VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

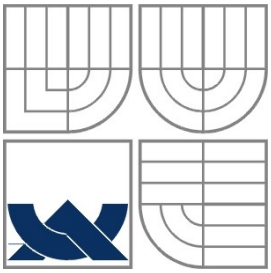
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

NÁSTROJ PRO PODPORU MIGRACE NASTAVENÍ GNOME

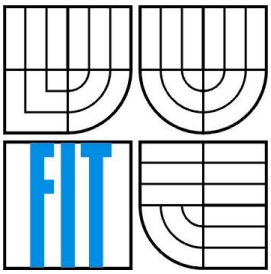
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE JIŘÍ CEPÁK
AUTHOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

NÁSTROJ PRO PODPORU MIGRACE NASTAVENÍ GNOME

TOOLS FOR MIGRATION OF GNOME'S SETTINGS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JIŘÍ CEPÁK

VEDOUCÍ PRÁCE
SUPERVISOR

ING. PETR ČÁSTEK

BRNO 2009

Abstrakt

Tato práce studuje způsob uložení konfiguračních souborů v unixových operačních systémech, tedy umístění i obsah systémových i uživatelských konfiguračních souborů. Zabývá se také historií operačních systémů Unix a GNU/Linux a popisuje aplikaci, která uvádí poznatky této práce do praxe a umožňuje zálohování konfiguračních souborů a také jejich obnovu ze zálohy.

Abstract

This work studies the way of saving configuration files in Unix-like operating systems, so the location and content system and also user configuration files. This work also studies the history of operating systém Unix and GNU/Linux and describes aplication, that implements gained experience and makes possible to save configuration files and also restore from backup.

Klíčová slova

Operační systém, Unix, GNU/Linux, konfigurační soubory, Python, GTK+, PyGTK, XML.

Keywords

Operating system, Unix, GNU/Linux, configuration files, Python, GTK+, PyGTK, XML.

Citace

Jiří Cepák: Nástroj pro podporu migrace nastavení GNOME, bakalářská práce, Brno, FIT VUT v Brně, 2009

Nástroj pro podporu migrace nastavení GNOME

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Petra Částka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jiří Cepák
24.2.2009

Poděkování

Na tomto místě bych rád poděkoval mému vedoucímu Ing. Petru Částkovi za odborné vedení, za poskytnuté rady a za čas, který mi při tvorbě práce věnoval.

Obsah

1 Úvod

1.1 Zadání

Navrhněte nástroj s grafickým uživatelským rozhraním pro import-export nastavení aplikací a některých nastavení GNOME.

1.2 Doplnění zadání

Aplikace má běžet volitelně také bez grafického uživatelského rozhraní.

1.3 Obsah a cíl práce

Cílem této práce je implementace nástroje s grafickým uživatelským prostředím, umožňujícím zálohování systémových a uživatelských konfiguračních souborů v unixových operačních systémech s prostředím GNOME. Obsahem této práce je také teorie, popisující způsob ukládání konfiguračních souborů v unixových systémech a dále popis použitých nástrojů.

První kapitola obsahuje zadání, formulaci cíle a popis obsahu práce.

Druhá kapitola se zabývá teorií, potřebnou k řešení práce a použitými nástroji. Na začátku se věnuji pojmu operační systém a historii operačních systémů Unix a GNU/Linux. Dále popisuji použité nástroje, jmenovitě programovací jazyk *Python*, knihovnu *GTK+*, port knihovny GTK+ do jazyka Python *PyGTK*, textový formát *XML* a program pro vývoj grafický návrh GUI aplikace *Glade User Interface Builder*. Na konci této kapitoly rozebírám způsob ukládání konfiguračních souborů v unixových operačních systémech, tedy jakým způsobem a kam se ukládají systémové a uživatelské konfigurační soubory.

Třetí kapitola se zabývá konfiguračními soubory jednotlivých aplikací, rozebírá obsah adresářů s konfiguračními soubory v domovském adresáři uživatele, určuje obsah jednotlivých souborů a vybírá ty, které mají být zálohovány. Tato kapitola dále obsahuje popis obsahu adresáře */etc*, který obsahuje systémové konfigurační soubory a také se zabývá inicializačním souborem shellu.

Čtvrtá kapitola se zabývá samotnou aplikací, popisuje důvody použití zvolených nástrojů, principy fungování aplikace, některá implementační fakta a použitá řešení.

V poslední páté kapitole, kterou je závěr, dochází k sumarizaci poznatků, zhodnocení dosažených výsledků a popisu dalších možných vylepšení a úprav aplikace.

1.4 Poznámka

V unixových operačních systémech se používá pojem *directory – adresář*, kdežto ve windowsových operačních systémech je běžné hovořit o *složkách – folders*. Z tohoto důvodu používám v celé práci (technické zprávě i v komentářích programu samotného) pojem *directory – adresář*.

2 Teorie

2.1 Operační systémy

Operační systém je součástí každého běžného výpočetního systému. Jedná se o komplexní programové vybavení vytvářející mezivrstvu mezi hardware počítače a jeho aplikační vybavením a uživateli. Primární funkcí operačního systému je poskytování podpory pro realizaci počítačových programů. Operační systémy vždy měly, mají a pravděpodobně i budou mít značný vliv na vývoj hardware i software [3] [1].

Operační systém může být jednoduchý a minimalizovaný (například DOS) nebo velký a složitý (například OS/2 nebo VMS). Operační systém *Unix* patří ke středně velkým systémům. Velké systémy poskytují o něco více zdrojů a prostředků než jednoduché operační systémy, a to v takové formě, aby s nimi bylo možno řešit prakticky libovolnou úlohu [1].

Klíčovou částí operačního systému je tzv. j*ádro* (*kernel*). Ve většině operačních systémů, jako je Unix, OS/2 nebo VMS, plní jádro systému takové funkce, jako je spouštění programů, přidělování systémových zdrojů, přidělování času procesoru současně běžícím aplikacím a podobně. Jádro systému je program, běžící na počítači ihned po nastartování (po inicializaci hardware pomocí *BIOSu* a výběru z jaké části hardisku se bude načítat), realizuje všechny konfigurační funkce a jako poslední program před vypnutím počítače realizuje všechny potřebné funkce k zastavení systému [1].

2.2 Unix

V roce 1965 pracovaly společnosti *Bell Telephone Laboratories* (divize *AT&T*) a *General Electric* na projektu *Mac of MIT*, jehož cílem bylo vytvořit operační systém *Multics* (*Multiplexed Information and Computing Service*). Později společnost Bell Telephone Laboratories od tohoto projektu odstoupila a v důsledku tohoto rozhodnutí neměla k dispozici kvalitní operační systém. Proto se programátoři *Ken Thompson* a *Denis Ritchie* rozhodli navrhnout operační systém, který by této společnosti vyhovoval. Ken Thompson tento návrh realizoval a Brian Kernighan tento operační systém pojmenoval Unix (slovní hříčka na Multics) [1] [7].

Původně bylo jádro napsáno v assembleru, z tohoto důvodu bylo nepřenositelné mezi různými platformami. V roce 1973 byl *Unix* kompletně přepsán do programovacího jazyku C (tento jazyk vytvořil Denis Ritchie), kernel měl v té době 11 000 řádků. Koncem 70. let byla společnosti AT&T antimonopolním úřadem zakázána činnost v oblasti počítačového průmyslu, proto se tato společnost

rozhodla vzhledem k výhodným finančním podmínkám převést licenci Unixu na některé university. Unix se stal populární v akademických kruzích a postupem času se prosadil i v komerční sféře [1] [7].

Akademické prostředí bylo pro Unix velmi přínosné, mezi nevýznamnější počiny patří:

- Editor vi - Bill Joy, univerzita Berkeley, na této universitě vzniklo i mnoho dalších věcí důležitých pro rozvoj software obecně, např. regulární výrazy, výzkum v oblasti databází apod.
- Tisk - University of Toronto.
- Shell - Yale University.
- Zlepšování výkonu, podpora více uživatelů a sítě - Purdue University (Electrical Engineering Department) [1] [7].

V současnosti existují dvě základní varianty Unixu: System V od společnosti *USL* (*Unix System Labories*) a *BSD* (*Berkeley Software Distribution*). Kromě těchto dvou základních verzí existuje spousta dalších verzí operačního systému Unix. Komerční verze jsou zpravidla odvozeny od jedné ze základních verzí USL nebo BSD, ale existují i verze, kombinující vlastnosti obou verzí [1] [7].

2.3 GNU/Linux

Projekt *GNU* byl založen v roce 1983 programátorem *Richardem Stallmanem*. Tento projekt je založen na myšlence zcela svobodného operačního systému unixového typu, tedy jádra včetně aplikací, nezbytných pro pohodlné používání systému. Jedním z důvodů vzniku byla potřeba zpřístupnění programového vybavení původně napsaného pro systém Unix uživatelům ostatních operačních systémů a také vytvoření přenositelného operačního systému, který bude mít podobné vlastnosti jako Unix. V roce 1984 začal vlastní vývoj jednotlivých aplikací a roku 1994 se podařilo prvotní cíl naplnit, když bylo ke *GNU* aplikacím přidání jádro *Linux*, vyvinuté finským studentem *Linusem Torvaldsem*. Výsledný operační systém byl pojmenován *GNU/Linux*. Tento operační systém byl vyvinut a zdokonalován bez počtem lidí po celém světě a to bez zásahu společností Unix systém Labories a Berkley Software Distribution. Projekt GNU je v současnosti spravován nadací Free Software Foundation [1] [8].

GNU/Linux je i v současnosti zdokonalován vývojáři po celém světe. Jedná se o verzi operačního systému Unix pro osobní počítače s procesory Intel, Alpha a dalšími. Systém je navržen tak, aby maximálně využíval vlastnosti procesorů (preemptivní multitasking). Licence, vztahující se

na tento operační systém, se nazývá General Public License (GPL) a jednou z jejích největších předností je povinnost zpřístupnění zdrojových kódů [1].

Operační systém *GNU/Linux* podporuje většinu programového vybavení, napsaného pro Unix, včetně systému *X Window*. Tento systém byl vytvořen v Massachusettském institutu technologie tak, aby umožňoval operačním systémům Unix vytvářet grafická okna a interaktivně spolupracovat mezi sebou [1].

V současnosti je *GNU/Linux* součástí trhu s osobními počítači a je plnohodnotným operačním systémem se všemi funkcemi a aplikacemi, potřebnými pro pohodlnou a efektivní práci. Na serverech má *GNU/Linux* pověst spolehlivé, výkonné a stabilní platformy, na které běží rozličné služby společností jako je Amazon, americká pošta nebo německá armáda. Ve velké míře je také využíván v oblasti poskytování internetu a internetových služeb jako web server, firewall apod. Za zmínku také stojí jeho využití v různých vestavěných zařízeních, mobilních telefonech a PDA [2].

2.3.1 Tar

Tar je součástí balíku aplikací *GNU*. Jedná se o aplikaci v příkazové řádce, umožňující archivaci souborů a to pomocí sloučení velkého množství souborů do jediného archivního souboru, přičemž zachovává veškeré informace o souborech (např.: uživatelská práva). Jméno tar je zkratkou „*tape archive*“, protože tento nástroj byl původně používán pro archivaci na magnetické pásky [1] [8].

2.3.2 Gzip

Gzip je součástí balíku GNU. Jedná se o aplikaci v příkazové řádce, umožňující kompresi souborů za použití kódování *Lempel-Ziv* (*LZ77*). Zachovává veškeré informace o souborech (např.: uživatelská práva a časy modifikace) [8].

2.4 Python

Python je interpretovaný objektově orientovaný programovací jazyk. Vznikl v roce 1990 Stichtingském matematickém centru jako nástupce jazyka ABC, autorem je Holanďan *Guido van Rossum*. Python je vyvíjen jako open source projekt, tedy je zdarma a jeho zdrojový kód je otevřený. Jedná se o multiplatformní programovací jazyk, dostupný ve většině operačních systémů (Unix, Windows, Mac OS). Název Python vznikl podle názvu britského seriálů Monty Pythonův létající cirkus.

Vytyčené cíle Pythonu:

- Intuitivní,
- snadný,
- mocný programovací jazyk.
- Otevřený zdrojový kód.
- Kód, který je srozumitelný jako běžná angličtina.

Díky tomu, že je Python interpretovaným jazykem, není tak výkonný jako kompilované programovací jazyky, avšak díky dobrým možnostem skriptování a velkým množstvím modulů je velmi mocným nástrojem pro pohodlný vývoj aplikací.
Zdroj: [12]

2.5 GTK+

GTK+ neboli *The Gimp Toolkit* je open-source knihovna, umožňující rychlí vývoj grafického uživatelského rozhraní (*GUI*). Jedná se o multiplatformně kompatibilní knihovnu se snadno použitelným rozhraním pro programování aplikace (*API*).

GTK+ je jednou ze dvou nejpoužívanějších knihoven pro vývoj grafického uživatelského rozhraní v prostředí unixových operačních systémů. Tato knihovna společně s knihovnami *Qt* nahradilo dříve velmi používaný *Motif*.

GTK+ jsou napsány v jazyku C, ale jsou portovány i do dalších jazyků, jako je C++, Python, C# a další. Tyto knihovny jsou šířeny pod licencí *GNU LGPL*, umožňující vývoj placených (proprietárních) i neplacených aplikací.

Knihovny *GTK+* byly původně vyvinuty v roce 1997 členy Experimental Computing Facility (*XFC*) na Kalifornské univerzitě v Berkley.

Architektura *GTK+* se skládá ze čtyř základních částí:

- *Glib:* základní knihovna, jádro GTK+, struktury pro jazyk C, zajišťuje přenosnost
- *Pango:* stará se o rozvržení a vyobrazení textu
- *Cairo:* knihovna pro 2D grafiku
- *ATK:* rozhraní

Zdroj: [13]

2.6 PyGTK

PyGTK je port knihovny *GTK+* do jazyka Python, nabízející komplexní sadu grafických prvků a jiných potřebných programovacích zařízení, jako je lokalizace textu a další. *PyGTK* je součástí projektu *GNOME* a je stejně jako knihovna *GTK+* šířen pod licencí *GNU LGPL* [14].

2.7 XML

Extensible Markup Language je jednoduchý, velmi snadno modifikovatelný textový formát, odvozený od *SGML*, obsahující strukturované informace, které obsahují data samotná, ale také informaci o jejich významu. Jedná se o značkovací jazyk, který byl vyvinut a standardizován konsorciem *W3C*. Umožňuje popis struktury dokumentu z hlediska obsahu jednotlivých částí, nezabývá se tím, jak bude výsledný dokument zobrazen, tedy popisem jeho vzhledu. Vzhled dokumentu se definuje pomocí jiného souboru se stylem. *XML* umožňuje transformaci do jiného typu dokumentu nebo struktury *XML*. V *XML* formátu se běžně ukládá vektorová grafika, meta-data různých objektů, matematické rovnice, rozhraní pro programování aplikací (*API*) a spousty dalších strukturovaných informací [5].

2.8 Glade User Interface Builder

Glade Interface Designer je program pro grafický návrh GUI aplikací pro knihovnu GTK+, umožňující použití přídavných komponent prostředí GNOME. V současnosti existuje již čtvrtá verze tohoto programu. Jedná se o aplikace nezávislou na programovacím jazyku, která negeneruje kód v programovacím jazyku, ale soubor s popisem GUI ve formátu XML (GladeXML). Tento XML soubor může být načten do aplikace pomocí *libglade* knihoven. Mezi podporované jazyky patří: C, C++, Java, Perl, Python, C#, Pike, Ruby, Haskell a další.

Glade Interface Designer je šířen pod licencí GNU GPL. První verze tohoto programu vyšla roku 1998 a poslední, čtvrtá, 6.4.2009, GUI k aplikaci, kterou popisuje tato práce, bylo vyvíjena pomocí třetí verze, vzniklé v roce 2006.

Zdroj: [17]

2.9 Způsob ukládání konfiguračních souborů v unixových operačních systémech

Filosofie operačního systému Unix se od filosofie ostatních operačních systému podstatně liší. Autoři Unixu se nepokoušeli předvídat potřeby všech uživatelů, místo toho se pokoušeli navrhnout operační systém tak, aby si každý uživatel mohl své pracovní prostředí operačního systému jednoduše upravit podle svých potřeb [1].

Konfigurace jednotlivých programů operačního systému Unix se definuje prostřednictvím konfiguračních souborů. Aplikace v unixových operačních systémech striktně oddělují uživatelské data od systémových. To se váže i k nastavení aplikací a ukládání jejich konfiguračních souborů, ty jsou označovány také jako „*ini-files*“, „*rc files*“ nebo „*dot files*“. Veškeré konfigurační soubory aplikací jsou ukládány do domovského adresáře uživatele. Dodržovaným standardem je nastavení adresářů s konfiguračními soubory jako skrytých, kdy jejich název vždy začíná znakem „.“ [14].

3 Způsob uložení konfiguračních souborů jednotlivých aplikací

V této části budeme zkoumat způsob, jakým jsou ukládány konfigurační soubory konkrétních aplikací, tzn. podíváme se na obsah skrytých adresářů v domovském adresáři uživatele a na složku se systémovými konfiguračními soubory (*/etc*). Předmětem zkoumání budou pouze ty aplikace, jejichž konfiguraci program umožňuje zálohovat.

3.1 Evolution

Evolution (původně *Ximian Evolution*) je poštovní, PIM a Groupwarový klient. Obsahuje email, kalendář, poznámky, úkoly, kontakty. Lze jej propojit s Microsoft Exchange a Novell GroupWISE. Je možné jej synchronizovat s Palm OS zařízením [16].

Disponuje mnoha funkcemi jako je inteligentní filtrování spamu, pokročilé vyhledávání, integrace do prostředí GNOME, synchronizace adresáře se seznamem kontaktů programu Pidgin, webový kalendář, šifrování a elektronické podepisování zpráv, vFolders, podpora pluginů (Eplugin system) a další [16].

Konfigurační soubory se nacházejí ve složce *$HOME/.evolution* a zálohován je její celý obsah vyjma adresáře *cache*. Jedná se o soubory a adresáře s konfigurací kalendáře, kategoriemi, adresářem, emaily samotnými apod. Samozřejmostí je možnost zálohy pouze některé části (kalendář, emaily samotné a vše ostatní).

3.2 Firefox

Mozilla Firefox je bezpochyby jedním z nejpoužívanějších prohlížečů v linuxových operačních systémech. Je v základní instalaci většiny nejpoužívanějších distribucí. Ještě než se posuneme dále, rád bych upozornil, že všechna konstatovaná fakta, obsažená v tomto odstavci, se vztahují striktně na osobní počítače a nikoliv servery, kde je situace, vzhledem k časté absenci grafického uživatelského rozhraní (*X Window System*), odlišná.

Mozilla Firefox je velmi komplexní program a proto obsahuje i několik nástrojů k úpravě konfiguračních souborů. Prvním a nejznámějším z nich je umístěn v menu (Předvolby – Preferences).

Druhým a daleko podrobnějším je integrovaný formulář, na který se dostaneme po zadání *about:config* do adresového formulářového políčka přímo v programu Mozilla Firefox. Jedná se o podrobný výčet možných nastavení, který lze změnit pouhým kliknutím [4].

Konfigurační soubory nalezneme na standardním místě, v domovském adresáři uživatele, ve skrytém adresáři s název *.mozilla*. Tento adresář obsahuje podadresář extensions s rozšířeními Firefoxu (např. napojení na emailový klient Mozilla Thuderbird) a firefox s nastavením uživatelského profilu, konfiguračními soubory, záložkami, historií a dočasnou pamětí (cache).

Záložky ve Firefoxu jsou ukládány do databáze. Jedná se o binární soubory s příponou *sqlite*. *Sqlite* je databázový relační systém napsaný v jazyku C. Při změně hodnoty proměnné *browser.bookmarks.autoExportHTML* na *True* v integrovaném konfiguračním formuláři Firefox, dochází k automatickému generování záložek do souboru *bookmarks.html*. Toto nastavení je však bohužel výchoze nastaveno na False, takže je nutno zálohovat celou databázi, které je poměrně velká. Řešení ve stylu dočasné automatické změny proměnné naší aplikací by se minulo účinkem, protože k vygenerování html souboru z databáze je třeba restartovat Firefox, což by paradoxně bylo pomalejší než zálohování celé databáze [6].

3.3 Gimp

GIMP je velmi kvalitní multiplatformní nástroj pro úpravu fotografií a rastrové grafiky. Název *GIMP* je zkratka z anglického *GNU Image Manipulation Program*. Jedná se o aplikaci s mnoha funkcemi, lze ho využít pro retušování fotografií, dávkové zpracování obrázků, konvertor obrazových souborových formátů atd. Je snadno rozšiřitelný. Je navržen tak, aby mohl být pomocí zásuvných modulů a skriptů snadno doplňován novými funkcemi. Pokročilé skriptovací rozhraní umožňuje automatizovat vše od nejjednodušších až po ty nejsložitější úlohy. Jednou z předností programu GIMP je jeho volná dostupnost pro nejrůznější platformy a operační systémy. Většina distribucí GNU/Linuxu obsahuje GIMP jako standardní součást systému. GIMP je k dispozici i pro další systémy, jako např. Microsoft Windows nebo Apple Mac OS X (Darwin) [11].

Konfigurační soubory se nacházejí v adresáři */home/user/gimp-x.y*, kde *x.y* je číslo verze. Obsah jednotlivých souborů [11]:

- *colorrc*: naposledy použité barvy
- *gimprc*: globální konfigurace Gimp (paměťová náročnost, cesta k fu-skriptům ...)
- *gtkrc*: specifikace GTK stylu
- *menurc*: konfigurace menu
- *profilerc*: historie barev profilu

- *toolrc*: nástroje Gimpu

V adresáři programu GIMP se nacházejí také adresáře s uživatelsky vytvořenými paletami, přechody apod.

3.4 GNOME Commander

GNOME Commander je dvou panelový grafický souborový manager pro prostředí *GNOME*, navržený podle *Norton* a *Midnight Commanderu*. Je založen na knihovnách *GTK+* a *GnomeVFS*. Tato aplikace si klade za cíl pokrýt škálu funkcí, požadovaných pokročilými uživateli. Funkčnost je zajištěna díky užívání speciálních aplikací a spouštění efektivních příkazů [10].

Základní funkce [10]:

- GTK-2 grafické rozhraní
- GNOME mime typy
- FTP (díky GnomeVFS modulu)
- přístup na SAMBA servery
- menu po kliknutí na pravé tlačítko myši
- kontextové menu myši, snadno použitelné k volání externích aplikací
- nástroje pro hledání souborů a porovnávání adresářů

Konfigurační soubory GNOME Commanderu se stejně jako u ostatních aplikací nacházejí v domovském adresáři uživatele a to v adresáři .gnome-commander. Jedná se o adresář s pluginy, soubor s připojeními (connections) a další konfigurační soubory. Kam si GNOME Commander ukládá např. nastavení vzhledu se mi zjistit nepodařilo, v *$HOME/.gnome-commander*, */usr/share* ani v */etc* to podle mého průzkumu není.

3.5 Pidgin

Pidgin (dříve známý jako Gaim) je multiprotokolový a multiplatformní klient pro zasílání zpráv v reálném čase (instant messaging). Tento program podporuje mnoho komunikačních protokolů: OSCAR (AIM/ICQ/.Mac), MSN, Yahoo, Gadu-Gadu, IRC, XMPP (Jabber, Google Talk), MyspaceIM, OpenNAP, SILC, Zephyr, Napster a GroupWise. Pidgin je svobodný software šířený pod licencí GNU GPL. GUI program je napsáno v knihovně Gtk+ 2 a k dispozici jsou verze pro Linux, Windows, BSD a další operační systémy [15].

Funkce:

- Přidání kontaktů.
- Konference.
- Metakontakty.
- Dokáže hlídat kontakty (připojení, návrat z nečinnosti apod.).
- Přenositelná (portable) verze.
- Kompletně v češtině [15].

Pidgin slouží jako klient pro několik komunikačních protokolů, které umožňuje uživateli využívat zároveň. Program je možno rozšířit množstvím pluginů. Pidgin podporuje textovou komunikaci ve stylu ICQ, posílání souborů přes některé protokoly, chaty a IRC kanály. Uložené kontakty je možno sledovat, Pidgin je schopen automaticky odeslat zprávu kontaktu při jeho připojení, upozornit uživatele, že se daný kontakt přihlásil, vrátil z nečinnosti, odhlásil se atp. Jednotlivá okna chatů, kanálů a rozhovorů mezi uživateli je možno sdružovat do jednoho okna se záložkami [15].

Pidgin umožňuje automatické zaznamenávání veškeré komunikace do historie, ve které je možno vyhledávat klíčová slova. Program podporuje Zprávy o nepřítomnosti i různá kódování textové komunikace [15].

Konfigurační soubory se nachází v adresáři *$HOME/.purple/*. Obsah jednotlivých podadresářů a souborů:

- podadresář *cerfificates*: obsahuje certifikáty pro zabezpečenou (šifrovanou komunikaci),
- podadresář *icons*: obsahuje ikony (avatary) uživatelů ze seznamu přátel,
- podadresář *logs*: obsahuje záznamy historie, každý uživatel má svůj podadresář s číslem účtu a v něm log s názvem identifikujícím datum rozhovoru,
- podadresář *smileys*: obsahuje dodatečně přidané témata smajlíků,
- soubor *accels*: automaticky generován, obsahuje definici cest,
- soubor *accounts.xml*: obsahuje obecné informace o účtu jako je protokol, číslo účtu, heslo apod.,
- soubor *blist.xml*: obsahuje skupiny kamarádů,
- soubor *pounces.xml*: obsahuje pouze verzi,
- soubor *prefs.xml*: obsahuje globální konfiguraci programu (písmo, nastavení proxy apod.),
- soubor *status.xml*: obsahuje uložené stavy.

Zálohováno je vše kromě podadresáře *icons* (zbytečně by zvětšovalo velikost zálohy, automaticky se stáhne po spuštění aplikace) a automaticky generovanému souboru *accels*.

3.6 Rhythmbox

Rhythmbox je komplexní nástroj pro přehrávání a management hudby, je navržen podle programu iTunes firmy Apple a založen na frameworku Gstreamer. Mezi jeho základní funkce patří [9]:

- snadné procházení hudby na pevném disku
- hledání a třídění
- komplexní podpora audio formátů (díky Gstreameru)
- podpora internetových rádií včetně streamů last.fm
- fronta přehrávání
- audio visualizace
- přenos hudby z a do iPodu, MTP a USB hudebních přehrávačů
- zobrazování obalu alba a textu písně, automatické stahování z internetu
- přehrávání, konverze do mp3 formátu a vypalování audio CD

Rhythmbox je základní aplikací pro přehrávání audio souborů v prostředí GNOME. Jeho konfigurační adresář se nachází v */home/user/.gnome2/rhythmbox/* a obsahuje dva konfigurační soubory a podadresář s obrázky obalů jednotlivých alb, které jsou automaticky stahovány z internetu (pokud je tato funkce povolena). První konfigurační soubor s názvem *playlists.xml* obsahuje aktuální frontu k přehrávání, nejlépe hodnocené a nedávno přidané skladby. Druhý soubor *rhythmboxdb.xml* obsahuje kompletní soupisku hudebních souborů, přidaných do aplikace [9].

I přes malou velikost stáhnutých obalů alb jsem se rozhodl, vzhledem k možnosti existence jejich velkého počtu, k zálohování pouze konfiguračních souborů bez adresáře s obaly. Zbytečné navyšování velikosti záloh by mohlo mít negativní vliv na jejich přenositelnost a navíc se opětovné stáhnutí obrázků alb provede automaticky a vzhledem k jejich velikosti a rychlosti dnešního internetu i velmi rychle.

3.7 Screenlets

Screenlets jsou malé, jednoduché aplikace, napsané v Pythonu, které slouží k zobrazení informací o času, vytížení procesoru nebo disků a podobně na ploše. Při použití správce oken Compiz Fuzion je možno umístit jednotlivé mini aplikace na samostatnou vrstvu a tu v případě potřeby skrýt. Cílem projektu Screenlets je vytvoření jednoduché vývojové základny (základní třídy), která umožní jednoduchý a rychlý vývoj veškerých potřebných mini aplikací na plochu moderního linuxového prostředí [18].

Mezi základní kameny tohoto projektu patří body jako jednoduchý a rychlý vývoj, stovky hotových mini aplikací ke stáhnutí, plná spolupráce s prostředím Compiz a funkčnost i bez tohoto prostředí, vývoj pod licencí GPL a další [18].

Adresář s konfiguračními soubory se nachází v domovském adresáři uživatele ve skrytém souboru *.screenlets*, jeho obsahem jsou podadresáře s jednotlivými mini aplikacemi a soubor *config.ini* s globální konfigurací aplikace (zobrazování ikony v panelu GNOME, nastavení fyzikálních vlastností a další parametry).

Zálohovat budeme celý adresář, protože ruční opětovné stáhnutí všech mini aplikací by mohlo být, zvláště při jejich větším počtu, časově náročné. Návrhem na vylepšení by mohlo být nepřenášení všech mini aplikací z konfiguračního adresáře, nýbrž záloha pouze těch, které nejsou v nějaké centrální databázi (*gnome-look.org*) volně ke stažení a vytvoření souboru s výčtem těch chybějících. Při obnově ze zálohy by aplikace potřebné soubory stáhla a rozbalila na příslušné pozice. Vhodné by bylo také umístění volby pro zapnutí a vypnutí této funkce do nastavení.

3.8 Xine

Xine je multimediální přehrávač, přehrávající CD, DVD, VCD, dekódující video v mnoha různých formátech a také zobrazující streamované video. Jedná se o rychlý a výkonný přehrávač, fungující na rozličných platformách a šířený pod licenci GPL. Xine sám o sobě je knihovna, která je využívána mnoha aplikacemi (*Kaffeine*, *Totem* ...), přehrávajícími multimediální soubory. Funkčnost Xine je možno rozšířit pomocí různých pluginů [19].

Adresář s konfiguračními soubory se nachází v *$HOME/.xine* a zálohován bude tento adresář celý, kromě automaticky generovaného souboru *catalog.cache*.

3.9 Globální systémová konfigurace /etc

Adresář */etc* obsahuje velké množství souborů a to především systémové konfigurační soubory. V této práci uvedu jen některé základní, pro širší pohled je nutno prostudovat síťové konfigurační soubory, které vynechám a také manuálové stránky jednotlivých programů.

Adresář */etc/rc[0-5].d* resp. */etc/rcS.d* osahuje skripty, které se spouští při startu systému nebo se mění úroveň běhu systému (tzv. *runlevel*). Adresáře */etc/rc[0-5].d* obsahují skripty pro jednotlivé runlevely 0-6 a adresář */etc/rcS.d* obsahuje skripty, které se spouští vždy při startu systému, bezprostředně po zavedení jádra.

Pozn 1.: Tyto adresáře obsahují většinou pouze odkazy (*simlinky*) na skutečné skripty (popř. binární soubory), které jsou většinou v adresáři */etc/init.d* - mají speciální název udávající, zda se jedná o start/kill, prioritu a název programu.

Pozn 2.: Počet runlevelů nemusí být vždy 6 (*viz. /etc/inittab*).

Soubor */etc/passwd* obsahuje databázi uživatelů systému s položkami, v nichž je uloženo uživatelské i skutečné jméno uživatele, domovský adresář, šifrované heslo a některé další informace. Formát uložených dat je popsán v manuálové stránce programu *passwd*.

Pozn.: V moderních systémech se šifrovaná hesla ukládají do souboru */etc/shadow*, soubor *passwd* obsahuje všechny ostatní informace.

Soubor */etc/fdprm* obsahuje tabulku parametrů disketové jednotky a popisuje jak vypadají různé formáty disket.

Soubor */etc/fstab* obsahuje seznamy souborových systémů, připojovaných automaticky při startu systému příkazem *mount -a* (většinou v */etc/rc*) a také informace o odkládacích oblastech.

Soubor */etc/group* je velmi podobný souboru */etc/passwd*, ale místo uživatelů popisuje pracovní skupiny. Formát uložených dat je popsán v manuálové stránce programu *group*.

Soubor */etc/inittab* určuje chování systému při bootování. Obsahuje např. číslo defaultního runlevelu, které určuje do kterého runlevelu se má nabootovat. Dále obsahuje informace, jak se má systém chovat při výpadku napájení. Jedná se o textový soubor, upravující chování procesu *init* (*/sbin/init*). *Init* je první proces, spouštěný jádrem po bootování, jeho PID je vždy 1. V souboru *inittab* je obvykle, definováno spouštění konzolí a skriptů jednotlivých runlevelů.

Soubor */etc/issue* obsahuje výstup programu *getty*, který se zobrazí před výzvou přihlášení uživatele. Obvykle obsahuje stručný popis systému nebo uvítací hlášení. Obsah určuje správce systému.

Soubor */etc/magic* je konfigurační soubor programu file, který obsahuje popisy různých formátů souborů, podle kterých pak program file tyto typy rozpoznává.

Soubor */etc/motd* je tzv. zpráva pro tento den – automatický výstup na terminál uživatele po úspěšném přihlášení do systému. Obsah volí správce systému a často se využívá k předávání informací všem uživatelům, např. upozornění na plánovanou výluku systému.

Soubor */etc/mtab* uchovává seznam aktuálně připojených souborových systémů. Jeho obsah po zavedení systému a připojení určených souborových systémů prvotně nastavují inicializační skripty, v běžném provozu pak automaticky příkaz mount. Tento soubor se používá v případech, kdy je potřeba zjistit, které souborové systémy jsou připojené, např. při zjišťování volného místa *df -h*.

Soubor */etc/shadow* je databáze stínových hesel uživatelů systémech, které mají nainstalovanou podporu stínových hesel. Kódovaná bezpečnostní hesla jsou z bezpečnostních důvodů přenesena ze

souboru */etc/passwd* do souboru */etc/shadow*, ze kterého může číst pouze superuživatel. Snižuje se tak pravděpodobnost odhalení některého z přístupových hesel.

Soubor */etc/login.defs* je konfiguračním souborem příkazu login.

Soubor */etc/printcap* je databází vlastností tiskáren.

Soubory */etc/profile*, */etc/csh.login*, */etc/csh.cshrc* jsou spuštěny při přihlášení uživatele nebo startu systému interprety příkazů *Bourne shell* nebo *C shell*. Umožňují správci systému stanovit globální nastavení stejná pro všechny uživatele.

Soubor */etc/securitty* identifikuje zabezpečené terminály, tedy ty, ze kterých se může přihlašovat superuživatel. Typicky jsou v seznamu uvedeny pouze virtuální konzoly, takže je nemožné získat oprávnění superuživatele přihlášením po modemu nebo ze sítě. Nepovoluje přihlášení superuživatele přes síť, rozumnější (a z hlediska bezpečnosti lepší) je přihlásit se jako běžný uživatel a pak získat superuživatelská práva příkazy *su* nebo *sudo*.

Soubor */etc/shells* udržuje seznam důvěryhodných interpretů příkazu.

Soubor */etc/termcap* je databáze vlastností terminálu a opisuje kterými escape sekvencemi se řídí různé typy terminálů.

Zdroj: [1]

3.10 Inicializační soubor shellu .bashrc

Soubor *.bashrc* je inicializačním souborem shellu, jedná se o uživatelský soubor, tzn. každý uživatel má svůj vlastní. Nachází se v domovském adresáři. Obsahem tohoto souboru jsou tzv. *aliasy*, tedy přezdívky a systémové proměnné.

Přezdívkou je myšlena úprava původního příkazu do formátu s rychlejším zápisem nebo lepší možností správného zápisu. Typickým příkladem je příkaz: *alias ll="ls -l"*, který funguje tak, že když zadáme příkaz *ll*, shell jej rozvine v příkaz *ls -l* a až poté interpretuje.

Systémové proměnné definují důležitá konfigurační nastavení systému. Typickým příkladem jsou proměnné *HOME* (domovský adresář uživatele), *USER* (jméno uživatele) nebo *TERM* (typ terminálu). Seznam všech systémových proměnných lze získat příkazem *env*.

Zdroj: [1]

4 Popis aplikace

V této části práce odůvodňuji použití jednotlivých nástrojů, tedy použití jazyka Python a knihoven GTK+. Dále popisuji princip fungování aplikace, tedy způsob vytváření a obnovy záloh a zmiňuji některá implementační fakta a použitá řešení.

4.1 Implementační jazyk, vývojové nástroje pro tvorbu GUI

Jako implementační jazyk jsem po dohodě s vedoucím práce zvolil Python a to i přes původní návrh implementovat tento projekt v jazyku C, protože co možná nejvyšší rychlost a výkonnost aplikace zde není na místě, nýbrž jsou, podle mého názoru, potřeba dobré možnosti skriptování a možnost efektivního vývoje.

Vzhledem k povaze a orientaci Bakalářské práce jsem grafické prostředí aplikace vyvíjel pomocí knihovny GTK+, respektive PyGTK. GTK+ bylo zvoleno z důvodu použití této knihovny v drtivé většině aplikací, nativně vyvíjených pro prostředí (window manager) GNOME, z důvodu běhu celého prostředí na této knihovně a z důvodu situování programu (dle zadání Bakalářské práce) výlučně do tohoto prostředí. Zároveň však nevylučuji funkčnost programu i v rozdílném prostředí a to za předpokladu existence knihoven, potřebných pro běh. Nutnou podmínkou pro úspěšné fungování aplikace je její spuštění výhradně v unixovém operačním systému, a to z důvodu odlišného způsobu ukládání konfiguračních souborů aplikací v jiných operačních systémech a dále z důvodu závislosti aplikace na některých nástrojích z balíku aplikací GNU. Aplikace je bez problému spustitelná na školním serveru *eva.fit.vutbr.cz*, který běží na operačním systému FreeBSD .

K návrhu GUI aplikace byl použit program *Glade Interface Designer*, který umožňuje grafický návrh rozhraní aplikace a automatické vygenerování XML souboru s popisem tohoto rozhraní. Tento soubor je následně pomoc modulu Pythonu *gtk.glade* dynamicky načten (za běhu aplikace). Výhodou takového přístupu je snadná úprava grafického rozhraní a možnost načtení i do jiných programů, psaných odlišným jazykem. Vývoj také velmi zefektivňuje existence velkého množství funkcí v modulu *gtk* pro změnu a zjištění stavu widgetů v XML souboru.

4.2 Princip fungování aplikace

Aplikace funguje jak v příkazové řádce, tak s grafickým uživatelským rozhraním a disponuje dvěma základními funkcemi – vytvářením a obnovou záloh. Při běhu s grafickým uživatelským prostředím běží jádro aplikace a grafické rozhraní zvlášť, je programově ošetřeno, že při ukončení jedné části dojde i k ukončení části druhé.

4.2.1 Vytváření záloh

V příkazové řádce jako první parametr zadáme přepínač -s (--save), který určuje vytváření záloh a vybereme aplikace, které chceme zálohovat pomocí dalších parametrů (bez zadání parametrů aplikace zálohuje vše). Aplikace zpracuje zadané parametry a nastaví příslušné proměnné.

Při použití GUI se aplikace vybírají pomocí zaškrtnutí příslušného tlačítka (checkboxu). Aplikace, s kterými pracujeme, jsou poté definovány nastavení parametru active widget na hodnotu True.

Po spuštění úlohy jsou příslušné konfigurační soubory zabaleny pomocí programů tar a gzip, vygenerováno jméno a záloha je uložena do aktuálního adresáře. Možným rozšířením funkčnosti aplikace by byla implementace možnosti volby cílového adresáře nebo možnosti uploadu zálohy např. na ftp server.

4.2.2 Obnova záloh

V příkazové řádce jako první parametr zadáme přepínač -r (--restore), který určuje obnovu zálohy a vybereme aplikace, které chceme obnovit pomocí dalších parametrů (bez zadání dalších parametrů aplikace obnoví vše ze zálohy). Aplikace zpracuje zadané parametry a nastaví příslušné proměnné.

Při použití GUI se pomocí menu přepneme na obnovu zálohy. Bezprostředně po stisknutí příslušného tlačítka je spuštěn skript, který vyhledá v aktuálním adresáři soubory, jejichž formát odpovídá formátu záloh a podle přihlášeného uživatele a hodnoty ID v názvu zálohy vybere tu nejnovější, vytvořenou přihlášeným uživatelem. Získaný název poté vyplní do widgetu výběru souboru.

Vyplněním widgetu výběru souboru je zaslán signál, upozorňující na tuto skutečnost. Záloha je pomocí programu tar analyzována a je zjištěn její obsah. Podle obsahu jsou zaškrtána příslušná tlačítka, umožňující obnovu zálohy příslušných aplikací. Tlačítka, definující obnovu zálohy ostatních aplikací jsou šedá, manipulace s nimi není umožněna a obnova těchto aplikací není možná.

Po spuštění akce pomocí tlačítka OK jsou ze záloh rozbaleny vybrané aplikace na příslušné místa. Původní konfigurační soubory jsou automaticky nahrazeny.

4.2.3 Rozdíly ve funkčnosti aplikace spuštěné s a bez GUI

Z důvodu jednoduchého ovládání aplikace i ze shellu a také přehledné a jasné nápovědy není umožněna záloha ani obnovení pouze části zálohy adresáře /etc, jak je tomu při spuštění s grafickým rozhraním. Danou funkci jsem neimplementoval, protože řešením by bylo zadávání parametrů parametru parametru programu (tři úrovně parametrů programu) nebo vytvoření nějakého dotazovacího mechanizmu, který by byl komplikací při použití této aplikace jako součást nějakého skriptu. Samozřejmě i toto by se dalo řešit, avšak implementace takové funkce není podle mého názoru v současnosti nezbytná a může být námětem dalších verzí programu.

Aplikace, spuštěná v shellu, obecně také méně zatěžuje procesor a má menší paměťové nároky. K většímu vytěžování procesoru při použití GUI dochází především při funkcích, usnadňujících uživateli práci s aplikací. Například při kliknutí v hlavním menu na Backup → Restore backup dojde k výpisu všech souboru v aktuálním adresáři, vybrání záloh a z nich je vybrána ta nejnovější, vytvořená uživatelem a její název je vložen do pole pro vybrání zálohy, které má být obnovena. Vliv na celkový spotřebovaný čas procesoru má samozřejmě také vykreslování jednotlivých oken, menu apod. Na velikost alokované operační paměti má vliv především vykreslení obrázků při použití GUI a také spuštění služeb systému (výpisy adresářů apod.).

Protože mě zajímal rozdíl nároků při použití aplikace s a bez GUI, provedl jsem s pomocí programu *time* experimentální měření. Nejprve jsem vytvořil zálohu všech možných aplikací, kterou jsem posléze obnovoval oběma způsoby tak, aby bylo měření co nejpřesnější (žádné zbytečné operace při použití GUI). Výsledky jsou uvedeny v následující tabulce 1.

Tabulka 1: výstup programu time

	real	user	sys
shell	0m2.679s	0m0.644s	0m0.256s
GUI	0m22.621s	0m7.096s	0m1.120s

Z měření vychází řešení s GUI jako daleko náročnější na systémové prostředky, spotřebovaný čas procesoru se navíc bude nadále zvyšovat s přibývajícím počtem záloh v adresáři, proto by námětem na další verzi programu mohla být možnost vypnutí automatického předvyplnění vybírané zálohy. Z pohledu celkového trvání operace v reálném čase nedošlo k žádnému překvapení, používání

GUI je obecně pomalejší a k propastnému rozdílu také přispívá to, že program time měří reálný čas od spuštění aplikace, takže nezohledňuje výběr parametrů a zápis celého příkazu.

Z celkového pohledu pracuje, podle mého názoru, aplikace velmi svižně a to i při použití na slabších počítačích.

4.3 Způsob pojmenování záloh

Zálohy jsou pojmenovány systematicky, tak aby byly snadno a jednoznačně identifikovatelné uživatelem i počítačem a přitom neporušovaly zásady vhodného pojmenovávání souborů (v názvu se nevyskytují mezery, lomítka, dvojtečky ani jiné další nevhodné znaky).

Systém pojmenovávání je následující:

př: `GSMUbackup[user]-02.03.2009_14h29m58s_ID1236000598.tar.gz`

Název se skládá z několika částí:

- nápis `GSMUbackup` (zkr. Gnome Settings Migration Utility)
- `$USER`: jméno uživatele, jehož data jsou zálohována (v hranatých závorkách)
- datum ve formátu `DD.MM.RRRR`
- čas ve formátu `HHhMMmSSs`
- ID zálohy, systémový čas (počet sekund od 1.1.1970)
- koncovka `tar.gz`

Datum je poskládáno z dne (*%d*), měsíce (*%m*) a roku (*%y*). Záměrně nebyla použita sekvence, definující datum podle lokalizace (*%X*), protože v některých případech by v takovémto datu mohly být použity nevhodné znaky, které by byli shellem chybně interpretovány. Např. na serveru *eva.fit.vutbr.cz* po použití sekvence *%X* dojde k vygenerování data ve formátu *DD/MM/RRRR*, které je pro pojmenování souboru krajně nevhodné, shell bude interpretovat znak „/“ jako konec názvu adresáře (cestu). Oproti tomu na mém počítači s českou lokalizací systému dochází ke generování data ve formátu *DD.MM.RRRR*, který je pro pojmenování souborů vhodný více.

4.4 Implementační fakta

V této části uvedu některá implementační řešení, použité v aplikaci, tzn. jakým způsobem jsem řešil některé komplikovanější části nebo jak jsem při řešení uvažoval.

4.4.1 Stavový řádek

V průběhu volání služeb operačního systému (např.: zabalování souborů a adresářů) je uživatel informován o průběhu činnosti aplikace pomocí stavového řádku. Vzhledem k povaze projektu a konkrétním volaným službám nebylo nutné implementovat volání služeb operačního systému v samostatném podprocesu, vytvořeného například pomocí funkce *fork()*. Místo toho jsou tyto služby volány v aktuálním procesu, funkce tlačítek je blokována a uživatel je o průběhu činnosti informován pomocí příkazového řádku.

4.4.2 Zpracování parametrů příkazové řádky

Aplikace po spuštění prochází předané parametry, k tomu využívá knihovny *getopt*. Při nalezení nesmyslného parametru vytiskne nápovědu a ukončí se, pokud se jedná o parametr, určující akci, zvolenou akci vykoná. Pokud není zadán žádný parametr, spustí aplikaci s grafickým uživatelským rozhraním.

Pokud je zadán parametr nějaké akce v příkazové řádce (ukládání nebo obnova zálohy), aplikace otestuje, zda byly předány parametry, definující konkrétní aplikace. Pokud ano, dochází ke zpracování tohoto parametru a hledání konkrétních aplikací. To je implementováno tak, že je pomocí regulárního výrazu v parametru hledáno konkrétní písmeno, určující aplikaci. Při nálezu je nastavena proměnná, definující výskyt aplikace, na *True*.

4.4.3 Vytváření seznamu aplikací pro zálohování

Každá aplikace má svojí proměnnou typu string s názvem *<jméno aplikace>PATH*, která obsahuje výčet absolutních cest k souborům a adresářům, které mají být zálohovány. Dále je v programu proměnná taktéž typu string s názvem *commandPATH*, která udržuje výčet všech aplikací k zálohování (všech jejich cest). Před samotným zálohováním je v cyklu zjišťováno zda má být daná aplikace zálohována a pokud ano, do proměnné, udržující seznam všech takových aplikací je pomocí konkatenace řetězců přidána požadovaná cesta. Po průchodu všemi aplikacemi dochází k zálohování samotnému, kdy je programu tar jako parametr file předán kompletní výčet cest.

Ekvivalentním řešením na úrovni programu tar by bylo postupné přidávání jednotlivých aplikací do archivu, což tar umožňuje, avšak volání služeb systému nepatří z pohledu systémových zdrojů k nejlevnějším činnostem, proto jsem implementoval raději systém, uvedený v předchozím odstavci.

U složky /etc, definující konfiguraci systému, dochází k zálohování jen některých částí. Zvláštním požadavkem byla záloha inicializačních souborů, proto jsem tento adresář rozdělil na inicializační soubory (každý zvlášť) a ostatní soubory a adresáře. Výčet souborů a adresářů se může měnit, proto program zjistí výčet obsahu tohoto adresáře a pomocí regulárního výrazu vybere jeho požadovanou část.

Pokud je do pole vstupního textu (*Option files(s)*) vyplněna cesta k volitelnému souboru nebo adresáři, je daný text pomocí konkatenace řetězců připojen k proměnné, určující aplikace k zálohování. Cesty volitelných aplikací k zálohování je nutno oddělovat mezerou, jinak program *tar* neinterpretuje danou cestu správně.

4.4.4 Zjišťování obsahu zálohy

Když jsem uvažoval jak řešit zjišťování obsahu záloh, prvotní myšlenkou samozřejmě bylo rozbalení zálohy do adresáře /tmp, zjištění jejího obsahu a její smazání. Toto řešení by bylo mimořádně neefektivní, proto jsem důkladně prostudoval manuál k programu tar a objevil parametr -t, který umožňuje vypsat obsah archivu (při přidání parametru -z i archivu, zabaleného programem gzip). Obsah zálohy načítám do proměnné přímo (pomocí volání služeb operačního systému) a s ní dále pracuji.

Předchozím způsobem získávám proměnnou s výčtem obsahu zálohy, jeden soubor se svojí absolutní cestou na každém řádku. Množství souborů v záloze může být velmi velké, proto je nutné pracovat s proměnnou obsahu zálohy velmi obezřetně a co nejvíce optimalizovat.

Zjištění konkrétních aplikací, nalézajících se v záloze, jsem prováděl tak, že jsem určil klíčová slova, jednoznačně identifikující každou aplikaci a při postupném procházení obsahu zálohy (po řádkách) jsem pomocí regulárního výrazu toto klíčové slovo hledal. Klíčové slovo není nic víc, něž např.: „\.*firefox*“, „\.gimp-[0-9].[0-9]“ nebo „/etc/rcS\.d“ (zápis pomocí konkrétních regulárních výrazů), díky těmto výrazům se dá správně identifikovat konkrétní řádek. Zobecňující klíčové slovo vyžadovala záloha adresáře /etc, kdy bylo možno zálohovat některé části a „zbytek“ tohoto adresáře. Tento „zbytek“ musel být nějak identifikován a proto jsem jako jeho zástupce zvolil podadresář *acpi*, který je vždy součástí adresáře /etc. Při zaškrtnutém políčku *all other* jako součást adresáře /etc při použití GUI aplikace dochází k identifikaci při výskytu /etc/acpi a poté k rozbalení a to pomocí parametru programu tar *exclude*, kdy je rozbalen celý adresář /etc vyjma inicializačních rc-souborů.

Optimalizaci jsem prováděl pomocí proměnné *founded*, která zamezovala testování řádku, který již byl přiřazen jiné aplikaci, tzn. každý řádek je testován na klíčové slovo aplikace, dokud není nalezena, poté se testuje další řádek. Dále pokud je výskyt aplikace potvrzen, nedochází k jejímu

opětovnému hledání. Tímto postupem jsem se snažil snížit počet kroků cyklu a testování podmínek na minimum.

Pokud je zatrženo zaškrtávací tlačítko obnovy volitelně přidaných aplikací, je ze zálohy obnoveno vše kromě předvolených aplikací. Podnětem k dalším vylepšením by mohlo být zobrazení přehledu volitelných aplikací, uložených v záloze. Toto by se dalo implementovat například pomocí konfiguračního souboru s metadaty archivu, přibaleného v záloze.

4.4.5 Obnova zálohy programu Gimp

Program Gimp ukládá svoje konfigurační soubory na standardní umístění do domovského adresáře uživatele, avšak do adresáře *gimp-x.y*, kde *x.y* je číslo verze aplikace. Konfigurační soubory jsou zpětně kompatibilní.

Obnovu této části zálohy jsem rozdělil do několika kroků a to:

1) zjištění verze aplikace v záloze
2) rozbalení adresáře s konfiguračními soubory
3) test, zda je v domovském adresáři více adresářů gimp-x.y (různé verze)
4) pokud ano, zkopírování obsahu adresáře ze zálohy do druhého adresáře (novější nebo starší verze) a smazání adresáře ze zálohy

Tento postup se možná zdá zbytečně složitý, ale i přes důkladné hledání se mi nepodařilo zjistit jak pomocí taru rozbalit obsah adresáře - pouze obsah, bez cesty. Předpokládám, že tato funkce není v programu tar implementována.

4.4.6 Způsob kontroly práv při práci s adresářem /etc

S adresářem /etc může manipulovat pouze superuživatel a proto i v aplikaci pro přenos nastavení systému musí být tento fakt zohledněn.

Při použití aplikace bez grafického uživatelského rozhraní proběhne nejprve zpracování parametrů, kdy se testuje jaká činnost byla zvolena a případná specifikace aplikací k zálohování/obnově ze zálohy. Každá aplikace má svojí booleovskou proměnnou, určující, zda byl zadán její přepínač, tedy zda s danou aplikací chceme pracovat. Před prací s adresářem /etc proběhne test na uživatele, který aplikaci spustil, pokud se jedná o superuživatele, je umožněna záloha i obnova zálohy adresáře /etc. Pokud se pokouší obyčejný uživatel o práci s adresářem /etc je upozorněn na

nedostatečná práva výpisem na standardní datový výstup a provedou se všechny jím zvolené akce nad zvolenými soubory vyjma tohoto adresáře.

Při použití aplikace s grafickým uživatelským rozhraním jsou aplikace, se kterými chceme pracovat, určeny nastavením parametru *active* zaškrtávacího tlačítka na hodnotu *True* v XML souboru, definujícím vzhled aplikace. Každá zálohovaná aplikace má svoje zaškrtávací tlačítko. Při přepínání v menu jsou vždy všechna tlačítka nastavena na výchozí hodnotu (parametr *active* i *sensitive*) a zaškrtávání už řeší glade knihovny samy (voláním funkcí *set_active()* a *set_sensitive()* nastavujeme příslušné parametry). Při spuštění akce tlačítkem OK už jenom program ověří, které aplikace uživatel zaškrtal (funkce *get_active()*) a s nimi se posléze pracuje. Základem je správné nastavování výchozích hodnot při přepínání se v aplikaci. Po spuštění je aplikace v módu uložení zálohy a při spuštění superuživatelem umožňuje interakci s tlačítkem etc a tlačítky, určujícími jednotlivé části adresáře /etc. V opačném případě jsou tlačítka prošedlá a práce s nimi není umožněna.

Dalším možným omezením by bylo povolení obnovy pouze zálohy vytvořené stejným uživatelem, který ji chce obnovit. Toto omezení by se ovšem opíralo pouze o jméno, uvedené v názvu zálohy a které lze jednoduše změnit. Navíc by se toto omezení minulo účinkem, protože v některých situacích může být vhodné obnovovat cizí zálohu, případně při změně uživatelského jména (například z *Franta* na *František*) dokonce nutné.

4.4.7 Interakce s uživatelem - nastavování modifikovatelnosti a stavu zatržení tlačítek

Velmi důležitým aspektem aplikace z pohledu přehlednosti a jednoduché a intuitivní práce uživatele je správné nastavování modifikovatelnosti a stavu zatržení tlačítek. Aplikace musí vhodně nastavovat stavy jednotlivých tlačítek tak, aby pozitivně působily na pocit uživatele z práce s aplikací.

Způsob, který jsem použil a který mi přišel nejuniverzálnější a nejlépe odolný modifikacím zdrojového kódu se odrážel od nastavování výchozích hodnot při vhodných okamžicích, jako je například přepnutí v menu.

Základní nastavení, které má být zobrazeno hned po spuštění aplikace, je definováno již v XML souboru s definicí GUI, při postupné práci dochází k jeho modifikaci. Při přepnutí v menu jsou vždy všechna tlačítka nastavena na výchozí hodnoty, kdy jsou „prošedlá“ ta, s kterými není interakce uživatele povolena a stav zaškrtnutí je „nezaškrtnuto“. Při zaškrtnutí tlačítka, majícího další podtlačítka, určující soubory a podadresáře v daném adresáři, dochází k zaškrtnutí všech podtlačítek, tato situace platí i naopak při odškrtnutí tohoto tlačítka.

Při výběru zálohy, která má být obnovena dochází k zaškrtnutí tlačítek, určujících jaké aplikace záloha obsahuje. Pokud se jedná o tlačítko s podtlačítky, jsou zatrženy části aplikace, obsažené v záloze, ale při odškrtnutí „rodičovského“ tlačítka nedojde k jejich odškrtnutí, nýbrž pouze k jejich „prošednutí“. Toto chování je z důvodu udržení informace o obsahu zálohy.

4.4.8 Zálohování volitelných souborů a adresářů

Aplikace umožňuje zálohování volitelných souborů a adresářů, tato funkce je přístupná pouze při použití GUI. Ve spodní části aplikace se nachází widget textového vstupu, který slouží k vepsání absolutní cesty nebo více cest k souboru/adresáři, oddělených mezerou. Obsah tohoto pole je předán jako parametr programu *tar* při samotném vytváření zálohy a dané soubory/adresáře jsou přibaleny do archívu.

Detekce volitelných souborů v archívu není v současné verzi implementována, možnost rozbalení volitelných souborů je prostřednictvím zaškrtávacího tlačítka, které na rozdíl od ostatních tlačítek stejného typu není zatrženo podle obsahu dané části v archivu, ale jeho stav je vždy „zaškrtnutelný“ (sensitive). Samotné rozbalení volitelných souborů se provádí pomocí parametru *exclude* programu *tar*, kdy je rozbaleno z archivu vše, kromě předdefinovaných souborů a adresářů. Jak vyplývá z předchozího souvětí, rozbalení jen některých volitelně zabalených souborů není umožněno, vždy je rozbaleno vše nebo nic.

5 Závěr

V programu je poměrně velký výběr aplikací k zálohování, včetně možnosti zálohy o obnovy uživatelsky definovaných aplikací. Jako kladný krok hodnotím rozšíření o funkčnost bez grafického rozhraní, i když takovéto použití má v současné verzi jistá omezení. Aplikace pracuje velmi svižně a ovládání je velmi jednoduché a přehledné.

Z pohledu teorie jsem se, díky proniknutí do problematiky, dozvěděl mnoho zajímavých informací a faktů o unixových operačních systémech a jejich historii. Z pohledu praktického jsem si vyzkoušel jaké to je, vyvíjet kompletní software, testovat běh na více platformách a ošetřovat nejrůznější výjimky.

Námětů na další rozšíření je mnoho, některé jsme již uvedl v předešlém textu. Jedná se například o přidání sekce nastavení do hlavního menu, kde by bylo možno vypínat a zapínat některé funkce programu (např.: automatické předvyplňování formuláře výběru souboru při obnově dat) a implementace možnosti výběru části adresáře /etc i ostatních aplikací k zálohování, při použití v příkazové řádce.

Jedním z větších námětů k rozšíření by byla úprava zdrojového textu pro snadnější přidávání programů k zálohování (rozdělení do tříd, automatizace některých mechanizmů) a případně umožnění uživateli samostatně si libovolné programy přidávat a to buď stylem výběru libovolného souboru/adresáře pomocí widgetu výběru souboru a nebo jednoduchou úpravou celého grafického rozhraní (po výběru příslušného souboru/adresáře by bylo automaticky vytvořeno zaškrtávací tlačítko v dialogu zálohování). Toto řešení by také muselo nutně obsahovat možnost dynamického vytváření zaškrtávacích tlačítek na základě obsahu zálohy.

Pro případné geografické rozšíření programu by bylo také vhodné oddělit popisky a texty v aplikace od zdrojového kódu na úroveň např. databáze, aby byla snadnější a efektivnější úprava aplikace pro použití v jiných jazycích.

Aplikaci je díky volitelné funkčnosti i bez GUI možno začlenit jako součást skriptu pro automatické zálohování, kdy je její použití, především pro méně zkušené uživatele, jednodušší než samotné používání program tar a gzip.

Seznam použitých zkratek

API – Aplication Programming Interface

GNOME – GNU Network Object Model Environment

GNU – GNU's Not Unix

GUI – Graphical User Interface

GTK+ - The GIMP Toolkit

SGML – Standard Generalized Markup Language

XML – Extensible Markup Language

Literatura

[1] Kolektiv autorů: *Linux dokumentační projekt, 3. aktualizované vydání.* Brno, Computer Press, 2003, ISBN 80-7226-761-2

[2] Kolektiv autorů: *Linux dokumentační projekt, 4. aktualizované vydání.* Brno, Computer Press, 2008, ISBN 978-80-251-1525-1

[3] Tomáš Kašpárek, Radek Kočí, Petr Peringer, Tomáš Vojnar: *IOS – Studijní opora*. Brno, 2007.

[4] Livingston, B., Thurrott, P.: Windows Secrets News [online]. 2.12.2004 [cit. 9.3.2009]. Dostupný z WWW: <http://windowssecrets.com/comp/041202>

[5] Walsh, N.: A Technical Introduction to XML [online]. 3.10.1998 [cit. 18.3.2009]. Dostupný z WWW: <http://www.xml.com/pub/a/98/10/guide0.html>

[6] Hipp, R.: knihovna SQLite [online]. 4.2.2009 [cit. 18.3.2009]. Dostupný z WWW: <http://www.sqlite.org>

[7] Žák, K.: Historie OS Unix [online]. 28.6.2001 [cit. 19.3.2009]. Dostupný z WWW: <http://www.root.cz/clanky/historie-os-unix/>

[8] Fry, S.: The GNU Operating System [online]. 14.3.2009 [cit. 19.3.2009]. Dostupný z WWW: <http://www.gnu.org>

[9] Matthew, J.: Rhythmbox – The music management application for GNOME [online]. 25.3.2009 [cit. 25.3.2009]. Dostupný z WWW: <http://projects.gnome.org/rhythmbox/>

[10] Staringlnacke, M.: GNOME Commander [online]. 28.6.2008 [cit. 25.3.2009]. Dostupný z WWW: <http://www.nongnu.org/gcmd/index.html>

[11] The GIMP Documentation Team: GNU Image Manipulation Program [online]. 2009 [cit. 25.3.2009]. Dostupný z WWW: <http://docs.gimp.org/en/>

[12] Kuchling, A.: Python documentation [online]. 12.3.2009 [cit. 16.3.2009]. Dostupný z WWW: <http://www.python.org>

[13] The GTK+ Team: The GTK+ Project [online]. 2007 [cit. 8.4.2009]. Dostupný z WWW: <http://www.gtk.org>

[14] PyGTK Team: About PyGTK [online]. 7.4.2006 [cit. 11.4.2009]. Dostupný z WWW: <http://www.pygtk.org>

[15] Pidgin Team: About Pidgin [online]. 2009 [cit. 13.4.2009]. Dostupný z WWW: <http://pidgin.cz>

[16] Novell Inc.: Evolution [online]. 2008 [cit. 13.4.2009]. Dostupný z WWW: <http://projects.gnome.org/evolution/>

[17] Glade Team: About Glade [online]. 2009 [cit. 13.4.2009]. Dostupný z WWW: <http://glade.gnome.org/>

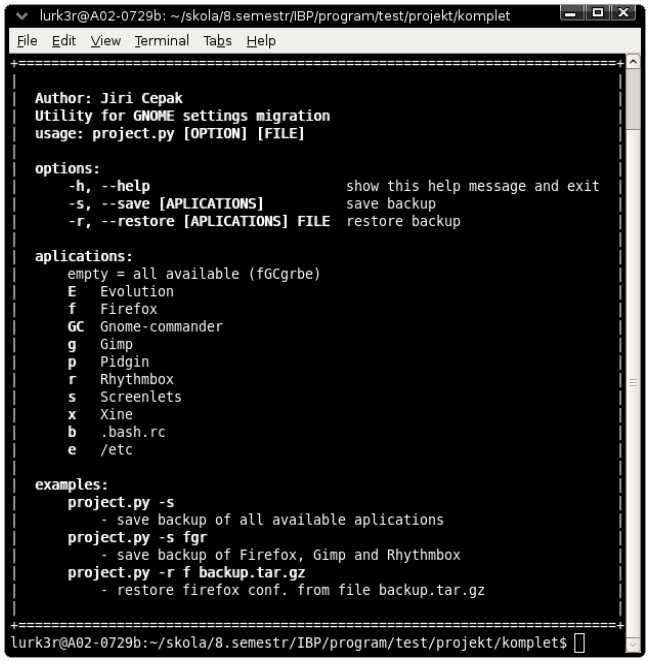
[18] Rico Pfaus: Screenlets Informations [online]. 2.4.2009 [cit. 20.4.2009]. Dostupný z WWW: <http://www.screenlets.org/index.php/Information>

[19] Xine Team: About Xine [online]. 2009 [cit. 20.4.2009]. Dostupný z WWW: <http://www.xine-project.org/about >

Seznam příloh

Příloha 1.: Vybrané snímky aplikace

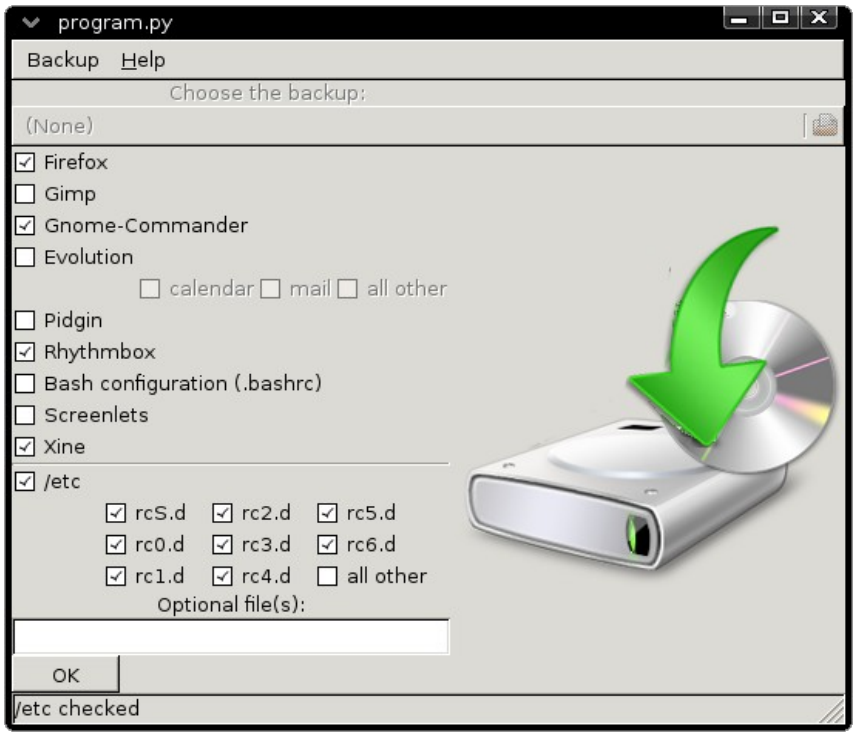
Na obrázku 1 je zobrazena nápověda aplikace pro použití v příkazové řádce. Toto okno se zobrazí při spuštění aplikace s parametrem -h nebo --help a také při zadání chybných parametrů a zobrazuje správné parametry a příklady použití.



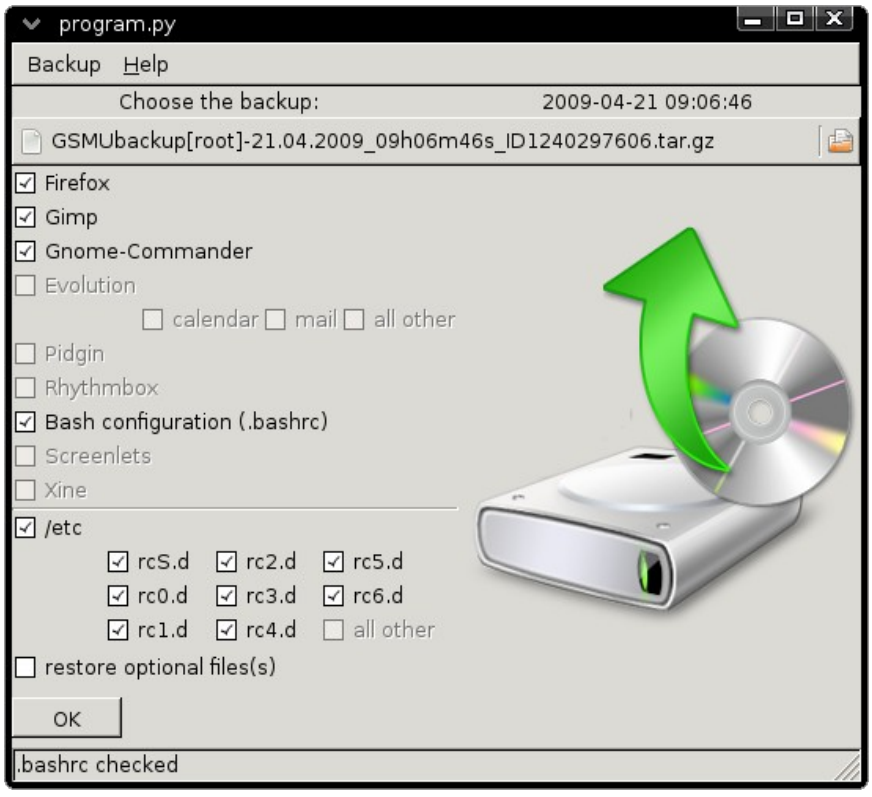
Obr. 1: nápověda aplikace pro použití v příkazové řádce

Na obrázku 2 je zobrazeno GUI při zálohování. Na vrchní části aplikace je hlavní menu s možností výběru zálohování/obnovy záloh a okna About. Výběr zálohy je prošedlý, prosvícen je až při obnově zálohy. Zaškrtávací tlačítka určují jaké aplikace je možno zálohovat, na snímku je GUI aplikace spuštěné se superuživatelskými právy a proto je možno zálohovat i adresář /etc. Při spuštění bez těchto práv je tato možnost prošedlá. Dále aplikace obsahuje tlačítko, které spustí zvolenou akci a stavový řádek, kde jsou zobrazovány informace o průběhu pro činnosti. GUI je doplněno obrázkem, doplňujícím vzhled aplikace a usnadňujícím orientaci uživatele (šipka směrem k CD/externímu hdd říká, že chceme zálohovat).

Na obrázku 3 je zobrazeno GUI při obnově zálohy. Ve vrchní části je zobrazeno datum a čas jejího pořízení a hned pod tímto údajem je vybraný soubor. Aplikace, které jsou prošedlé nejsou součástí zálohy a není je možno obnovit. Ostatní tlačítka, označující jednotlivé aplikace, jsou prosvícené a zaškrtnuté, to značí že je lze obnovit. Jedná se o snímek obrazovky programu, spuštěného se superuživatelskými právy. Obrázek šipky směrem od CD/externího hdd značí obnovení ze zálohy.



Obr. 2: GUI pro zálohování, spuštěno jako root – je možno zálohovat /etc



Obr. 3: GUI při obnovení zálohy-aplikace, neobsažené v záloze, jsou prošedlé