**UNIVERZITA KOMENSKÉHO V BRATISLAVE**

**FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY**

**Zálohovanie súborového systému do obsahom adresovaného úložiska**

**Bratislava 2015  
Michal Molec**

**UNIVERZITA KOMENSKÉHO V BRATISLAVE**

**FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY**

**Zálohovanie súborového systému do obsahom adresovaného úložiska**

**Diplomová práca**

Študijný program: Aplikovaná informatika   
Študijný odbor: 2511 Aplikovaná informatika

Školiace pracovisko: Katedra aplikovanej informatiky

Školiteľ: Mgr. Michal Molec, PhD.

**Bratislava 2015  
Michal Molec**

Naskenované zadanie záverečnej práce

**Čestné vyhlásenie**

Čestne vyhlasujem, že som diplomovú prácu "Zálohovanie súborové systému do obsahom adresovaného úložiska" vypracoval samostatne s použitím uvedenej literatúry a zdrojov dostupných na internete.

V Bratislave dňa 1.6.2015

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Michal Molec

**Poďakovanie**

Chcel by som sa poďakovať vedúcemu diplomovej práce Mgr. Jánovi Kľukovi, PhD. za cenné rady a usmernenie pri vypracovaní tejto práce.

**Abstrakt**

MOLEC, Michal: Zálohovanie súborového systému do obsahom adresovaného úložiska - Univerzita Komenského v Bratislave. Fakulta matematiky, fyziky a informatiky, Katedra aplikovanej informatiky. Vedúci diplomovej práce: Mgr. Ján Kľuka, PhD. Bratislava, 2015.

Cieľom diplomovej práce je navrhnúť vhodné obsahom adresované úložisko dát (content-addressable storage) a implemetovať nad ním zálohovanie súborového systému unixového OS (napr. Linux), údržbu a prezeranie záloh a obnovu dát. Návrh samotného úložiska môže vychádzať z už existujúceho (napr. git), treba ho však upraviť s ohľadom na požadovaný účel. Očakáva sa kreatívne využitie (a prípadná úprava) vhodne zvolených techník kompresie (klasická bezstratová, rozdielová) a garbage collection, technológie FUSE, prípadne sledovania zmien v súborovom systéme. Implementácia by mala byť objektová, modulárna a rigorózne otestovaná (unit testing pokrývajúci funkcionalitu kritickú pre spoľahlivosť zálohovania). Preferovaným implementačným jazykom je python..

**Kľúčové slová:** Zálohovanie, FUSE, Rsync, Linux, Python

**Abstract**

MOLEC, Michal: File-system backup into content-addressable storage – Comenius University in Bratislava. Faculty of mathematics, physics and informatics, Department of aplied informatics. Dissertation supervisor: Mgr. Ján Kľuka, PhD. Bratislava, 2015.

The goal of this work is to design suitable content-addressable storage (CAS) and implement backup, backup maintenance and browsing, and data recovery of a Unix file system over the CAS. The design of the CAS may be based on existing ones (e.g., Git), it should, however, be adjusted for backup purposes. The student is expected to creatively choose, use, and adjust suitable methods of compression (standard lossless compression, delta compression) and garbage collection, the FUSE technology, and, possibly, the file-system event notification mechanism (inotify). The implementation should be object-oriented, modular, and rigorously tested (unit testing covering components critical to reliability of backup). The preferred implementation language is python.

**Keywords:** Backup, FUSE, Rsync, Linux, Python

**Obsah**

[Úvod 10](#_Toc405922246)

[1 Úvod do problematiky 11](#_Toc405922247)

[1 Návrh riešenia 11](#_Toc405922248)

[2 Implementácia 12](#_Toc405922249)

[Záver 10](#_Toc405922250)

[Bibliografia 11](#_Toc405922251)

[Prílohy 12](#_Toc405922252)

# Úvod

# Východiská

V tejto kapitole budú uvedené existujúce riešenia zálohovania dát a ich základný princíp fungovania.

## Existujúce zálohovacie riešenia

Duplicity: <http://www.nongnu.org/rdiff-backup/>

Time Machine: <https://support.apple.com/en-us/HT201250>

AMANDA: http://www.amanda.org/

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | rdiff-backup | duplicity | Time Machine | AMANDA | Git |
| Licencia | GPL | GPL | Apple Inc. | BSD | GPL |
| Windows/OS X/ Linux | ✓/✓/✓ | ✓/✓/✓ | 🗶/✓/🗶 | ✓/✓/✓ |  |
| Príkazový riadok | ✓/✓ | ✓/✓ | ✓/✓ | 🗶/✓ |  |
| GUI | ??? |  |  |  |  |
| Prog. jazyk | Python | Python | ? | C, Perl |  |
| Verziovanie | opačné inkrementálne | inkrementálne | inkrementálne | Inkrementálne | opačné inkrementálne |
| Kompresia |  |  |  |  |  |
| Delta kompresia |  |  |  |  |  |
| Jednoduché prezeranie záloh | najnovšia |  | všetky |  | všetky |
| Zmazanie ľubovoľnej zálohy |  |  |  |  |  |

### Rdiff-backup

Rdiff-backup[[1]](#footnote-1) je zálohovací software pre plne automatické inkrementálne zálohovanie a spätné zrekonštruovanie zálohovaných dát. Obnovovanie dát môže prebiehať na konkrétne zvolenom súbore alebo celej zálohe. Implementovaný je v jazyku Python a pri svojom behu využíva algoritmus rsync (viď kap. 1.3.2) implementovný v knižnici librsync.

Základné vlastnosti:

* Kopírovanie súborov lokálne alebo po sieti cez protokol SSH
* Zachovanie práv súborov a médií (zálohovanie na flash disk so súborovým systémom FAT)
* Obnovenie zmazaných súborov
* Štatistiky a záznamy o zálohách

Samotné zálohovanie prebieha tak, že na cieľovom adresári je vytvorená presná kópia zálohovaného adresára. Ak nastanú na zálohovaných súboroch zmeny, tak rdiff-backup pri ďalšom zálohovaní ukladá do hlavného adresára novú verziu zmenených súborov a do špeciálneho podadresára aj *opačné rozdiely* (*reverse diff*, teda rozdiely popisujúce rekonštrukciu staršej verzie z novšej) vypočítané algoritmom. Tie môžu byť neskôr využité pri obnove starších verzií súboru. Výhodou takéhoto prístupu je, že aktuálne verzie zálohovaných dát sú ihneď k dispozícii a stačí ich len prekopírovať.

Spolu s dátami sú k jednotlivým súborom udržiavané aj práva, vlastník a skupina, čas zmeny. Rdiff-backup dokáže zálohovať aj súbory zariadení, pipe, hardlink a symlink. Údržba práv umožňuje zálohovanie na ne-Unixové súborové systémy (napr. flash disk vo formáte FAT). …lokálne alebo SSH… …štatistiky…

Open source … možnosť obnovy bez rdiff-backup.

### Duplicity

Duplicity je software pre zálohovanie, ktorý vytvára šifrované (prípadne kryptograficky podpísané) tar archívy. Implementovaný je v jazyku Python a pri svojom behu taktiež využíva algoritmus rsync (knižnica librsync).

Základné vlastnosti:

* Kopírovanie súborov lokálne alebo po sieti cez protokol SSH
* Kryptovanie a podpisovanie vytvorených tar archívov pomocou GnuPG
* Inkrementálne zálohovanie

Princíp zálohovania v Duplicity je nasledovný. Každá záloha je ukladaná do tar archívu. Duplicity poskytuje aj možnosť kryptovania a podpisovania zálohovaných dát pomocou GnuPG. Pre ich využitie je potrebné si najskôr vygenerovať samotný GPG kľúč, prípadne digitálny podpis.

Postup zálohovania v Duplicity je nasledovný: Počiatočná záloha obsahuje všetky dáta (full backup). Pri ďalšej zálohe sa ukladajú len rozdiely (diffy) vypočítané algoritmom rsync voči pôvodným súborom na základe signatúr vypočítaných pri predchádzajúcich zálohovaniach, ktoré sa ukladajú spolu zo zálohou ale aj lokálne na zálohovanom počítači. Nevýhodou takéhoto prístupu môže byť čas, potrebný pre obnovenie starších záloh. Napríklad pri prístupe k najnovšej verzii zálohy je potrebné prejsť všetkými inkrementálnymi zálohami, ktoré boli vykonané.

***Sem pokec o dôraze na bezpečnosť: šifrovaní, podpisovaní a rôznych možnostiach ukladania záloh.***

Samotné Duplicity je nástroj ovládateľný cez príkazový riadok. Existuje k nemu však kvalitná grafická nadstavba s názvom Déjà Dup. Medzi jej hlavné výhody patrí napríklad plánovanie záloh, integrácia v Linuxovom grafickom rozhraní GNOME (Nautilus) alebo podpora pre cloudové zálohovanie. Vďaka integrácii v GNOME si v súborovom manažéri Nautilus dokážete jednoducho obnoviť akýkoľvek súbor bez toho, aby ste museli obnovovať celú zálohu.

### AMANDA (Advanced Maryland Automatic Network Disk Archiver)

AMANDA***[footnote s URL]*** **[citácia knihy o Amande]** je zálohovací software, ktorý umožňuje zálohovanie dát po sieti LAN. Funguje na princípe jedného zálohovacieho servera, ktorý sa stará o zálohovanie viacerých klientov (Windows, Linux, Unix, BSD, Mac OS-X). Používa k tomu natívne nástroje ako dump a GNU tar. Dump je Unixový nástroj vyvinutý pre zálohovanie diskových oddielov na úrovni blokov, nie súborov a adresárov. Kvôli spätnej kompatibilite ukladania záloh na pásky, sú jeho výstupom bloky pevných veľkostí. GNU tar je Unixový nástroj, ktorý slúži na archiváciu dát na úrovni súborov a adresárov. Oba nástroje možno kombinovať s kompresnými programami ako gzip, bzip2, xz.

Základné vlastnosti:

* Inkrementálne aj úplne zálohovanie
* Podpora šifrovania dát
* Zálohovanie viacerých platforiem – využíva k tomu buď samba protokol alebo svoj vlastný, natívny klient s názvom amanda-client
* Open Source
* Dáta su obnoviteľné aj bez samotného softwaru Amanda

### Time Machine

Time Machine***[footnote s URL]*** je vstavaná zálohovacia služba systému Mac OS X, ktorá vytvára zálohy na externý disk alebo na sieťové úložiská dát podporujúce protokol AFP.

Základné vlastnosti:

* Podpora šifrovania dát
* Záloha celého systému, vrátane systémových súborov, aplikácií, používateľských účtov atď.
* Kompletný obraz systému pre daný čas
* Nezmenené súbory i celé adresáre ukladá ako odkazy (hardlinky) k pôvodným súborom alebo adresárom, do zálohy kopíruje len nové a zmenené súbory a adresáre
* Možnosť vyhľadávania súborov a rýchly náhľad do súboru
* Udržiava hodinové zálohy za posledných 24 hodín, denné zálohy za posledný mesiac a týždenné zálohy

Time Machine má taktiež podporu pre zálohovanie na NAS zariadenia a servery podporujúce protokol AFP. Okrem zariadení od Apple tento protokol implementuje napríklad Unixový balík netatalk.***[pridať poznámku pod čiarou s URL]*** Záleží však na samotnej verzii softvéru. Kým staršie verzie dokázali spolupracovať s veľkým množstvom NAS serverov, najnovšie verzie vyžadujú server podporujúci AFP (Apple Filling Protocol).

### Git

Git[[2]](#footnote-2) ***[citácia knihy]*** je decentralizovaný / distribuovaný systém riadenia revízií (DVCS). Primárne slúži pre správu verzií zdrojových kódov. Bol vyvinutý pre vývoj Linuxového jadra. Väčšinou je však využívaný vývojármi pri práci na softwarových projektoch, ale dá sa využívať na akékoľvek dáta. Dá sa tak jednoducho zistiť, kedy sa daný súbor zmenil, ako sa zmenil a taktiež kým bol zmenený.

***Sem by som dal mostík k zálohovaniu, napríklad:*** Aj keď git nie je určený na zálohovanie, používa viacero techník, ktoré sa stali inšpiráciou pre náš zálohovací softvér: obsahom adresované úložisko dát, kombináciu deflate kompresie a delta kompresie, garbage kolekciu.

Základné vlastnosti:

* Git je distribuovaný systém, čo znamená, že nepotrebuje žiadny centrálny bod, s ktorým by musel udržiavať spojenie. Všetky základné operácie sú vykonávané lokálne.
* Kompletná história bez možnosti zmeny
* Kompatibilita s existujúcimi systémami a protokolmi: HTTP, FTP, SSH, rsync, alebo komunikácia priamo cez socket
* Garbage kolekcia v obsahom adresovanom úložisku
* Delta kompresia
* Deflate kompresia
* Viaceré vetvy vývoja

Pojem garbage kolekcia bude zadefinovaný a vysvetlený neskôr v kapitole 1.2.4. V Gite je využívaná garbage kolekcia hlavne za účelom exportovania súborov, ktoré už nie sú viac využívané a označí ich ako „stratené“ objekty (loose objects). Pri hľadaní týchto súborov garbage kolekcia mení samotnú štruktúru repozitára na efektívnejšiu a snaží sa tak redukovať diskový priestor a zvýšiť výkonnosť.

Delta kompresia (delta kódovanie) je mechanizmus ukladania prípadne prenášania dát vo forme rozdielov medzi jednotlivými verziami. Rozdiely sú ukladané do súborov, ktoré sú nazývané taktiež delty alebo diffy. Git si vždy ukladá celý obsah najnovších súborov a rozdiely medzi súbormi sú ukladané pomocou delta kompresie len ak sú označené ako „stratené“ súbory (loose objects). Tieto súbory sú následne komprimované do binárnych súborov nazývaných tiež balíčkové súbory (packfile). Vďaka tomuto prístupu dokáže Git šetriť diskový priestor a byť efektívnejší.

## Predchádzajúca bakalárska práca

Táto diplomová práca vychádza z bakalárskej práce *Zálohovanie v Linuxe s úsporným ukladaním histórie*, ktorej autorom je Patrik Kemény ***[odkaz do literatúry]***. Jej cieľom bolo navrhnúť a implementovať v operačnom systéme Linux nástroje, ktoré budú vykonávať zálohovanie dát na vzdialený server a následne umožnia ich sprístupnenie. Kľúčové vlastnosti týchto nástrojov mali byť: úsporný prenos dát na vzdialený server, úsporné ukladanie záloh s ich históriou (kompresia dát), vymazávanie starších záloh a sprístupnenie záloh formou postupnosti úplných obrazov systému súborov. Pre dosiahnutie úsporného prenosu dát mali byť využité techniky nástroja rsync a git. Následné sprístupňovanie záloh malo byť realizované pomocou FUSE. Ako inšpirácia pre túto prácu mohli byť použité aj existujúce zálohovacie nástroje ako duplicity alebo rdiff-backup.

Autor však splnil len niektoré body zo stanoveného cieľu tejto bakalárskej práce. Vyvinutý zálohovací nástroj dokáže vytvárať a obnovovať plné alebo inkrementálne zálohy. Úsporné ukladanie záloh bolo implementované pomocou inkrementálneho zálohovania. Ak už v danom obsahom adresovanom úložisku existuje súbor s rovnakým obsahom ako sa snažíme zálohovať, tak bude uložený len odkaz (link) na tento súbor. Žiadne ďalšie techniky pre úsporné ukladanie dát neboli implementované.

## Techniky a použitý softvér

Východiskami pre náš prístup k zálohovaniu a obnove dát sú techniky a existujúce softvérové nástroje, ktorých prehľad uvádzame v tejto sekcii. Najdôležitejšie techniky sú obsahom adresované úložisko, algoritmus rsync, delta kompresia, garbage kolekcia na základe počítania referencií a virtuálne súborové systémy. Využili sme tiež konkrétne implementácie: na delta kompresiu používame program rdiff implementujúci algoritmus rsync, na bezstratovú kompresiu formát gzip, v ktorom sú dáta komprimované algoritmom deflate, pre virtuálny súborový systém využívame techonológiu FUSE.

### Obsahom adresované úložisko

Obsahom adresované úložisko dát (*content-addressed storage*) ***[zdroj!]*** funguje na nasledovnom princípe. Na strane klienta, je buď aplikačným serverom alebo samotným klientom vytvorený objekt na základe jeho obsahu (content address). Ten je následne odoslaný do úložiska a je pre neho vypočítaná adresa (hash), podľa ktorého je neskôr daný objekt jednoznačne identifikovateľný. Vypočítaná adresa je následne odoslaná späť aplikácií, ktorá pre ďalšie referencie na daný objekt, pracuje už len s ňou.

Výhody a nevýhody:

* Najvhodnejšie na dátach, ktoré sa často nemenia (kvôli častému počítaniu / prepočítavaniu adries pre jednotlivé súbory)
* Rýchle vyhľadávanie pre obsah daného súboru
* Nikdy neexistuje viac ako jedna kópia daného súboru v úložisku (dva také isté súbory majú rovnakú obsahovú adresu)

### Delta kompresia

*Delta kompresia* (rozdielová kompresia, delta compression) ***[citácia]*** je technika kompresie dátových objektov (napríklad súborov), pri ktorej sa namiesto obsahu objektu ukladá iba *rozdiel* (*delta*, *diff*) voči obsahu iného dátovému objektu. Obyčajne ide o *dopredný* rozdiel (forward delta) voči chronologicky predchádzajúcej verzii príslušného objektu alebo *spätný* rozdiel (reverse delta) voči jeho nasledujúcej verzii.

Pre textové dáta možno rozdiel počítať napríklad algoritmom ***ZISTITE AKÝM A CITUJTE***, ktorý je implementovaný v štandarnom Unixovom nástroji *diff* a knižnici ***DOPLŇTE***. Pre textové a binárne dáta je možné na výpočet rozdielu použiť algoritmus rsync, opísaný v nasledujúcej podsekcii.

### Algoritmus rsync

Algoritmus rsync vyvinul Andrew Tridgell ***[citácia]*** na zefektívnenie prenosu dát, ak sa na cieľovom počítači už nachádza ich predchádzajúca verzia.

Algoritmus efektívne počíta rozdiely medzi súbormi pomocou signatúr. V algoritme sú použité dva typy signatúr, silná („strong checksum“) a slabá („weak checksum“ alebo „rolling checksum“) signatúra. Slabá signatúra musí byť veľmi rýchla a lacná na výpočet. Silná signatúra musí mať veľmi nízku pravdepodobnosť vzniku kolízie a je počítaná len vtedy, keď sa dve slabé signatúry zhodujú. Princíp fungovania rsync algoritmu je nasledovný. Predstavme si, že máme dva počítače *A* a *B*. V počítači A sa nachádza súbor, ktorý pozostáva ai bajtov a v počítači B sa nachádza súbor, ktorý pozostáva z bi bajtov. B rozdelí týchto bi bajtov na N rovnakých blokov a vypočíta slabú aj silnú signatúru ku každému z nich. Následne sú tieto signatúry odoslané na počítač A. Tu sa vypočítajú pre každý bajtový posun i v ai slabé signatúry pre každý blok začínajúci na pozícii i. Takto získané signatúry sú porovnané s každou slabou signatúrou získanou z počítača B. Ak nastane prípad, že sa dve signatúry rovnajú, A ďalej vypočíta aj silnú signatúru a porovná ju so silnou signatúrou z B, ktorá patrí danému bloku. Ak sa zhodujú aj tieto dve signatúry, tak A odošle správu B o mieste zhody a ktoré bloky sa zhodujú. Ak sa nezhodujú, tak sú odoslané len samotné bajty. Napokon sú na počítači B použité tieto informácie na rekonštrukciu ai. K výpočtu oboch signatúr je používaný kryptografický hašovací algoritmus MD4, ktorý generuje 128 bitové signatúry.

Samotná rekonštrukcia súboru je jednou z najjednoduchších častí rsync algoritmu. Akonáhle sú na počítač B odoslané všetky informácie o zhodujúcich sa blokoch alebo samotné bajty, tak rekonštrukcia súboru sa začne vykonávať. Pri rekonštrukcii súboru sa postupne zapisujú zmenené bajty získané z počítača A alebo jednotlivé zhodujúce sa bloky bajtov v pôvodnom súbore, nachádzajúcom sa na počítači B.

Ako sa rsync dá využiť na delta kompresiu, čo je *rozdiel/diff/delta*, ktorý spomíname v predchádzajúcej sekcii.

### FUSE

FUSE (Filesystem in Userspace) ***[odkaz na knihu o FUSE, footnote s URL],*** teda súborový systém v užívateľskom priestore, je technológia, ktorá umožňuje bežným používateľom Unixových operačných systémov (Linux, FreeBSD, NetBSD, OpenSolaris, Mac OS X) pripájať do stromu priečinkov a súborov vlastné súborové systémy, implementované programami, ktoré bežia mimo jadra (kernelu) operačného systému. Umožňuje tak okrem iného vytváranie virtuálnych súborových systémov z akýkoľvek dát a poskytovaných služieb, nie len z dát uložených na pamäťových zariadeniach priamo pripojených k počítaču.

Súborový systém funguje tak, že kernel dostane požiadavku od klientskej aplikácie, napríklad „vypíš adresár“, a tú prenechá ovládaču súborového systému. Ten následne požiadavku obslúži a spracuje odpoveď, ktorú kernel vráti aplikácii.

Ovládač súborového systému (napr. ext3) je kernelový modul, ktorý s kernelom komunikuje cez rozhranie VFS (virtual file system) a je pevnou súčasťou samotného kernelu. Takýto súborový systém môže obsluhovať len privilegovaný používateľ (root) a ten prideľuje práva k jednotlivým pripájacím bodom (mount points).

FUSE vystupuje v kerneli ako obyčajný súborový systém (viď obrázok 1). Nezaujíma sa však o obslúženie požiadavky a spracovanie odpovede, ale namiesto toho, pomocou knižnice libfuse komunikuje s bežnou používateľskou aplikáciou a tá vykoná všetko potrebné. Akonáhle táto obslužná aplikácia vráti odpoveď v požadovanej forme, FUSE ju prenechá kernelu a kernel ju vráti klientskej aplikácii.

FUSE a aplikácie obsluhujúce súborové systémy musia pre prácu so súbormi implementovať dôležité metódy ako napríklad open, close, read, write alebo seek, definované skupinou štandardov POSIX (Portable Operating System Interface).

Obrázok 1. Tok riadenia FUSE



Výhody:

* Celá logika súborového systému je zapísaná v programe, prípadne skripte, ktorý je nezávislý na platforme či konkrétnom operačnom systéme, pokiaľ implementuje FUSE
* Súborové systémy môže obsluhovať aj bežný uživateľ.
* Stabilné API pre FUSE – nie je potrebné prepisovať programy pri zmene kernelovej časti súborového systému.

Ako môžeme vidieť na obrázku 1, FUSE a taktiež všetky iné súborové systémy využívajú VFS (virtual file system). VFS poskytuje pre súborové systémy abstraktné rozhranie pre komunikáciu s kernelom. FUSE musí preto pre prácu so súbormi implementovať dôležité metódy ako napríklad open, close, read, write alebo seek. Toto rozhranie je definované skupinou štandardov s názvom POSIX (Portable Operating System Interface).

Najznámejšie existujúce FUSE aplikácie sú napríklad sshfs, ntfs-3g, exfat-fuse. Sshfs (secure shell file system) je súborový systém založený na SFTP (secure file transfer protokol) protokole, ktorý umožňuje bezpečný prístup k súborom na vzdialenom počítači. Vďaka tomu, že väčšina SSH serverov podporuje tento protokol, tak samotné používanie sshfs je veľmi jednoduché. Stačí len vykonať pripojenie (mount) takéhoto súborového systému.

Ntfs 3g a exfat-fuse sú open source read – write ovládače (drivery). Ntfs 3g je implementáciou súborového systému NTFS od Microsoftu. Medzi jeho hlavné výhody patrí napríklad žurnalovanie, kompresia alebo podpora šifrovania.

Exfat-fuse je implementáciou súborového systému exFAT (Extended File Allocation Table), ktorý je vhodný hlavne pre flash disky. Využíva namiesto NTFS, ktorý je nevhodný pre takéto zariadenia, kvôli množstvu dátových štruktúr, ktoré obshauje.

Využitie FUSE je však skutočne široké. Existujú zaujímavé virtuálne súborové systémy pre Wikipédiu (WikipediaFS), Gmail (GmailFS) alebo Dropbox (DropboxFS). Pomocou WikipediaFS môžu používatelia Wikipedie priamo editovať články ako lokálne súbory vo vlastnom textovom editore. GmailFS je súborový systém, pomocou ktorého dokážete využívať svoje Gmailové konto ako úložisko dát.

### Garbage kolekcia na základe počítania referencií

Garbage kolekcia je mechanizmus, ktorý slúži na automatickú správu zdrojov v počítači. Najčastejšie sa však využíva na správu pamäte. Jeho úlohou je v tomto prípade získavať pamäť, ktorá už nie je využívaná prostriedkami (programy a procesy). Uľahčuje tak aj prácu programátorom, ktorí by vždy museli manuálne definovať, ktoré objekty môžu byť dealokované a uvoľniť tak pamäť systému.

Najdôležitejší problém, ktorý riešia všetky garbage kolektory, je priradenie objektov do dvoch skupín. Na objekty, ktoré sú dosiahnuteľné a na objekty, ktoré nie sú dosiahnuteľné.

Najjednoduchšou implementáciou garbage kolektora je pomocou metódy počítania referencií (reference counting). Funguje na nasledovnom princípe. Ku každému objektu je asociovaný počet referencií na daný objekt z iných objektov. Ak hodnota klesne na 0, objekt je následne označený ako nedosiahnuteľný a je alebo bude uvoľnený z pamäte. Táto metóda však nedokáže pracovať s cyklickými štruktúrami.

## Gzip a kompresia Deflate

GZIP je označením pre typ súboru, ale aj pre konkrétnu implementáciu softvéru slúžiaceho na kompresiu a dekompresiu súborov.

Ako formát súboru, je GZIP založený na bezstratovom kompresnom algoritme Deflate, ktorý je kombináciou slovníkového algoritmu LZ77 a Huffmanovho kódovania. Jeho autorom je Phil Katz. Samotná štruktúra súboru typu gzip je definovaná nasledovne. Na začiatku sa nachádza 10 bajtová hlavička, ktorá obsahuje „magické“ číslo (magic number), číslo verzie, časový údaj a môže obsahovať aj pôvodné meno súboru. Ďalej nasledujú samotné dáta, ktoré sú kompresované pomocou algoritmu Deflate. Na konci sa nachádza 8 bajtová pätička, ktorá obsahuje kontrolný súčet a údaj o dĺžke pôvodných nekompresovaných dát.

Algoritmus Deflate v porovnaní s alternatívami ako Burrows-Wheeler (bzip2) alebo LZMA (xz) zvyčajne neposkytuje najlepší kompresný pomer, ale jeho výhodou je rýchla kompresia aj dekompresia, malé pamäťové nároky a dostupnosť v rôznych operačných systémoch. Preto sa často používa, napríklad aj v kompresii šifrovanej komunikácie cez SSH alebo pri prenose hypertextových dát protokolom HTTP.

## Rdiff

Rdiff je Unixový nástroj, ktorý slúži na počítanie signatúr a ich následné využitie pre zisťovanie rozdielov medzi dvoma súbormi. Je založený na rovnakom princípe ako Unixový nastroj rsync, ktorý je priamou implementáciou algoritmu rsync, avšak dáva používateľovi priamu kontrolu nad jednotlivými operáciami. Rdiff dokáže vykonávať tieto operácie:

* Výpočet a vytvorenie súboru so signatúrou pre zadaný súbor
* Výpočet a vytvorenie delta súboru na základe nového súboru a signatúry jeho pôvodnej verzie
* Aplikovanie delta súboru na pôvodný súbor a následné vytvorenie nového súboru (patch)

### Databáza typu kľúč-hodnota CodernityDB

Databázy kľúč-hodnota patria k najjednoduchším typom databáz. Z pohľadu systému manažujúceho databázu má každý záznam iba dve polia – kľúč, ktorý slúži na adresovanie záznamu a hodnotu, ktorá je s kľúčom asociovaná. Kľúčom i hodnotou často môže byť ľubovoľná postupnosť binárnych dát. Pre efektívne adresovanie sú kľúče indexované v B-strome alebo hašovacej tabuľke. Takéto databázy často nemajú architektúru klient-server, ale sú implementované ako embedované. Celá funkcionalita databázy je teda implementované ako knižnica, ktorá sa prilinkuje ku klientskej aplikácii. K implementáciám patria napríklad BDB (Berkeley DataBase),

CodernityDB je databáza typu kľúč-hodnota (key-value) s otvoreným zdrojovým kódom, ktorá disponuje viacerými pokročilejšími funkciami. Je naprogramovaná v jazyku Python, a jej kľúčovými vlastnosťami sú napríklad rýchlosť (100 000 vložení alebo výberov za sekundu), multiplatformovosť či podpora viacerých indexov. Samotná databáza si udržiava informácie o indexoch a väčšinu operácií vykonáva na nich. CodernityDB poskytuje používateľom štyri rôzne druhy databáz:

* Database – pre využitie v jednoprocesovom, jednovláknovom prostredí
* DatabaseThreadSafe – pre využitie s viacero vláknami (čítanie neblokuje zápis)
* DatabaseSuperThreadSafe – obmedzenie na jednu databázovú operáciu v danom čase
* CodernityDB-HTTP – HTTP verzia databázy pre v prostrediach s viacerými procesmi / vláknami

# Návrh riešenia

V tejto kapitole bude popísaný návrh riešenia samotnej aplikácie.

## Východiská

Hlavné východiská tejto práce sú:

* Navrhnúť obsahom adresované úložisko dát
* Implementovať prezeranie, údržbu a obnovu záloh / dát
* Využitie kompresie
* Implementovať garbage kolekciu pre údržbu záloh
* Implementovať FUSE pre prezeranie záloh a možnosť jednoduchého kopírovania dát

Keďže táto práca čiastočne vychádza z pôvodnej bakalárskej práce, využijeme z nej už navrhnuté obsahom adresované úložisko, pre ďalšiu implementáciu a rozvoj.

## Obsahom adresované úložisko

Pôvodne navrhnuté obsahom adresované úložisko pozostávalo zo samotných zálohovaných objektov a zo súborov, ktoré uchovávali časové údaje jednotlivých záloh. Túto štruktúru sme využili a ďalej rozšírili o novú funkcionalitu. V našej navrhovanej implementácii obsahom adresovaného úložiska pribudne žurnálovací systém a interná databáza, ktorá bude využitá pri implementácii garbage kolekcie. Ako ďalšie pribudnú hlavičkové súbory k jednotlivým zálohovaným objektom, ktoré budú definovať spôsob uloženia daného objektu.

## Prezeranie, údržba a obnova záloh

V pôvodnej implentácii bolo prezeranie záloh možné len pri úplnej obnove dát a chýbala akákoľvek údržba dát. Preto sme sa rozhodli pre tieto účely navrhnúť vlastnú implementáciu FUSE a využiť existujúce techniky garbage kolekcií.

### FUSE

Jedným z hlavných východísk aplikácie je prezeranie záloh. Doposiaľ sme si obsah záloh však mohli prezerať len po ich úplnej obnove. Takýto prístup je však veľmi nepraktický z viacerých dôvodov:

* Potreba voľného ukladacie priestoru na obnovu dát (nevýhoda hlavne pri veľkých zálohách)
* Čas obnovy pri veľkých zálohách
* Kvôli jednému súboru je potrebné obnoviť celú zálohu

Preto sme sa v našom riešení rozhodli využiť a implementovať FUSE. Vďaka jeho využitiu dokážeme eliminovať všetky vyššie spomenuté nevýhody. Používateľ si tak bude môcť pripojiť svoju zálohu ako virtuálny disk, prezerať si jeho obsah a následne si z neho vyextrahovať len tie dáta, o ktoré bude mať záujem.

### Garbage kolekcia

Ďalším hlavným východiskom aplikácie je údržba záloh. V pôvodnom riešení však nebol navrhnutý ani implementovaný žiadny spôsob údržby záloh.

V existujúcej implementácii rozlišujeme dva druhy záloh. Plnú zálohu a inkrementálnu. Inkrementálna záloha je vytváraná tak, že ak sa obsah daného súboru voči pôvodnému nijak nezmenil, tak je uložený len odkaz (link) na pôvodný súbor. Inak sa na súbore vykoná záloha. Novšie zálohy sa tak stávajú závislými na starších. Výhodou takéhoto prístupu je šetrenie ukladacím priestorom. Problém však nastáva pri mazaní jednotlivých záloh. Ak by sme sa napríklad rozhodli vymazať najstaršiu zálohu, avšak najnovšia záloha by stále obsahovalo pôvodný nezmenený súbor z najstaršej. A práve pre tieto účely sme sa rozhodli využívať techniky garbage kolekcie. Jej hlavnou úlohou bude udržiavanie záloh, teda mazanie už viac nepotrebných súborov alebo celých záloh.

## Využitie kompresie

Pre využitie kompresie dát sme sa rozhodli využiť dve existujúce techniky, ktoré boli podrobnejšie popísané v predchádzajúcej kapitole. Objem dát sa budeme snažiť najskôr redukovať pomocou algoritmu rsync a jeho Unixovej implementácie rdiff. Ak sa v úložisku už bude nachádzať predchádzajúca verzia daného súboru, tak pomocou nástroje rdiff vytvoríme len rozdielový súbor a ten bude sa uložení. Následne budú všetky súbory uchované v úložisku podliehať kompresii pomocou nástroja GZIP. Týmto spôsobom by sme mali zaručiť rozsiahle šetrenie ukladacím priestorom.

## Žurnálovací systém

Pri zálohovaní dát môže nastať situácia, pri ktorej je zálohovací proces prerušený, či už systémom alebo samotným používateľom. V takomto prípade môže nastať nekonzistencia dát v úložisku a teda aj v samotnej zálohe. Aby sme sa vyhli takýmto situáciám, rozhodli sme sa navrhnúť jednoduchý žurnálovací systém, ktorý by mal fungovať nasledovne. Pri vytváraní záloh sa budú všetky zmeny ukladať a zapisovať do tohoto systému. Ak nastane prerušenie zálohovacieho procesu, v žurnálovacom systéme sa vyhodnotí, či boli všetky dáta úspešne zálohované alebo nie. Ak nie, systém takúto nekonzistentnú zálohu aj s dátami odstráni.

# Implementácia

V tejto kapitole bude rozoberaná konkrétna implementácia navrhnutých riešení z predchádzajúcej kapitoly a pre lepšiu predstavivosť a pochopenie budú uvedené aj konkrétne príklady.

# Záver

# Bibliografia

[1] ANDREW, Tridgell. Efficient Algorithms for Sorting and Synchronization [Dátum Júl 1999] http://gan.anu.edu.au/~brent/pd/Tridgell-thesis.pdf

[2] SCOTT, Chacon. Pro Git [Dátum 8. Február 2010] <http://labs.kernelconcepts.de/downloads/books/Pro%20Git%20-%20Scott%20Chacon.pdf>

[3] CURTIS, Preston. Backup & Recovery (4. kapitola)

[Dátum Január 2007]

[4] <https://www.kernel.org/doc/Documentation/filesystems/fuse.txt>

[5] L. PETER Deutsch, DANIEL G. Bobrow. An efficient, incremental, automatic garbage

collector

[Dátum 9. September 1976]

<http://dl.acm.org/citation.cfm?id=360345>

[6] HENRY Lieberman, CARL Hewitt. A real-time garbage collector based on lifetimes of

objects

[Dátum 6. Jún 1983]

http://dl.acm.org/citation.cfm?id=358147

# Prílohy

1. CD so zdrojovými kódmi aplikácie a bakalárskou prácou vo formáte PDF

1. <http://www.nongnu.org/rdiff-backup/> [↑](#footnote-ref-1)
2. http://git-scm.com [↑](#footnote-ref-2)