst)	Transformační soubor
	Definuje chování instalátoru (vhodné pro bezobslužnou instalaci)
MSM (*.msm)	Merge modul
	Umožňuje připojit instalaci nějaké komponenty do MSI balíčku
MSU (*.msu)	Aktualizační balíček
	Slouží k nainstalování aktualizací (přidruženo k Windows Update)

Tab. 5.1 – přehled souborů přidružených k MSI balíčku

Pro vytvoření MSI balíčku neexistuje nějaký obecný postup jako tomu je u DEB či RPM balíčků v OS Linux. Pokud chceme vytvořit MSI balíček, budeme na to potřebovat nějaký nástroj. Těchto nástrojů je ale mnoho a ne všechny vyhovují našim představám. Například Microsoft na svých oficiálních stránkách píše, že pro vytvoření MSI balíčku je třeba nástroj Veritas Software Console. Tento nástroj je ale poměrně zastaralý a vzhledem ke konkurenci i dost nedostačující.

Jak je uvedeno výše, nástrojů pro tvorbu MSI balíčků je mnoho, a proto bylo třeba udělat průzkum, který by jednotlivé nástroje porovnal. Na základě testování pak bude možné vybrat nástroj takový, který bude nejvhodnější jak pro případné uživatele, tak pro náš záměr, tedy bezobslužné sestavení MSI balíčku (pro testovací program OpenControl) [5].

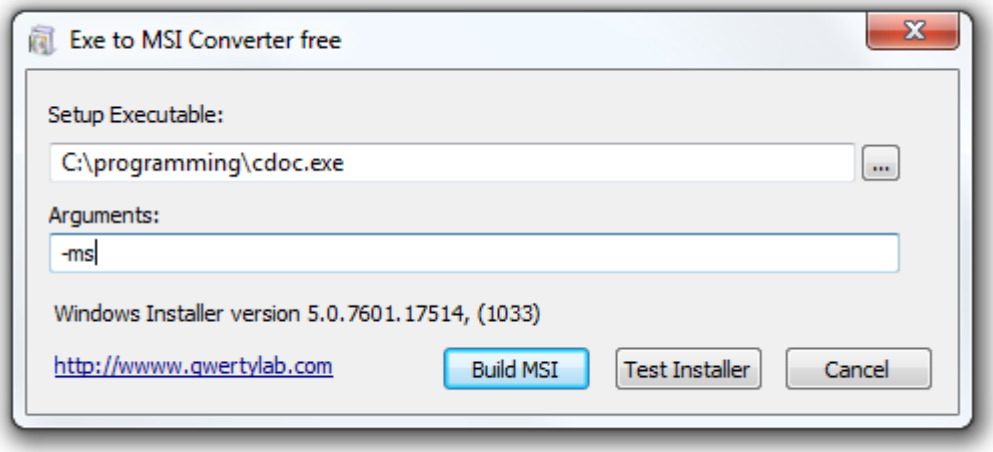
5.1 Nástroje pro vytváření MSI balíčků

Pro testování jsem se snažil vybrat nástroje, které jsou hodně odlišné, co se funkcionality týče. Dále pak všechny vybrané nástroje jsou ve formě freeware. K tomuto kroku jsem přistoupil, jelikož například u shareware formy bych nemohl otestovat veškeré funkce, které nástroj nabízí a tím pádem ho ani korektně ohodnotit. Důležité je, se také zmínit, že všechny nástroje byly testovány jak na MS Windows 7, tak i na starší verzi MS Windows XP.

Níže budu uvádět jednotlivé testované nástroje s příslušným komentářem a snímkem obrazovky (angl. screenshot). Na konci pak bude shrnutí, které bude mimo jiné obsahovat tabulku s hodnocením. Na základě tohoto hodnocení bude zvolen nástroj pro praktickou část.

5.1.1 Exe to MSI Converter

Nástroj Exe to MSI Converter lze jednoduše nainstalovat i s ním posléze pracovat. Je velmi jednoduchý, jediné co lze zadat je vstupní soubor ve formátu *.exe a dále pak argumenty, které mohou pozměnit nastavení výsledného MSI balíčku. Kromě samotného sestavení MSI balíčku nabízí tento nástroj ještě možnost jeho otestování. Na obrázku č. 1 lze vidět pracovní okno nástroje.

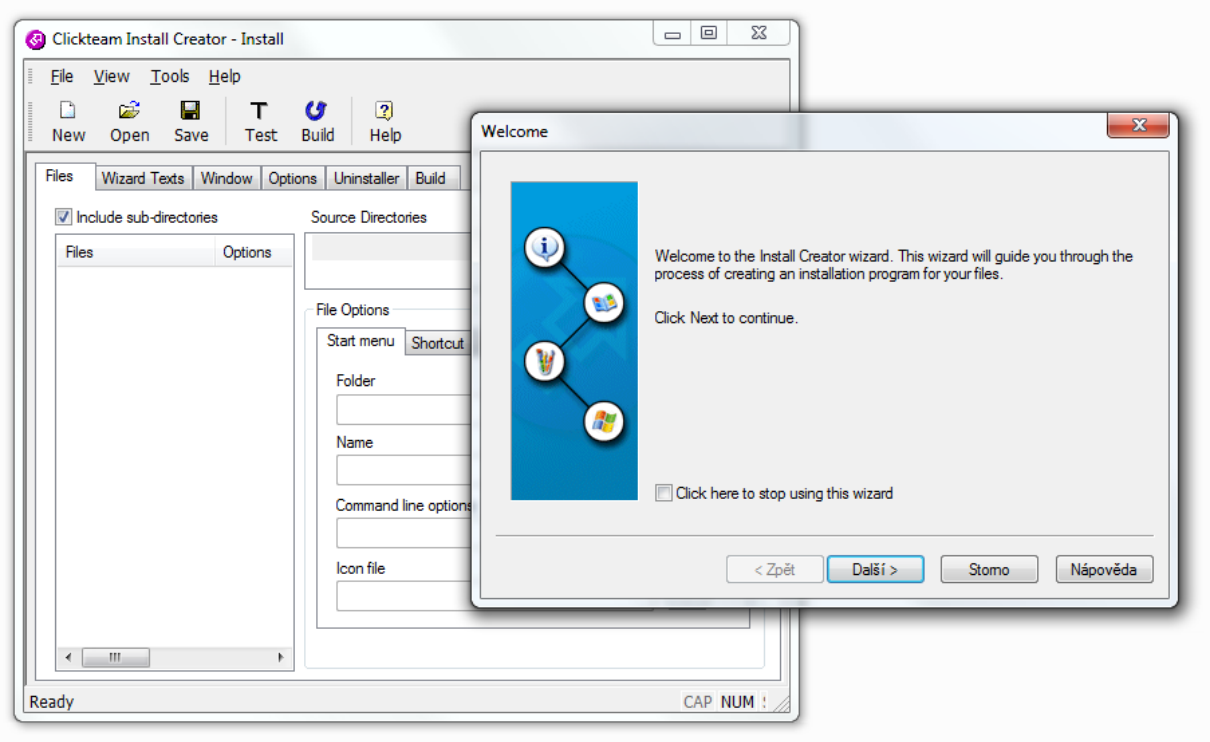


Obr. 5.1 – Nástroj Exe to MSI Converter

Pokud bych měl tento Exe to MSI Converter shrnout, jedná se nástroj pro nenáročné uživatele, kteří chtějí snadno a rychle vytvořit základní MSI balíček.

5.1.2 Clickteam Install Creator

Instalace nástroje Clickteam Install Creator je opět jednoduchá. Prostředí nástroje nám nabízí velkou škálu možností. Uživatel určitě ocení možnost „dvojího“ rozhraní. Jedno rozhraní je ve formě průvodce a dovoluje tak uživateli vytvářet MSI balíček krok po kroku. Druhé rozhraní je klasické, kdy je na výběr několik oken pro danou operaci. Na obrázku č. 5.2 můžeme vidět na levé straně klasické rozhraní a na straně pravé rozhraní ve formě průvodce. Nástroj disponuje funkcemi, které dokáží vytvořit profesionální MSI balíček (lze přidat grafiku, odkazy, licenční ujednání, logo a další).

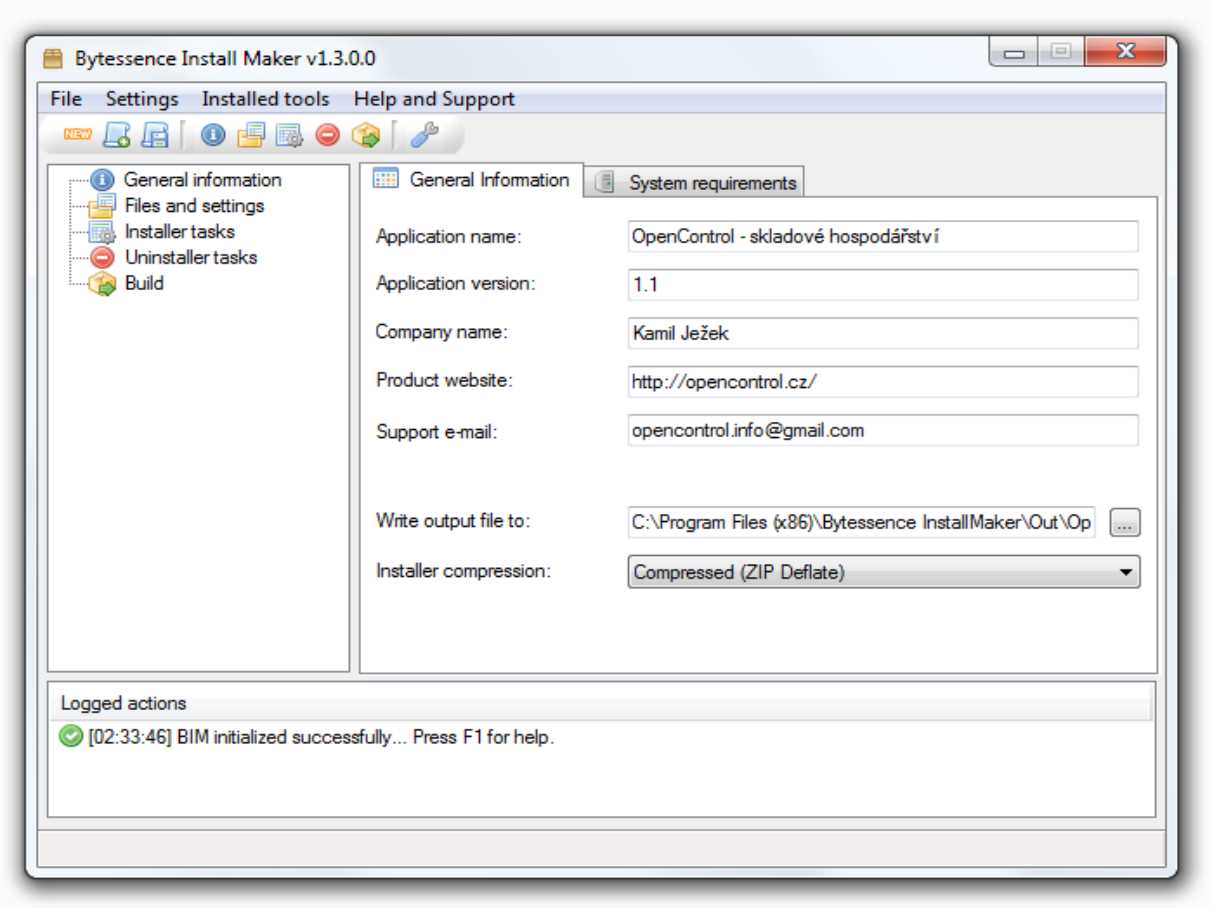


Obr. 5.2 – Nástroj Clickteam Install Creator

Shrnutím, jedná se o nástroj určený pro uživatele, kteří chtějí vytvářet profesionální MSI balíčky, jednoduše a zároveň kvalitně. Vzhledem k rozhraní, které je ve formě průvodce, lze vytvořit výstupní soubor i za poměrně krátký časový úsek.

5.1.3 Bytessence Install Maker

Jako další testovaný nástroj byl Bytessence Install Maker. Instalace opět proběhla bez problémů a prostředí nástroje vypadá dost přehledně. Co se týče funkcionality nástroje, tak je zde kladen velký důraz na možnost přizpůsobení si výsledného MSI balíčku přesně podle uživatelovy představy. Je možné tedy použít grafiku, upravit text, přidat licenční ujednání, přidat vytvoření ikony na ploše či zapsat požadovanou hodnotu do registrů. Na obrázku č. 5.3 je možné vidět pracovní okno nástroje, při nastavování hlavních informací pro výstupní MSI balíček.

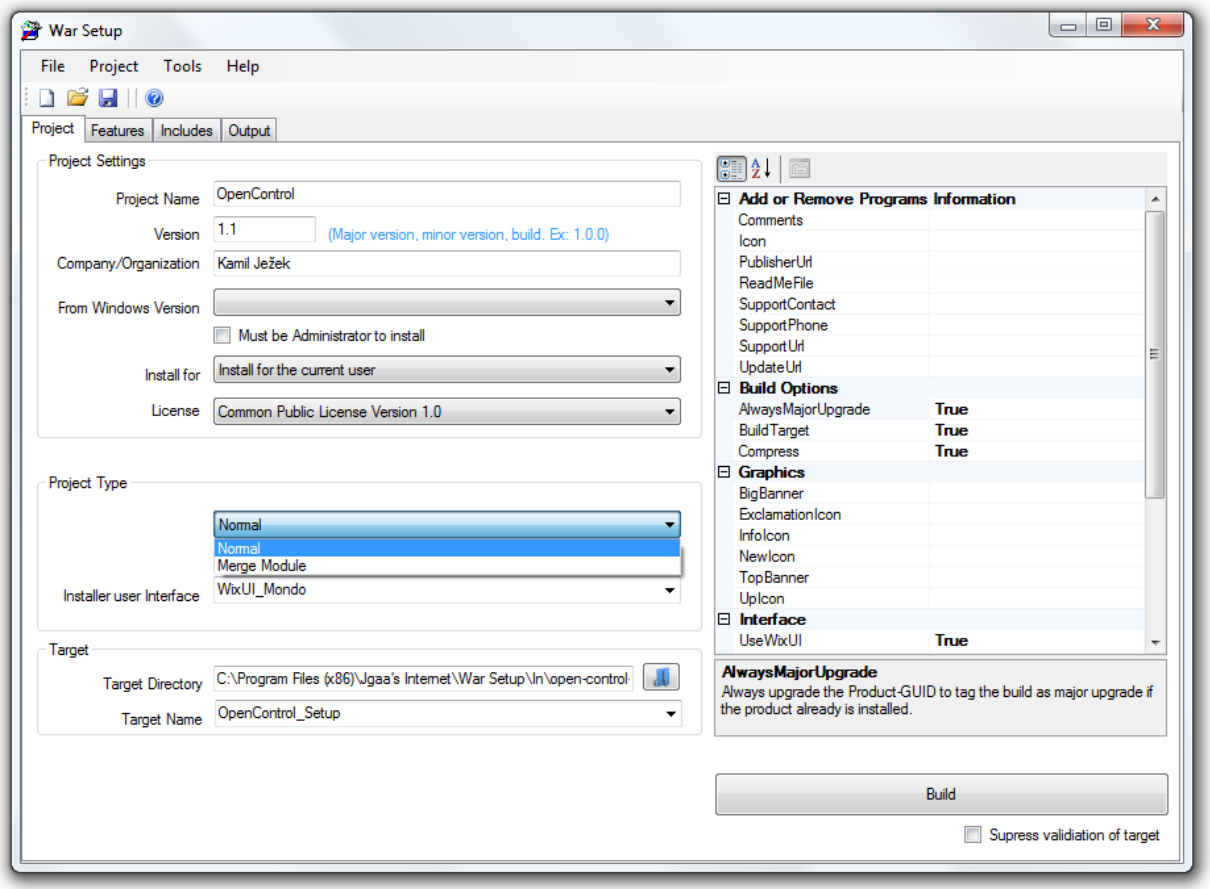


Obr. 5.3 – Nástroj Bytessence Install Maker

Nástroj je velmi komplexní a je určen uživatelům, kteří chtějí připravit výsledný MSI balíček do detailu přesně dle jejich představ. Prostředí je opět přehledné a práce s nástrojem je vcelku jednoduchá. Hodit se může i možnost kontroly požadavků pro instalaci (př. verze OS)

5.1.4 War Setup

Nástroj War Setup má opět standardní jednoduchou instalaci, která není nijak komplikovaná. Grafické rozhraní má přehledné a není problém s ním vytvářet kvalitní výstupní MSI balíčky. Nástroj dovoluje kromě MSI balíčků vytvářet i merge moduly. Dále je zde možnost přidání WiX modulu (sestavení MSI balíčků z XML dokumentu). Výhodná je i provázanost s nástrojem MS Visual Studio. Na obrázku č. 5.4 je zobrazeno pracovní okno nástroje při nastavování projektu.



Obr. 5.4 – Nástroj War Setup

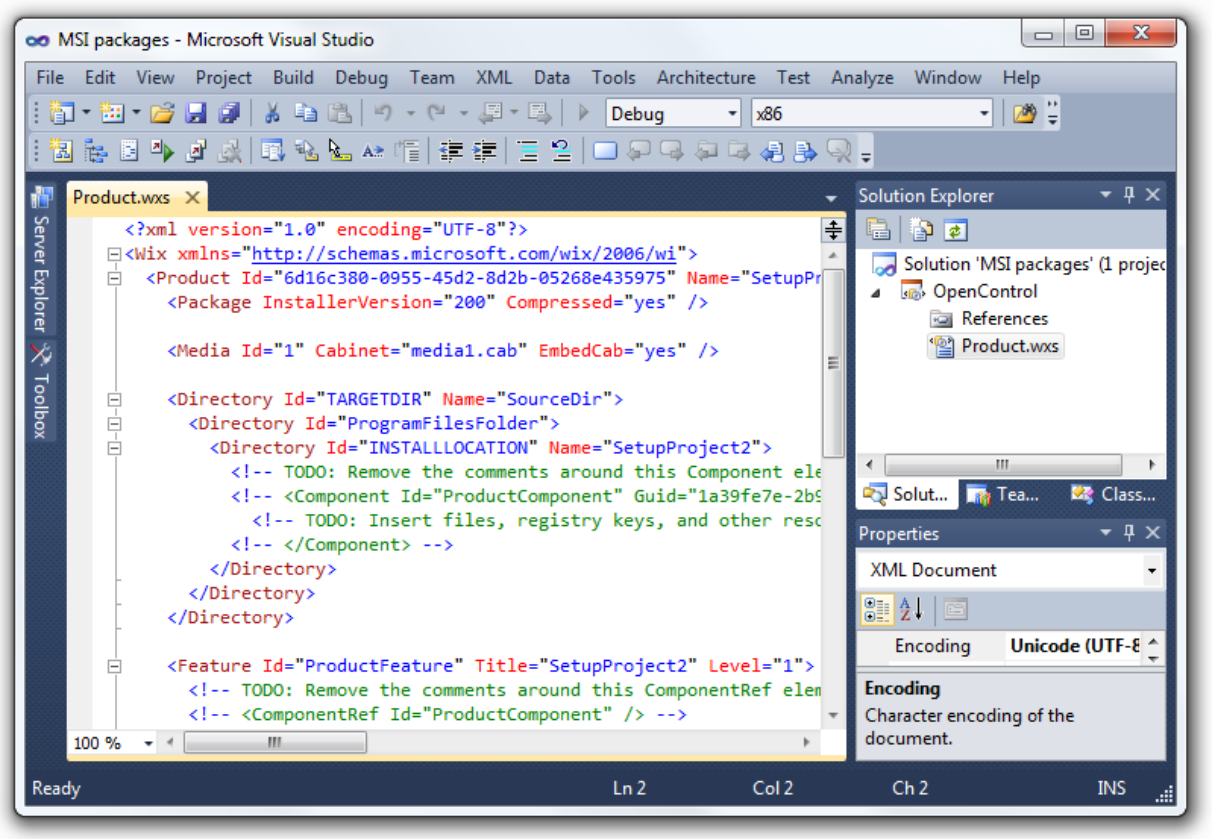
Nástroj War Setup je určen uživatelům, kteří mají širší znalosti s vytvářením MSI balíčků. Nabízí totiž pokročilé funkce, které pro vytvoření běžného MSI balíčku nejsou potřeba. Ideálně by tento nástroj byl vhodný pro uživatele s předchozími zkušenostmi s nástrojem Windows Installer XML (WiX).

5.1.4 Windows Installer XML toolset (WiX)

Nástroj WiX je open-source software od firmy Microsoft. Instalace je trochu složitější, nejprve je třeba stáhnout vlastní WiX, který obsahuje jednotlivé moduly. Jedná se o tyto moduly:

- Candle – kompilátor určený pro vstupní XML dokument
- Dark – dekompilátor, který převede MSI balíček zpět na XML dokument
- Light – linker se stará o správné sestavení výsledního MSI balíčku z binárních souborů
- Lit – knihovna (jedná se o volitelný modul)
- Tallow – generátor WiX kódu obsahující adresářovou strukturu

V další fázi je dobré použít nějaký nástroj, ve kterém budeme psát XML kódy. Pro toto se skvěle hodí nástroj MS Visual Studio (ovšem je třeba doinstalovat podporu pro WiX). Pokud máme vše výše uvedené splněno, můžeme založit nový XML dokument přímo pro WiX. Na obrázku 5.5 je uveden základní XML kód od WiXu.



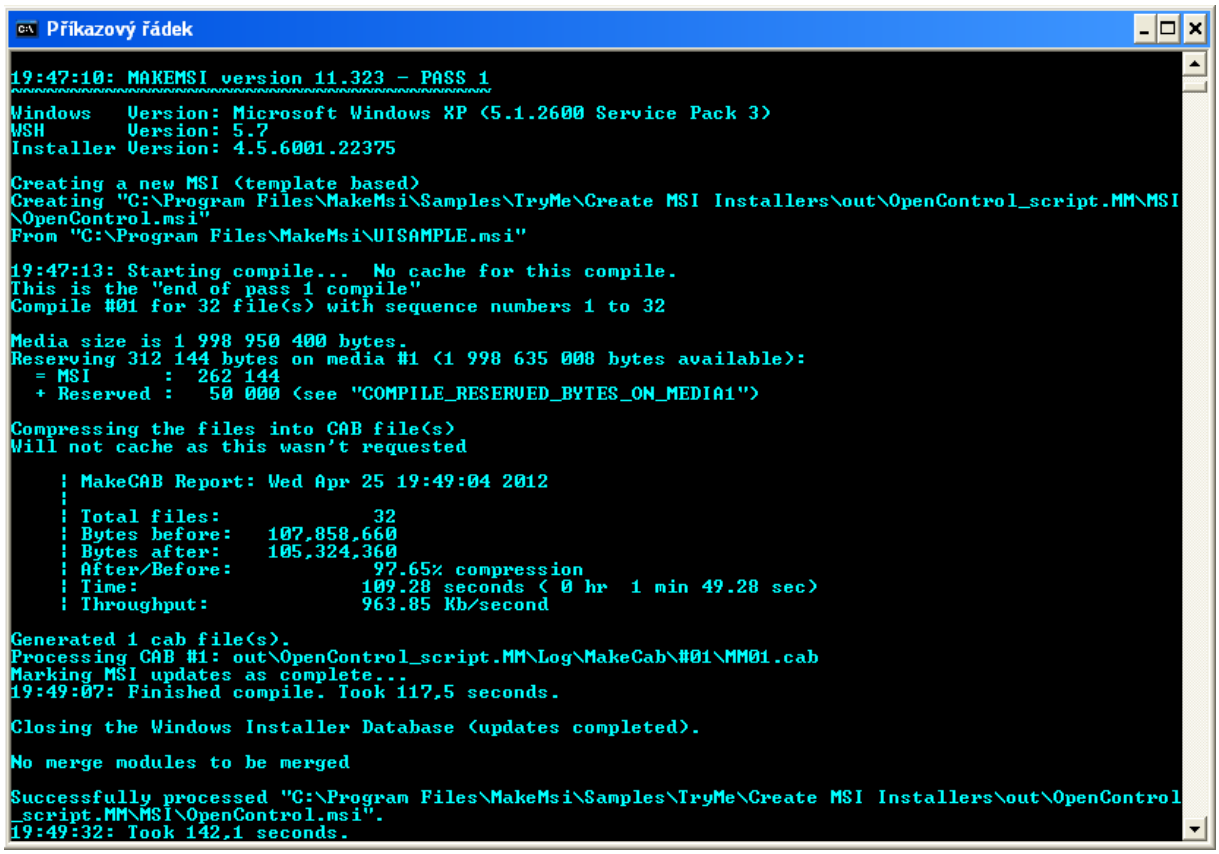
Obr. 5.5 – XML kód nástroje WiX (spuštěno na MS Visual Studio)

Co se ovládání nástroje WiX týče, je třeba se naučit alespoň základy XML. Veškeré nastavení pro konstrukci výsledného MSI balíčku není nikde pevně určeno. Je tedy teoreticky možné zvolit jakékoliv nastavení – vše závisí jen na přidání určitého úseku kódu v XML.

Nástroj WiX je na internetu dost známý. Není ale moc vhodný pro uživatele vyžadující co nejjednodušší a nejpřehlednější software. Nástroj je vhodný spíše pro programátory, kteří chtějí vytvořit co nejkvalitnější a plnohodnotný MSI balíček pro daný produkt.

5.1.5 MakeMsi

Posledním testovaným nástrojem byl opět open-source software MakeMsi. Instalace není nijak složitá ani časově náročná. Problém ale začíná při práci s tímto nástrojem. Princip je podobný jako u nástroje WiX s tím rozdílem, že kromě XML dokumentů se zde musí vytvořit speciální skripty, jejichž syntaxi určuje sám autor. Po zkonstruování skriptů je třeba vytvořit výsledný balíček MSI – to se provádí pomocí příkazové řádky (viz obr. č. 5.6).



Obr. 5.6 – Nástroj MakeMsi při sestavování MSI balíčku

Prostředí nástroje MakeMsi nám umožní vytvořit MSI balíčky, které mohou být použity jako profesionální distribuce pro daný produkt. Svými vlastnostmi dovoluje uživateli sestavit MSI balíček bezobslužně – uživatel po zadání příkazu nemusí již nijak komunikovat s nástrojem. Veškerý software pro sestavení je automaticky přidán a nakonfigurován při instalaci nástroje MakeMsi, takže není třeba žádných přídavných nástrojů.

MakeMsi v sobě zahrnuje vše, co uživatel potřebuje pro vytvoření výsledného MSI balíčku. Veškerá modifikace instalačního průvodce se dělá pomocí skriptů, takže lze přizpůsobit obsah přesně tak, aby vyhovoval představám uživatele. Syntaxe skriptů je ale složitější a tak bude vytvoření rozsáhlejších skriptů časově náročné.

5.1.6 Shrnutí testovaných nástrojů

Po otestování všech výše uvedených nástrojů jsem sestavil tabulku č. 5.2, kde jsem uvedl hodnocení. Hodnocení (rank) jsem určil podle tří faktorů – složitost instalace, složitost práce s nástrojem a funkcionalita, přičemž největší procento jsem přidával právě funkcionalitě nástrojů.

Testované nástroje	Webové stránky nástrojů	Rank
Exe to MSI Converter	http://www.qwertylab.com/	60 %
Clickteam Install Creator	http://www.clickteam.com/	85 %
Bytessence Install Maker	http://www.bytessence.com/	75 %
War Setup	http://warsetup.jgaa.com/	70 %
Windows Installer XML	http://wix.sourceforge.net/	80 %
MakeMsi	http://dennisbareis.com/	90 %

Tab. 5.2 – Seznam testovaných nástrojů v OS MS Windows 7 a MS Windows XP

Jak lze vidět z tabulky č. 2, nejvyšší rank má nástroj MakeMsi, který je ve formě open-source. Jeho funkcionalita je nejlepší ze všech uvedených nástrojů a jediná nevýhoda spočívá ve složitosti syntaxe skriptů. Z nástrojů, které nejsou ve formě open-source má nejvyšší rank Clickteam Install Creator. Opět má vynikající funkcionalitu, ale i přívětivé pracovní prostředí, které podstatně ulehčuje vytváření MSI balíčků.

Vzhledem k těmto výsledkům jsem se rozhodl, že pro praktickou část použiji nástroj MakeMsi. Splňuje totiž hlavní požadavek – žádná interakce uživatele s nástrojem při samotném průběhu sestavování MSI balíčku. Dalším důvodem je dosažení nejvyššího ranku při testování a výše popisovaná funkcionalita nástroje.

6 Realizace distribuční formy

Tato kapitola se zabývá realizací skriptů pro jednotlivé distribuční balíčky. Jsou zde uvedeny skripty pro vytváření DEB/RPM balíčků určené pro OS Linux a MSI balíčky pro OS MS Windows. Hlavním požadavkem pro realizaci je řešení, které umožní vytvoření výsledného balíčku bez zásahu uživatele do vlastního průběhu vytváření. Zaobírám se zde tedy nástroji, které lze spouštět z příkazové řádky. Veškeré podrobnosti jsou uvedeny přímo v podkapitolách**.**

6.1 Nástroj DPKG-DEB (ALIEN) – realizace skriptů pro vytváření DEB/RPM balíčků

V této realizační části nejdříve uvedu jaký je můj záměr, popíši nástroje, které jsem při vytváření výsledného skriptu použil a uvedu i některé úseky kódů, které jsou ve skriptu obsaženy.

6.1.1 Zadání úlohy

Realizační část je zaměřená na vytváření DEB a RPM balíčků. Hlavním úkolem této části je tedy zkonstruování skriptu (skriptů), který vytvoří DEB či RPM balíček. Podmínkou je žádná interakce s uživatelem během samotného sestavování. Další podmínkou je přidání libovolných komponent do běhu instalace. Důležité také je, aby výsledný skript byl univerzální, tedy aby fungoval pro jakýkoliv program.

6.1.2 Analýza úlohy

Původní plán byl vytvoření dvou skriptů. První měl sloužit k vytváření DEB balíčků a spouštěl by se z OS Linux Ubuntu. Druhý by pak sloužil k vytváření RPM balíčků a spouštěl by se z OS Linux OpenSUSE.

Zkonstruoval jsem tedy první skript, který za pomoci nástroje DPKG-DEB sloužil k vytváření DEB balíčků a pokračoval jsem ve druhém – vytváření RPM balíčků. Zde ale nastal problém, jelikož u každé distribuce operačního systému Linux byla trochu odlišná adresářová struktura (jak je popisováno konstrukce RPM balíčků). Zhotovený skript by tedy fungoval pouze na dané distribuci a nebyl by přenositelný. Proto jsem přistoupil k jinému řešení. Objevil jsem nástroj ALIEN (konvertor), který umí převést DEB balíček na RPM. Napadlo mě tedy tento nástroj zabudovat do prvního skriptu a přidat tak kromě DEB balíčku i možnost vytváření RPM balíčku. Jelikož testování nástroje ALIEN proběhlo úspěšně, přistoupil jsem k tomuto řešení a vytvořil tak skript, který umí vytvářet oba formáty balíčků. Skript je navíc přenositelný, takže lze použít na jakékoliv distribuci OS Linux, kde jsou nainstalovány nástroje DPKG-DEB a ALIEN. Díky tomuto řešení jsem také ulehčil uživateli ovládání, protože pro konstrukci DEB, či RPM balíčku stačí vyplnit jen kontrolní soubor (tedy o speciální soubor se starat nemusíme). Jediná nevýhoda použitého řešení je v času běhu skriptu, protože je logické, že pro zkonstruování RPM balíčku bude třeba vytvořit i balíček DEB. Časová ztráta je ale přijatelná (viz testování doby běhu v závěru).

6.1.3 Popis implementace

Výsledný skript je napsán ve skriptovacím jazyce BASH (Bourne again shell). Tento jazyk jsem zvolil kvůli tomu, že je již v základu obsažen v každé Linuxové distribuci a jedná se o standard POSIXu (je tedy přenositelný). Ke skriptu jsem přiložil i kontrolní soubor pro vytvoření DEB balíčku (implicitně se vytvářejí binární DEB/RPM balíčky – lze změnit).

BuildDEB&RPM_script.sh

Tento skript očekává data ve vstupním adresáři, z nichž pak provede sestavení do DEB balíčku a to za pomoci přiloženého kontrolního souboru. Pokud uživatel zadá požadavek pro vytvoření RPM balíčku, je pak proveden převod z DEB právě do RPM balíčku. Výsledné data jsou nám pak k dispozici ve výstupním adresáři.

Jako první jsou ve skriptu definována jména komponent (jedná se pouze o komponentu JAVA, ovšem skript je konstruován tak, aby přidání dalších komponent nečinil žádné problémy). V další fázi skript kontroluje, jestli je zadán správný formát parametrů a v případě, že ano, tak nastaví slovo zvolené v parametru, jako název programu. Zde bych chtěl poukázat na konstrukci příkazu echo (konstrukce spočívá ve výpisu textu v červené barvě), viz níže:

```
echo -e "\033[31mError - Wrong format of arguments!\033[0m"
```

V další fázi se kontroluje výběr daného balíčku. Pokud uživatel nezadá správné parametry (viz uživatelská příručka), bude vypsáno chybové hlášení a skript se ukončí. Dále probíhá kontrola, jestli je přiložen kontrolní soubor (control), bez něj by nebylo možné DEB balíček sestavit, takže v případě, že není přiložen, skript se opět ukončí. Vstupní data jsou tedy zkontrolována a nyní je třeba vytvořit výstupní složku (může už být vytvořená, skript kontroluje i tuto možnost). V další fázi proběhne kontrola výstupních dat. Tato kontrola je ve skriptu implementována z důvodu, aby nedošlo k přepsání výsledného DEB/RPM balíčku či aby se nepoškodila dočasná data (dočasná adresářová struktura pro DEB balíček). Pokud je vše výše uvedené v pořádku, skript začne generovat adresářovou strukturu pro DEB balíček a zkopíruje do ní vstupní data, což lze vidět níže:

```
cp  -r in/$PROGRAM_NAME/* out/$PROGRAM_NAME/$PROGRAM_NAME/
mkdir out/$PROGRAM_NAME/DEBIAN/
chmod 666 control
cp control out/$PROGRAM_NAME/DEBIAN/
```

Hned pod touto konstrukcí je další, která umožňuje do adresářové struktury přidat i soubory jako:

- copyright (licenční ujednání – nutno uvést typ licence např. GPL-2)

- prerm (obsahuje příkazy pro odstranění vstupních souborů)
- postinst (obsahuje příkazy, které definují chování balíčku po instalaci)
- md5sum (ověření, zda byly data správně přeneseny – pomocí 128 bitové šifry)

Tyto soubory nejsou k sestavení DEB balíčku třeba, avšak uživatel je může použít (soubory fungují na stejném principu jako kontrolní soubor, tudíž pro ně platí i stejná pravidla, např. v tom kam se mají vložit).

V dalším kroku je konstrukce, která v případě zadání správného parametru nainstaluje spolu s program obsaženým v balíčku i komponentu – JAVA (tato konstrukce je připravena tak, aby šla jednoduše rozšířit i o další komponenty). Pro komponentu JAVA jsem použil ruční instalaci (kterou zpracuje skript) => žádná interakce s uživatelem. Vše potřebné pro sestavení DEB balíčku je připraveno a proto se může zavolat nástroj, který provede potřebné operace (opět uvádím, viz níže).

```
dpkg-deb -b out/$PROGRAM_NAME out/$PROGRAM_NAME.deb
```

Pokud sestavení proběhne úspěšně, tak se nám ve výstupní složce vytvoří DEB balíček. Skript ale pokračuje dál a to až k podmínce, která v případě, že je zvolen parametr pro RPM, sestaví RPM balíček a DEB balíček odstraní (viz níže).

```
alien -r --scripts $PROGRAM_NAME.deb
rm -f $PROGRAM_NAME.deb
```

K sestavení je použit právě nástroj ALIEN, který pouze převede DEB balíček na RPM [2].

6.1.4 Uživatelská příručka

Pro korektní průběh skriptu je nejdříve třeba vytvořit správné uspořádání souborů. Adresář, ve kterém budeme chtít spustit skript, by měl tedy obsahovat tyto položky:

- in (adresář, určený pro vstupní data)
- out (adresář, který po sestavení bude obsahovat výstupní data)
- control (kontrolní soubor pro DEB balíček)
- DEBRPM_Builder_script.sh (soubor s vlastním skriptem)

Do adresáře in je třeba vložit další adresář, který bude obsahovat data (program) pro sestavení výsledného balíčku. Adresář s těmito daty (programem) musí být pojmenovaný stejně jako parametr, který budeme zadávat při spuštění skriptu (formát parametrů viz dále). V závislosti na vstupních datech je pak třeba upravit přiložený kontrolní soubor.

Nyní již přejdeme k samotnému spuštění skriptu. Jelikož skript obsahuje nástroje, které nejsou implicitně obsažený v OS Linux, tak je třeba tyto nástroje nainstalovat. Pro nástroj DPKG-DEB poslouží příkaz níže:

```
sudo apt-get install dpkg
```

A pro nástroj ALIEN použijeme následující příkaz:

```
sudo apt-get install alien
```

V terminále se pak pomocí příkazu cd přemístíme do adresáře se souborem DEBRPM_Builder_script.sh a zadáme následující příkaz (**pozor** musí být uveden i příkaz sudo, průběh jinak nebude korektní):

```
sudo ./DEBRPM_Builder_script.sh OpenControl -rpm --java
```

Tím dosáhneme vytvoření RPM balíčku pro program OpenControl, s tím, že při instalaci tohoto balíčku se nainstaluje i komponenta JAVA. Formát zadávání parametrů je totiž následující:

```
<Název_programu> <Formát_balíčku> [<Název_komponenty>]
```

Název programu může být jakýkoliv krom názvů parametrů (kvůli kontrole pro uživatele, je totiž možné že se uživatel splete a použije nějaký parametr na nesprávném místě). Formát balíčku lze vybrat z parametrů -deb nebo –rpm. Název komponenty JAVA je pak nepovinný parametr a zadává se buď jako –j nebo --java. Když se podíváme na obrázek č. 6.1, můžeme vidět, jak skript reaguje na výše uvedený příkaz.

```
benny@ubuntu:~/Desktop/BPINI$ sudo ./DEBRPM_Builder_script.sh OpenControl -rpm -j
====================================
Building DEB/RPM package for program: OpenControl
====================================
Program name set to 'OpenControl' - OK
Package format set to RPM (*.rpm) - OK
Input data available: 'in/JAVA'- OK
Input data available: 'in/OpenControl'- OK
Control file found - OK
Copying...
Files successfully copied - OK
Adding component...
Component 'JAVA' added - OK
Starting tool 'DPKG-DEB'...
dpkg-deb: building package `opencontrol' in `out/OpenControl.deb'.
DEB package created - OK
Converting...
OpenControl-1.1-2.i386.rpm generated
Successfully completed
====================
```

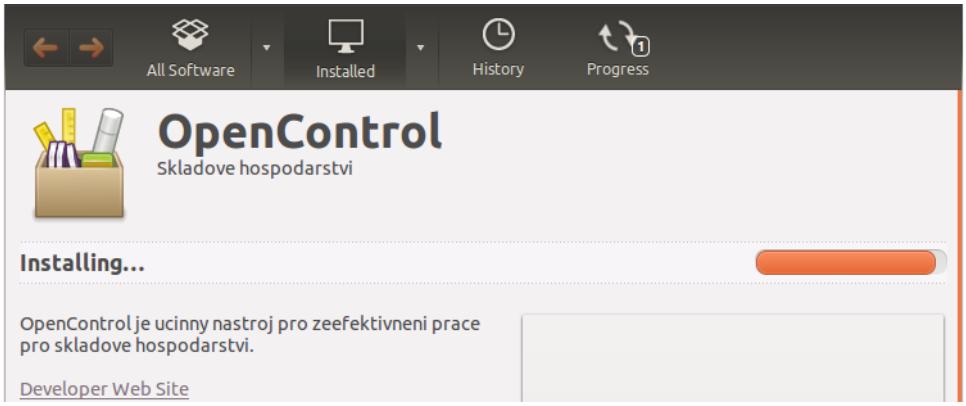
Obr. 6.1 – Průběh skriptu DEBRPM_Builder_script

U každé sekce kódu jsou přidány výpisy hlášek, takže uživatel má po celou dobu běhu skriptu přehled o tom, co se zrovna vykonává. Je ale třeba vědět, co je za problém, když se uživateli zobrazí nějaká chybová hláška. Proto je viz níže uvedena tabulka, která tyto hlášky konkretizuje (tabulka č. 6.1).

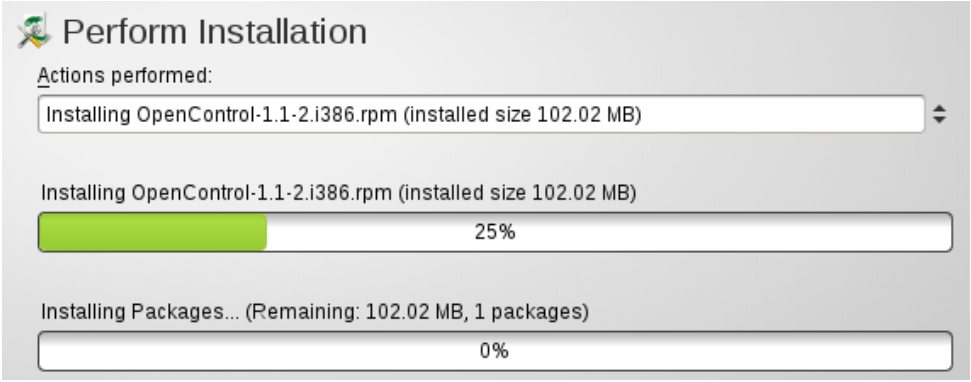
Chybová hláška:	**Možné řešení:**
Error - Wrong format of arguments!	Zadejte správný formát parametrů
Error - Input data not available!	Vložte data do vstupního adresáře
Error - Control file not found!	Přiložte kontrolní soubor
Error - Folders detected (output)!	Odstraňte adresáře z výstupního adresáře
Error - Files detected (output)!	Odstraňte soubory z výstupního adresáře

Tab. 6.1 – Chybová hlášení skriptu

Vytvořený DEB/RPM balíček pak není problém korektně nainstalovat, viz obr. č. 6.2 a 6.3.



Obr. 6.2 – Průběh instalace DEB balíčku obsahující program OpenControl



Obr. 6.3 – Průběh instalace RPM balíčku obsahující program OpenControl

6.1.5 Závěr

Výsledek realizační části splňuje zadání – vytvořený skript korektně vytváří DEB či RPM balíčky a to bez interakce s uživatelem. Skript je plně univerzální, takže funguje na jakémkoliv programu - záleží jen na správném nastavení kontrolního souboru, či souboru prerm, postinst atd. Skript také není problém spustit na většině Linuxových distribucí, takže je přenositelný.

Doba běhu skriptu je uvedena v tabulce č. 6.2. Měřena byla při sestavování DEB/RPM balíčku a dále pak při použití parametru pro přidání komponenty JAVA. Lze vidět, že se nejedná o nějak velkou časovou ztrátu (zvláště vzhledem k dnešním moderním systémům). Je také logické, že doba běhu závisí na vstupních datech, zde byly vstupními daty program OpenControl.

Použité parametry:	**Doba běhu:**			**Průměrná doba běhu:**
	1. sestavení:	*2. sestavení:*	*3. sestavení:*	
-deb	13,42 s	10,67 s	17,28 s	13,79 s
-rpm	26,07 s	21,61 s	31,37 s	26,35 s
-deb --java	55,11 s	49,91 s	59,07 s	54,69 s
-rpm --java	1 min 54,88 s	1 min 43,03 s	1 min 57,25 s	1 min 48,72 s

Tab. 6.2 – Doba běhu skriptu DEBRPM_Builder_Script

Skript byl vytvářen pomocí nástroje SciTE v operačním systému Linux Ubuntu. Veškeré distribuce Linuxu, se kterými jsem se díky této práce setkal byly nainstalovány ve virtualizačním softwaru pomocí nástroje VMware Player. V průběhu testování a seznamování se s nástroji pro tvorbu DEB či RPM balíčků jsem tak pracoval s distribucemi jako Fedora, MandrivaLinux, OpenSUSE, Red Hat, Debian a Ubuntu. Krom nabytých znalostí s DEB/RPM balíčky jsem tedy získal i základní přehled o Linuxových distribucích.

Závěrem bych se ještě zmínil o souboru Debian Policy Manual, což je seznam „pravidel“ pro vytváření DEB balíčků. Je zde například uvedeno jaké všechny položky můžeme použít v kontrolním souboru a co znamenají (nejen v kontrolním souboru). Dále jsou tu také dobře popsány soubory, potřebné pro zdrojové balíčky (použijeme, pokud chceme DEB balíček sestavit ze zdrojových kódů nějakého programu). Webové stránky jsou uvedeny níže:

http://www.debian.org/doc/debian-policy/

6.2 Nástroj MakeMsi – realizace skriptů pro vytváření MSI balíčků

V této realizační části nejdříve uvedu jaký je můj záměr, popíši detailněji nástroj, ve kterém jsem výslednou práci vytvořil a závěrem pak uvedu některé úseky kódů, které jsou obsaženy ve výsledných skriptech.

6.2.1 Zadání úlohy

Realizační část je zaměřená na vytváření MSI balíčku pro testovací produkt – program OpenControl. Hlavní podmínkou je, aby během vlastního průběhu sestavení MSI balíčku uživatel nemusel nijak interagovat s nástrojem. Další podmínkou je přidání libovolných komponent do instalátoru tak, aby byly instalovány spolu s programem. V našem konkrétním případě se bude jednat o komponentu Java SDK. Samozřejmostí je vytvoření takových postupů, které se nebudou vázat jen na program OpenControl samotný, ale budou univerzální, tzn. v případě potřeby by měly jít lehce modifikovat na jakýkoliv jiný program.

6.2.2 Analýza úlohy

Jak již nadpis napovídá, pro realizaci úlohy jsem se rozhodl použít nástroj MakeMsi. Důvody, proč jsem přistoupil právě k tomuto nástroji, je několik. Dle požadovaného zadání jsem měl v podstatě možnost vybrat pouze z nástrojů ve formě open-source. Krom testovaných nástrojů MakeMsi a WiX jsem vybíral ještě z nástroje Inno Setup. Všechny jsem důkladně prostudoval a zjistil, že právě nástroj MakeMsi má nejlepší funkcionalitu a tudíž se i nejlépe hodí pro tuto úlohu. Dalším důvodem je možnost sestavení MSI balíčku z příkazové řádky a i možnost nezasahování do vlastního průběhu vytváření MSI balíčku (žádná interakce uživatele s nástrojem).

Řešení úlohy pomocí tohoto nástroje znamenalo detailně se seznámit s připravenými skripty a po pochopení pak sestavit výsledné skripty pro program OpenControl (připravené skripty = testovací skripty od autora nástroje MakeMsi). Co se připravených skriptů týče, jsou to jednotlivé moduly, kde každý vykonává nějakou operaci (např. přidání merge modulu, přidání grafiky, atd). Jednotlivé moduly budu dále komentovat viz dále.

Nejprve je třeba uvést, jaké skripty vlastně budu vytvářet. Nástroj MakeMsi umožňuje vytvářet tři základní typy skriptů a to:

- MM (*.MM)
- MMH (*.MMH)
- VER (*.VER)

Co se týče syntaxe, tak ta je u všech stejná, rozdíl je pouze v obsahu. Skripty ve formátu MM jsou základní a očekává se, že v nich budou použity konstrukce pro vlastní běh instalátoru (určení adresářové cesty, přidání komponent do instalačního běhu, vytvoření ikon, atd.). Skripty v MMH formátu jsou hlavičkové a měly by zde být konstrukce, které mají definiční charakter (definování grafiky, definování informací o vývojáři produktu, atd.). Jako poslední představím skripty typu VER. Tyto skripty slouží pro zápis změn a dále pak definují popis výsledného MSI balíčku. Zápisem změn je myšlen zápis různých modifikací skriptů pod určitou verzí a příslušným datem. Do popisu výsledného MSI balíčku pak spadá i název balíčku, určení souboru s licenčním ujednáním či definice pro jaké operační systémy je obsah balíčku určen (u skriptů typu VER je i mírně odlišná syntaxe).

Pokud budeme chtít vytvořit základní, jednoduchý MSI balíček, postačí nám k tomu pouze úprava skriptu typu MM. V případě rozsáhlejší práce, kdy budeme chtít vyřešit i některé detaily, bude třeba zasáhnout i do skriptů typu MMH či VER.

Nyní se podíváme na připravené MM skripty, které jsou v základu k dispozici v adresáři nástroje MakeMsi. Z těch hlavních se jedná se o tyto moduly (systémová cesta MakeMsi\Samples\TryMe\):

- TryMeAddVersionKeywordAndLaunchCondition.MM
- TryMeConditionDialogs.MM
- TryMeCreate[START]OfMsiFromScratch.MM
- TryMeCreateIisSite.MM
- TryMeDllCustomAction.MM
- TryMeLoadDirTreeMaintainingAttributes.MM
- TryMeShortcuts.MM
- TryMeTaskSchedules.MM
- TryMeUpdateRegistry00-99.MM
- TryMeUseMergeModule.MM
- TryMeUserInputUsed2CreateFile.MM
- TryMeWithFixedGuids.MM
- TryMeWithMultipleMedia.MM
- TryMeWithNonAdvShortcutPlusMore.MM
- TryMeWithNoRootFeature.MM
- TryMeWithUserDialog.MM
- TryMeCreateMergeModule.mm

Krom těchto modulů je v nabídce ještě modul TryMe.MM, který má v sobě zabudované základní konstrukce pro vytvoření holého MSI balíčku (pouze vytvoří výsledný balíček bez jakéhokoliv nastavení instalátoru).

Popis modulů, k čemu daný modul slouží je u většiny zřejmý z názvu souboru. Pokud by nebyl, stačí daný modul otevřít a v hlavičce tohoto modulu je pak uveden popis (v angličtině). Pozornost bych věnoval modulu TryMeCreateMergeModule.mm a s tím souvisejícím TryMeUseMergeModule.MM. Merge modul, jak je uvedeno již v tabulce č. 1, slouží k přidání komponent. Je ale třeba vědět, že slouží k přidání dodatečné komponenty, ve stavu, kdy už máme MSI balíček zhotovený – tzn. už nezasahujeme do původního instalátoru, pouze k němu přidáme další vstupní data. Ještě podotknu, že pro modul TryMeDllCustomAction.MM je třeba mít nainstalovaný nástroj MinGW (včetně nastavení proměnného prostředí) a pro modul TryMeCreateIisSite.MM je třeba v OS MS Windows povolit internetovou informační službu. Poslední poznámku bych zmínil o modulu TryMeTaskSchedules.MM, kde je třeba, pakliže budeme pracovat v OS MS Windows XP, nahradit plánovač jt.exe za novější schtasks.exe.

Závěrem bych pak chtěl upozornit, že nástroj MakeMsi nefunguje zcela korektně v OS MS Windows 7, proto je lepší použít OS MS Windows XP (stačí použít nějaký virtualizační software a starší operační systém do něj nainstalovat – nástroj MakeMsi tomu neklade žádný odpor) [6].

6.2.3 Popis implementace

Pro vytvoření výsledného MSI balíčku pro testovací produkt, program OpenControl jsem použil celkem tři moduly obsahující skripty, soubor s licenčním ujednáním a dále pak soubory obsahující grafiku. Popis modulů uvádím níže:

OpenControl_script.MM

Tento skript typu MM nejprve určí adresářové cesty, kam se mají instalovat data a poté určí, kde se zdrojová data nacházejí, a o jaká data se jedná. Níže uvádím, jak vypadá konstrukce pro určení adresářové cesty pro přídavnou komponentu Java SDK:

```
<$DirectoryTree Key="INSTALLDIR_JAVA" Dir="c:\program files\OpenControl\java\"
CHANGE="\" PrimaryFolder="Y">
```

Konstrukce pro určení zdrojových dat, která se následně budou kopírovat je viz níže:

```
<$Files "in\Java JDK\jxpiinstall6.25.exe" DestDir="INSTALLDIR_JAVA">
```

Pokud chceme do instalačního procesu přidat více dat, je to možné, avšak pro každý adresář musíme vytvořit novou konstrukci DirectoryTree. Jednotlivé soubory pak už lze přidat do jednoho DirectoryTree. V další části je úsek kódu, který zajistí, aby se po skončení instalačního procesu instalátoru spustila instalace přídavné komponenty (v našem případě Java SDK). Konstrukce je nastavená tak, aby nečekala na návratový kód, ale ihned se spustila (jakmile se nakopírují vstupní data, je spuštěna instalace komponenty). Tuto konstrukce lze vidět viz níže:

```
#(
	<$ExeCa

		EXE="[INSTALLDIR_JAVA]jxpiinstall6.25.exe" Args=^"jxpiinstall6.25"^
		WorkDir="INSTALLDIR_JAVA"
		SEQ="InstallFinalize-"   Type="immediate ASync AnyRc"
		Condition="<$CONDITION_INSTALL_ONLY>"

	>
#)
```

Po těchto operacích následují konstrukce, které mají za úkol vytvořit požadované zástupce i s příslušnou grafikou ikon. Po nastavení adresářových cest a makra se pak použije níže uvedená konstrukce, která umí definovat i základní grafiku použitou pro ikony.

```
#(
	<$Shortcut

		Dir="SHORTCUT_FOLDER"
		Feature="."
		Title="Open start.bat (to compile)"
		Description=^This advertised shortcut simply open "start.bat" to compile^
		Icon="@.\Program_icon.ico"
		WorkDir="INSTALLDIR_MAIN"

	>
#)
```

Všechny tyto typy konstrukcí, uvedené u modulu OpenControl_script.MM uvádím proto, že jsou univerzální, tzn. lehkou modifikací je lze přizpůsobit i jinému programu než OpenControl.

Modul OpenControl_script.MM tedy určí vstupní data, nakopíruje je do instalačního adresáře a po dokončení instalačního procesu spustí přídavnou komponentu. Mimo to vytvoří všechny potřebné zástupce (včetně grafiky ikon), přidá nástroj pro odinstalování či správu nainstalovaného programu a nadefinuje makra.

OpenControl_script.VER

V tomto modulu je obsažen skript typu VER, který určuje název instalovaného produktu, popis, který bude ve vlastnostech výsledného MSI balíčku, či soubor s licenčním ujednáním. Dále je zde definováno pro jaký operační systém je produkt určen. Poslední část obsahuje popis změn s příslušným datem a verzí (pouze informativní, nemá vliv na běh nástroje).

ME.MMH

Modul ME.MMH je již v základu obsažen v nástroji MakeMsi, takže zkonstruování tohoto modulu znamenalo modifikování toho původního. Z toho vyplývá, že původní modul bude třeba přepsat tím nynějším (řešeno automaticky – viz uživatelská příručka).

Tento modul obsahuje konstrukce pro definování specifikací v úvodním okně instalátoru. Jedná se o jméno vývojáře, webové stránky, e-mail apod. Dále je zde pak definovaná veškerá grafika, ať už v oknech instalátoru, či v ikonách určených pro MSI balíček. Pro grafiku ikony jsem použil vlastní logo a pro banner taktéž. Z použitých konstrukcí bych věnoval pozornost definování postranní barvě textu v instalačním okně. Jedná se o tuto konstrukci:

```
#define? UISAMPLE_LEFTSIDE_TEXT_FONT_COLOR          &H7F0000
```

Konstrukce obsahuje hodnotu &H7F0000, která definuje danou barvu. Důležité je, že pokud bychom chtěli barvu změnit, musí se použít jiná hodnota dle syntaxe **VBscriptu.**

Použití grafiky pak znamená pouze nadefinovat, kam do okna instalátoru chceme požadovanou grafiku umístit a vybrat adresářovou cestu k obrázku ve vhodném formátu, viz níže:

```
#define? UISAMPLE_BITMAP_WHITE_BANNER      Graphics\White_Banner.bmp
```

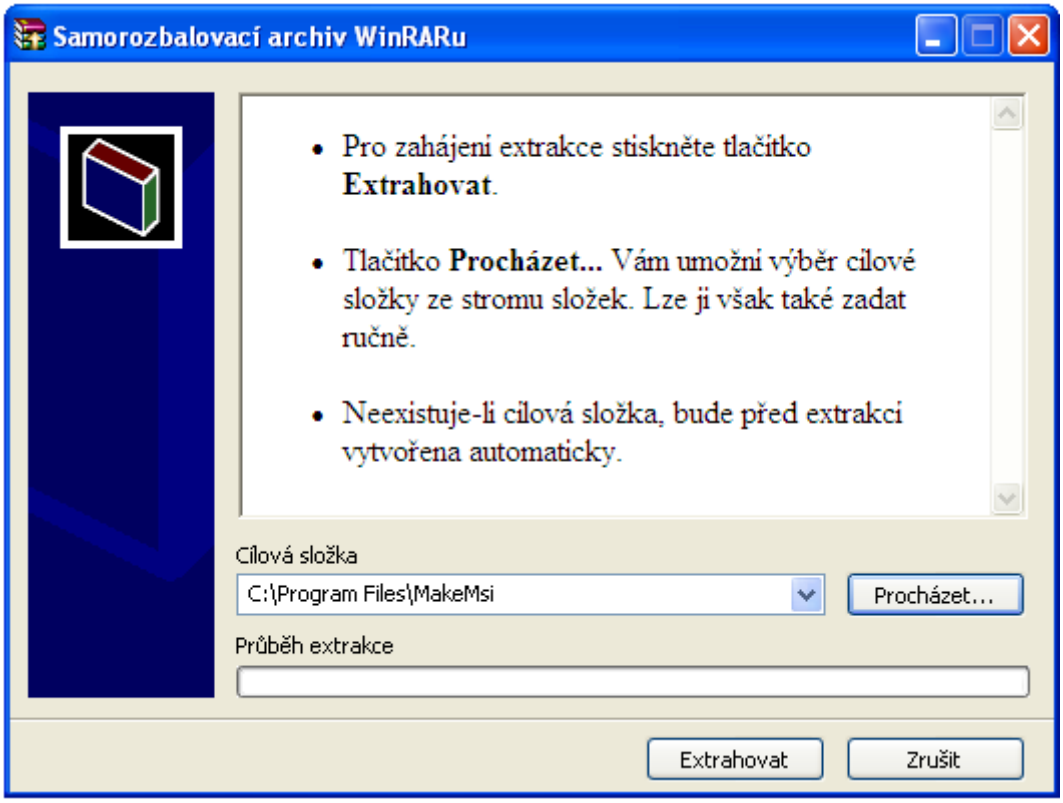
Opět jsem uvedl základní typy definic, které se ve výsledném skriptu vyskytují, a lze vidět, že jdou lehce upravit i pro jiný program než testovací OpenControl.

6.2.4 Uživatelská příručka

V uživatelské příručce popíši jak z vytvořených skriptů za pomoci nástroje MakeMsi sestavit výsledný MSI balíček pro testovací produkt OpenControl.

Použité skripty, grafika a testovací produkt musejí být nakopírovány do speciálních složek k tomu určených. Abych uživatele nezatěžoval těmito informacemi, připravil jsem samorozbalovací archiv, který se postará o správné nakopírování (či přesunutí) souborů do

daných složek. Stačí tedy spustit přiložený samorozbalovací archiv *OpenControl_source.exe* a nastavit správnou cestu k nástroji MakeMsi (viz obr. 6.4).



Obr. 6.4 – Nastavení adresářové cesty v samorozbalovacím archivu WinRAR

Při extrakci budeme dotázáni, jestli chceme přepsat modul ME.MMH, je třeba tuto hlášku potvrdit. Poté již můžeme přejít k samostatnému vytvoření MSI balíčku. Spustíme si příkazový řádek (v OS MS Windows standardně Start→Spustit→cmd), ve kterém si nejdříve příkazem cd zvolíme správný adresář:

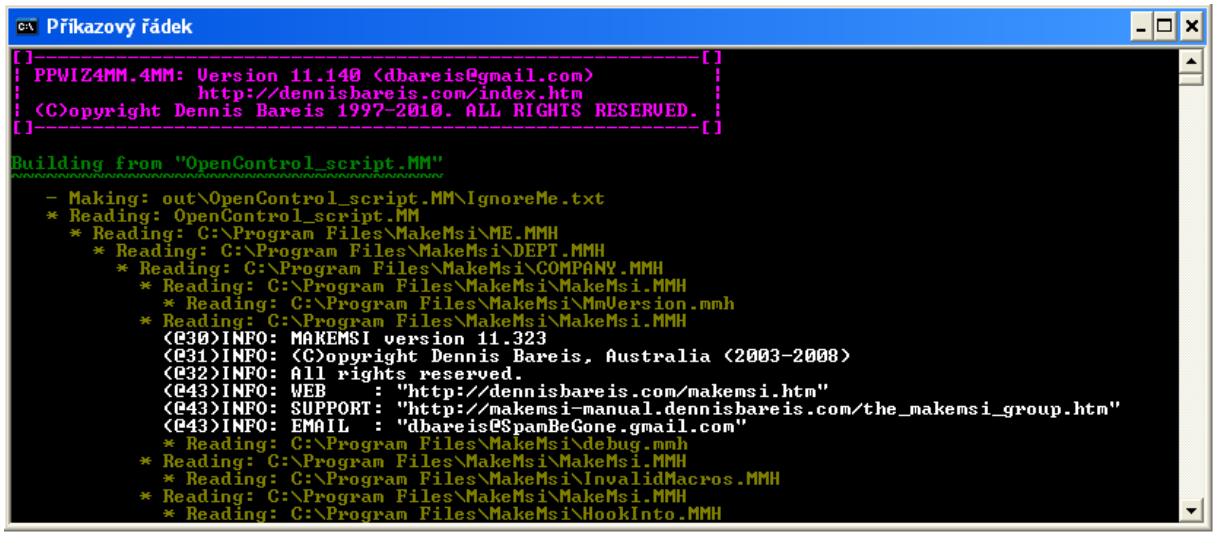
```
C:\Program Files\MakeMsi\Samples\TryMe\Create MSI Installers>
```

A dále pak zadáme příkaz, uvedený níže:

```
mm OpenControl_script.MM
```

Mělo by se spustit samotné sestavení MSI balíčku (obr. 6.5). V první fázi sestavování se nejprve přečtou veškeré skripty obsažené v nástroji MakeMsi. Proběhne kontrola skriptů, a pokud je vše v pořádku, přejde se k archivování. V tomto kroku se obsah vstupních dat (program OpenControl) zarchivuje do komprimovaného CAB formátu. Tento formát je vyvinutý společností Microsoft, a proto se také používá u MSI balíčků. V neposlední řadě

nástroj ještě zkontroluje, jestli nejsou přiložené nějaké merge moduly. Pokud při běhu nástroje nenastal žádný problém, bude následně vygenerován výsledný MSI balíček.

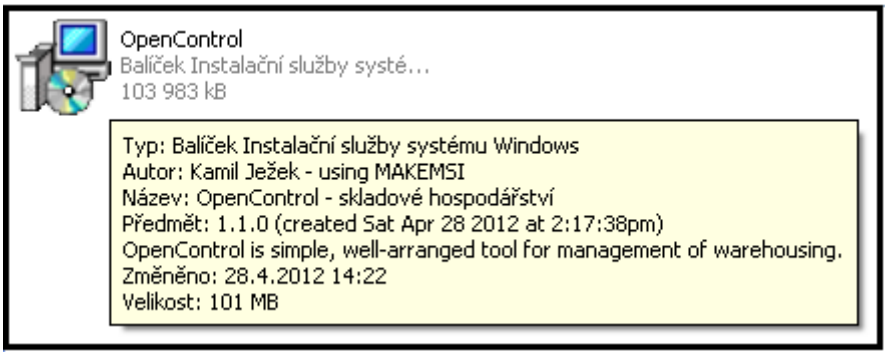


Obr. 6.5 – Sestavení MSI balíčku pomocí nástroje MakeMsi

Výsledný MSI balíček, včetně logů nalezneme na této adresářové cestě:

```
C:\Program Files\MakeMsi\Samples\TryMe\Create MSI Installers\out\
```

Na obrázku č. 6.7 lze vidět výsledný MSI balíček včetně vytvořených popisů. Takto vytvořený balíček lze použít na všech operačních systémech Microsoft Windows, kde je nainstalovaná podpora pro formát MSI (v OS MS Windows 2000 a vyšších je podpora nainstalovaná implicitně).



Obr. č. 6.7 – Výsledný MSI balíček pro program OpenControl

6.2.5 Závěr

Výsledek realizační části splňuje zadání – vytvořené skripty umožňují sestavení MSI balíčku, kdy při vlastním běhu sestavování uživatel nemusí nijak interagovat s nástrojem. Navíc konstrukce, které jsou ve skriptech použity, umožňují univerzální přístup, tedy je možné je lehkou modifikací upravit i na jiný program než ten testovací OpenControl. Přidání libovolné komponenty do balíčku (v našem případě Java SDK) funguje také korektně.

Doba běhu nástroje při použití vytvořených skriptů je uvedena v tabulce č. 6.3 (doba za jakou se sestaví MSI balíček). Závisí hodně na velikosti vstupních dat. Nejnáročnější na čas je totiž samotný proces komprimování.

Sestavení:	Doba běhu:	Průměrná doba běhu:
1.	2 min 53,01 s	3 min 7,72 s
2.	3 min 19,75 s	
3.	3 min 10,43 s	

Tab. 6.3 – Doba běhu nástroje MakeMsi

Uvedl bych ještě, že doba běhu nástroje byla testována ve virtualizačním softwaru – nástroji VMware Player, kde byl nainstalován operační systém Microsoft Windows XP. Výsledné skripty pro program OpenControl byly taktéž vytvářeny v tomto operačním systému, za použití nástroje SciTE (nástroj je třeba použít i pro korektní zobrazení skriptů).

Nástroj MakeMsi byl ideální pro realizaci této úlohy. Jediný problém nastal se syntaxí skriptů, kde se jednalo o „syntaxi určenou autorem“. Musel jsem si tedy detailně pročíst dokumentaci a v případě problémů vyhledávat na internetových fórech. Výhodou tohoto nástroje je ale velká funkcionalita a dostatek testovacích skriptů. Díky otestování všech těchto skriptů jsem se pak dozvěděl další užitečné vlastnosti MSI balíčků. Myslím si tedy, že nástroj MakeMsi byl dobře zvolený.

7 Celkový závěr

Vzhledem ke koncipování této bakalářské práce uvedu spíše obecnější pohled, veškeré detaily, co se zhodnocení a závěru týče, jsou uvedeny v jednotlivých kapitolách. Část o propagaci softwarových produktů jsem zpracovával spíše teoreticky, snažil jsem se ale prostudovat veškerou problematiku a vypracovat tak kvalitní přehled. Propagaci jsem hodně soustředil na SEO metody, které jsou rozebrány jak obecně, tak konkrétně i s praktickou ukázkou. Abych ale skutečně pojal širší část propagace, uvedl jsem i nástin, jak by měla vypadat reklama na požadovaný produkt. Část distribuce softwarových produktů pak byla spíše praktická, jelikož jsem otestoval poměrně dost nástrojů a závěrem pak i vytvořil požadované skripty.

Velmi zajímavá mi přišla část o distribuci softwarových produktů, jelikož jsem měl možnost seznámit se s různými distribucemi Linuxu. Při realizaci skriptů jsem si pak zopakoval znalosti s jazykem BASH a získal nové zkušenosti s testovanými nástroji. Celkově si myslím, že mi byla tato práce přínosem a že jsem jí zpracoval dle požadovaných pokynů.

Do budoucna by bylo přínosné prozkoumat detailněji všechny konfigurační soubory, které jsou třeba k vytváření DEB/RPM balíčků. Zajímavá se mi též jeví možnost licencování u těchto balíčků. Pokud bych tedy někdy měl možnost, například v nějakém školním projektu se k tomuto tématu vrátit, rád bych tyto možnosti prozkoumal.

Přehled zkratek

- **Addon**
  forma označení pro doplňky do určitého programu

- **Tag**
  značka, která obsahuje nějakou informaci

- **Dependecy hell**
  stav, kdy vlivem závislostí jednotlivých balíčků dojde k zacyklení

- **Open-source**
  forma softwaru, která má volně k dispozici svůj zdrojový kód

- **POSIX**
  rozhraní, zajišťující přenositelnost programů mezi více platformami

- **Virtualizační software**
  Software, pomocí kterého můžeme nainstalovat na již stávající operační systém další, bez nutnosti fyzického přístupu

Literatura

[1] Jak psát web – SEO metody

URL: http://www.jakpsatweb.cz/seo/ (aktuální k 5.5.2012)

[2] Debian Policy Manual – policy requirements for the Debian distribution

URL: http://www.debian.org/doc/debian-policy/ (aktuální k 5.5.2012)

[3] Rukověť baliče RPM – manuál Davida Nečase k RPM balíčkům

URL: http://www.abclinuxu.cz/autori/david-necas (aktuální k 5.5.2012)

[4] Linuxové noviny – tvorba RPM balíků

URL: http://www.linux.cz/noviny/1998-05/clanek10.html (aktuální k 5.5.2012)

[5] Microsoft TechNet – extensive technical resources about MS products

URL: http://technet.microsoft.com/en-us/ (aktuální k 5.5.2012)

[6] DennisBareis – MakeMsi tool

URL: http://dennisbareis.com/ (aktuální k 5.5.2012)

[7] HelpMark – SEO, marketing

URL: http://www.helpmark.cz/seo (aktuální k 5.5.2012)

[8] Hertzog Raphaël, Roland Mas

E-BOOK: The Debian Administrator's Handbook (aktuální k 5.5.2012)

[9] Smička Radim, Ing.

E-BOOK: Optimalizace pro vyhledávač – SEO (aktuální k 5.5.2012)

[10] Kubíček Michal, Linhart Jan

333 tipů a triků pro SEO (Computer Press, 2010)

[11] Langville, Amy N.

Google's PageRank and beyond (Princeton University Press, 2006)

[12] Bíbr Ivan

Mandriva Linux 2007.1 CZ (Computer Press, 2007)

Obsah DVD

Přiložené DVD obsahuje tyto složky:

- ***Instalační_formy_skriptů***
  Obsahuje archivy s potřebnými daty a zajišťuje tak snadnou instalaci potřebného prostředí pro skripty

- ***Použité_nástroje***
  Obsahuje nástroje, které byly použity k testování tvorby distribučních balíčků

- ***Reklamní_brožury***
  Obsahuje reklamní brožury od společnosti Konica Minolta (na tyto materiály se odkazuji v části o propagaci)

- ***Screenshoty_výsledku***
  Obsahuje screenshoty průběhu skriptů a následné instalace vytvořeného distribučního balíčku

- ***Text_práce***
  Obsahuje text této práce ve formátu DOCX, DOC a PDF

- ***Vytvořené_skripty***
  Obsahuje zdrojové kódy vytvořených skriptů