VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

SYNCHRONIZACE DOKUMENTŮ ULOŽENÝCH NA 3 A VÍCE POČÍTAČÍCH

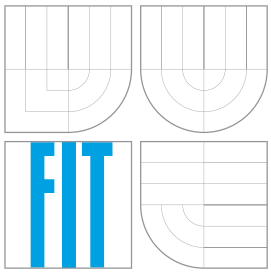
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

LUKÁŠ NYKRÝN

BRNO 2010

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

SYNCHRONIZACE DOKUMENTŮ ULOŽENÝCH NA 3 A VÍCE POČÍTAČÍCH

FILE SYNCHRONIZATION ON 3 OR MORE PCS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

LUKÁŠ NYKRÝN

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PETER JURNEČKA

BRNO 2010

Abstrakt

Práce pojednává o návrhu a implementaci aplikace pro sdílení souborů mezi více počítači za pomoci externího disku. Aplikaci je možné provozovat jak na systému GNU/Linux, tak na systému MS Windows. V prostředí systému Windows aplikace provádí synchronizace v pravidelných intervalech. V případě běhu v systému GNU/Linux nabízí navíc možnost hlídat stav souborů a ihned po každé změně soubor automaticky synchronizovat. Celý systém hlídá a ošetřuje veškeré možné kolizní stavy souborů.

Abstract

This document descripes the design and implementation of application for file sharing between multiple computers using an external drive. Application can be run on GNU/Linux and MS Windows. In Windows application performs synchronization at regular intervals. Under GNU/Linux offers extra choice to watch the status of files and immediately synchronize file after change. The whole system monitors and cares for all possible collision states of files that can arise.

Klíčová slova

synchronizace souborů, hlídání změn souborů na disku, detekce připojených externích disků, SQLite, inotify, HAL, wxWidgets

Keywords

file synchronization, report file changes, detection of connected extern disks, SQLite, inotify, HAL, wxWidgets

Citace

Lukáš Nykrýn: Synchronizace dokumentů uložených
na 3 a více počítačích, bakalářská práce, Brno, FIT VUT v Brně, 2010

Synchronizace dokumentů uložených na 3 a více počítačích

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Petera Jurnečky.

.....
Lukáš Nykrýn
16. května 2010

Poděkování

Děkuji svému vedoucímu Ing. Peteru Jurnečkovi za veškerou pomoc a rady, které mi poskytl během vypracování této práce.

Obsah

1 Úvod **4**
- 1.1 Synchronizace dat mezi počítači . . . 4
- 1.2 Obsah práce . . . 4

2 Zadání **5**
- 2.1 Popis aplikace . . . 5
- 2.2 Základní cíle aplikace . . . 5

3 Existující řešení **6**
- 3.1 Open source . . . 6
  - 3.1.1 Unison . . . 6
  - 3.1.2 Conduit . . . 7
  - 3.1.3 FreeFileSync . . . 7
  - 3.1.4 DirSync Pro . . . 8
- 3.2 Komerční programy . . . 9
  - 3.2.1 Allway Sync . . . 9
  - 3.2.2 Synchronize It! 3 . . . 9
  - 3.2.3 Dropbox . . . 10

4 Návrh **11**
- 4.1 Výběr implementačních prostředků . . . 11
  - 4.1.1 Programovací jazyk . . . 11
  - 4.1.2 Grafické uživatelské rozhraní . . . 11
  - 4.1.3 Ukládání trvalých dat . . . 12
  - 4.1.4 Ukládání dočasných dat . . . 13
  - 4.1.5 Práce se soubory . . . 13
  - 4.1.6 Sledování změn v souborovém systému u GNU/Linux . . . 13
  - 4.1.7 Detekce připojení a odpojení disků u GNU/Linux . . . 13
- 4.2 Návrh implementace . . . 14
  - 4.2.1 Synchronizace . . . 14
  - 4.2.2 Ukládání dat . . . 15
  - 4.2.3 Chování aplikace . . . 15
- 4.3 Implementační nástroje . . . 17
  - 4.3.1 Vývojové prostředí . . . 17
  - 4.3.2 Verzovací nástroje . . . 17

5 Implementace **18**
5.1 Synchronizace . . . 18
5.2 Práce se soubory . . . 20
5.3 Databáze . . . 21
5.4 Detekce připojených externích disků . . . 21
5.4.1 GNU/Linux . . . 21
5.4.2 Windows . . . 22
5.5 Spouštění synchronizace . . . 22
5.5.1 Na pokyn uživatele . . . 22
5.5.2 Automatické . . . 22
5.6 Grafické uživatelské rozhraní . . . 24

6 Testování a porovnání s podobnými programy **26**
6.1 Synchronizace složky s dokumenty . . . 26
6.2 Synchronizace velkého množství malých souborů . . . 27
6.3 Synchronizace malého množství velkých souborů . . . 28
6.4 Využití operační paměti . . . 29
6.5 Shrnutí . . . 29

7 Závěr **30**

A Obsah CD **32**

Seznam obrázků

3.1 Unison . . . 6
3.2 Conduit . . . 7
3.3 DirSync Pro . . . 8
3.4 Synchronize It! 3 . . . 9
3.5 Dropbox . . . 10

4.1 Návrh vzhledu hlavního okna aplikace . . . 12
4.2 ER diagram . . . 15
4.3 Sekvenční diagram . . . 16

5.1 Dialog pro přidání synchronizace . . . 19
5.2 Dialog pro řešení kolizí . . . 19
5.3 Třídy synchro a synchroWorker . . . 20
5.4 Třída dir . . . 20
5.5 Třídy sql a sqlTable . . . 21
5.6 Třída media . . . 22
5.7 Třídy synchroCtrl, fileWatch, mediaWatch . . . 23
5.8 Hlavní okno aplikace – svazky . . . 24
5.9 Hlavní okno aplikace – média . . . 24
5.10 Třídy mainWindow, MyTaskBarIcon, dialColl, dialSynAdd a dialExtAdd . . . 25

6.1 Synchronizace složky s dokumenty – první synchronizace . . . 26
6.2 Synchronizace složky s dokumenty – po změně souborů . . . 27
6.3 Synchronizace velkého množství malých souborů – první synchronizace . . . 27
6.4 Synchronizace velkého množství malých souborů – po změně souborů . . . 28
6.5 Synchronizace malého množství velkých souborů – první synchronizace . . . 28
6.6 Synchronizace malého množství velkých souborů – po změně souborů . . . 29
6.7 Využití operační paměti . . . 29

Kapitola 1

Úvod

1.1 Synchronizace dat mezi počítači

V současné době není neobvyklé, že člověk pracuje během dne na několika různých počítačích. Dopoledne jsem ve školní laboratoři, cestou domů pracuji na notebooku a večer si sednu k počítači doma. Samozřejmě, při práci na nějakém projektu, bych rád měl své soubory na všech třech strojích. Existuje několik metod jak toho docílit.

Můžeme si někde na internetu založit úložiště, kam si budeme své soubory ručně kopírovat, či využijeme nějaké služby, jako je například níže zmíněný Dropbox. To ovšem přináší tu nevýhodu, že pokud se dostanu do míst bez internetového připojení, moje data budou nedosažitelná.

Druhou možností je použít přenosný disk a soubory kopírovat na něj. Ovšem ruční kopírování je velmi nepohodlné, ale pokud postup automatizujeme nějakým „běžným kopírovacím nástrojem“ jako například rsync, dostáváme se do problémů v případě, že mezi synchronizacemi se upraví jak soubor na externím disku, tak v počítači a jednu verzi pak při kopírování bez varování ztratíme.

Řešením je tedy aplikace, která za nás bude automaticky kopírovat změněná data z počítače na externí disk s tím, že bude hlídat, aby takové problémy nenastaly.

1.2 Obsah práce

Tuto práci lze volně rozdělit do čtyř hlavních částí. Nejprve je v kapitolách „Úvod“ a „Zadání“ rozebrána modelová situace, která zobrazuje problém synchronizace souborů mezi více počítači a z něho je vyvozeno zadání a cíle této práce.

Druhá část popisuje některá existující řešení jak z oblasti open source, tak komerční aplikace. U každého programu jsou uvedeny základní informace a popis kladů a záporů.

Třetí část popisuje vyvíjenou aplikaci. Kapitola „Návrh“ seznamuje čtenáře s implementačními prostředky vhodnými pro vytvoření programu a zdůvodňuje použití konkrétních z nich. Navazuje oddíl navrhující chování aplikace. Sekce „Implementace“ osvětluje vnitřní strukturu celého programu a jeho interakci s uživatelem.

Poslední část zhodnocuje výsledek celé práce, ukazuje možnosti budoucího rozšíření a srovnává program s některými již existujícími.

Kapitola 2

Zadání

2.1 Popis aplikace

Aplikace má zajišťovat synchronizaci dat na více počítačích za pomoci přenosného disku (např. usb flash paměti). Bude se tedy jednat o program, který zvládá synchronizovat data mezi počítačem a externí pamětí, přičemž zohledňuje všechny možné stavy, které mohou během synchronizace nastat.

2.2 Základní cíle aplikace

- **Multiplatformnost**
  Aby bylo dosaženo co možná nejširšího uplatnění, měla by aplikace fungovat minimálně pod operačními systémy MS Windows a GNU/Linux, přičemž její chování by mělo být pod oběma OS shodné.

- **Jednoduchost ovládání**
  GUI aplikace a její nastavení musí být co možná nejjednodušší, aby se v nich uživatel neztrácel. Příliš velké množství různých voleb většinu běžných uživatelů od aplikace odradí a vzrůstá riziko, že kvůli špatnému nastavení dojde ke ztrátě dat.

- **Automatický běh**
  Aplikace by měla během svého běhu vyžadovat minimum zásahů uživatele. V ideálním případě by měl uživatel synchronizace na začátku nastavit a pak být pouze upozorňován pokud vznikne nějaký problém. Pokud synchronizace bude probíhat automaticky a ne na pokyn uživatele, přináší to tu výhodu, že uživateli se nemůže stát, že synchronizaci zapomene spustit.

Kapitola 3

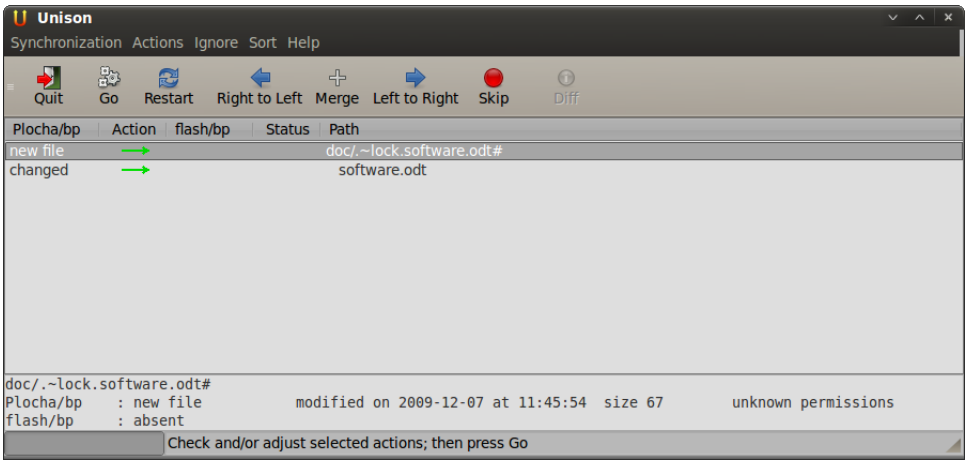
Existující řešení

3.1 Open source

3.1.1 Unison

- Licence: GPL
- Programovací jazyk: OCaml
- Platformy: Windows, GNU/Linux, OS X
- Detekce kolizí: Ano
- Poslední verze: 2.27.57 (20.1.2008)
- Adresa: http://www.cis.upenn.edu/~bcpierce/unison/

Unison je multiplatformní aplikace pro synchronizaci dat na lokálně připojených discích nebo po síti přes protokol ssh. Má velmi dobře řešené grafické rozhraní, které před synchronizací zobrazuje jaké akce se provedou a uživatel má možnost, kteroukoliv z nich ovlivnit. V současné době již není dál vyvíjen.

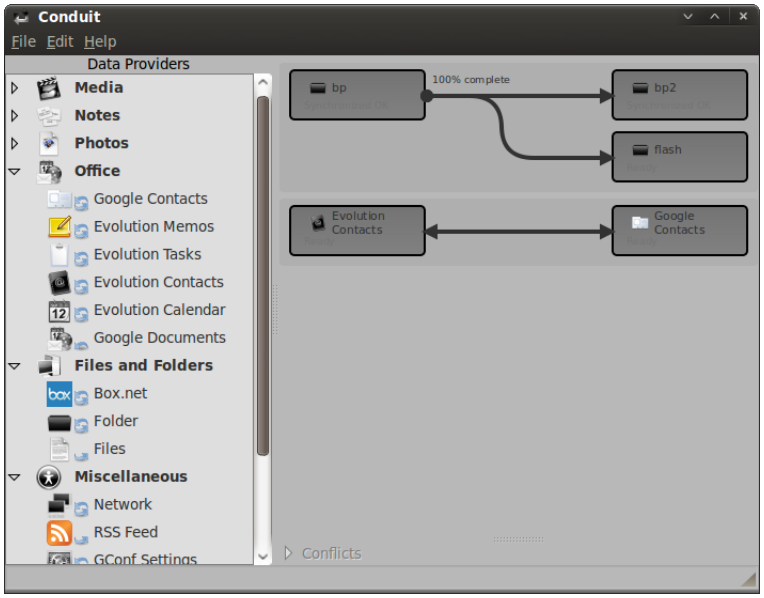


Obrázek 3.1: Unison

3.1.2 Conduit

- Licence: GPL
- Programovací jazyk: Python
- Platformy: GNU/Linux
- Detekce kolizí: Ano
- Poslední verze: 0.3.16 (1.10.2009)
- Adresa: http://live.gnome.org/Conduit

Conduit je víceúčelový nástroj pro synchronizaci souborů, kontaktů, kalendáře,... Je vyvíjen pouze pro GNU/Linux a prostředí Gnome, Má výborně řešené rozhraní pro vytváření synchronizačních párů. Program umí běžet jako démon v systray. Bohužel současná verze není plně stabilní, program občas přestává reagovat a nepovedlo se mi zprovoznit synchronizaci složky na disku a složky na fat32 usb disku.



Obrázek 3.2: Conduit

3.1.3 FreeFileSync

- Licence:GPL
- Programovací jazyk: C++
- Platformy: Windows, GNU/Linux
- Detekce kolizí: Ano
- Poslední verze: 3.6 (31.3.2010)

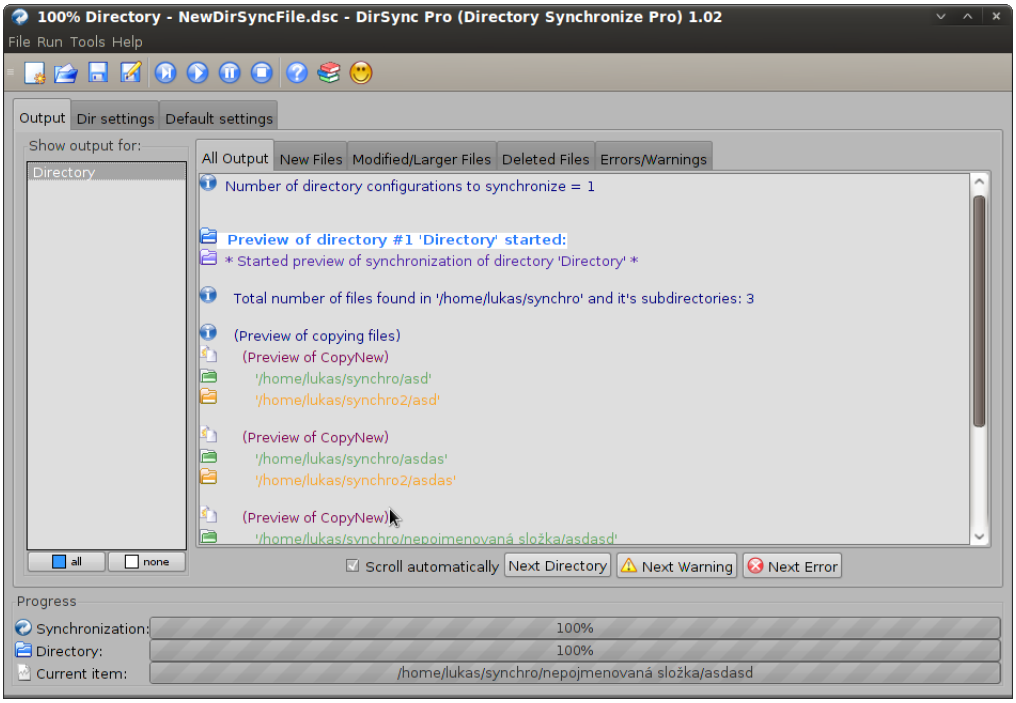
- Adresa: http://sourceforge.net/projects/freefilesync/

FreeFileSync je multiplatformní aplikace na porovnávání obsahu dvou adresářů a jejich synchronizaci. Zobrazuje obsah obou adresářů a podle pokynů provede potřebné akce. Program je spíše určen k jednorázovému použití než k pravidelnému zálohování.

3.1.4 DirSync Pro

- Licence: GPL
- Programovací jazyk: Java
- Platformy: Windows, GNU/Linux, Mac OS X
- Detekce kolizí: Ano
- Poslední verze: 1.24 (14.3.2010)
- Adresa: http://directorysync.sourceforge.net/

DirSyncPro je program na synchronizaci dvou adresářů běžící na jakémkoliv operačním systému kde je dostupný virtuální stroj pro Javu. Obsahuje velké množství nastavení (vynechávání podsložek, zacházení s odkazy, nastavení implicitních akcí při kolizích, ...). Velkým plusem je možnost nastavení zálohování souboru před mazáním nebo změnou. Není ovšem určen pro rezidentní běh a zabírá poměrně velké množství paměti kvůli nutnosti běhu virtuálního stroje Javy.



Obrázek 3.3: DirSync Pro

3.2 Komerční programy

3.2.1 Allway Sync

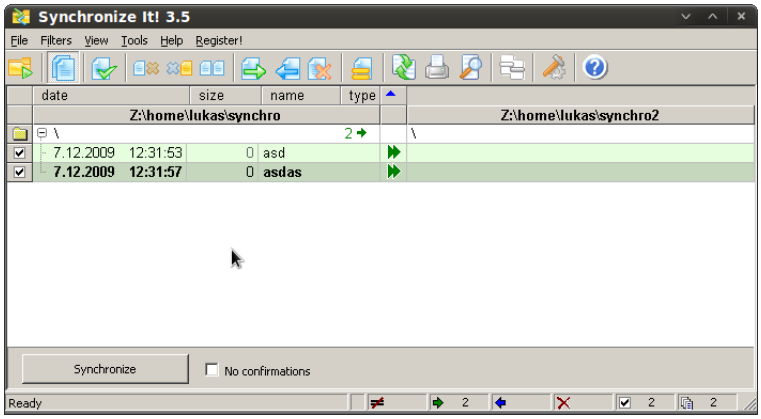
- Platformy: Windows
- Detekce kolizí: Ano
- Freeware: Ano (základní verze)
- Poslední verze: 10.3.8 (9.5.2010)
- Adresa: http://www.allwaysync.com/

Allway Sync je komerční synchronizační program pro Windows. Podporuje synchronizaci mnoha různých zdrojů jako USB disků, síťových disků, internetových úložišť. . . . Před synchronizací přehledně zobrazuje akce, které se budou provádět, rozdělené do kategorií. Každou akci si může uživatel přenastavit. Neobsahuje možnost automatického pravidelného spouštění.

3.2.2 Synchronize It! 3

- Platformy: Windows
- Detekce kolizí: Ano
- Freeware: Ne
- Poslední verze: 3.5 (7.12.2009)
- Adresa: http://www.grigsoft.com/wndsync.htm

Synchronize It! je komerční program pro synchronizaci adresářů umístěných na lokálním počítači, síti, nebo USB disku. Obsahuje několik synchronizačních režimů (synchronizace, zálohování, duplikování a update) podle, kterých se vypořádává s kolizemi. Před každou synchronizací zobrazí veškeré akce, které bude se soubory provádět a uživatel může tyto akce ovlivnit. Zajímavostí je, že jej lze provozovat i GNU/Linux za použití Wine.

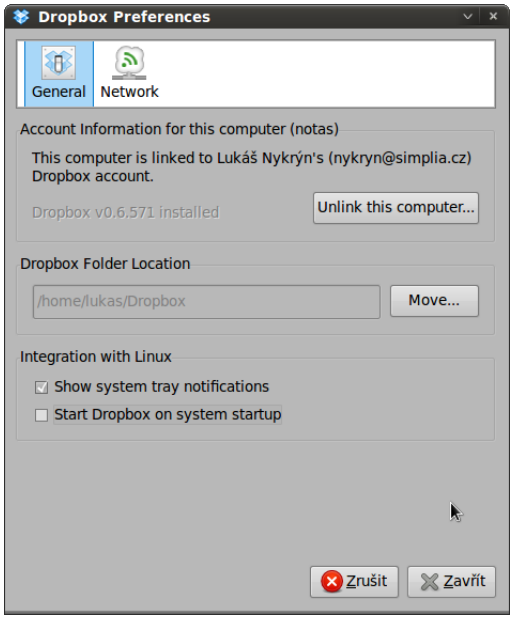


Obrázek 3.4: Synchronize It! 3

3.2.3 Dropbox

- Platformy: Windows, GNU/Linux, OS X, iPhone
- Detekce kolizí: Ano
- Freeware: Ano (základní verze)
- Poslední verze: 0.7.110 (25.2.2010)
- Adresa: http://www.dropbox.com/

Dropbox na rozdíl od předchozích programů nenabízí synchronizaci dvou adresářů, ale synchronizaci jednoho pevně daného adresáře v systému s serverem poskytovatele aplikaci. K těmto datům pak lze přistoupit buď přes jiného dropbox klienta se stejnými přihlašovacími údaji a nebo přes webové rozhraní. V základní variantě zdarma je poskytován prostor 2GB. Po zaplacení jde rozšířit na 50GB (99$ ročně) nebo 100GB (199$ ročně).



Obrázek 3.5: Dropbox

Kapitola 4

Návrh

4.1 Výběr implementačních prostředků

Kvůli požadavku na multiplatformnost je vhodné zvolit implementační prostředky tak, aby byli softwarové požadavky pro běh aplikace co nejmenší.

4.1.1 Programovací jazyk

Vzhledem k použití na platformě MS Windows jsem nejprve vyloučil skriptovací jazyky. Jejich použití by sice zajistilo jednoduchou implementaci a nezávislost na platformě, ale vyžadují přítomnost interpretu, což výrazně zvyšuje softwarové nároky. Nakonec tedy rozhodování skončilo mezi jazykem Java a C++.

- Java
  Pro jazyk Java hovoří jednoduchá tvorba GUI a nezávislost kódu na platformě, což ale opět v důsledku znamená, že vyžaduje k spuštění interpreta mezikódu (i když ten vzhledem k rozšířenosti Javy bývá na většině strojů nainstalovaný). Hlavní nevýhodou tohoto jazyka je ovšem vysoká paměťová náročnost pro běh virtuálního stroje, kvůli které není příliš vhodný pro rezidentní aplikace.

- C++
  Jazyk C++ má tu nevýhodu, že bude třeba pro každou platformu vytvořit rozdílné spustitelné binární soubory, ovšem to přináší nižší požadavky na hardware. Zároveň aplikace nebude závislá na instalaci jiného softwaru. Tento jazyk má navíc velmi dobře zpracovanou spolupráci s jádrem systému, což zvláště u programování systémově specifických částí aplikace (např. detekování připojení paměti) zjednodušuje jejich implementaci.

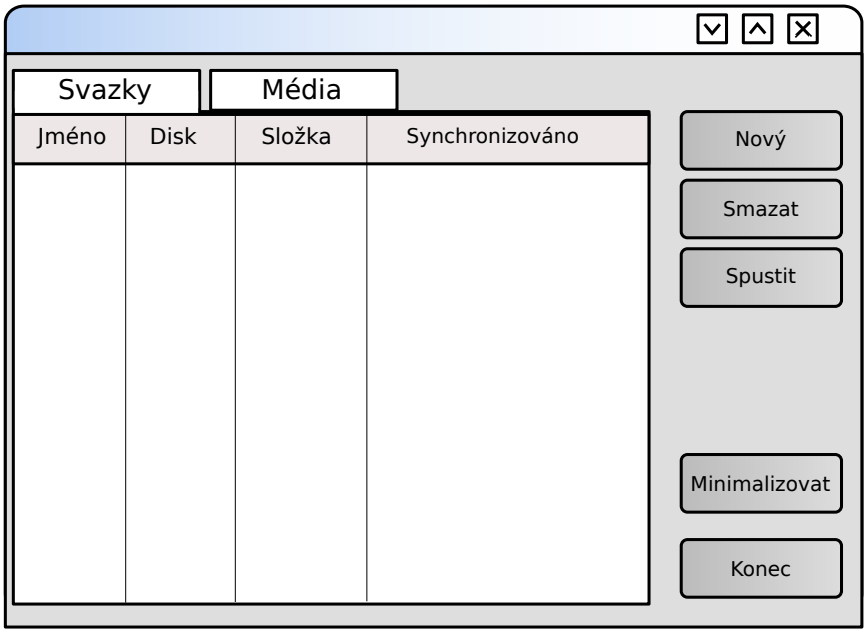
Jako programovací platforma byl tedy vybrán jazyk C++.

4.1.2 Grafické uživatelské rozhraní

Při výběru knihoven pro vytváření grafického rozhraní jsem nejdříve vyloučil platformněspecifické knihovny jako WinAPI, u nich by bylo nutné vytvořit pro každý operační systém vlastní GUI, což je velmi pracné a aplikace by vypadal pod každým systémem jinak.

Nakonec se výběr zúžil na GTK, QT, wxWidgets. GTK a QT jsou sice více rozšířené a existuje pro ně návrhářský software, který by ulehčil vývoj, ale pod MS Windows vyžadují

nainstalované prostředí. Volba tedy padla na wxWidgets, hlavně kvůli nulovým softwarovým požadavkům ze strany MS Windows.[4]



Obrázek 4.1: Návrh vzhledu hlavního okna aplikace

4.1.3 Ukládání trvalých dat

Aplikace bude pro svůj běh vyžadovat ukládání informací například o synchronizovaných zařízeních nebo o proběhnutých synchronizacích. Zde jsem se rozhodoval mezi použitím běžných textových souborů, XML či SQL databáze.

- Textové soubory
  Ukládání informací do běžných textových souborů je nejjednodušším způsobem, který nevyžaduje žádné dodatečné knihovny a strukturu dat si určuje sám programátor. Ovšem jejich použití je značně nepraktické vzhledem k nutnosti vytvoření celého rozhraní pro práci s nimi.

- XML soubory
  Pro práci s XML soubory už je hotová celá řada knihoven, které jejich použití usnadňují. Zároveň poskytuje pohodlnou práci se strukturovanými daty. Nevýhodou je pomalejší přístup k informacím, pokud je v souboru hodně záznamů.

- SQL databáze
  Databáze poskytuje výborné prostředky pro práci s daty které lze uložit formou tabulky. Zde lze jednoduše záznamy vyhledávat a provádět filtrování dat podle parametrů. Nevýhodou je ovšem nutnost přítomnosti nějakého databázového systému. Programy typu MySQL či PostgreSQL jsem vyloučil, protože jejich běh by zabral mnohonásobně víc prostředků než celá aplikace. Nejvhodnější se jeví použití databáze SQLite, která funguje na principu knihovny, která se přilinkuje k aplikaci. [1] [5]

Nejvhodnější způsob ukládání informací o souborech a synchronizačních párech je zcela jistě uložení do SQL databáze. Ta zaručí rychlý přístup k datům a snadnou implementaci.

4.1.4 Ukládání dočasných dat

Během běhu aplikace je potřeba si udržovat informace o adresářích na lokálním a externím disku. Tyto data lze uchovat buď v kontejnerech jazyka C++ a nebo použít dočasnou tabulku v sqlite.

Použití standartních prostředků C++ by mělo být při správné implementaci rychlejší, ale práce s nimi je nepohodlná a při velkém počtu souborů budou veškerá budou data uložená v paměti. Oproti tomu využití dočasných tabulek bude pravděpodobně pomalejší, ale práce s nimi bude komfortnější, např. získání změněných dat na lokálním disku je otázkou jednoho dotazu a nevyužívaná data jsou uložena v souboru na disku, takže nezabírají místo v paměti.

4.1.5 Práce se soubory

Akce se soubory (jako je kopírování, přesouvání, mazání) jdou v C++ provádět za použití knihoven operačního systému. Tento přístup není příliš vhodný. Znamenal by napsat dvě různé třídy pro práci se soubory pod Windows a GNU/Linux a musel by se vyřešit problém se zápisem cest, protože oba systémy používají jiná lomítka pro oddělení názvů složek v cestě.

Proto byla v aplikaci použita knihovna Boost.Filesystem, která poskytuje jednotné rozhraní pro práci s daty nad oběma platformami.

4.1.6 Sledování změn v souborovém systému u GNU/Linux

Pro sledování změn v souborovém systému poskytuje jádro od verze 2.6.13 podsystém inotify. V aplikace ho lze volat přímo nebo pomocí některé knihovny. Přímé volání nepokládám za vhodné, protože práce s ním není příliš komfortní a hlavně neumožňuje rekurzivní sledování celého stromu složky.

Rozhodl jsem se tedy použít některou z externích knihoven. Výběr nakonec skončil u knihoven **inotify-cxx** a **inotify-tools**. Hlavní výhodou **inotify-cxx** je objektové rozhraní a velmi snadné používání, ovšem sledování celého podstromu složky zatím není implementováno. Nakonec tedy volba padla na **inotify-tools**, což je balík utilit a knihoven založených na inotify, který na rozdíl od ostatních poskytuje možnost rekurzivního sledování. [7]

4.1.7 Detekce připojení a odpojení disků u GNU/Linux

U detekce připojení je na výběr ze dvou odlišných možností. Buď si napsat pravidlo pro udev, které nějakým způsobem běžící aplikaci upozorní a nebo v aplikaci komunikovat s démonem HAL pomocí d-bus.

Použití pravidla pro udev je velmi jednoduché, stačí napsat jeden řádek, který spustí shellový skript na poslání signálu aplikaci. Pro vytvoření pravidla jsou ovšem potřeba superuživatelská práva, což by značně snížilo využitelnost aplikace.

V druhém případě je potřeba se připojit na sběrnici d-bus a zažádat si o zprávy od démona HAL. Tento způsob je plně integrovaný do aplikace a nevyžaduje žádné externí zásahy do systému. [2]

4.2 Návrh implementace

4.2.1 Synchronizace

Synchronizace probíhá ve třech fázích:

1. Porovnávání změn souborů
2. Řešení kolizí
3. Přenos souborů

Nejprve je vytvořen seznam souborů lokální složky a složky na přenosné paměti a časů jejich posledních změn. K nim se přidá seznam souborů a časů změn lokální složky po poslední synchronizaci. Tento seznam se porovnán podle tabulky 4.1 a je vybrána příslušná akce. Pokud dojde ke kolizi, je akce vybrána podle implicitního nastavení nebo je dotázán uživatel.

lokální disk		externí disk	akce
stav po minulé synchronizaci	aktuální stav	aktuální stav	
1	n	n	nic
1	1	1	nic
n	1	1	nic
n	1	n	nahrání na externí disk
1	2	1	nahrání na externí disk
n	n	1	nahrání do složky
1	1	2	nahrání do složky
1	n	1	smazání z externího disku
1	1	n	smazání ze složky
n	1	2	kolize
n	2	1	kolize
1	2	n	kolize
1	n	2	kolize
1	2	3	kolize
1	3	2	kolize
2	1	3	chybový stav
2	3	1	chybový stav
2	1	n	chybový stav
2	n	1	chybový stav
2	1	1	chybový stav

n – soubor neexistuje
číslo – označuje datum poslední změny

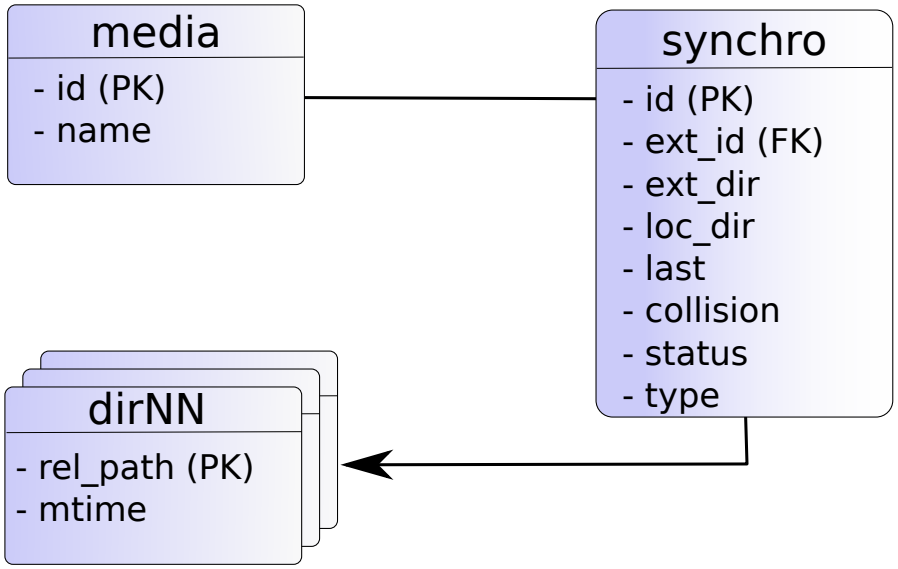
Tabulka 4.1: Tabulka stavů souborů

4.2.2 Ukládání dat

Na lokálním disku je uložena databáze s informacemi o synchronizačních svazcích a médiích. Každý svazek zde má definováno jméno, cestu k složce, id média, čas poslední synchronizace, stav, způsob synchronizace a řešení kolizí.

U média je pouze ukládáno jeho jméno a identifikační číslo. Současně se v ní nachází seznam souborů z jednotlivých svazků a jejich stav po poslední synchronizaci. Strukturu dat v databázi popisuje ER diagram 4.2.

Na externí paměti je pouze v textovém souboru uloženo její jméno.



Obrázek 4.2: ER diagram

4.2.3 Chování aplikace

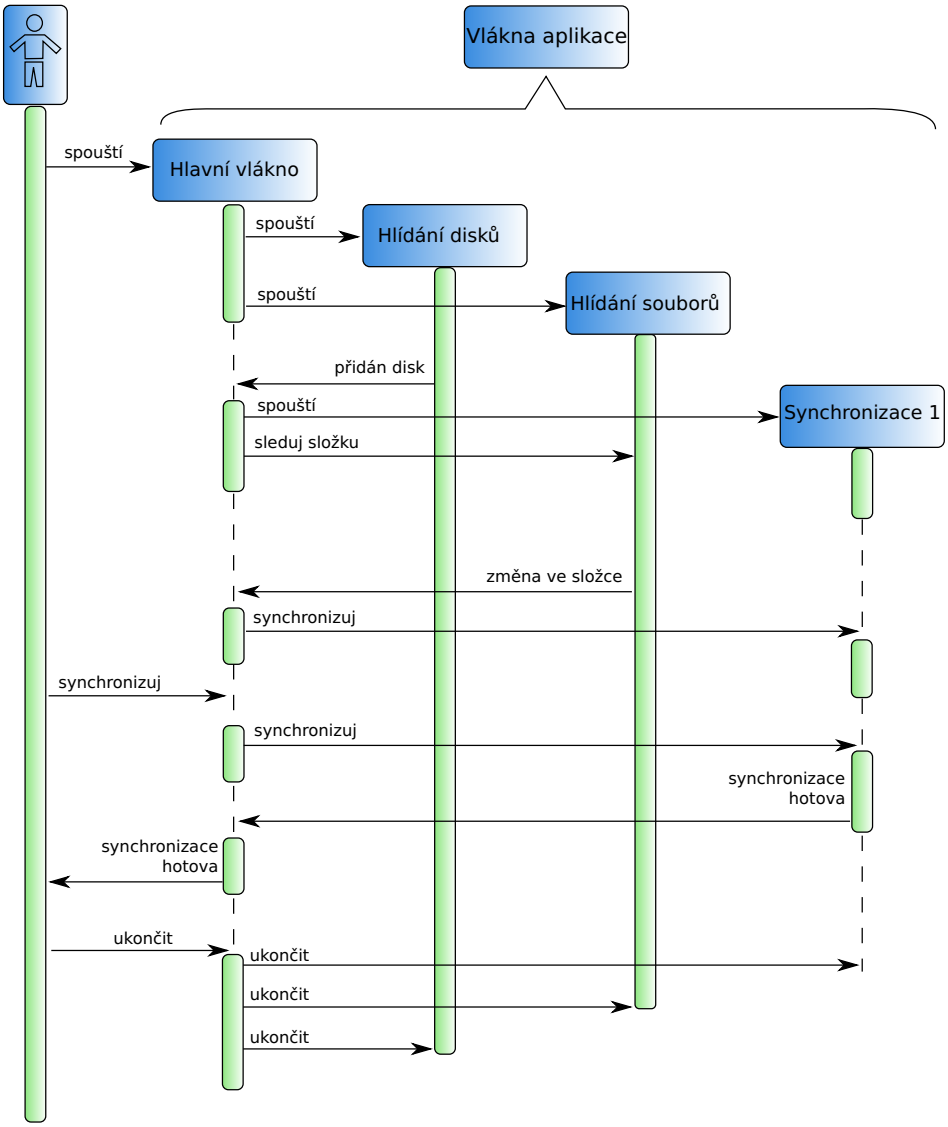
Aplikace by měla fungovat tak, že po spuštění GUI nebo při připojení přenosné paměti tuto paměť prohlédne a zjistí, zda se používá k synchronizaci. Pokud ano, projde synchronizované soubory a zjistí zda se vyskytují nějaké problémy, například uživatel od poslední synchronizace změnil soubory jak na pevném disku, tak na přenosné paměti a podle konfigurace buď proveden předdefinovanou akci nebo se zeptá uživatele. Nakonec celou synchronizaci provede a obnoví záznamy v databázi. Program dál běží ve režimu démona a periodicky kontroluje, zda nebyl připojen externí disk či nedošlo ke změnám na disku a podle potřeby synchronizaci opakuje.

Příklad běhu aplikace je vidět na obrázku 4.3. Uživatel zapne program a jeho hlavní část ihned po startu spouští vlákna starající se o detekování disků a indikaci změn souborů. Hned poté je nalezen disk se synchronizačním svazek a je tedy spuštěno vlákno jej obsluhující. Za běhu je pak detekována změna souboru v sledované složce a zpráva o této události je poslána hlavnímu vláknu, které ji předá samotné synchronizaci.

V další fázi uživatel sám zadá popud k synchronizaci svazku a následuje stejný postup jako v předchozím případě, s tím rozdílem, že synchronizační vlákno vrací informaci o

dokončení pro uživatele.

Přidávaní nových synchronizačních párů probíhá tak, že uživatel vybere složku na lokálním disku, externí disk a jméno synchronizace, které bude použito jako jméno složky na externím disku. Zároveň může nastavit výchozí akce, které se provedou při kolizích. Tyto údaje se pak uloží do databáze a sdílení je přiděleno unikátní identifikační číslo.



Obrázek 4.3: Sekvenční diagram

4.3 Implementační nástroje

U složitějších aplikací je také důležité pečlivě zvolit nástroje, ve kterých bude program vyvíjen.

4.3.1 Vývojové prostředí

Primární vývoj aplikace byl prováděn pod operačním systémem Linux, kde k tomuto účelu posloužilo prostředí NetBeans doplněné o C++ pack. Pro kompilaci byl použit kompilátor GCC ve verzi 4.4.1 a knihovny WxWidgets 2.8.10, Boost 1.40.0, SQLite 3.6.16 a Inotifytools 3.13.

Pod operačním systémem Windows patří mezi nejlepší prostředí MS Visual Studio (konkrétně byla použita verze 2008). To bylo doplněno o wxPack v2.8.10.06 (pro vytváření projektů založených na WxWidgets) a BoostPro 1.42.0. Pro práci s databází bylo opět použita knihovna SQLite 3.6.16.

4.3.2 Verzovací nástroje

Během vývoje je vhodné si aktuální stav práce ukládat a zálohovat, aby bylo možno se během vývoje vrátit o kousek zpátky nebo abychom dokonce o celý projekt nepřišli např. kvůli havárii pevného disku. Zároveň správné prostředky usnadní transfer zdrojových kódu při testování na různých platformách. Pro verzování kódu jsem použil systém Mercurial, hlavně pro jeho integraci s prostředím NetBeans a nezávislosti na samostatném repozitáři na rozdíl např. od SVN.

Pro zajištění přenosu kódu mezi platformami a vytváření zálohy jsem se vybral DropBox. Toto řešení je poněkud netradiční, ale díky monitorování změn souborů dochází k nahrání k nahrání dat na server ihned po uložení, takže v případě havárie disku jsou ztraceny pouze neuložené změny.

Kapitola 5

Implementace

Celý program lze rozdělit do několika následujících částí:

- Synchronizace
- Práce se soubory
- Databáze
- Detekce připojených externích disků
- Spouštění synchronizace
- Grafické uživatelské rozhraní

Krom automatického spouštění synchronizace a nastavování času změny souboru je zbytek kódu multiplatformní.

5.1 Synchronizace

Základní část programu tvoří třída `synchro`. Ta si v konstruktoru z databáze zjistí cestu složky na lokálním disku a určí složku na externím disku z jména disku a sdílené složky.

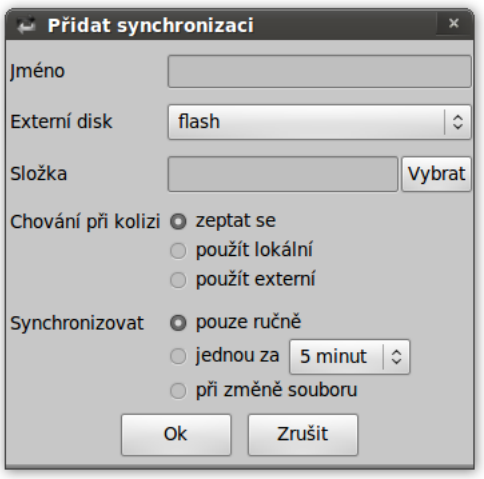
Vytvoří se dočasné tabulky v SQLite pro uchování seznamů souborů na lokálním a externím disku a pokud se jedná o první synchronizaci je vytvořena i tabulka se záznamem stavu souborů po poslední synchronizaci. Vzhledem k tomu, že každá instance třídy `synchro` má své vlastní připojení k databázi, nejsou dočasné tabulky z ostatních míst programu dostupné a není třeba jim vytvářet unikátní jména.

Z těchto tří tabulek je pak vytvořen pohled, který slouží pro zjednodušení sql dotazů, které generují seznamy změněných či smazaných souborů. Dále je vytvořen mutex, který se stará o zajištění správného přístup z více vláken a je spuštěno vlákno starající se o provádění synchronizace. Samotná synchronizace probíhá ve čtyřech krocích.

Nejprve je třeba obnovit seznamy souborů, o tuto akci se stará metoda `update`, ta volá aktualizační funkce tříd pro práci se soubory. Buď je obnoven seznam pro celou složku, to v případě první, časově spuštěné nebo uživatelem vyvolané synchronizace, a nebo jsou obnoveny jen záznamy z konkrétní podsložky či konkrétního souboru v případě synchronizace řízené sledováním změn na disku.

Dále je třeba zjistit, zda během synchronizace nemůže dojít ke kolizi. To kontroluje metoda `hasCollision`, která provede sérii dotazů na databázi, které hledají takové soubory,

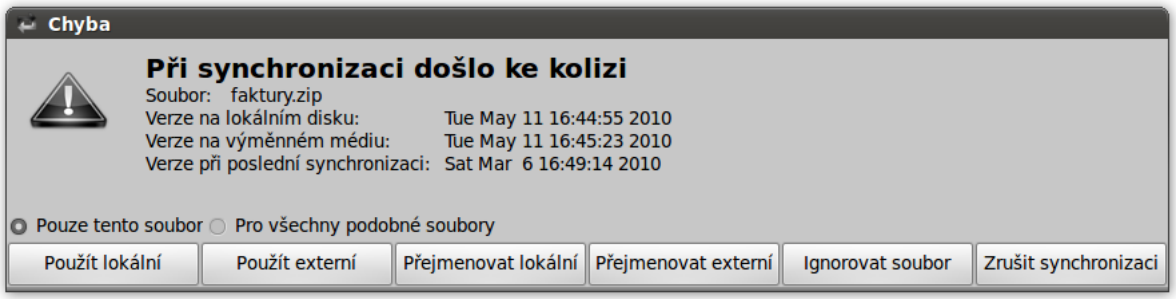
které byly změněny či smazány od poslední synchronizace jak na lokálním tak externím disku. Pokud se kolize vyskytnou je třeba je vyřešit, zda existuje uživatelem předdefinované řešení lze zjistit metodou `hasCollisionDefault`, tato metoda má čistě informační charakter a na vlastní synchronizaci nemá vliv.



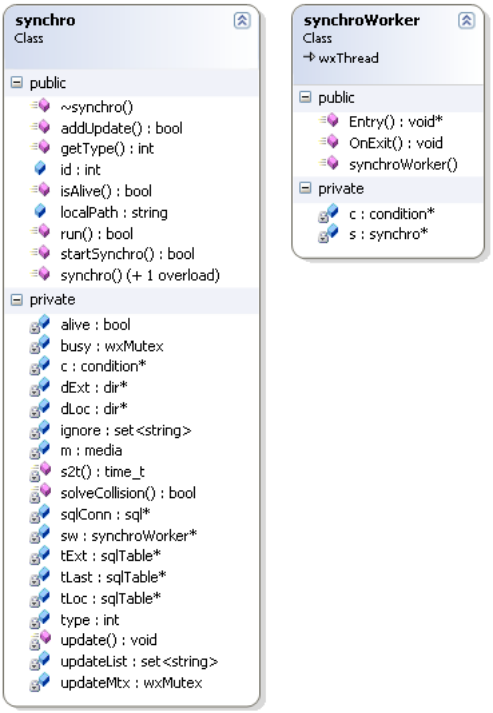
Obrázek 5.1: Dialog pro přidání synchronizace

Řešení kolizí provádí metoda `solveCollision`. Ta pokládá dotazy na databázi pro jednotlivé typy kolizí a buď se řídí podle předdefinované akce a nebo se zeptá na řešení uživatele.

Posledním krokem se synchronizace souborů u nichž nedošlo ke kolizi. Tam se opět provádějí dotazy, které určují co se se soubory provede. Během provádění změn se průběžně obnovují záznamy o poslední proběhnuté synchronizaci. Třída také obsahuje metodu `isAlive`, která určuje zda lze synchronizaci dále provádět nebo zda byl svazek odpojen, v tom případě metoda ukončí vlákno starající se o spouštění synchronizace.



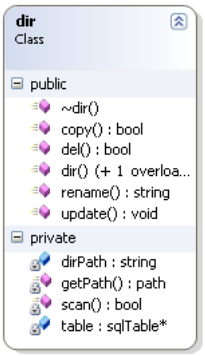
Obrázek 5.2: Dialog pro řešení kolizí



Obrázek 5.3: Třídy synchro a synchroWorker

5.2 Práce se soubory

Třída `dir` poskytuje rozhraní pro operace se soubory na disku a stará se o záznamy složky v databázi. Úpravy souborů zajišťují metody `copy` (kopírování ze složky do složky), `del` (mazání souborů) a `rename` (přejmenování soubor na soubor.kopie). Dále třída poskytuje metodu `update`, pro aktualizaci celé složky nebo její části. Interně je pro obnovování informací o souborech využívána metoda `scan`, která rekurzivně prochází celý strom složky a přepisuje záznamy o souborech do databáze.



Obrázek 5.4: Třída dir

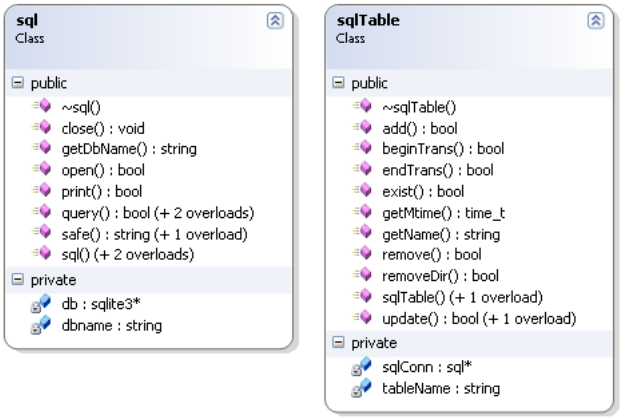
5.3 Databáze

Program obsahuje dvě třídy využívané k práci s databází: `sql` a `sqlTable`.

Třída `sql` poskytuje základní abstrakci pro připojení k **sqlite3** databázi. V konstruktoru se připojí k souboru zadanému parametrem a pokud neexistuje, databázi vytvoří. Provádění dotazů na db zajišťuje třída query, která výsledek vrací ve formě vektoru řetězců. Pro zjednodušení vytváření dotazů lze použít metodu `safe`, která upravuje zadaný řetězec na tvar vhodný pro použití jako hodnotu v sql dotazu a zabraňuje provedení sql injection.

Třída `sqlite` obsahuje metody pro operace přidávání, mazání a úpravu nad tabulkou se záznamy o souborech ve složce a to jak v transakcích (atomická operace nad více záznamy, kdy k zapsaní dojde až po provedení celé transakce), tak bez nich.

Soubor s databází je uložen podle nastavení operačního systému, v GNU/Linux je to většinou složka /home/$UŽIVATEL/.$PROGRAM a v českých Windows C:\Documents and Settings\$UŽIVATEL\Data aplikací\$PROGRAM



Obrázek 5.5: Třídy sql a sqlTable

5.4 Detekce připojených externích disků

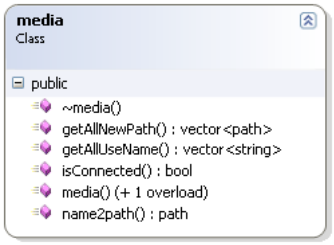
Určování, zda-li a kde je nějaký externí disk připojen je závislé na operačním systému a vyžaduje tedy rozdílný přístup na platformách GNU/Linux a Windows. Obecně v programu existuje třída `media`, která vrací seznamy připojených medií používaných k synchronizaci (`getAllUseName`), a zařízeních, které by se potencionálně mohli stát prostředkem ke sdílení (`getAllNewPath`).

5.4.1 GNU/Linux

Na platformě GNU/Linux je toto chování implementováno čtením ze souboru */etc/mtab*. Ten obsahuje název, typ, místo a parametry připojení. Z tohoto seznamu stačí vybrat všechny fyzické disky, krom disku primárního. Následně je zjištěno, zda se na disku nalézá konfigurační soubor oznamující, že je disk využíván.

5.4.2 Windows

V operačním systému Windows jsou využity funkce `GetLogicalDrives` a `GetDriveType` z knihovny `windows.h`. `GetLogicalDrives` vrací 32b číslo, které udává stav jednotek, řazený podle abecedy. Tedy 1011 0100 0000 0000 0000 0000 0000 0000, oznamuje, že jsou připojeny jednotky A, C, D a F. Připojené disky předáme funkci `GetDriveType`, která vrací ID typu disku. Program vyhledává všechny disky označené číslem 2 – výměnné médium. Nakonec určí využívání disku k synchronizaci stejně jako ve variantě pro GNU/Linux.[3]



Obrázek 5.6: Třída media

5.5 Spouštění synchronizace

O řízení synchronizací se stará třída `synchroCtrl`. Ta udržuje seznam všech instancí třídy `synchro`. O aktualizaci tohoto seznamu se stará metoda `refresh`, která je volána na základě aktivity uživatele nebo automaticky. V případě varianty pro Windows je kontrola spouštěna periodicky, zatímco v GNU/Linux na základě zprávy od démona HAL přes systém d-bus.

5.5.1 Na pokyn uživatele

Pokyn k provedení synchronizace je vyvolán z GUI aplikace. Metoda `runSynchro` si zjistí, zda je připojeno potřebné médium a zda-li již nebyla synchronizace spuštěna automaticky, v tom případě je uživateli vrácena chyba. Jinak je synchronizace provedena a uživatel je informován o jejím dokončení.

5.5.2 Automatické

Periodické

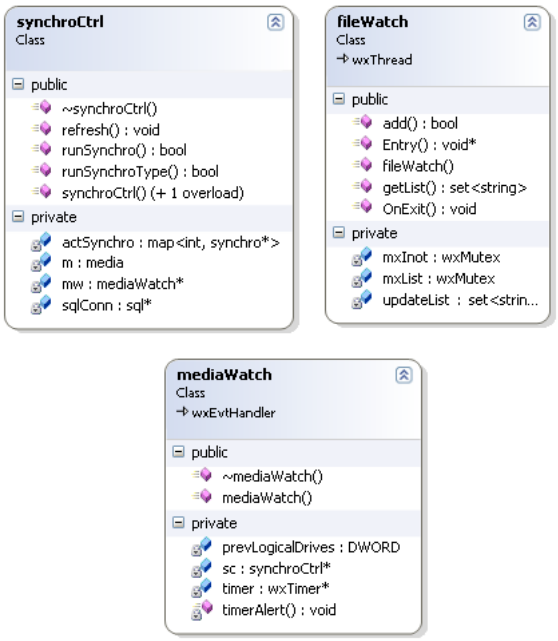
Při spuštění programu je vytvořen časovač, který pravidelně po 30 sekundách odešle signál hlavnímu vláknu. Program zjistí, zda u časově řízených synchronizací již uplynul čas k další synchronizaci. Pokud ano, zažádá o zámek, obnoví databázi souborů a zjistí zda nedošlo ke kolizím. Při jejich výskytu a existenci předdefinovaného řešení jsou kolize vyřešeny, jinak je vyvolána událost žádající hlavní vlákno o vyřešení problému uživatelem. Nakonec se provede synchronizace nekolidujících souborů.

Hlídáním změn na souborovém systému

Tato možnost spouštění synchronizací je implementována pouze ve verzi pro GNU/Linux. Třída `main` spustí vlákno, které využívá systému inotify pro sledování změn nad všemi

sledovanými složkami. Vlákno si při startu zažádá o hlášení změn z celé podsložky a pokud dojde k chybě je uživateli doporučeno buď přejít na periodickou synchronizaci a nebo zvětšit limit pro počet „inotify hlídačů“ v systému.

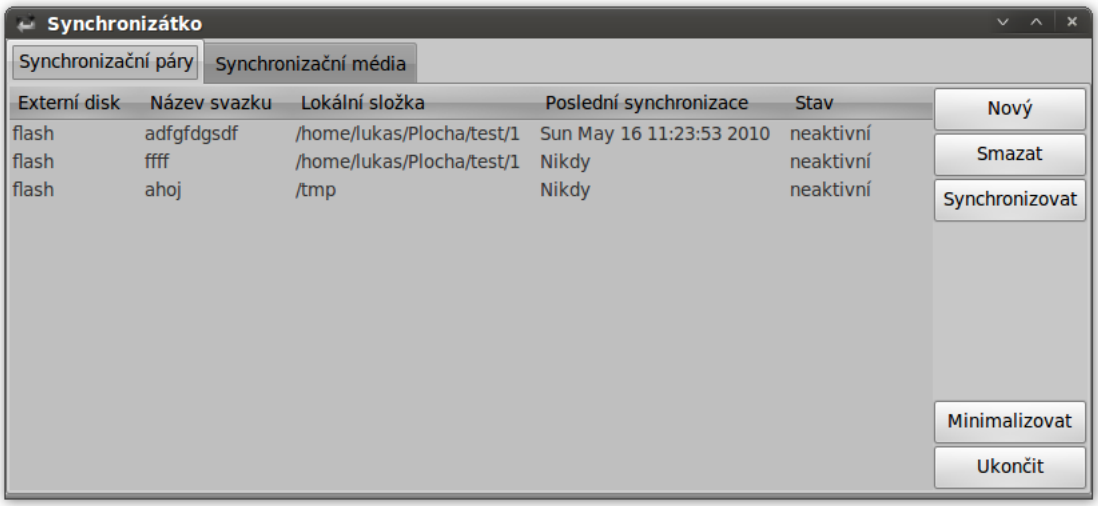
Vlákno si při běhu hlídá možnost vytvoření nové složky v sledovaném prostoru a ta je přidána do sledování. V případě nalezení změny jsou obnoveny v databázi pouze ty záznamy jichž se změna týkala a zbytek synchronizace je proveden stejně jako v případě periodického spouštění.



Obrázek 5.7: Třídy synchroCtrl, fileWatch, mediaWatch

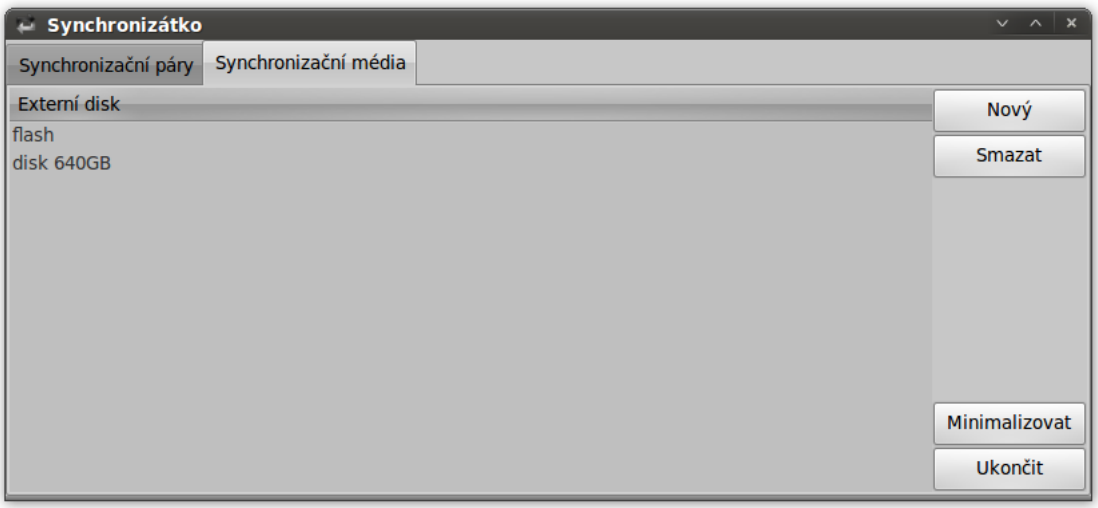
5.6 Grafické uživatelské rozhraní

Hlavní okno aplikace tvoří třída `mainWindow`. V něm uživatel vidí všechna média používaná pro synchronizaci a synchronizační dvojice a má možnost zde synchronizaci ručně spouštět.



Obrázek 5.8: Hlavní okno aplikace – svazky

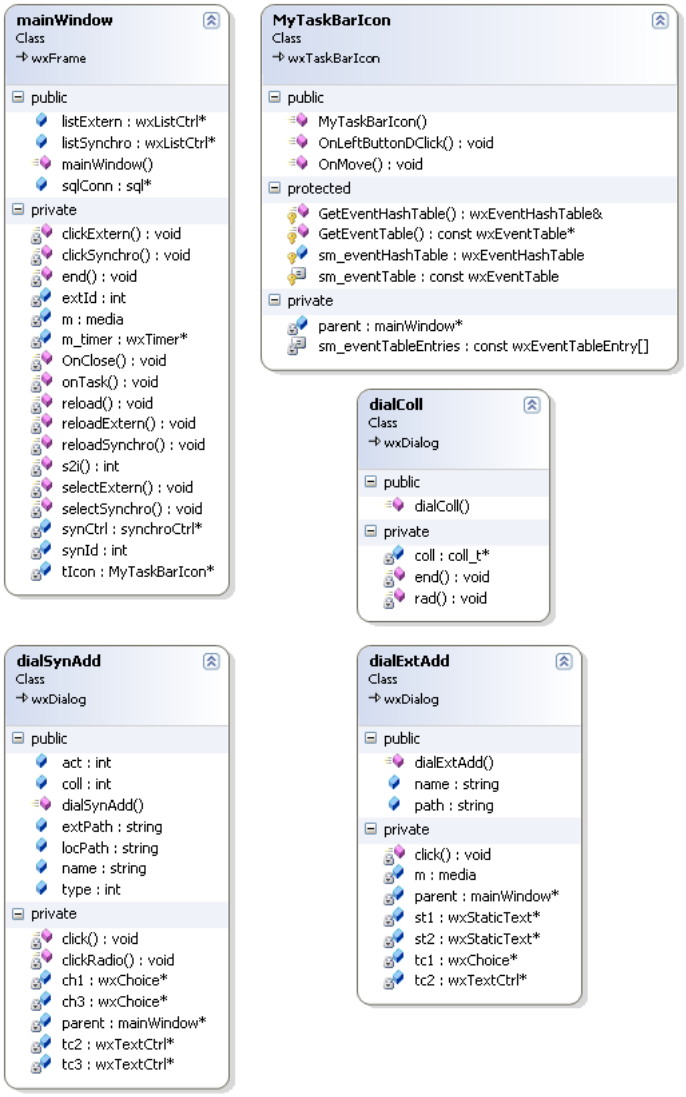
Uživatel může v obou těchto seznamech přidávat a odebírat položky, přičemž odstranění celého média vede zároveň i ke smazání synchronizačních párů, které ho využívaly. Při přidání nového disku dostane uživatel seznam aktuálně připojených disků nepoužívaných k synchronizaci a vyžaduje se po něm zadání jména pro tento disk. Toto jméno se pak objevuje v seznamu použitelných disků ve volbě přidání synchronizačního páru, kde uživatel krom disku vybírá jméno synchronizace (současně složka na externím disku), lokální složku, automatické spouštění a chování při kolizi.



Obrázek 5.9: Hlavní okno aplikace – média

Hlavní okno lze schovat do systémové lišty buď zavřením okna, nebo kliknutím na tlačítko minimalizovat. Aplikace se vypne po stisknutí tlačítka ukončit.

Druhou částí grafického rozhraní je dialog o hlášení kolizí. V něm je zobrazen název souboru, čas jeho poslední změny na lokálním a externím disku a stav při minulé synchronizaci. Uživatel dostává na výběr z možností jak lze kolizi vyřešit (viz. tabulka 4.1) nebo může soubor ignorovat či synchronizaci zastavit. Rovněž se zde také nabízí možnost uplatnit rozhodnutí pouze pro daný soubor a nebo pro všechny podobné kolize.



Obrázek 5.10: Třídy mainWindow, MyTaskBarIcon, dialColl, dialSynAdd a dialExtAdd

Kapitola 6

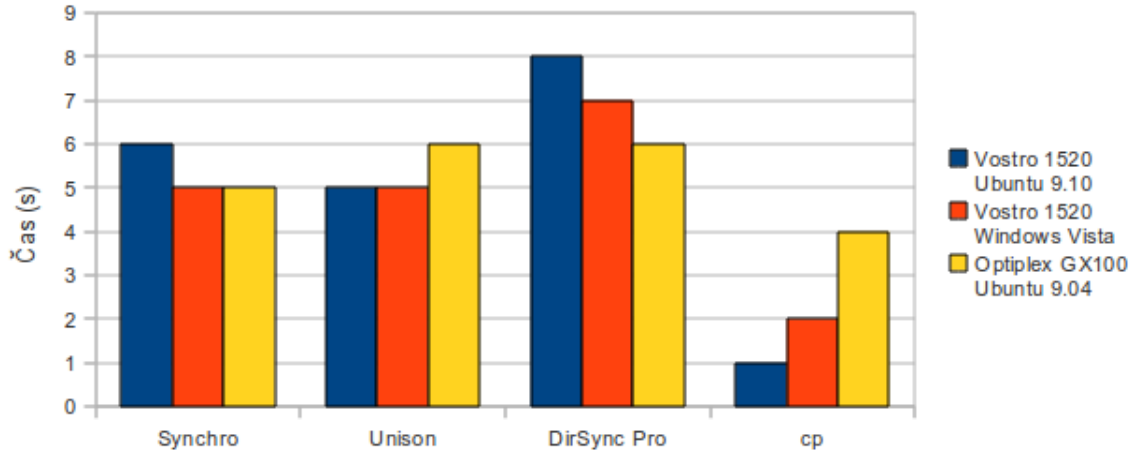
Testování a porovnání s podobnými programy

Pro otestování programu jsem zvolil tři scénáře. Synchronizace byla použita na „běžnou složku s dokumenty“, velké množství malých souborů a několik větších souborů. Aby bylo možno výsledky s něčím porovnat, byly testy provedeny i na programech Unison 3.1.1 a DirSync Pro 3.1.4.

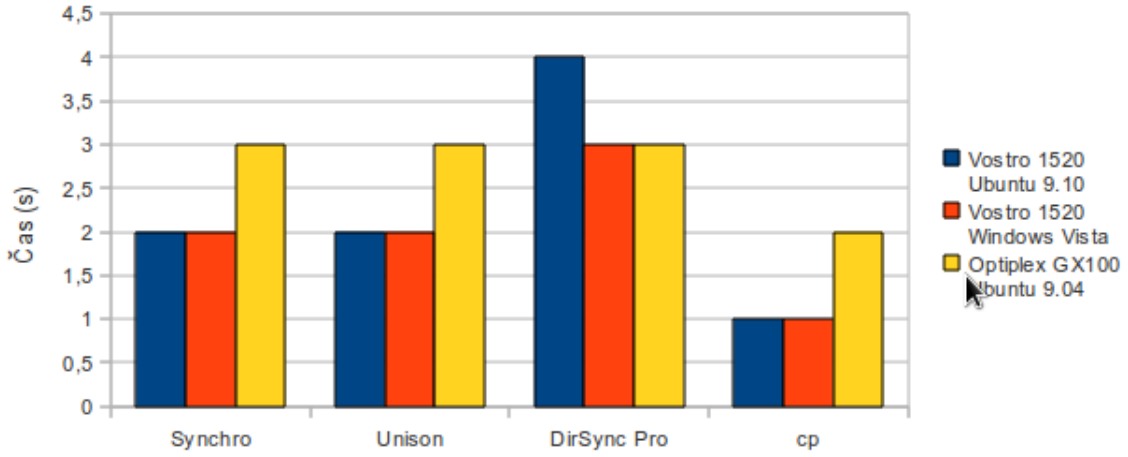
Jako testovací stroje posloužili DELL Vostro 1520 s operačními systémy Ubuntu 9.10 Karmic Koala a Windows Vista a starší DELL Optiplex GX100 s Ubuntu 9.04 Jaunty Jackalope.

6.1 Synchronizace složky s dokumenty

Zde byla pro testovací účely použita moje složka s účetnictvím. Ta obsahuje 50 souborů (převážne pdf) s celkovou velikostí 50 MB. V prvním pokusu byl vytvořen na usb flash disku nový sdílený svazek a byla spuštěna synchronizace. V druhém kroku byla změněna polovina souborů a synchronizace proběhla znovu.



Obrázek 6.1: Synchronizace složky s dokumenty – první synchronizace

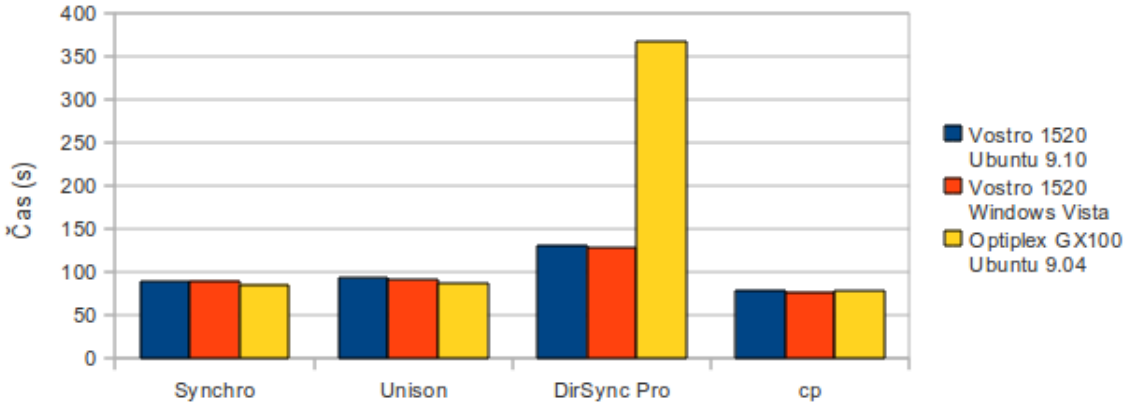


Obrázek 6.2: Synchronizace složky s dokumenty – po změně souborů

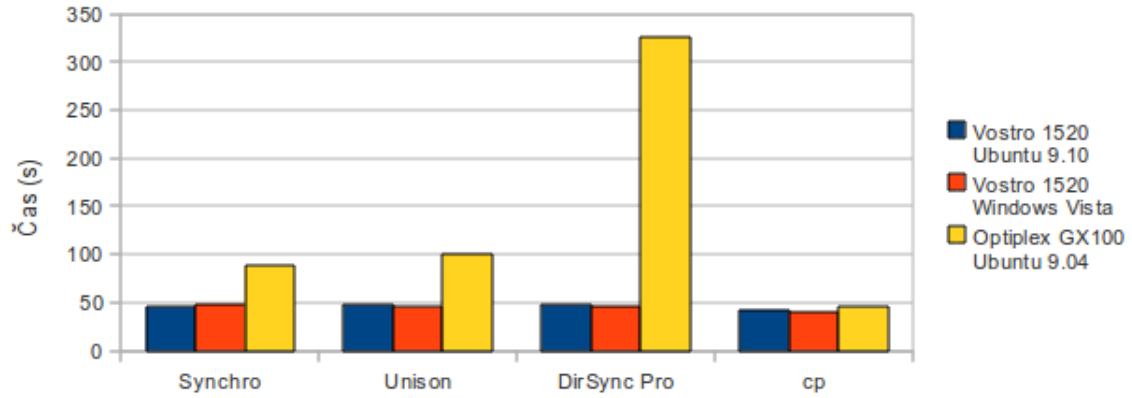
První test neukázal žádné podstatné rozdíly mezi programy, pouze se projevil rozdíl proti kopírování v systému, které přesouvá data ihned, zatímco synchronizační aplikace potřebují čas na počáteční inicializaci.

6.2 Synchronizace velkého množství malých souborů

Druhou zkouškou bylo sesynchronizovat složku obsahující 2000 5B souborů. Tento test jsem považoval jako klíčový. Režie na samotný přesun souboru je zde minimální a mělo by se projevit, zda program nepotřebuje příliš času porovnáváním souborů. Nejdříve byla opět přenesena celá složka, poté změněno 50% souborů a synchronizace proběhla znovu.



Obrázek 6.3: Synchronizace velkého množství malých souborů – první synchronizace

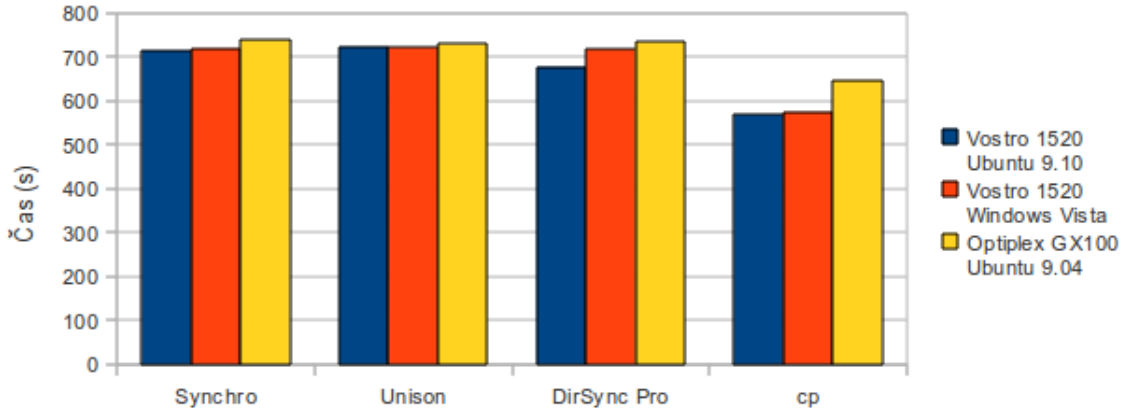


Obrázek 6.4: Synchronizace velkého množství malých souborů – po změně souborů

Výsledky druhého testu ukazují na nevhodnost používaní programů v Javě na starších počítačích. Režie takovéto aplikace je velmi vysoká a operace se soubory se zpomalí.

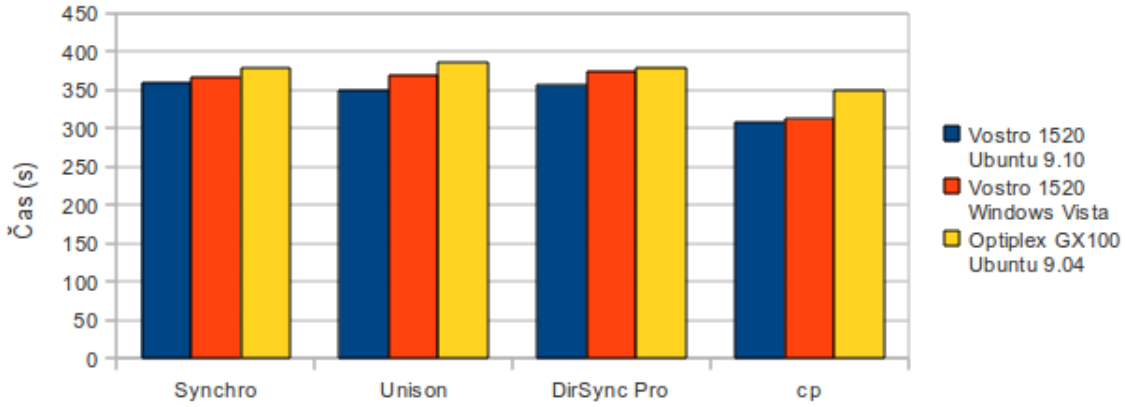
6.3 Synchronizace malého množství velkých souborů

Posledním testem na přenos souborů bylo zadání stejné jako v předchozích dvou, s tím že bylo použito 12 souborů s velikostí 200MB. Zde již nelze očekávat velké rozdíly, neboť většinu času aplikace spotřebuje na přenos souboru.



Obrázek 6.5: Synchronizace malého množství velkých souborů – první synchronizace

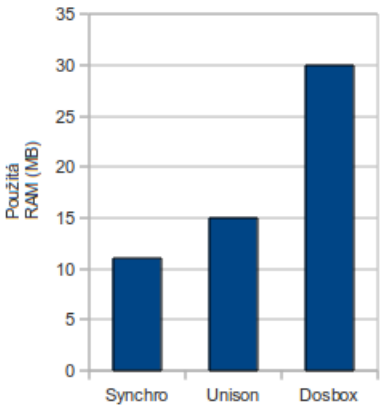
Test potvrdil původní očekávání a aplikace zde byly velmi vyrovnané. Po většinu času program stráví čekáním na kopírování souborů a jeho vlastní režie nemá na dobu běhu příliš velký vliv.



Obrázek 6.6: Synchronizace malého množství velkých souborů – po změně souborů

6.4 Využití operační paměti

Důležitým parametrem programu je také množství alokované operační paměti. Protože Dirsync Pro není přímo určen pro rezidentní běh a využívá virtuální stroj Javy, který sám o sobě používá řádově desítky MB paměti, byl místo něj pro referenci použit Dropbox 3.2.3. Ten v předchozích testech nebylo možno použít, protože nezálohuje na externí disk nýbrž na síť. Pro porovnání měli tyto programy přidány k synchronizaci složky z prvního 6.1 a druhého 6.2 testu. Měření proběhlo na stroji Dell Vostro 1520 pod OS Ubuntu 9.10.



Obrázek 6.7: Využití operační paměti

6.5 Shrnutí

Z výše uvedených měření je patrno, že tato aplikace, co se týče rychlosti, nijak nevyniká a ani nezaostává za podobnými programy. Velmi dobrou volbou se ukázalo použití jazyka C++ oproti Javě, díky němuž aplikace obstojně běží i na starších strojích. Co se týče použité paměti je na tom například oproti Dropboxu mnohem lépe.

Kapitola 7

Závěr

Aplikace splňuje požadavky na funkčnost tak jak byly definovány v zadání. Program je zaměřen na uživatele, který pracuje na několika počítačích a nemá možnost být neustále připojen k internetu nebo nemůže na počítače instalovat vlastní software.

Díky využití multiplatformních knihoven je spustitelná jak pod operačním systémem Windows, tak pod GNU/Linux, přičemž její rozhraní je uzpůsobeno tak, aby uživatel jedné verze mohl okamžitě začít pracovat s druhou a přitom aby aplikace vzhledem zapadla do systému.

Program má intuitivní ovládání, kde přidání nové synchronizace je velmi jednoduché a na první pohled je vidět, v jakém stavu jsou jednotlivé složky. Hlavní výhodou je automatický běh. Při přidání svazku lze nastavit jak často se má synchronizace provádět, či v případě verze pro GNU/Linux je možnost nechat soubory přenášet hned po změně.

Díky návrhu, lze program v budoucnu rozšířit o mnoho dalších funkcí, například pro přidání podpory šifrování a komprimace, by stačilo upravit metodu používanou ke kopírování souborů a dopsat do ní např. šifrovací algoritmus. Další možností je zahrnout i podporu pro jiná datová úložiště (síťové disky, přístup přes ssh,...), jedinou podmínkou je, aby bylo schopno poskytnout seznam souborů a čas jeho poslední úpravy. Po té by bylo možno program používat i ve zcela jiném prostředí, například ve škole, kde učitel nakopíruje soubory do serverové složky a automaticky se žákům vytvoří kopie na jejich počítačích poté, co ho zapnou.

Literatura

[1] SQLite Documentation. 2010, [Online; navštíveno 20. 4. 2010].
URL http://www.sqlite.org/docs.html

[2] Masters, J.; Blum, R.: *Linux profesionálně – programování aplikací*. Zoner Press, 2008, iSBN 978-80-86815-71-8.

[3] msdn: Volume Management Functions. 2010, [Online; navštíveno 13. 05. 2010].
URL http://msdn.microsoft.com/en-us/library/aa365730(v=VS.85).aspx

[4] Smart, J.; Hock, K.: *Cross-Platform GUI Programming with wxWidgets*. Prentice Hall, 2006, iSBN 0-13-147381-6.

[5] Stephens, R. K.; Plew, R. R.: *Naučte se SQL za 21 dní*. Computer Press, 2004, iSBN 97-880-722-6870-2.

[6] Wikipedia: File synchronization — Wikipedia, The Free Encyclopedia. 2010, [Online; navštíveno 13. 05. 2010].
URL http://en.wikipedia.org/w/index.php?title=File_synchronization&oldid=361346339

[7] Wikipedie: Inotify — Wikipedie: Otevřená encyklopedie. 2008, [Online; navštíveno 13. 05. 2010].
URL http://cs.wikipedia.org/w/index.php?title=Inotify&oldid=3228708

Dodatek A

Obsah CD

- Složka `src`:
  - Zdrojové kódy aplikace
- Složka `bin`:
  - Předkompilované spustitelné binární soubory
- Složka `doc`:
  - Dokumentace ve formátu PDF
  - Zdrojové kódy dokumentace v LaTeXu