VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

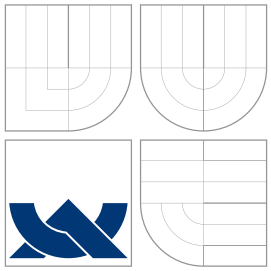
PODPORA SNAPSHOTU A ROLLBACKU PRO KONFIGURAČNÍ SOUBORY V DISTRIBUCI FEDORA

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

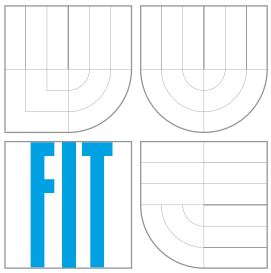
AUTOR PRÁCE
AUTHOR

Bc. MICHAL JEŽEK

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

PODPORA SNAPSHOTU A ROLLBACKU PRO KONFIGURAČNÍ SOUBORY V DISTRIBUCI FEDORA

SNAPSHOT AND ROLLBACK SUPPORT FOR CONFIGURATION FILES ON FEDORA DISTRIBUTION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MICHAL JEŽEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. ALEŠ SMRČKA

BRNO 2008

Abstrakt

Cílem této diplomové práce je navrhnout a implementovat nástroje pro podporu snapshotu a rollbacku konfiguračních souborů na distribuci GNU/Linuxu. Sada nástrojů umožňuje pravidelné/automatické ukládání konfiguračních souborů do zvoleného umístění. Aktualizace záloh reagují na události na souboru sledováním změn pomocí podsystému jádra inotify. Nástroje umožňují návrat k libovolné vybrané záloze. Způsob aktualizace záloh je konfigurovatelný. Nástroj umožňuje porovnávat data z libovolných záloh, zobrazit rozdíly v konfiguracích a případně provést sloučení mezi aktuálními souboury a vybranou zálohou. Nástroje také umožňují porovnání jak konfigurace z jednoho klienta, tak i konfigurace klientů mezi sebou. Mezi klienty je zároveň možné zobrazit rozdíly a případně provést sloučení.

Klíčová slova

Snapshot, rollback, inotify, zálohování, konfigurace, událost, Linux, podsystém jádra, systém správy verzí, SCM

Abstract

The purpose of this thesis is to design and implement tools for support of a snapshot and a rollback for configuration files on the GNU/Linux distribution. The set of the tools enables an automatic/periodical saving of the configuration files into the selected placement. The creation of backups reacts to file events by watching the changes with kernel subsystem inotify. Tools are enabling to return to the selected backup. The way of the backup actualization is configurable. This tool permits the data comparison from selected backups, to show the differences in configurations and eventually to manage a merge among actual and selected backup. Tools also allows a comparison of a configurations of one client or configurations among clients, and to display the mutual differences, eventually to manage their merge.

Keywords

Snapshot, rollback, inotify, backup, configuration, event, Linux, kernel subsystem, source code management, SCM

Citace

Michal Ježek: Podpora snapshotu a rollbacku pro konfigurační soubory v distribuci Fedora, diplomová práce, Brno, FIT VUT v Brně, 2008

Podpora snapshotu a rollbacku pro konfigurační soubory v distribuci Fedora

Prohlášení

Prohlašuji, že jsem tento diplomový projekt vypracoval samostatně pod vedením pana Ing. Aleše Smrčky. Další informace mi poskytl pan Ing. Radek Vokál ze společnosti RedHat Inc. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

. . . . . .
Michal Ježek
19. května 2008

Poděkování

Děkuji vedoucímu diplomové práce, panu Ing. Aleši Smrčkovi, za cenné rady poskytnuté při vytváření této práce. Rovněž bych chtěl poděkovat za spolupráci panu Ing. Radku Vokálovi.

Obsah

Kapitola 1

Úvod

Téma této práce jsem vybral, neboť úzce souvisí s operačním systémem[1] GNU/Linux. Tento OS byl pro mne vždy příležitostí naučit se mnoho nového. Svojí otevřeností mi umožnil nahlédnout do každé jeho části na kterékoliv úrovni. Přístupem k zdrojovým kódům tohoto OS jsem se mnoho naučil o programování.

Jedním z aspektů používání OS GNU/Linux je nastavení pomocí textových souborů. Častokrát jsem při používání tohoto OS s nastavením značně experimentoval. Tyto experimenty vyžadovaly zálohování funkčního nastavení. Následně přehled v těchto zálohách nebyl snadný. Vytvořením nástrojů, které řeší toto zálohování, bych se chtěl tento problém usnadnit. Projekt by měl pomoci dalším uživatelům při spravování OS GNU/Linux.

Tyto nástroje by měly pomoci řešit problémy, na které uživatel narazí během hledání nového nastavení. Nástroje sice nepomohou uživateli vybrat nastavení nové, ale umožní uživateli vycházet z předchozího nastavení, případně se k některému nastavení v případě potřeby vrátit. Nástroje by měly být nápomocné ať už začínajícím uživatelům nebo zkušeným uživatelům OS GNU/Linux.

První distribuce GNU/Linux, pomocí které jsem se seznámil s tímto OS, se nazývala RedHat Linux. V současnosti už je tato distribuce poskytována jako komerční produkt. Pro otevřený vývoj softwaru zde byla ponechána distribuce Fedora, která navazuje na počáteční otevřený vývoj distribuce RedHat Linux. Práce na tomto projektu s podporou z této společnosti mi umožňuje nahlédnout do vývoje softwaru pro distribuci Fedora.

Kapitola 2 popisuje mou představu činností, které by měly navrhované nástroje provádět. Tuto představu se pokusím nastínit uvedením do problému zálohování nastavení a zároveň uvedením okruhu funkčnosti nástrojů. Dále také uvádím některá další řešení se stejným zaměřením. Informace o těchto řešeních jsou stručně shrnuta v poslední části této kapitoly.

Kapitola 3 prezentuje navrhované řešení. Popisuje mou představu nástrojů,, pomocí kterých by bylo možné dosáhnout činností uvedených v kapitole předchozí. Popisuje návrhy jednotlivých částí, z kterých se projekt skládá.

Kapitola 4 poskytuje náhled do implementačních detailů, které jsem během projektu řešil. Tato kapitola vysvětluje výběr hlavních programových součástí použitých pro implementaci nástrojů tohoto projektu.

Kapitola 5 obsahuje testovací případy, kterými jsem ověřoval funkčnost vytvořených nástrojů. Tyto testy pokrývají činnosti, které jsou vyžadovány po vytvořených nástrojích. Testy jsou rozděleny podobně jako je rozdělen návrh těchto nástrojů.

[1] V dalším textu budu používat pro označení operační systém zkratku OS.

Kapitola 2

Specifikace a současný stav

V této kapitole se pokusím specifikovat jaké činnosti bude diskutovaná aplikace vykonávat. Z jakých problémů vychází toto řešení a také některé z předpokladů, které mne vedly k výslednému návrhu.

2.1 Nástin problému

Administrace OS GNU/Linux je velkou skupinou správců chápána jako uživatelsky nepřívětivá, což je ve většině případů způsobeno tím, že konfigurace systému se provádí pomocí textových souborů. Tyto textové soubory je nutné pro změnu konfigurace ručně editovat, což je nepohodlné pro většinu uživatelů, kteří běžně používají okenní aplikace. Pro tento OS je možné nalézt mnoho praktických aplikací s grafickým uživatelským rozhraním[1], které usnadňují nastavování tohoto systému.

Přestože tyto aplikace pracují s GUI, samotné nastavení se ukládá opět do textových konfiguračních souborů. Možnost přístupu k nastavení bez GUI má i své výhodné stránky. Například možnost jednoduchého přístupu k této konfiguraci vzdáleně přes síť, kdy nemusí být GUI vždy dostupné. Pro zkušenější uživatele je také mnohem rychlejší editovat nastavení v oblíbeném textovém editoru, než procházet mnoha okny.

Jedná se o soubory s malým objemem, nicméně s velkým významem. Obsah souborů je velice důležitý pro běh OS a používaných aplikací. Tyto soubory obsahují často velice komplexní údaje o tom, jak se mají OS nebo aplikace během provozu chovat. Například konfigurační soubory webového serveru Apache nebo konfigurační soubor grafického rozhraní oken X. Nastavení se skládá z mnoha komentářů, klíčových slov a jejich odpovídajícím hodnotám. Popis těchto klíčových hodnot mnoho usnadní, nicméně situaci to příliš nezlepší.

Nastavení se provádí změnou hodnot těchto klíčových slov, které nastavovaná aplikace rozezná. Podle těchto hodnot je pak ovlivněn běh a chování aplikace. Běh a chování aplikace vycházející z nastavení nemusí vždy odpovídat nárokům uživatele, který nastavení provedl. Když není nastavení zapsáno správně, mohou vzniknout problémy. Někdy může nastavení znemožnit běh aplikace, případně změní její chování do té míry, že aplikace není použitelná.

U většiny aplikací může hledání funkčního nastavení zabrat mnoho času. Tato funkčnost může být dosažena i po několika po sobě následujících změnách nastavení před dosažením ideálního stavu, pokud je takovýto stav vůbec dosažitelný.

V průběhu času se nastavení mění do takové míry, že někdy může být obtížné stanovit, jaké bylo nastavení před několika dny. V horším případě se může vyskytnout nutnost určit jaké nastavení bylo před týdny, měsíci atd.

[1] Dále už jen GUI. V anglické i české literatuře se běžně používá výraz GUI (anglicky Graphic User Interface).

OS GNU/Linux je také velice často využíván v síťovém prostředí a je možné předpokládat, že jeden uživatel bude spravovat více než jeden systém. Dále je možné, že v těchto sítích je více uživatelů, kteří se starají o jeho nastavení. To vytváří další obtíže v práci s konfiguračními soubory a zpětném sledování změn konfigurací.

Zálohování těchto konfigurací, přes takto dlouhá období, případně uchovávání různě funkčních nastavení, může být obtížné. Tento problém je možno řešit kopírováním do jiného umístění, případným přejmenováváním nebo přidáváním originálních přípon za jméno souboru. Problém může nastat pokud počet těchto záloh přeroste v neúnosnou mez.

V případě, že po delší době se provede další změna nastavení, po které přestane zvolené nastavení poskytovat požadovanou funkčnost, může být opětovné hledání funkčního nastavení značně složité. Bez záznamů o tom, za jakým účelem záloha vznikla, je pak vyhledání funkčního nastavení značně problematické. Informace umožňující orientaci v zálohách nemusí být zcela komplexní údaj. Nicméně je to už dostatečně rozsáhlý údaj nato, aby byl uváděn v názvu souboru, případně příponě. Záznamy o těchto údajích do dalšího souboru může situace také znepřehlednit.

2.2 Navrhovaná aplikace

Navrhovaný systém se zabývá zálohováním[2] těchto konfiguračních souborů. Zálohováním jednotlivých změn, které v průběhu sledování konfiguračních souborů nastaly. Tím je míněna změna hodnot u klíčových slov. Tyto změny mohou být malé jednořádkové editace, ale také změny celých bloků, případně i přidání a odebrání souborů. Systém by měl provádět zálohování způsobem, který je možné provádět neomezeně dlouho a stále udržovat určitý přehled o posloupnosti záloh.

Provádění záloh by také nemělo klást nároky na uživatele, u kterého lze očekávat, že sám má plno práce se samotným nastavením systému. Provádění záloh by mělo usnadnit uživateli práci se systémem a ne přidávat další práci navíc. Největším nárokem by mělo být nastavení aplikace a poté už jen používání výhod, které tato aplikace poskytuje.

Uživatel by měl mít k dispozici možnost vytvořit zálohu manuálně. Tímto způsobem by se mělo nahradit řešení, jakým způsobem a kam uložit konfigurační soubory, u kterých lze předpokládat návrat ke stavu před změnou jeho obsahu. Počet záloh by neměl být omezen na jeden záložní soubor tak, jak je tomu například u některých textových editorů. Počet záloh by měl být neomezený, ale každá z těchto záloh by měla být dohledatelná a obnovitelná. Tato dohledatelnost by měla spočívat v možnosti vyhledání informací uložených spolu se zálohou. Tedy například pořadí zálohy, případně, co která záloha obsahuje. Při ukládání záloh by měl uživatel mít možnost přidat údaj, proč tuto zálohu vytváří. Údaj o času a datu vzniku zálohy, stejně jako údaj, ke kterému nastavení v konfiguračnímu souboru se záloha vztahuje.

Velkou službou uživateli by měla být možnost provádět zálohy automaticky bez nutnosti opakovaně provádět manuální zálohy. Automatické ukládání záloh podle nastavení, které provedl uživatel. Toto nastavení by přehledně a účelně umožňovalo editaci, kdy a za jakých okolností vytvářet automatickou zálohu. Nastavení by mělo být přiřazeno k okruhům souborů. Tyto okruhy by určovaly, které soubory, případně adresáře, budou obsaženy v záloze. Dále pak také, které soubory naopak v zálohách být nemají. U každého tohoto okruhu by mělo být možné upřesnit při jaké události k záloze dochází.

Jednou z těchto událostí je zápis do souboru, který byl přednastaven pro sledování. To znamená, že uživatel změnil nastavení v souboru a je tedy vhodná doba pro vytvoření zálohy změněného

[2] V dalším textu bude používáno označení snapshot (snímek obsahující zálohovaná data). Pro obecné popsání návrhu postačí uvést, že se jedná o zálohu.

souboru. Tato záloha bude provedena při každém zápisu do souboru, jedná se tedy o vhodné řešení pro soubory, které jsou editovány po dlouhém časovém období.

Další takovouto událostí je časový interval, tedy jeho konec. Uživateli by mělo být umožněno přednastavit zálohování v určitý čas, například provést zálohy jednou denně. Takové nastavení může být vhodné například na produkčním serveru, na kterém se mění nastavení každý den. Jako další příklad může být nově instalovaný OS, kde se nastavení může měnit každých pár minut.

V obou případech je ukládání po přednastavené události podstatným ulehčením situace pro uživatele, který nemusí ztrácet čas manuálním udržováním souvislé řady záloh. U každé takové automatické zálohy by měla být uvedena událost, při které tato záloha vznikla. Podobně jako u manuálních záloh, je údaj o času a datu vzniku zálohy nutností. Dále pak také nastavení, ke kterému se záloha vztahuje. Následně, udržovaní představy o tom, jak se toto nastavení časem měnilo, může být také výhodnou a zajímavou vlastností.

Vytvořené zálohy by měly být uloženy v jednom úložišti přístupném uživateli. Přistup k obsahu tohoto úložiště, by měl být dostupný přes nástroj umožňující zobrazení jednotlivých záloh. Další možností jak prohlížet uložené soubory, je porovnání verzí záloh mezi sebou. Zobrazení rozdílu (ať už aktuálního souboru a některé verze, případně některých verzí záloh mezi sebou) umožňuje lepší orientaci v nastavení systému.

Další důležitou vlastností je pro uživatele možnost obnovení nastavení do předchozího stavu. Prohlížení dostupných záloh umožní získat představu, která z dostupných verzí záloh je vhodná pro obnovu. Pomocí jedinečného označení by pak uživatel mohl obnovil vybranou verzi konfigurace. Tato obnova by se provedla do konfigurace a tím by se uvedly v platnost hodnoty nastavené v době pořízení zálohy. Soubory přepsané obnovou zálohy by bylo možné obnovit zpět. Pokud by obnova zálohy nesplnila očekávaný výsledek. Případně pokud by obnova neproběhla podle představ uživatele.

Uživatel by měl mít možnost prohlížet zálohy i z dalších umístění, nejen z lokálního počítače. Nástroj prohlížení by umožňoval zobrazit rozdíly mezi jednotlivými umístěními. Tato umístění by měla být jasně oddělena, ale zároveň by umístění mělo být stejné, jako umístění lokálních záloh. Všechny tyto zálohy by tedy měly být umístěny v jednom úložišti. Tato možnost by měla být i v síti o několika počítačích, u kterých lze předpokládat podobné, nebo dokonce stejné nastavení. Prohlížení by bylo možné provádět podobně jako je tomu na úložišti pro lokální počítač. Při obnově verze zálohy ze vzdáleného počítače by tato obnova nebyla provedena do lokální konfigurace. Obnova by se provedla pouze v rámci úložiště do části vyhrazené odpovídajícímu vzdálenému umístění. Toto úložiště pak bude možné přenést do dalšího umístění a obnovit konfiguraci v dalším umístění. To by bylo možné, neboť soubory označené jako aktuální, by byly jednoduše odděleny od záloh.

2.3 Funkční požadavky

Aplikace splňuje následující funkční požadavky vycházející z důvodů popisovaných výše. Některé požadavky vycházejí zároveň z předpokladů, běžných u takovýchto aplikací.

2.3.1 Povinné požadavky

Tato sekce obsahuje požadavky na aplikaci, kterých má být dosaženo. Požadavky jsou rozděleny do funkčních okruhů, které odpovídají činnosti nástrojů navrhované aplikace.

Zálohování:

- vytvářet zálohy konfiguračních souborů
- bude uchovávat rozdíly mezi jednotlivými zálohami
- tyto zálohy budou uchovávat také údaj:
  - o datu pořízení
  - o verzi a unikátní označení
- k těmto zálohám bude možné přiřadit zprávu zadanou uživatelem
- záloha bude obsahovat zprávu, kterým nástrojem byla vytvořena (z této zprávy lze určit po jaké události tato záloha vznikla)

Nastavení:

- dodávat další nastavení pomocí parametrů dodaných při spuštění z příkazové řádky
- upřesnit různými způsoby v konfiguraci, které soubory zahrnout do zálohy:
  - jednotlivé soubory
  - celé adresáře
  - zjednodušených regulárních výrazů[3]
- nastavit okruh zálohovaných souborů pomocí záznamu v konfiguraci[4]
- upřesnit, které soubory nemají být zahrnuty do žádné zálohy
- kontrolovat syntaxi konfiguračního souboru (tím je myšlena správnost klíčových slov)
- kontrolovat, jestli soubory uvedené v konfiguraci existují (zahrnuje i kontrolu existence externího nástroje)
- při chybě v syntaxi by měl nástroj tuto chybu vypsat a skončit
  - upřesnit nastavení pro každý záznam
  - nastavit při jaké události se bude konkrétní záznam vytvářet
  - upřesnit, které soubory se nemají zahrnout do zálohy (zde jde o upřesnění pro záznam)
- kontrola by zároveň měla umožňovat neuvést parametr do konfigurace
- aplikace by měla umožnit pracovat se symbolickými odkazy následovně:
  - ukládat existenci symbolického odkazu
  - zálohovat cíl odkazu
  - umožnit vypnout sledování symbolických odkazů (na úrovni celé konfigurace a zároveň pro každý specifický záznam)

[3] Obdobné expanzním výrazům shellu zahrnujícím symboly „?,∗,[],{}“.
[4] Záznamy z konfigurace jsou vysvětleny dále v textu (viz. 3.1)

Chybový výstup:

- chybový výstup do souboru
- nastavit pro tento výstup, který soubor má být použit
- chybový výstup do příkazové řádky
- nastavit tento výstup na různé úrovně výřečnosti
- úplně vypnout chybový výstup do příkazové řádky

Prohlížení záloh:

- umožnit zobrazení zprávy k odpovídající záloze
- rozlišit, ke kterému záznamu z konfigurace soubor náleží
- zobrazit všechny:
  - verze záloh obsažené v úložišti
  - soubory umístěné v úložišti
  - záznamy z konfigurace
  - úložiště pro klienty
  - verze vybraného souboru
- zobrazit obsah konkrétní verze souboru
- zobrazit rozdíl mezi dvěma verzemi souboru
- vytvořit sloučení aktuální a zálohované verze souboru
- umožnit porovnat konfiguraci mezi různými klienty

Obnova záloh:

- obnovit konkrétní zálohu pomocí nástroje jehož vstupem bude:
  - jedinečný identifikátor
  - jméno záznamu v konfiguraci
- uchovat soubory přepsané zálohou v přejmenovaných souborech

Ostatní:

- způsob, jakým je aplikace navržena, by měl umožnit pozdější rozšíření o jiné systémy úložiště

2.3.2 Nepokryté požadavky

V následujícím seznamu jsou uvedeny požadavky, které tato aplikace neřeší. Některé požadavky uvedené v tomto seznamu mohou být možností jak projekt rozšířit v dalších verzích.

- nastavovat chybový výstup pomocí konfiguračního souboru (aplikace potřebuje chybový výstup už během čtení konfiguračního souboru)
- sledovat změny v konfiguraci během spuštění
- zavádět nastavení na klientech a toto nastavení přenášet
- provádět automatický merge souborů (jde o konfigurační soubory kde by tento merge mohl způsobit problémy)
- ignorovat soubory vytvořené po obnově zálohy, toto nastavení je ponecháno na uživateli
- umožnit prohlížení zálohy z klientů pomocí upřesnění záznamů z konfigurace
- sledovat oprávnění uživatele pro vstup do adresářů které nástroje potřebují pro svou funkci
- přenášet zálohy na další počítač (v dalším textu je uvedeno jakým způsobem je možné tyto přenosy řešit
- sledovat větvení verzí záloh a strukturu rodičů a jlistů
- umožňovat neautorizovanému uživateli zápis do adresářů pro která nemá oprávnění (tedy i adresář se zálohami)
- nesleduje přejmenování souborů, případně další operace, které poskytuje systém správy verzí pro tyto operace
- nebude umožňovat zobrazení rozdílů pro celé adresáře (některé nástroje pro prohlížení rozdílů by mohly způsobit problémy s touto možností)
- automaticky provádět aktualizaci archivního adresáře
- nebude mazat soubory, které podle obnovy nemají být v konfiguračním souboru (toto mazání konfigurace by mohlo být rizikové)
- nebude umožňovat zálohovat další repositář systému správy verzí
- nebude řešit pokud v případě příliš intenzivních zápisů do konfiguračního adresáře nebude docházet k zálohování (V případě tak intenzivních zápisů se může jednat o vážný problém v OS.)

2.4 Požadavky na systém

Zde je uvedeno jaké má aplikace nároky na systém, na kterém bude používána. Tyto požadavky jsou rozděleny na dvě části. V první je uvedeno jaké softwarové balíky[5] aplikace vyžaduje, aby ji bylo možné používat. V části druhé je pak uvedeno jaké jsou hardwarové nároky na systém.

[5] V OS GNU/Linux je software šířen v tzv. balících. Podrobnější vysvětlení se dá nalézt v 3.1.15.

2.4.1 Software

Balíky softwaru požadovaných aplikací jsou:

- Jádro OS GNU/Linux verzi 2.6.13-rc3 a pozdější. Tato verze jádra obsahuje rozšíření o podsystém inotify. Toto rozšíření je nutné pro sledování změn v souborovém systému.
- dev-lang/python nejméně verze 2.3, požadavkem je, aby rozšíření pyinotify bylo schopné fungovat. Jedná se o hlavní interpret programu, v kterém je aplikace naprogramována.
- dev-python/pyinotify je modul jazyka Python umožňující ovládat sledování změn v souborovém systému.
- dev-util/mercurial verze 0.9.5 a vyšší. Mercurial je distribuovaný systém zprávy verzí a je použit jako úložiště pro zálohy souborů.
- některý ze systémů cron například sys-process/vixie-cron jedná se o službu systému, která umožňuje spouštět další úlohy ve zvolený čas.
- některý z nástrojů umožňující prohlížení rozdílů souborů a sloučení těchto rozdílů. Některé z dostupných nástrojů jsou popsány v části 4.4.

2.4.2 Hardware

Hardwarové nároky lze odvodit z požadavku, aby na počítači běžel OS GNU/Linux. Ten je možné provozovat na různých procesorových architekturách. K nejvíce rozšířené procesorové architektuře patří patří x86. Nároky pro tuto architekturu jsou uvedeny jako následující:

Procesor Minimum: Procesor třídy Pentium

Pevný disk Pro 32-bit x86 systémy:
Minimální instalace: 620MB
Pro 64-bit x86_64 systémy:
Minimální instalace: 900MB

Paměť Pro 32-bit x86 systémy:
Minimum pro textový-mód: 64MB
Pro 64-bit x86_64 systémy:
Minimum pro textový-mód: 128MB

Tyto údaje pocházejí ze stránek projektu České distribuce Fedora [17]. Jako další požadavek je možné označit nutnost síťového připojení, pokud se budou přenášet zálohované konfigurace na klienta. Nicméně, tato funkčnost není hlavním zaměřením této aplikace.

2.5 Současný stav

V této kapitole se pokusím popsat další řešení problému zálohování. Tyto aplikace jsou určeny pro zjednodušení práce při nastavování OS. Většinou se jedná o OS, projektované jako servery pro velké zatížení ze strany uživatelů.

2.5.1 Microsoft stínová kopie

Řešení od společnosti Microsoft je zaměřeno na zálohování souborů. Služba stínové kopie (anglicky Shadow Copy) je využívána k duplikování souborů na jednom disku. Tato služba stínové kopie vytvoří záznam, nebo-li snapshot souboru v předem zvolený časový interval. Záznamy jsou dostupné ve vlastnostech souboru, jako v předchozí verzi. Stínová kopie prezentuje list duplikovaných souborů, které byly zaznamenány touto službou. Stínová kopie provádí stínování na celém disku a není tedy možné specifikovat konkrétní adresář.

OS Microsoft Windows Vista je stínová kopie provázána s obnovou systému, která umožňuje uživateli vrátit zpět všechny instalace softwaru na počítači do předchozího stavu. Obnovení systému je zaměřeno na opravu problémů způsobených instalací softwaru, aktualizace, ovladače, nebo nějaké další události, která snížila funkčnost OS. Obnova systému provede vrácení stavu do některého z předem pořízených záznamů. Při obnově, je všechno co následuje po obnoveném stavu, smazáno [25].

2.5.2 IBM WebSphere rozšířené nasazení

Tento aplikační server (anglicky WebSphere Extended Deployment) obsahuje rozšíření, pojmenované rozšířená služba repositáře (anglicky Extended Repository Service). Tato služba umožňuje pomocí takzvaných kontrol (anglicky Checkpoint) ukládat nastavení. Pomocí těchto kontrol je možné obnovit stav, v kterém byly pořízeny.

V kontrolách se nachází celé nastavení systému, nebo pouze změny oproti předchozí kontrole. Plné zálohy jsou prováděny uživatelem, zatímco uložení změn probíhá automaticky. Zálohy lze později obnovovat. Nástroj použitý pro obnovování kontrol zároveň umožňuje obnovu krok za krokem, tak jak byly kontroly pořizovány. Tato funkce funguje podobně, jako funkce zpět v textovém editoru [12].

2.5.3 Oracle správce přístupu

Společnost Oracle poskytuje v rámci své aplikace na správu konfigurací, která je součástí softwaru Oracle správce přístupu (anglicky Oracle Access Manager). Aplikace je pojmenována Oracle správce přístupu konfigurační správce (anglicky Oracle Access Manager Configuration Manager). Aplikace slouží ke správě a migraci konfigurací pro některé další produkty společnosti Oracle.

Existuje zde možnost pro zálohování před provedením migrace nastavení a jeho případného obnovení. Oracle správce přístupu konfigurační správce obsahuje funkci, která ukládá záložní kopii celé konfigurace. Tato záloha je označena jako snapshot celého konfiguračního stromu, do kterého Oracle správce přístupu konfiguraci ukládá.

Oracle doporučuje používat tuto funkci před importem další konfigurace, případně před provedením migrace na jinde vytvořenou konfiguraci. Aplikace poskytuje uživatelské nástroje pro prohlížení, tvorbu, rušení a obnovu těchto snapshotů [20].

2.5.4 RollBack Rx

Tento nástroj umožňuje provádět a obnovovat zálohy. Nástroj je vytvořen pro OS Microsoft Windows 98/Me/2000/XP/Vista. Je určen především pro domácí uživatele a IT profesionály. Dále umožňuje jednoduše obnovovat stav počítače do času, v kterém byla záloha pořízena. Funkce, které tento nástroj poskytuje, jsou obnova po zhroucení systému (pokud OS Microsoft Windows není možné nastartovat), návrat k předchozímu snapshotu celého systému, návrat do budoucího snapshotu, automatické vytváření snapshotů apod.

Jednou z funkcí, kterou tento nástroj poskytuje, je vytváření záloh konkrétního adresáře. Tímto způsobem je možné zálohovat adresář, v kterém se konfigurace na OS Microsoft Windows nachází. Za tento adresář je možné považovat `C:\Windows` a další podadresáře [1].

Kapitola 3

Návrh řešení

Cílem této kapitoly je vysvětlit, jakým způsobem lze dosáhnout požadované funkčnosti navrhovaného systému. Postupně budou vysvětleny hlavní programové části, které obstarají uložení a zobrazení snapshotů, které byly již vytvořeny. Na závěr jejich obnovení zpět do stavu, v kterém byly pořízeny. V první části budou vysvětleny některé pojmy, které budu dále v textu používat.

3.1 Vysvětlení používaných pojmů

V této části vysvětluji význam pojmů používaných v dalším textu. Popsáním těchto výrazů bych chtěl usnadnit chápání dalšího textu. Pojmy zde uvedené vycházejí většinou z dalších zdrojů, které jsou uvedeny v seznamu na konci dokumentu. Pojmy jsou abecedně seřazeny pro snadnější vyhledávání.

3.1.1 Archivní adresář

Archivní adresář je umístěn na cestě `/var/db/snaproll`. Zde budou pevné odkazy na sledované soubory konfiguračního adresáře, které budou určené jako obsah snapshotů. Pevné odkazy na soubory z konfiguračního adresáře umístěny v tomto adresáři spolu s repositáři. Tímto způsobem je dosaženo toho, že konfigurační adresář neobsahuje přebývající soubory, tedy repositáře.

3.1.2 Číslo identifikující proces

Číslo které umožňuje identifikovat konkrétní proces. Číslo je přiřazeno procesu automaticky při jeho vytvoření v OS. Proces je běžící instance programu. Tyto instance mají každá přiřazen jedinečný identifikátor, kterým je hodnota nezáporných čísel datového typu int. Pomocí tohoto identifikátoru je možné komunikovat s tímto procesem pomocí signálů OS [10].

3.1.3 Datový typ int

Tento datový typ obsahuje číslo patřící do množiny celých čísel, která se skládájí z přirozených čísel (1, 2, 3, ...), nuly a záporných čísel (-1, -2, -3, ...). Tato množina se v literatuře většinou označuje $\mathbb{Z}$ [30].

3.1.4 Datový typ seznam

Tento datový typ je tzv. složený datové typ, které slouží k uložení různých hodnot. Typ se skládá ze sekvence jednotlivých položek. Seznamy podporují mnoho operací. Například spojování, indexo-

vání, operace s podsekvencemi a podobně [36]. V této aplikaci jsou seznamy využity k předávání názvů souborů a jejich adresářových cest. Kdy je potřeba aplikovat nějakou operaci na všechny položky v tomto seznamu.

3.1.5 Datový typ slovník

Datový typ slovník je rozšířením klasického datového typu list. U tohoto datového typu je možné zapisovat hodnoty k přiřazenému klíči. Hodnoty jsou pak dostupné přes tento klíč. Používání je podobné, jako u klasického datového typu list, u kterého jsou hodnoty přístupné pomocí indexu hodnoty. U datového typu slovník jsou tyto hodnoty přístupné přes klíčové slovo. Klíčová slova je možné i vyhledávat a jinak používat k přístupu k hodnotám.

3.1.6 Démon, Služba

Tímto pojmem se označuje proces, který neustále běží a průběžne vykonává nějakou činnost bez zásahu uživatele. Tento proces není za běžných okolností ukončován ani přímo uživatelem spouštěn [28]. V tomto textu používám i výraz služba. Službou je míněna spíše přímá činnost démona.

3.1.7 Glob vzor

Řetězec, který může být porovnán s dalšími textovými řetězci. Pravidla pro tyto řetězce jsou zjednodušením regulárního výrazu. Řetězce jsou specifikovány pomocí „$*$“, „?“, „[a-z]“ a další. Tento typ výrazu je rozvíjen v shellu na OS Unixového typu pro filtrování souborů při prohlížení souborového systému [29]. Další informace lze získat z manuálové stránky `glob(7)`.

3.1.8 Konfigurační adresář

Konfiguračním adresářem je označován `/etc` a všechny jeho podadresáře. V tomto adresáři se nachází většina nastavení celého systému. V tomto adresáři bude umístěno i nastavení navrhované aplikace.

3.1.9 Live CD distribuce

OS uložený na CD, jenž z něj může být spuštěn bez nutnosti jeho instalace do pevné paměti, jakou je např. pevný disk. Systém se znovu vrátí ke svému původnímu OS když je Live CD vyjmuto z mechaniky a počítač je restartován. Toho je možné díky neukládání dat na pevný disk, ale jejich uložení do dočasné paměti, jakou je např. paměť RAM. Takovéto řešení však klade vysoké nároky na paměť počítače [31]. Spuštěním OS z tohoto CD výrazně snižuje výkon celého systému.

3.1.10 Pevný odkaz

Odkaz na fyzická data na úložném zařízení je označen jako pevný odkaz (anglicky hardlink). Na většině operačních systémech lze všechny pojmenované soubory považovat za pevné odkazy. Jméno asociované se souborem lze považovat za jmenovku, která odkazuje operační systém na konkrétní data. Těchto jmenovek může být víc pro jedna data. Přestože k těmto datům může být přistupováno přes různé jmenovky, vždy se změna projeví na aktuálních datech, nezávisle na tom jak bude k datům přistupováno později. Pevné odkazy mohou ukazovat pouze na data, která existují na stejném souborovém systému . Při zrušení odkazu se přeruší asociace na jméno a konkrétní data. Data jsou dostupná dokud existuje alespoň jeden pevný odkaz na tyto data. Pokud je i tento pevný odkaz odstraněn, jsou odstraněna i data [3].

3.1.11 Regulární výraz

Tento výraz je řetězcem popisující celou množinu řetězců, konkrétně regulární jazyk. V programu je využit pro porovnávání textových řetězců, konkrétně hlavně jako kontrola textových vstupů z konfigurace [32].

3.1.12 Repositář, revize

Úložiště, v kterém budou snapshoty uloženy, se v SCM se označuje repositář. Každý záznam v repositáři se nazývá revize. Pro ušetření prostoru na disku se ukládá vždy první verze revize. Při ukládání každé další revize se uchovává pouze rozdíl oproti předchozí verzi revize.

Jednou z dalších možností, jak ukládat revize je uložit vždy celý obsah znovu. Způsob uložení dat v repositáři se liší podle použitého SCM.

3.1.13 Rollback

Obnovení předchozí zálohy, v tomto případě snapshotu, pořízené v předchozí době. Touto obnovou se dá docílit obnovení obsahu souborů, nebo adresářů do podoby, v které byl snapshot pořízen.

Jednou z možností obnovení je sloučení snapshotu s dalším souborem. Pokud vybraný snapshot vyhovuje pouze částečně, je možné ho sloučit vybraným souborem. Sloučení se provede pomocí zobrazení rozdílů mezi vybranými verzemi souboru a nabídkou z který řádek případně odstavec použít. Vybrané řádky a odstavce následně budou v obsahu výstupního souboru.

3.1.14 Snapshot

Soubor, soubory, nebo adresář a jejich struktura zálohovaná pro pozdější použití je označován jako snapshot. Tento snapshot je možné použít k obnovení obsahu souborů, nebo adresáře do časového okamžiku, v kterém byl uložen [33].

Vytváření snapshotu může být prováděno jako reakce na událost. To znamená periodicky jednou za určitý čas, nebo po událost zápisu do souboru Tímto způsobem je možné dosáhnout průběžné řady záloh konfiguračních souborů v průběhu času.

3.1.15 Softwarový balík

V OS GNU/Linux je software běžně šířen pomocí takzvaných softwarových balíků. Tyto balíky bývají zkomprimované archivy, obsahující všechny soubory spolu s informacemi, které uvádějí jak tento balík nainstalovat, odinstalovat. Dále pak jsou zde informace o tom, jaké akce se mají odehrát před instalací, případně po instalci tohoto balíku. Jako textové informace přiřazené k tomuto balíku jsou domovské stránky balíčku, software, který tento balíček vyžaduje, aby mohl být spuštěn.

V distribuci GNU/Linuxu Fedora jsou tyto balíky označeny příponou `.rpm`. Přípona vychází z názvu celého systému těchto balíčků. Název vychází z anglického spojení „Resource Package Management“. Toto spojení nemá v češtině ustálený výraz. Je možné jej přeložit do češtiny, jako balíčková správa zdrojů.

3.1.16 Souborový systém

Souborový systém (anglicky filesystem) je označení pro způsob organizace informací (ve formě souborů) tak, aby bylo možné je snadno najít a přistupovat k nim. Souborové systémy mohou používat paměťová média jako pevný disk, nebo CD, mohou poskytovat přístup k datům uloženým na

serveru. Souborový systém umožňuje ukládat data do souborů, které jsou označeny názvy. Obvykle umožňuje vytvářet adresáře, pomocí kterých lze soubory organizovat do stromové struktury [34].

3.1.17 Symbolický odkaz

Odkaz na další soubory, nebo adresáře. Tyto odkazy jsou samy o sobě soubory bez obsahu. Pokud je soubor, na který symbolický odkaz ukazuje smazán, symbolický odkaz zůstává, ale ukazuje do neexistujícího umístění. Při dotazu pozůstalý odkaz vypíše, na který soubor ukazoval.

3.1.18 Systém pro správu verzí

Posloupnost snapshotů je uložena v systému pro správu verzí. Tento systém pro správu verzí je v anglické literatuře označován jako Source code management. Z tohoto názvu je odvozena zkratka SCM, která se používá v literatuře české. Dále v textu bude použita tato zkratka.

3.1.19 Událost

Část dat, která poskytují informaci o jednom, nebo více systémových prostředcích, se nazývá událost [3]. V případě navrhované aplikace jsou těmito prostředky systémové hodiny a souborový systém. Sledovanou událostí na souborovém systému je zápis do souboru. Tyto události jsou sledovány pomocí služeb `cron(8)` a podsystému jádra `inotify(7)`. Služba `cron(8)` spouští další příkazy v předem definovaný čas. `Inotify(7)` sleduje události přístupů k přednastaveným souborům.

3.1.20 Uživatel

Vzhledem k zaměření tohoto projektu na konfigurační soubory se předpokládá, že uživatelem je považován správce OS. V případě běžného uživatele vznikne problém s oprávněním pro přístup do adresářů, v kterých tyto nástroje pracují. Tento projekt si neklade za cíl řešit tyto oprávnění.

3.1.21 Vlákno

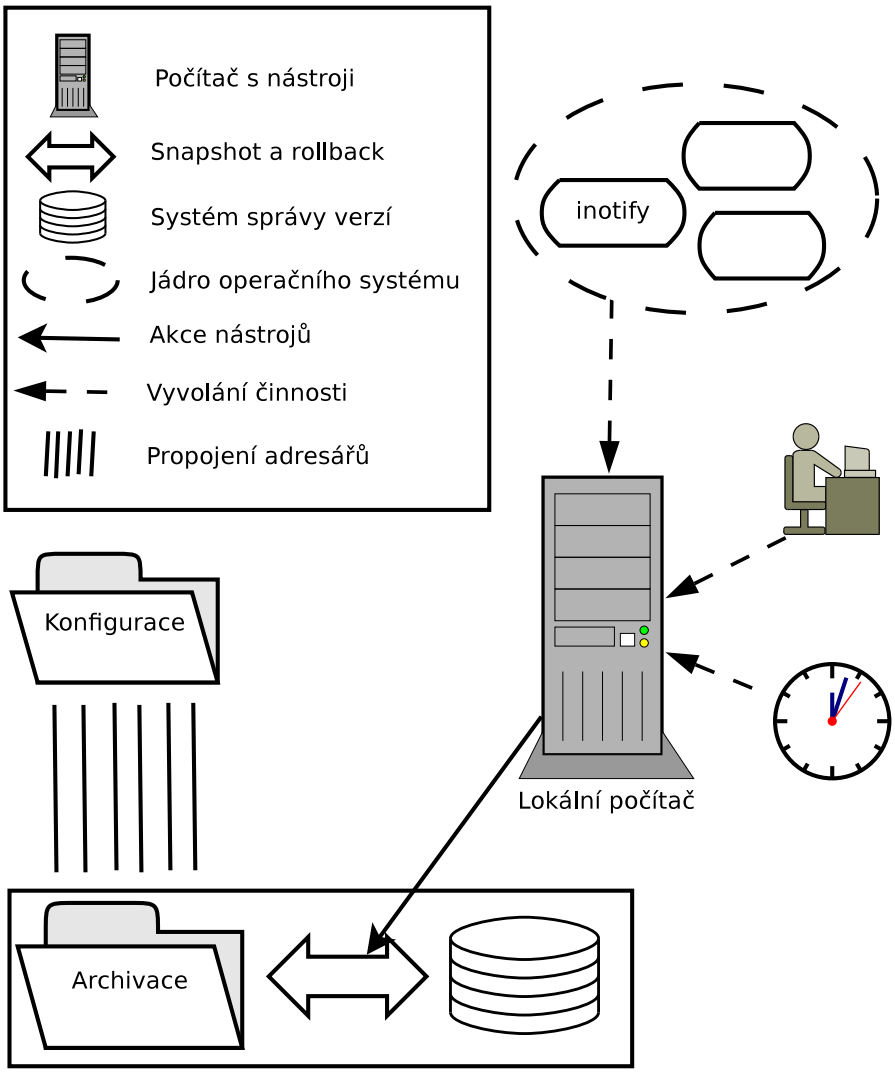
Vlákna jsou prostředkem známým z mnoha operačních systémů. Lze je chápat jako samostatné procesy. Jednotlivá vlákna běží nezávisle na sobě, přičemž ale sdílí společný adresový prostor. Změna jedné proměnné v jednom vlákně se ihned promítne do ostatních vláken. Vlákna tudíž potřebují určité prostředky pro synchronizaci. Pomocí těchto prostředků lze zajistit, že k určitým prostředkům (proměnným, souborům atd.) bude mít přístup vždy pouze jedno vlákno [35].

3.1.22 Záznam z konfigurace

Takto označuji okruh souborů v označených v konfiguračním souboru. Z těchto souborů jsou poté vytvářeny samotné snapshoty. Jedná se tedy o okruh souborů, které jsou určeny pro zálohování.

3.2 Architektura aplikace

V této části se pokusím nastínit architekturu navržené aplikace. Následující diagram 3.1 má ukázat vstupy a výstupy, spolu s hlavními částmi této aplikace. Diagram by měl napomoci vysvětlit hlavní děje odehrávající se během používání nástrojů.



Obrázek 3.1: Diagram architektury systému

V diagramu je naznačeno hlavní pole působnosti aplikace. Tím bude lokální počítač, na kterém se budou používat nástroje snapshotu, popsaném v části 3.1.14. Jedním z dalších nástrojů je rollback, jehož popis se nachází v části 3.1.13.

Tyto nástroje představují hlavní funkčnost celé aplikace, která zahrnuje vytváření a obnovování záloh konfiguračních souborů. Konfigurační soubory jsou standardně v OS GNU/Linux umístěny v adresáři `/etc`. Adresář je na diagramu 3.1 označen složkou s textem "Konfigurace". V této složce se nachází větší, či menší počet textových souborů. Nastavení je obsaženo v souborech textovou formou, ta se může mezi aplikacemi značně lišit.

Další složkou na obrázku je složka archivní. V ní jsou umístěny soubory určené pro zálohování a zároveň i repositář, v kterém se zálohy nacházejí. Soubory určené pro zálohování jsou v této složce, protože byly vytvořeny pomocí nástroje, který provádí propojení těchto adresářů. Soubory, které se mají propojit do archivního adresáře jsou vybrány na základě nastavení provedené uživatelem. Tento nástroj je popsán dále v části 3.7. Propojení, naznačené na obrázku řadou svislých čar, jsou ve skutečnosti pevné odkazy. Tyto odkazy umožňují souborům se vyskytovat v obou adresářích, bez

potřeby dalšího prostoru na souborovém systému. Princip pevných odkazů je popsán v části 3.1.10.

V archivním adresáři je dále umístěno úložiště, do kterého budou umisťovány zálohy souborů tj. snapshoty. Zároveň při obnově snapshotů jsou soubory obnovovány z repositáře do archivního adresáře a z něj poté do konfiguračního adresáře. Toto úložiště je tvořeno repositářem systému pro zprávu verzí. Podrobnější popis se nachází v jedné z dalších částí 3.12. Tento systém umožňuje uchovávat rozdíly mezi jednotlivými zálohami. Soubory úložiště je třeba umístit do stejného umístění, v kterém se nacházejí zálohované soubory. Z tohoto důvodu jsou soubory a repositář ve stejném adresáři. Tato skutečnost opodstatňuje existenci archivního adresáře.

Zálohování se provádí po externí události, případně po spuštění uživatelem. Pokud je zálohování spuštěno uživatelem, je provedeno nástrojem pro vytvoření snapshotů 3.2. Toto vytváření snapshotů jsem naznačil jedním koncem oboustranné šipky.

Události, při kterých se zálohování provádí, jsou změny v souborovém systému. Tyto události jsou sledovány pomocí podsystému jádra OS. Jádro OS je naznačeno přerušovaným oválem. Jednou z podčástí tohoto oválu je `inotify(7)`, což je podsystém jádra, který provádí zmiňované sledování. Zálohování při této události je popsáno více v části 3.11. Další externí událostí, při které je prováděno zálohováni, je vypršení časového intervalu. Událost po čase zprostředkovává služba `cron(8)`, která v přednastavený čas spustí požadované zálohování. Použití této služby je popsáno v části 3.8.

Druhým koncem oboustranné šipky je naznačen rollback 3.1.13. Tento nástroj umožňuje obnovu předchozích záloh do stavu, v jakém byly pořízeny. To znamená zpět do konfiguračního adresáře, s obsahem odpovídajícím obsahu během vytváření snapshotu.

Jedním z nástrojů, které jsou dostupné uživateli, je také přístup k repositáři a revizím, které jsou uložené v tomto úložišti. Tento nástroj umožní uživateli zobrazit seznam záloh. Případně zobrazit obsah některé z vybraných záloh, nebo rozdíly mezi obsahy ve vybraných revizích. Tímto nástrojem by uživatel měl přístup k samotnému úložišti a zálohám, které jsou v něm uloženy. Prohlížení těchto záloh umožňuje přístup do dalších repositářů umístěných v úložišti. Tyto další repositáře obsahující data podobná obsahu z jiných počítačů, umožňují pomocí nástroje prohlížení zobrazit rozdíly mezi lokálními konfiguračními soubory, případně mezi dalšími repositáři. Nástroj, umožňující prohlížení revizí, je popsán v jedné z dalších částí 3.14.

3.3 Návrh nástrojů umožňujících vytvoření a uložení snapshotu

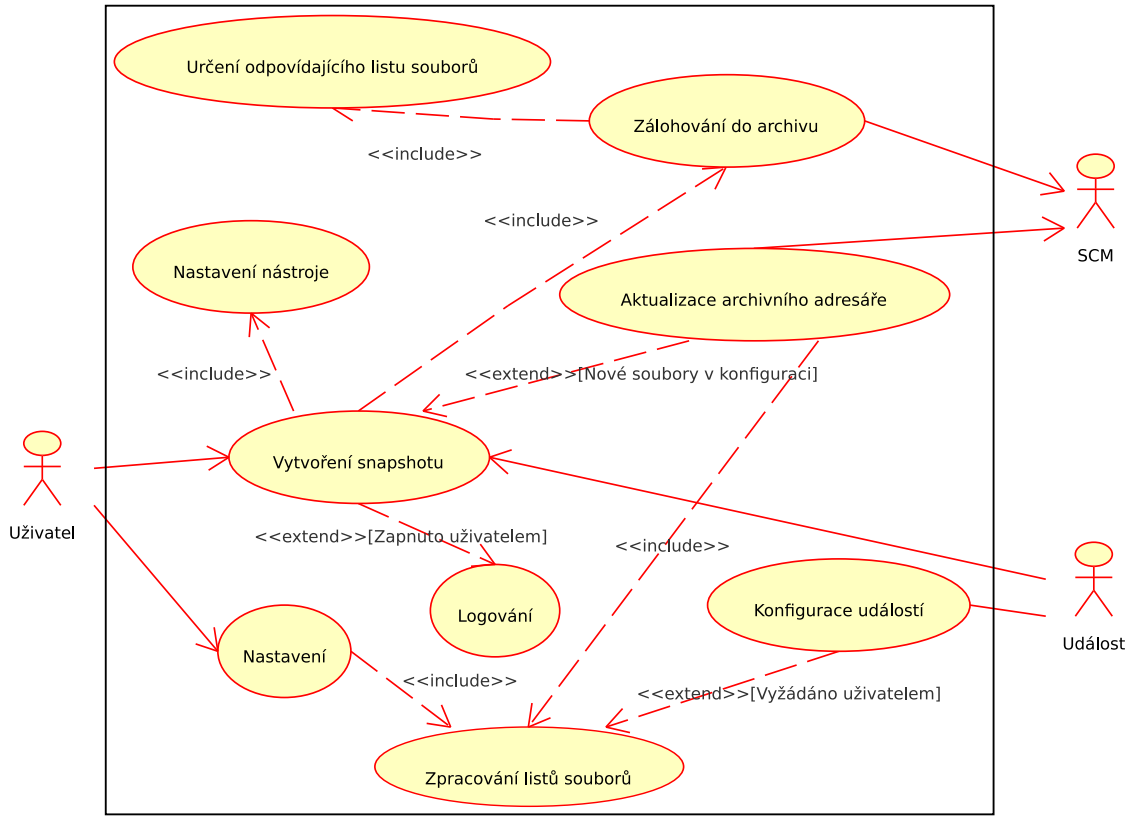
V této sekci je popsán návrh, jakým způsobem bude provedeno vytvoření prostředí, které umožní další funkčnost nástrojů. Vytvoření snapshotu a jeho uložení do úložiště. Grafický souhrn v diagramu užití je zobrazen na obrázku 3.2, který popisuje tento nástroj a jeho plánované způsoby užití.

V diagramu je naznačeno, které hlavní činnosti se budou odehrávat během vytvoření snapshotu. Jednou z hlavních činností je vytvoření prostředí, v kterém bude možné tyto snapshoty vytvářet. Prostředí se sestává ze dvou hlavních adresářů, jak už je naznačeno v diagramu 3.1. Tyto dva adresáře je možné mezi sebou aktualizovat pomocí nástroje, který bude obstarávat tuto činnost. Návrh nástroje je v sekci 3.7.

Tento nástroj, ale i všechny ostatní potřebují pro svou činnost nastavení uživatelem. Nastavení se provádí konfiguračním souborem `/etc/snaproll.conf`, případně vstupními parametry, dodanými během spouštění nástroje. Tato činnost je popsána v části 3.4.

Jednou z částí aplikace používanou všemi ostatními nástroji je chybový výstup nástrojů, to znamená logování. Tento výstup umožňuje nástrojům vypisovat oznámení o své činnosti. Chybový výstup je propojen se vstupními parametry nástrojů, tento výstup popsán v části 3.5.

Čtení tohoto nastavení, dále pak i samotné vytváření struktury archivního adresáře, případně určení odpovídajících seznamů souborů, pro které nastavení konkrétních souborů odpovídá, se pro-



Obrázek 3.2: Diagram případů užití nástroje pro vytvoření a uložení snapshotu

vádí za pomoci základní vrstvy. Tato vrstva je využívána všemi ostatními nástroji pro přístup k nastavení a případně k vytváření struktury a souborů adresářů. Vrstva je popsána v části 3.9.

Hlavní činností je vytváření snapshotů. Toto vytváření snapshotů je možné vyvolat různými spouštěči, jak je naznačeno v předchozím diagramu 3.1. Všechny spouštěče vytváření snapshotů používají jeden nástroj pro jeho vytvoření. Rozdílný je způsob, jakým je tento nástroj uveden do činnosti. Uvedení do činnosti pomocí spuštění uživatelem je provedeno z příkazové řádky, spuštěním nástroje s odpovídajícími vstupnímy parametry. Tento nástroj je rozepsán v sekci 3.10.

Dalším způsobem je spuštění pomocí služby `cron(8)`. Služba se nastavuje pomocí nástroje pro aktualizaci struktury adresářů. V jedné z podčástí návrhu tohoto nástroje 3.8.

Posledním způsobem je vytváření snapshotů po změně v sledovaných souborech. Sledování se provádí pomocí podsystému jádra, které takové sledování umožňuje. Sledování je provedeno přiřazením odpovídající akce k události, která je sledována. V tomto případě se jedná o akci zápisu do souboru, případně jeho smazání. Přístup k tomuto podsystému jádra zprostředkovává služba běžící na pozadí systému. Návrh této služby se nachází dále v textu v části 3.11.

3.4 Nastavení nástrojů

Nastavení je provedeno dvěma způsoby, konfiguračním souborem a parametry, které se připojí při spouštění za příkaz. Nastavení provedené oběma způsoby se v některých parametrech liší. To je

dáno především dostupností během spuštění nástroje a samozřejmě velikostí obsahu, který poskytuje každý z konfiguračních vstupů.

3.4.1 Parametry za příkazem

Vzhledem k dostupnosti těchto parametrů, hned po spuštění nástroje, jsou tyto parametry dobře použitelné pro nastavení chybového výstupu, to znamená logovacího souboru.

Každý nástroj má své specifické vstupní parametry, pomocí kterých uživatel určuje, jakým způsobem spouštěný nástroj poběží. Zde jsou uvedeny spouštěcí parametry, které jsou společné pro všechny nástroje. Jedná se především o nastavení chybového výstupu, který musí být nastaven už v době zpracovávání souboru s nastavením.

–logfile *soubor* Tento parametr nastavuje, do kterého souboru bude směrován chybový výstup spuštěného nástroje.

–verbosity <*debug*|*info*|*error*> Úroveň výřečnosti do příkazové řádky se nastavuje tímto parametrem. Jsou možné tři úrovně :

debug Při nastavení této úrovně jsou zobrazovány veškeré textové výstupy.

info Tato úroveň je standardně přednastavena. Zobrazuje všechna varování a chyby, které se udály za běhu programu.

error Potlačení všech, až na chybové výstupy, je možné nastavit touto úrovní.

–quiet Tímto parametrem lze kompletně vypnout výstup do příkazové řádky. Chybový výstup je stále směrován do logovacího souboru.

–conffile *soubor* Pomocí tohoto vstupu lze upřesnit jiný, než standardní konfigurační soubor.

–help Přes tento parametr je dostupný výpis s pomocným výpisem.

3.4.2 Konfigurační soubor

V konfiguračním souboru jsou uvedeny především jednotlivé záznamy, označující okruhy souborů, z kterých jsou poté vytvářeny snapshoty. Záznamy jsou uvedeny „názvem“ záznamu, který obsahuje názvy souborů, případně část cesty v konfiguračním adresáři. Přes tyto „názvy“ je možné odkázat se na konkrétní okruh souborů. V dalším textu označuji tyto okruhy jako záznamy v konfiguraci. Soubory je možné do těchto okruhů přidat třemi následujícími způsoby:

- Konkrétní soubor(y). Například `hosts sudoers`. V tomto případě tento záznam uvádí dva soubory.
- List souborů uvedený pomocí glob syntaxe[1] `adresar/*.conf`. Tímto způsobem je možné specifikovat například všechny soubory v adresáři. V tomto případě všechny soubory v adresáři `adresar/` s příponou `.conf`.
- Poslední možností je uvést celý adresář `acpi/`. Tímto způsobem jsou přidány všechny soubory v adresáři a dále všechny soubory v podadresářích.

[1] Vice je popsáno v části 3.1.7.

Výše uvedené zápisy je možné mezi sebou kombinovat v rámci jednoho záznamu. Jednotlivé zápisy jsou přidávány do seznamu souborů, které budou později obsaženy ve výsledném snapshotu. Příklad, jak tento záznam může vypadat, je uveden ve výpisu 3.1.

```
adresar/ exports {
    ontime */10 * * * *
    followsymlinks yes
    ignorelist *
    }
```

Příklad 3.1: Záznam v konfiguračním souboru.

Další řádky za názvem záznamu, umístěné ve složených závorkách, jsou nastavení, vztahující se ke konkrétnímu záznamu. V těchto nastaveních je možné specifikovat, jak se bude záznam ukládat, při které události, případně další vlastnosti.

3.4.3 Obecné nastavení

Příznaky uvedené v konfiguračním souboru, mimo konkrétní záznam, určují obecné nastavení. Obecně tyto příznaky ovlivňují nastavení nástrojů pro všechny záznamy konfigurace. Tato nastavení mají nejnižší prioritu při určování nastavení. Upřesnění nastavení popsané v další sekci 3.4.4 mají prioritu vyšší.

conf_dir *cesta* Uvádí, kde v souborovém systému je adresář, ve kterém jsou umístěny konfigurační soubory. Soubory uvedené v názvu záznamu začínají v tomto umístění. Například, při uvedení souboru `make.conf` a nastavení `conf_dir /etc`, se bude sledovaný soubor nacházet v umístění `/etc/make.conf`.

arch_dir *cesta* Upřesňuje, kde je umístěn archivní adresář, do kterého jsou propojeny pomocí pevných odkazů soubory z konfiguračního adresáře. Výsledně při uvedení `arch_dir /var/db/snaproll` a souboru `make.conf` v názvu záznamu. Soubor bude propojen do umístění v archivním adresáři s výslednou cestou `/var/db/snaproll/etc/make.conf`. Struktura obou adresářů je uvedena v kapitole 3.6.

ignorelist <*glob*> Tímto nastavením se specifikuje, za pomoci glob syntaxe, které soubory **nemají** být obsaženy v archivním snapshotu. Nastavení tohoto seznamu je možné dále upřesnit pro každý záznam, jak je to popsáno v sekci 3.4.4.

followsymlinks <*yes*|*no*> Tímto nastavením se upřesňuje, jestli mají být sledovány symbolické odkazy do umístění, na které ukazují.

- Pokud je nastaveno na *yes*, je sledován cíl všech symbolických odkazů, na který nástroj narazí. Tento cíl je pak propojen pevným odkazem na místě v archivním adresáři, které odpovídá umístění v konfiguračním adresáři.
- V případě, že je uvedeno *no*, symbolické odkazy nejsou sledovány do umístění, na které ukazují. Samotný symbolický odkaz je v archivním adresáři vytvořen, ale ukazuje na soubor, který se v archivním adresáři nenachází.

diffprogram *program* Upřesňuje, jakým nástrojem je prováděno zobrazení, případně sloučení rozdílů mezi revizemi souboru. Podrobnější popis je uveden v části 3.14.4. Program se nastaví uvedením spouštěcího souboru, kterým se nástroj spouští.

diffoptions *řetězec* Dodatečné parametry pro externí nástroj na prohlížení rozdílů. Tyto parametry jsou formulovány jako textový řetězec.

Nastavení tohoto sledování je možné dále upřesňovat, pro každý záznam zvlášť, jak je popsáno v následující sekci 3.4.4.

3.4.4 Upřesnění záznamu

Tyto příznaky umožňují specifikovat nastavení pro jednotlivé záznamy, tedy soubory uvedené ve jménu záznamu. Nastavení se vypisují do složených závorek za jménem záznamu. Jak je vidět na obrázku 3.1. Upřesněné záznamy mají větší prioritu, než obecné nastavení. Pokud je tedy u některého záznamu uveden některý z těchto příznaků, obecné nastavení je pro tento záznam změněno na to, které je uvedeno v záznamu.

manual Při tomto nastavení, nebude vytvářen snapshot po přednastavené události, ale pouze v případě spuštění nástroje `snapshot_create(1)`

ontime ∗ ∗ ∗ ∗ ∗ Tento příznak přednastavuje vytváření snapshotů v upřesněný čas, kdy bude spuštěno vytváření snapshotu. Použití příznaku je zobrazeno na obrázku 3.1.

onchange Tímto příznakem se nastavuje vytváření snapshotů při zápisu do souborů, ke kterým se toto nastavení vztahuje.

followsymlinks <*yes*|*no*> Tímto nastavením se přednastavuje chování nástroje pro aktualizaci archivního adresáře. Nastavení upřesňuje chování u symbolických odkazů.

- Pokud je nastaveno na *yes*, je sledován cíl všech symbolických odkazů, na který nástroj narazí. Tento cíl je pak propojen pevným odkazem na místě v archivním adresáři, kterému odpovídá umístění v konfiguračním adresáři.
- Je-li uvedeno *no*, symbolické odkazy nejsou sledovány do umístění, na které ukazují. Samotný symbolický odkaz je v archivním adresáři vytvořen, ale ukazuje na soubor, který se v archivním adresáři nenachází.

Toto nastavení se považuje za nastavení s nejvyšší prioritou. Nastavení se vztahuje ke konkrétnímu záznamu, u kterého je uvedeno. Vstupním parametrem toto nastavení není možné změnit.

ignore <*glob*> Za tímto příznakem se uvádí seznam souborů v *glob* syntaxy[2], které **nebudou** přidány při vytváření seznamu souborů určených pro snapshot. Tímto způsobem je možné dodatečně upřesnit které soubory se v snapshotu mají nacházet.

3.4.5 Standardní nastavení globálních proměnných

Proměnné, které jsou používané celou aplikací už nespadají do nastavení běžně prováděné uživatelem, jsou umístěné do dalšího souboru. Soubor je naimportován do každého zdrojového kódu, zde jsou proměnné nastaveny na hodnoty v tomto souboru uvedené.

Hodnoty nastavované tímto způsobem jsou například název souboru, do kterého se ukládá PID. Dalším dobrým příkladem je název a cesta souboru `/etc/crontab`, který byl umístěn z důvodů

[2]Tato syntaxe je popsána v předchozí části 3.1.7.

testování na jiném umístění. Dalším využitím tohoto souboru je nastavení klíčových slov, podle kterých jsou rozeznávány revize. To znamená co je v revizi uvedeno a proč tato revize vznikla.

Soubor je pojmenován `defaults`, s příponou odpovídající programovacímu jazyku. Tento soubor je možné naimportovat do programu a použít proměnné uvedené v tomto souboru.

3.5 Chybový výstup

Umožnit všem nástrojům vypisovat oznámení o jejich činnosti, nebo o neúspěchu, až po důvod ukončení činnosti umožňuje chybový výstup. Tento výstup je společný pro všechny nástroje. Jeho nastavení se provede pomocí vstupních parametrů nástrojů. Parametry jsou popsány v části 3.4.1. Je třeba, aby tento výstup byl už nastavený, a k dispozici během čtení konfiguračního souboru. Nastavení se týká především úrovně výřečnosti jednotlivých nástrojů. Úroveň je možné nastavit od velice informativního, až po tichý režim, kdy je výstup vypisován pouze do logovacího souboru[3]. Výstup do logovacího souboru je dostupný vždy, a je nastaven na velice informativní vypisování. Jsou zde tři úrovně výřečnosti. Jedna z možností umožňuje úplné vypnutí výstupu do konzole, z které byl nástroj spuštěn. Jedná se o následující úrovně:

info Vypisuje kompletně všechny výpisy, které nástroje oznamují. Tedy i informační výpisy.

warning Standardně přednastavená úroveň, která zobrazuje varování a chybové výpisy.

error Tato úroveň potlačí většinu výstupů, až na chybové výpisy.

Úrovně již byly zmíněny v předchozí části popisující vstupní parametry 3.4.1.

Chybový výstup je vždy inicializován při spuštění některého z nástrojů. Spuštění je naznačeno na sekvenčním diagramu 3.7. Při spuštění jsou třídě předány parametry, podle kterých třída nastaví chybový výstup. V diagramu 3.4 toto spouštění chybového výstupu není uvedeno. Tímto způsobem je možné udržet vyšší přehlednost diagramu. Zároveň se nejedná o příliš podstatné volání pro popis zbylého diagramu.

3.6 Struktura archivního adresáře

Tyto dva adresáře obsahují konfigurační soubory. V konfiguračním adresáři jsou umístěny konfigurace systému. V archivním adresáři jsou vytvořeny pevné odkazy na soubory z konfiguračního adresáře. Nejsou použity veškeré soubory, ale ty, které jsou upřesněny v názvu záznamů v konfiguračním souboru.

V archivním adresáři jsou umístěny soubory v jednotlivých adresářích, odpovídající počítači, na kterém jsou umístěny. Ve většině případů je tímto umístěním `localhost`, pokud se jedná o soubory nacházející se na místním počítači. V tomto adresáři je umístěn repositář SCM. Tento repositář je použit pro ukládání snapshotů, do kterého jsou umisťovány rozdílové zálohy souborů z podadresářů.

Cesta k archivnímu adresáři začíná v kořenovém adresáři systému a je upřesněna v konfiguračním souboru. Podadresáře odpovídají adresářům konfiguračního adresáře. Tam, kde u konfiguračního adresáře je kořenový adresář, je v archivním adresáři konec cesty archivního adresáře. Struktura je naznačena na následujícím příkladu 3.2. V příkladu je vidět konfigurační adresář na levé

[3] Tato možnost je výhodná v případě spouštění nástroje z jiného skriptu

straně. Na pravé straně se nachází archivní adresář obsahující klientské adresáře pro klienty s názvem `klient1` a `klient2`. Dále je v archivním adresáři umístěn adresář s repositářem pro lokální počítač.

Soubory a adresáře konfiguračního adresáře jsou umístěny na stejnou úroveň tak, jak odpovídají propojení do archivního adresáře.

```
                                    /var/db/snaproll/
                                            klient1/
                                            klient2/

                                            localhost/
                                                    .hg/
/etc/                                               etc/
acpi/                                               acpi/
wpa_supplicants/                                    wpa_supplicants/
exports                                             exports
```

Příklad 3.2: Struktura konfiguračního a archivního adresáře

Tímto způsobem odpovídá kořenový adresář systému a archivní adresář stejnému umístění. Takto je dosaženo možnosti ukládat i symbolické odkazy, které odkazují i mimo konfigurační adresář. Pokud je tedy, jako konfigurační adresář upřesněn, například `/home` je pak možné ukládat i nastavení uživatelských aplikací. Ukládání tímto způsobem je možné pokud je uživatel autorizován pro zápis do nastavení a archivního adresáře. Toho je možné dosáhnout specifickým nastavením nástrojů.

3.6.1 Klientské adresáře ve struktuře archivního adresáře

Klientské adresáře jsou dostupné v archivním adresáři. Adresář obsahující repositář pro lokální počítač se nachází v adresáři `/var/db/snaproll/localhost`, jak je uvedeno v příkladu 3.2. Pro klientské repositáře jsou zde další adresáře, do kterých se umístí klientské repositáře. Klienti jsou od sebe odlišeni názvem adresáře, který by měl odpovídat jejich síťovému jménu, případně jinému jménu, pro které se uživatel rozhodne. Volba názvů těchto adresářů není podstatná pro chod nástrojů. V podstatě je volba názvu adresáře čistě na uživateli. Součástí návrhu není nástroj pro přesuny těchto archivních adresářů mezi počítači. Nicméně je dobré se zmínit, jakým způsobem je možné tyto adresáře přesouvat mezi jednotlivými počítači.

Nejjednodušším způsobem je možnost zkopírovat archivní adresář z jiného umístění. Provedení by mohlo být provedeno jak je uvedeno v následujícím příkladu.

```
scp  -r root@klient:/var/db/snaproll/localhost /var/db/snaproll/klient
```

V tomto umístění je už možné tento adresář použít jako klientský, pro zobrazování obsahu, případně rozdílů mezi verzemi souborů. K přenosu je použit nástroj `scp(1)`. Tento nástroj umožňuje přenášet bezpečně[4] soubory, nebo adresáře přes síť.

Jedním z problémů, které se mohou vyskytnout, je bezpečnostní standard, který nedoporučuje umožnit přihlášení uživatele root tímto způsobem. Tento problém je jedním z důvodů, proč není nástroj navržen. Dále je nutné spustit tento přenos jako uživatel, který může zapisovat do adresáře `/var/db/snaproll`.

[4] Při přenosu je použito šifrování na vrstvě SSL.

Dalším vhodný způsob, jak zpřístupnit tyto klientské adresáře, může být využití síťového souborového systému[5]. Bližší informace mohou být nalezeny v manuálových stránkách `nfs(5)`.

3.6.2 Využití archivního adresáře

Archivní adresář umožňuje uložení repositáře SCM odděleně od konfiguračních souborů systému. Díky tomuto adresáři nevznikají žádné další podadresáře, ani soubory navíc v konfiguračním adresáři. Tento přístup je zvolen kvůli zpětné kompatibilitě. Vytvářením souborů, nebo adresářů v konfiguračním adresáři, by mohlo vést k zmatení uživatele. SCM potřebuje umístit svůj repositář do stejného místa, v kterém se nacházejí sledované soubory.

Současně je vhodné, aby v konfiguračním adresáři nevznikaly tyto repositáře. Tento problém je vyřešen pomocí pevných odkazů 3.1.10, které odkazují na soubory určené k zálohování. Repositář je umístěn na stejném místě, na kterém jsou tyto odkazy. Za takové místo lze označit adresář `/var/db/snaproll`. Adresář `/var` je zvolen z konvence. Do tohoto adresáře jsou většinou ukládány proměnné OS. Stejná konvence platí i pro umístění konfiguračních souborů v adresáři `/etc` [9].

Pevný odkaz nemůže být umístěn mezi rozdílnými diskovými oddíly. V případě, že adresáře `/etc` a `/var` jsou na rozdílných diskových oddílech, nebude možné využít adresář `/var`. Pak je možné umístit odkazy a repositář přímo do adresáře `/etc` na stejné místo, kde se bude nacházet nastavení navrhované aplikace. Případně do jakéhokoliv adresáře na stejném diskovém oddílu.

3.6.3 Vytvoření archivního adresáře

Archivní adresář se vytvoří pomocí nástroje pro aktualizaci struktury obou adresářů. Tento nástroj je popsán v části 3.7. Struktura konfiguračního adresáře by měla být vytvořena už po instalaci OS. V případě archivního adresáře je nutné použít aktualizační nástroj. Tento nástroj vytváří archivní adresář od části označující, o který klientský adresář se jedná. Část `/var/db/snaproll` již musí být vytvořena[6]. To samé platí i pro každý jiný adresář, uvedený v konfiguračním souboru.

Po upravení konfiguračního souboru `/etc/snaproll.conf` spustí uživatel tento nástroj a ten na základě získaných informací provede aktualizaci archivního adresáře. To znamená, že vytvoří pevné odkazy na soubory v konfiguračním adresáři, případně vytvoří adresáře na místech, která odpovídají konfiguračnímu adresáři.

3.7 Návrh nástroje pro aktualizaci struktury adresářů

Tento nástroj je použit pro aktualizaci obou adresářů a jejich struktury, která je popsána v sekci 3.6. Tyto aktualizace jsou součástí každého ukládání, nebo obnovování snapshotu v závislosti, který adresář byl změněn, a také podle toho, který nástroj byl použit. To znamená, že pokud je zahájeno vytváření snapshotu, je nejprve aktualizován archivní adresář. Naopak, při obnovení snapshotu, tedy rollbacku, je po této akci aktualizován adresář konfigurační. Tento nástroj aktualizuje kompletně celý adresář pro všechny záznamy uvedené v konfiguračním souboru. V případě dalších nástrojů je tato aktualizace prováděna pouze pro záznam, s kterým nástroj pracuje.

Těmito aktualizacemi se zaručí to, že v žádné záloze neskončí soubor, který by v ní neměl být umístěn. Podobně je také zaručeno, že v konfiguračním adresáři nebudou soubory z jiné, než obnovované zálohy. Posloupnost akcí, pomocí kterých se tato aktualizace provádí, je popsána dále v sekci 3.4.

[5] Název vychází z anglického „Network File System“.

[6] Tento adresář je vytvářen během instalace aplikace.

Nástroj vytvoří archivní adresář pomocí základní vrstvy pro zpracování konfiguračního souboru, která je popsána dále v textu 3.9. Tato základní vrstva zprostředkuje všechny operace nástroje pro aktualizaci adresářů. Nástroj je spouštěn přímo uživatelem po editaci nastavení v souboru `/etc/snaproll.conf`.

3.7.1 Vstupní parametry

Vstupní parametry, kterými je možné ovládat běh programu jsou v následujícím výpisu:

–symlinks <*yes*|*no*> Tímto parametrem je možné změnit nastaveni z konfiguračního souboru `/etc/snaproll.conf`. Parametr může změnit nastavení pro všechny soubory, které je popisované ve výpisu 3.4.3. Tímto parametrem není možné změnit chování u konkrétního záznamu.

–archdir Tento parametr upřesňuje činnost, kterou bude nástroj pro aktualizaci adresářů vykonávat. V tomto případě se jedná o aktualizaci archivního adresáře. Aktualizaci je nutné provádět po změně nastavení v souboru `/etc/snaproll.conf`. Spuštěním nástroje s tímto parametrem se docílí toho, že budou přidány a odebrány soubory do archivního adresáře, podle nastavení v souboru `/etc/snaproll.conf`. Dále je také aktualizován repositář úložiště, což je popsáno v části 3.12. Poslední činností je přidání záznamů souboru, který umožňuje vytváření snapshotů v předvolený čas. Toto přidávání je popsáno v části 3.8.

–confdir Tento parametr umožňuje aktualizovat konfigurační adresář podle struktury, která se nachází v archivním adresáři. Aktualizace konfiguračního adresáře by nemělo být nutné provádět, neboť tyto jsou provedeny nástrojem pro obnovu snapshotů. Obnovování je popsáno v části 3.15.

Dále je možné použít vstupní parametry, které jsou běžně dostupné pro všechny nástroje. Parametry jsou popsány v části 3.4.1. Pomocí těchto dalších parametrů lze přenastavit umístění konfiguračního souboru, změnit umístění logovacího souboru atd.

3.7.2 Diagram tříd

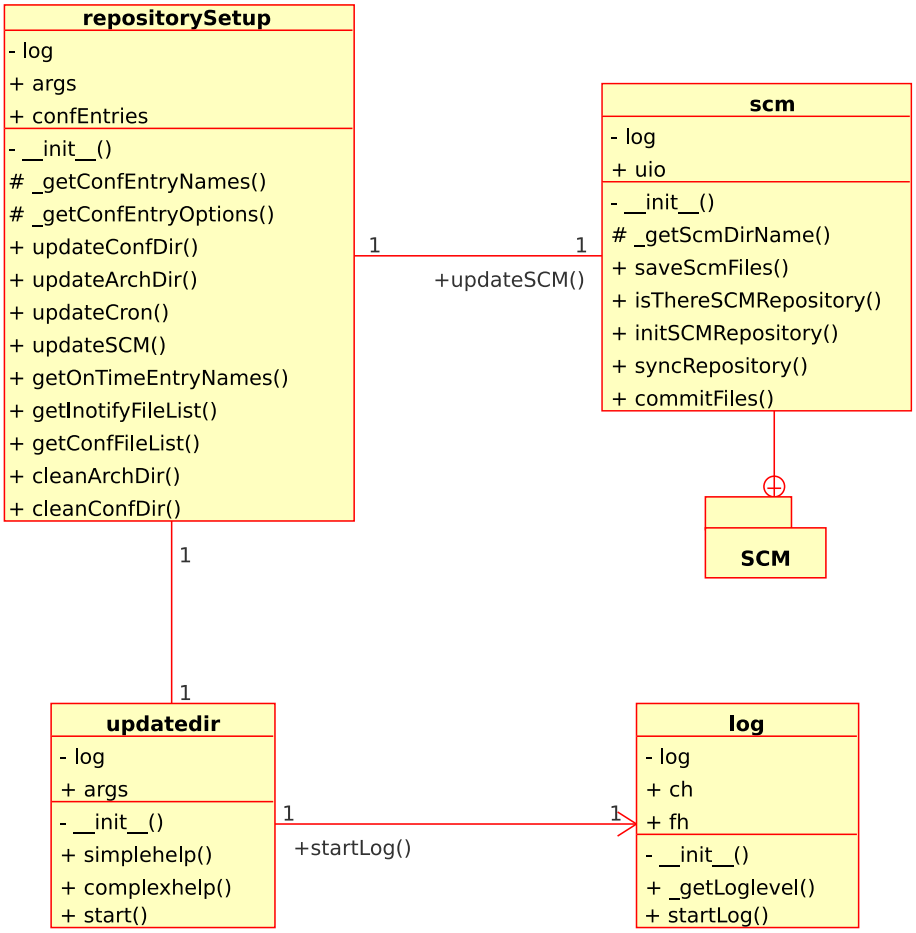
Následující diagram tříd 3.3 zobrazuje, jakým způsobem se bude nástroj skládat z jednotlivých tříd. Tyto třídy jsou rozděleny podle významu, který zastupují i pro ostatní třídy. Diagram je následován výpisem který má objasnit význam těchto tříd.

UpdateDir Po spuštění uživatelem je vytvořena třída, která zpracuje vstupní parametry a nastaví chybový výstup pomocí třídy log. Nakonec inicializuje třídu RepositorySetup. Metody této třídy jsou poté použity pro aktualizaci adresářů.

log Tato třída je po spuštění inicializována z každého spustitelného souboru, stejně jako u tohoto nástroje. Logování je nastaveno podle vstupních parametrů zadaných uživatelem. Tyto parametry jsou popsány v části 3.4.1.

RepositorySetup Tato třída zprostředkuje základní vrstvu, která umožňuje synchronizaci obou adresářů, přístup ke konfiguračnímu souboru `/etc/snaproll.conf`, případně dalším metodám, potřebným k zápisu do adresářů. Podrobnější popis této vrstvy se nachází v sekci 3.9.

Scm Tato třída je součástí balíku SCM, který je součástí snaproll aplikace. Třída slouží jako prostředek komunikace se systémem správy verzí. V tomto případě poskytuje možnost přidat a odebrat soubory z repositáře.



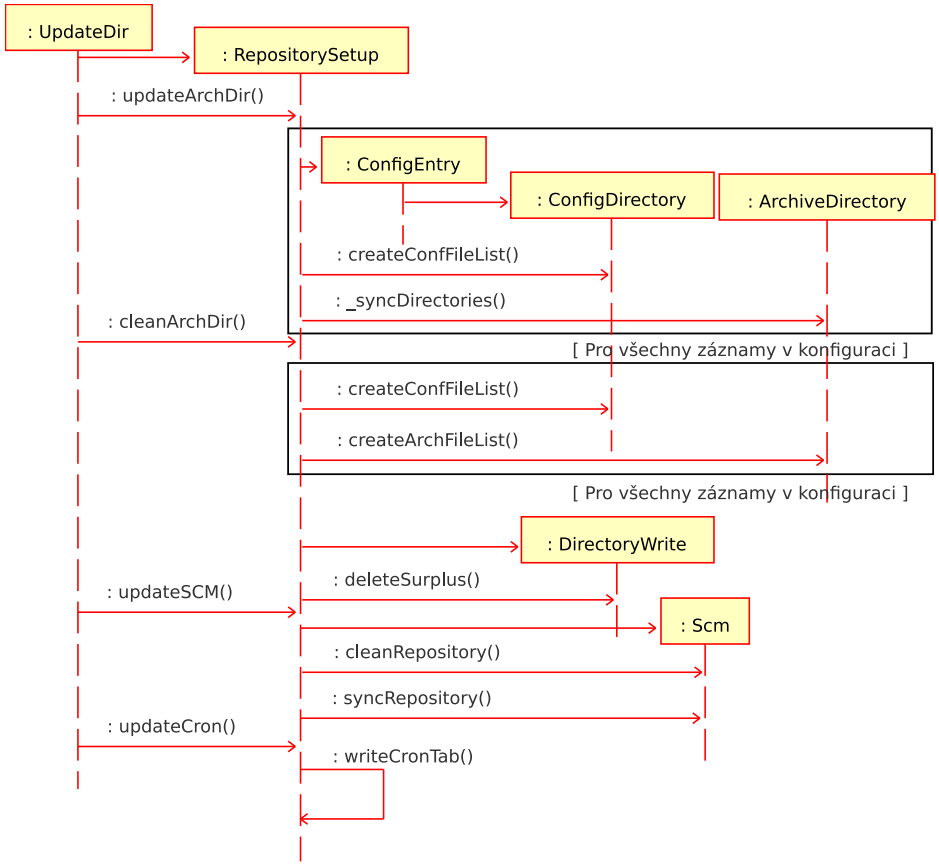
Obrázek 3.3: Diagram tříd s nástrojem pro aktualizaci archivního a konfiguračního adresáře.

3.7.3 Sekvenční diagram

Posloupnost inicializací a volání metod by měla být zřejmá ze sekvenčního diagramu 3.4. V tomto diagramu je názorně zobrazena posloupnost, pomocí které probíhá aktualizace archivního adresáře. Do sekvenčního diagramu je zároveň zahrnuta práce základní vrstvy, která je popsána v sekci 3.9.

V následujícím výpisu je možné vidět posloupnost provádění metod při aktualizaci archivního adresáře. Ve výpisu jsou postupně popsány jednotlivé metody. Každá z metod má svoje číslo, sloužící jako jednoznačné označení.

updateArchDir() Tato metoda vytváří soubory, které mají být součástí archivního adresáře. Části volání, které jsou označeny černým obdélníkem, jsou provedeny cyklicky několikrát za sebou. Každý průchod odpovídá jednomu záznamu z konfigurace, jako je například záznam uvedený v příkladu 3.1. Tímto způsobem se naplní seznamy souborů, adresářů a symbolických odkazů. Listy jsou vytvořeny na základě nastavení ze souboru `/etc/snaproll.conf`. Každý z těchto záznamů obsahuje jiné soubory. Proto je nutné vytvářet tyto seznamy pro každý záznam zvlášť. Následná aktualizace archivního adresáře se provádí pro každý z těchto seznamů.



Obrázek 3.4: Sekvenční diagram s aktualizaci archivního adresáře.

cleanArchDir() Touto metodou se provádí mazání, respektive odlinkování souborů, které nemají být v archivním adresáři. Oblast označená černým obdélníkem se provádí cyklicky několikrát za sebou, podobně jako tomu bylo u předchozí metody. V tomto případě je nutné, aby byly k dispozici seznamy souborů pro archivní i konfigurační adresář.

updateSCM() Metoda poskytuje možnost, jak přidat nový soubor do některého okruhu. SCM potřebuje pro svou činnost sledovat tyto soubory. Sledování souborů je zahájeno právě touto metodou. Metodou se docílí přidání souborů do SCM. Po přidání je možné s těmito soubory dále pracovat.

updateCron() Metoda slouží k vytvoření odpovídajícího zápisu do souboru, který nastavuje službu `cron(8)`. Tato operace je popsána v části 3.8. Po zapsání do nastavené služby `cron(8)` začne tato služba ihned při prvním uvedeném čase zálohovat.

createConfFileList() Metoda naplní seznamy obsahující soubory, linky a adresáře do seznamů, které se nacházejí v konfiguračním adresáři.

_syncDirectories() Metoda, která vytvoří všechny pevné odkazy na soubory do konfiguračního adresáře, případně na symbolické odkazy. V případě potřeby je vytvořena i adresářová cesta.

createArchFileList() Vytvoření seznamu souborů, symbolických odkazů a adresářů v archivním adresáři.

deleteSurplus() Metoda smaže všechny soubory, adresáře a symbolické odkazy podle seznamů, které byly vytvořeny v předchozích metodách. V seznamech jsou položky, které nejsou v konfiguračním adresáři.

cleanRepository() Odebrání souborů z repositáře SCM. Odeberou se soubory, které se už nenacházejí v archivním adresáři. To znamená, že byly smazány předchozí metodou **deleteSurplus()**.

syncRepository() Přidání nových souborů do repositáře SCM. Přidání souboru do repositáře je nutné, aby bylo možné s těmito soubory v repositáři pracovat. Nástroj pro vytváření snapshotů umožňuje rovněž přidat nové soubory, nicméně u tohoto nástroje se operace provádí pouze pro jeden záznam.

writeCronTab() Pomocí této metody jsou do souboru `/etc/crontab` přidány záznamy umožňující spustit zálohování v přednastavený čas. V tomto souboru se nachází nastavení externí služby `cron(8)`.

Po provedení této sekvence příkazů by měl být archivní adresář aktualizován podle uživatelského nastavení a obsahu konfiguračního adresáře. Dále by soubory měly být přidány mezi soubory, které sleduje SCM na změny. Ty jsou v případě vytváření snapshotu uloženy do vytvářené verze. Záznamy by měly být opatřeny příznakem `ontime`, uvedeny v nastavení služby `cron(8)`.

3.8 Návrh principu vytváření snapshotů v určený čas

V dolní části sekvenčního diagramu je uvedena poslední akce prováděná nástrojem pro aktualizaci struktury adresářů 3.4. Vzhledem k tomu, že tento nástroj je spouštěn po změně nastavení, je tímto nástrojem měněn i soubor `/etc/crontab`. Soubor je využíván službou `cron(8)`, která spouští další nástroje v přednastavený čas. Služba se běžně používá na OS GNU/Linux k spouštění dalších úloh v přednastavený čas [7].

Princip nastavení úlohy spočívá v uvedení naplánované úlohy ve zmiňovaném souboru `/etc/crontab`. Každý řádek, začínající upřesněním časového období, zavádí jednu úlohu do sledování. Každý řádek má specifickou syntaxi. Nastavení poskytuje velkou podrobnost a přesnost v uvedení času. Nastavení je možné s přesností na minuty, nebo na dny v týdnu, měsíci. Tímto způsobem je možné podrobně specifikovat, kdy bude úloha spuštěna. V souboru je uveden pro každý příkaz řádka postupně obsahující:

Čas
- minutu, v kterou bude příkaz spuštěn
- hodinu
- den v měsíci
- měsíc
- den v týdnu (0-neděle, 1-pondělí, ...)

Uživatel pod kterým bude příkaz spuštěn

Příkaz který se v uvedený čas spustí

Kterýkoliv z prvků upřesňující čas, je možné vyřadit z definice tak, že je místo hodnoty uvedena „∗“. Další možností, jak upřesnit čas, je uvedení několika čísel oddělených čárkou. Takto je upřesněno několik hodnot odpovídajících umístění v pětici možností pro určení času [7]. V případě následujícího příkladu je příkaz spouštěn každé pondělí a úterý. Okruh zálohovaných souborů na příkladu 3.3 se nachází v adresáři `acpi/`:

```
acpi/* {
    ontime * * * * 1,2
    }
```

Příklad 3.3: Příklad upřesnění záznamu pro vytváření snapshotu v určitý čas

V případě této aplikace bude vytváření snapshotu prováděno pomocí nástroje pro vytvoření snapshotu. Návrh nástroje se nachází v části 3.2. V případě vytvoření snapshotu vypadá výsledný řádek v souboru `/etc/crontab` tak, jak je uveden v následujícím příkladu 3.4. V příkladu je zobrazeno vytváření snapshotu každých deset minut. Okruh souborů, které bude snapshot obsahovat, je uveden v konfiguraci záznamem `"acpi/*"`. Konkrétní záznam je uveden jako příklad v sekci 3.1.

```
* * * * 1,2 root  /usr/bin/python /usr/sbin/snapshot_create.py "acpi/*" \
                  –m "crontab"
```

Příklad 3.4: Příklad řádku v souboru /etc/crontab

Pokud jsou v souboru uvedeny tyto řádky, `/etc/crontab`, je služba zodpovědná `cron(8)` za spouštění zálohování v požadovaný čas.

3.9 Základní vrstva pro čtení a zápis obou adresářů

Nástroj pro tyto aktualizace je navržen podle struktury záznamů v konfiguračním souboru `/etc/snaproll.conf`, a také podle struktury obou adresářů.

Část nástroje odpovídající jednomu okruhu souborů, který je vytvořen na základě záznamu z konfiguračního adresáře, je na následujícím diagramu 3.5. Tento nástroj je rozdělen do několika tříd, které jsou popsány v následujícím textu.

RepositorySetup Hlavní částí základní vrstvy je třída označená jako **repositorySetup**. Tato třída bude inicializována ze všech dalších nástrojů jako první, pokud bude třeba přistupovat ke konfiguraci `/etc/snaproll`. Tato třída poskytuje možnost číst konfigurační soubor pomocí zděděné třídy **FileReader**.

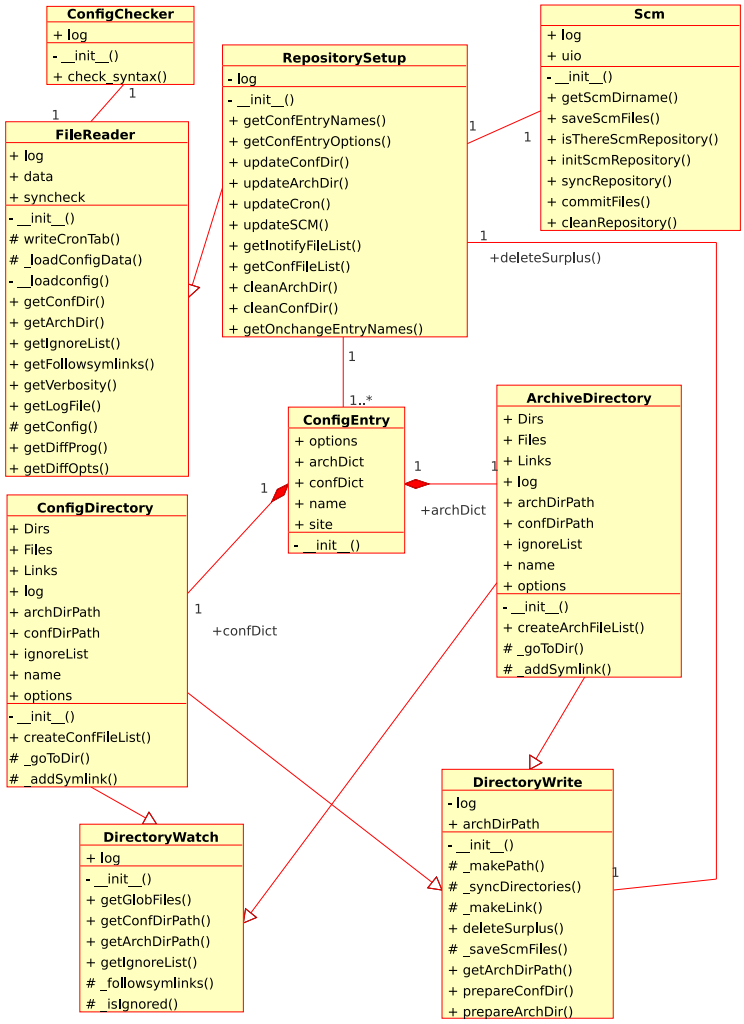
Zároveň je možné pomocí této třídy vytvářet strukturu obou adresářů. Archivní adresář se tímto způsobem synchronizuje s konfiguračním a také naopak umožňuje aktualizaci konfiguračního adresáře při obnově záloh.

Metoda **updateCron()** této třídy umožňuje aktualizovat soubor `/etc/crontab`, v kterém je nastavení služby `cron(8)`. Služba provádí spouštění úloh v přednastavený čas. Princip je popsán v části 3.8.

FileReader Tato třída řeší otevření a čtení konfiguračního souboru. Před předáním získané konfigurace je během čtení souboru prováděna kontrola syntaxe souboru. Kontrolu provádí třída **configChecker**, která v případě špatné syntaxe vypíše zprávu o tomto stavu a zároveň přidá k výpisu řádek, na kterém se tato chyba nachází.

ConfigChecker Třída provádějící kontrolu syntaxe, provede kontrolu vstupního řádku. Kontrola je provedena pomocí regulárních výrazů.

ConfigEntry Tato třída odpovídá jednomu konfiguračnímu záznamu, například záznamu, který je uvedený v příkladu 3.1. Pro každý takový záznam v konfiguračním souboru je vytvořena jedna instance této třídy. Jak vyplývá z diagramu tříd 3.5, třída **ConfigEntry** je kompozicí



Obrázek 3.5: Diagram tříd se základní vrstvou.

dvou dalších, které odpovídají konfiguračnímu a archivnímu adresáři. Třídy jsou pojmenovány **configDirectory** a **archiveDirectory**.

ConfigDirectory & ArchiveDirectory Obě třídy obsahují slovníky[7] souborů, adresářů a symbolických odkazů a dále pak metody určené pro práci s těmito slovníky. V každém z těchto slovníků jsou uvedeny jako klíč soubory v archivním, nebo konfiguračním adresáři a jako hodnoty jsou soubory v opačném adresáři. U slovníku s adresáři a symbolickými odkazy je řešení stejné.

DirectoryWrite Třída umožňuje zápis do souborového systému, to znamená vytváření adresářů, pevných odkazů a mazání nechtěných souborů. Tato třída se dědí oběma třídami **configDirectory** a **archiveDirectory**.

DirectoryWatch Účelem této třídy je získávat umístění adresářů a seznamy ignorovaných souborů, umístění obou adresářů, případně nastavení sledování pevných odkazů.

[7] Slovníky, jako datové typy, jsou popsány v předchozí části 3.1.5.

Scm K aktualizaci archivního adresáře patří také inicializace repositáře SCM, pokud tento repositář už inicializován není. Zároveň je nutné přidat a odebrat soubory, které budou ukládány do záloh úložiště.

Posloupnost provádění metod z diagramu tříd 3.5 je patrná ze sekvenčního diagramu s aktualizací archivního adresáře 3.4. Posloupnost, s kterou je prováděna aktualizace archivního adresáře, je vcelku podobná jako aktualizace konfiguračního adresáře. Tato aktualizace se provádí po obnově některé ze záloh. Popis obnovy bude v dalším textu 3.15.

3.10 Návrh nástroje pro manuální vytvoření snapshotu

Nástroj popisovaný v této části bude řešit samotné vytváření snapshotů. Vytvoření snapshotu je možné po předchozí aktualizaci archivního adresáře. Aktualizace je provedena pomocí nástroje, popsaného v sekci 3.7. Nástroj bude spouštěn přímo uživatelem, případně jsou třídy tohoto nástroje využity dalšími nástroji pro vytvoření snapshotu.

Tento nástroj bude používán pro vytváření snapshotů uživatelem z příkazové řádky. Dalším využitím je vytváření snapshotů po proběhlé časové události. Nástroj je uveden do souboru pro nastavení úloh spouštěných v přednastavený čas. Způsob spouštění je popsán v části 3.8. Poslední možností, jak využít tento nástroj, je spuštění pomocí služby pro sledování událostí na souborech.Služba sleduje okruhy souborů uvedených v konfiguračním souboru. Sledování je popsáno v jedné z následujících částí 4.2.

3.10.1 Vstupní parametry

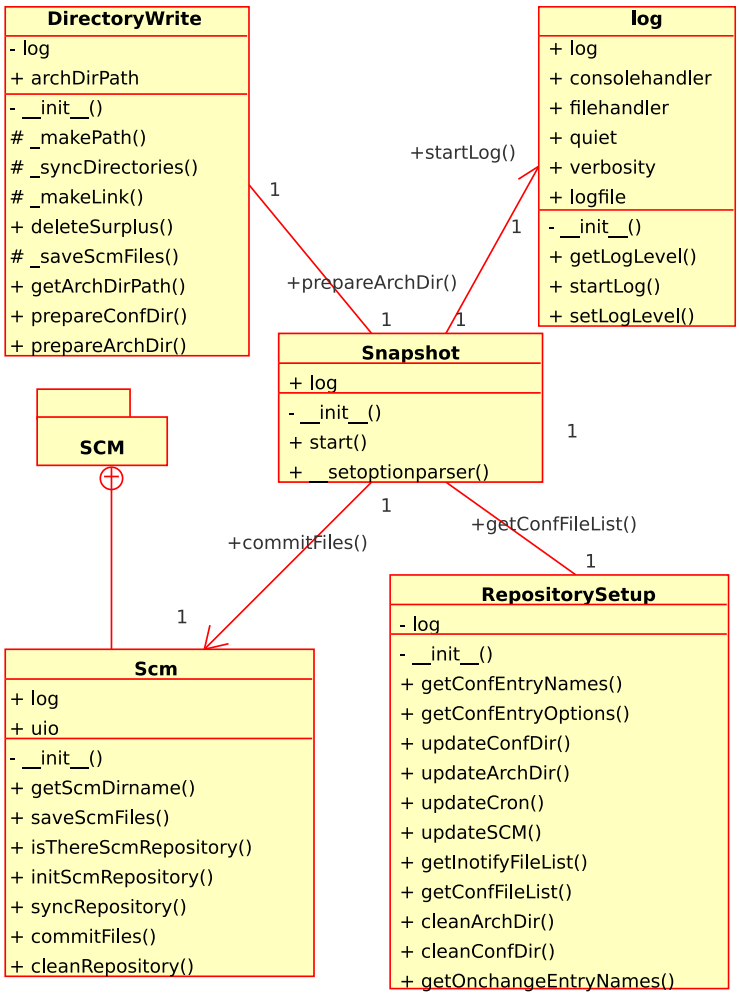
V případě, že nástroj je spouštěn uživatelem, je nutné dodat tomuto nástroji patřičné parametry, podle kterých nástroj zálohu vytvoří. Nástroj je možné spustit i s parametry uvedenými v části 3.4.1. K těmto základním parametrům patří další parametry uvedené níže:

–message "*text*" Tímto parametrem se nastaví zpráva, která bude později přiřazena k vytvořenému snapshotu.

"*záznam*" Tento vstup je povinný. S jeho pomocí je upřesněn záznam z nastavení, tedy okruh souborů upřesněných názvem záznamu. Soubory jsou uloženy do zálohy pokud se v některém vyskytne změna oproti předchozí záloze.

–symlinks <*yes*|*no*> Tímto parametrem je možné změnit nastaveni z konfiguračního souboru `/etc/snaproll.conf`. Tento parametr může změnit nastavení pro všechny soubory, které je popisované ve výpisu 3.4.3. Nastavením není možné změnit chování u konkrétního záznamu.

Pokud při spuštění není upřesněna žádná zpráva, je k záloze přiřazena zpráva obsahující informaci o tom, kterým nástrojem byla vytvořena. Tím je v tomto případě nástroj pro vytvoření snapshotu. Dalším vstupním parametrem je upřesnění, o který záznam z konfigurace se jedná. Tento okruh souborů se musí shodovat s jedním z uvedených v konfiguračním souboru. Příklad tohoto záznamu je na obrázku 3.1. Název záznamu je pomocí základní vrstvy 3.9 převeden na seznam souborů, které jsou následně použity pro vytvoření snapshotu. Vytvoření snapshotu obstarává API systém zprávy verzí, kterému je předán tento seznam souborů.



Obrázek 3.6: Diagram tříd popisující návrh nástroje pro vytvoření snapshotu.

3.10.2 Diagram tříd

Lepší představu může vytvořit následující diagram tříd 3.6, který ukazuje třídy používané tímto nástrojem. Třídy obsažené v diagramu tříd 3.6 jsou popsány ve výpisu níže:

Snapshot Tato třída zajišťuje spuštění nástroje a inicializaci dalších tříd. Zároveň je touto třídou spuštěn chybový výstup aplikace.

Scm Součást balíku SCM. Tato třída umožňuje zápis do úložiště systému správy verzí. V případě nástroje pro vytvoření snapshotu umožňuje uložení souborů do repositáře.

log Nastavení chybového výstupu na požadované úrovně a spuštění modulu, který poskytuje chybový výstup.

DirectoryWrite Tato třída poskytuje metodu, která ještě před aktualizací archivního adresáře provede kontrolu pevných odkazů mezi konfiguračním a archivním adresářem. Oba pevné odkazy v archivním a konfiguračním adresáři odkazují na stejný soubor. Popis pevných odkazů se nachází v předchozí části 3.1.10.

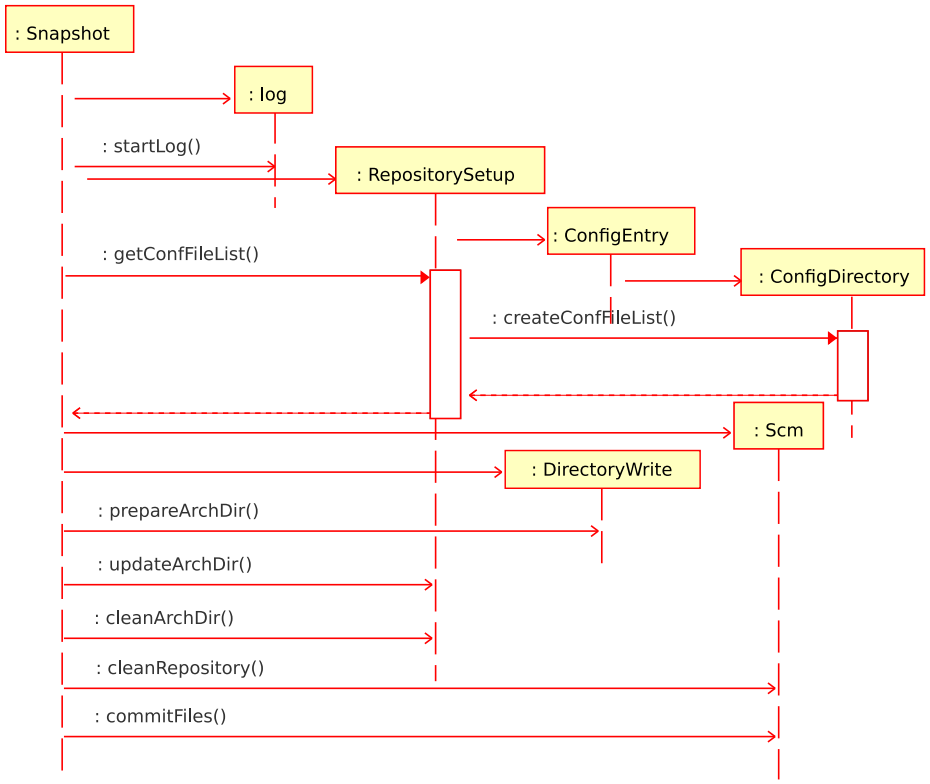
RepositorySetup Třída zajišťující přístup základní vrstvou. Vrstva je popsána v předchozí části 3.9, spolu s popisem dalších tříd. V případě nástroje pro vytvoření snapshotu zprostředkovává metody, pomocí kterých se provede aktualizace okruhu souborů do archivního adresáře. Dále pak poskytuje metody, pomocí nichž je možné získat seznam souborů určených pro zálohování. Tyto metody jsou dostupné pomocí níže uvedených tříd.

ConfigEntry Instance této třídy se vytvoří pro záznam, který je určen pro vytvoření zálohy. Poskytuje přístup k následující podtřídě.

ConfigDirectory Třída umožňuje vytvořit seznam souborů, které spadají do okruhu souborů vyžádaného záznamu.

3.10.3 Sekvenční diagram

V dalším diagramu 3.7 je možno vidět, v jaké posloupnosti se použijí třídy tohoto nástroje, případně další třídy z jiného nástroje.



Obrázek 3.7: Sekvenční diagram popisující návrh posloupnosti volání metod a tříd u nástroje pro vytvoření snapshotu.

V následujícím výpisu je možno vidět posloupnost provádění metod při manuálním vytváření snapshotu. Ve výpisu jsou postupně popsány jednotlivé metody. Metody jsou odkázány pomocí názvu, který se shoduje s názvy v diagramu.

startLog() Touto metodou je nastaven a spuštěn chybový pro nástroj a všechny jeho třídy.

getConfFileList() Metoda použitá pro získání seznamu souborů,odpovídající okruhu souborů, podle uvedeného záznamu z konfigurace.

createConfFilelist() Tato metoda naplní seznam souborů, linky a adresáře do seznamů. Obsahem jsou uvedené položky, které spadají do okruhu uvedeného jako název záznamu.

prepareArchDir() Kontrola pevných odkazů v archivním adresáři. V případě, že tyto pevné odkazy odkazují na jiné soubory, než jsou ty, které se nacházejí v konfiguračním adresáři.

updateArchDir() Aktualizace archivního adresáře souborů, které se nacházejí v konfiguračním adresáři.

cleanArchDir() Mazání souborů, které se už v konfiguračním adresáři nenacházejí. Jedná se například o soubory odebrané z konfiguračního adresáře.

cleanRepository() Touto metodou se provede odebrání souborů ze seznamu souborů sledovaných pomocí SCM. Odebrané soubory se ukládají do zvláštních revizí, které jsou označené jako revize s odebíranými soubory.

commitFiles() Metoda uloží seznam souborů z uvedeného okruhu, předaných jako parametr do úložiště SCM.

Po dokončení této sekvence je v úložišti o jeden záznam více. Záznam bude obsahovat datum, zprávu uživatele, případně zprávu označující tento nástroj jako prostředek, který vytvořil tuto zálohu. Hlavním obsahem snapshotu by měly být soubory určené okruhem, který vychází ze záznamu v konfiguraci. Výpis revize je zobrazen na příkladu v jedné z dalších částí 3.14.1.

3.11 Návrh služby sledování změn v souborovém systému

Dalším nástrojem, který umožňuje vytvořit snapshot je služba, která sleduje změny v souborovém systému. Sledování těchto změn je umožněno pomocí podsystému jádra OS. Tento podsystém je nazván `inotify(7)` a umožňuje aplikacím zažádat o sledování sady souborů na sadu událostí. Aplikace je upozorněna, pokud některá z těchto událostí nastane. V tomto případě je touto událostí zápis do souboru. Zápis do souboru je vhodná událost pro vytvoření snapshotu. Pokud je v nastavení uveden záznam s tímto způsobem ukládání, jsou tyto události v tomto případě sledovány.

Pokud událost nastane, je provedeno vytvoření snapshotu podobně, jako je to popsáno v části 3.10. V tomto případě není nástroj pro vytvoření snapshotu volán z příkazové řádky, ale přímo je volána hlavní třída nástroje. Třída inicializuje třídu snapshot a po předání názvu záznamu tato třída provede vytvoření snapshotu.

3.11.1 Spuštění služby

Služba je spouštěna pomocí skriptu umístěného v adresáři `/etc/rc.d/init.d`. V adresáři jsou umístěny i další skripty pro odpovídající služby. V tomto případě je touto službou sledování změn v souborovém systému. Skript se spouští s následujícímy parametry:

start Při tomto parametru skript otestuje, jestli je dostupný soubor `/var/run/snaproll.pid`. V souboru je uvedeno číslo identifikující proces[8]. Tento identifikátor je použit pro zastavení spuštěného procesu, případně pro získání stavu procesu. Pokud tento soubor neexistuje, je poté spuštěna služba sledování změn v souborovém systému.

[8]Pro číslo identifikující proces se běžně v literatuře používá zkratka PID (anglicky Process Identification Number). Z tohoto důvodu bude v textu dál použita tato zkratka.

stop Tento parametr je určen k zastavení spuštěné služby, respektive procesu. Zastavení je provedeno přečtením PID v souboru `/var/run/snaproll.pid` a zastavení procesu pomocí příkazu `killproc(8)`, případně jeho ekvivalentem poskytnutým distribucí GNU/Linux.

restart Tímto procesem je možné spuštěnou službu restartovat, to znamená zastavit a opět znovu službu spustit. Tak jak je to uvedeno v předchozích dvou případech.

status Tímto parametrem[9] je možné získat stav v jakém se služba nachází. Tedy jestli služba běží nebo je zastavena.

Pomocí těchto parametrů je možné ovládat běh služby, která je spuštěna jako démon, více je popsáno v části 3.1.6. Jakmile je tato služba spuštěna, umožňuje sledování souborů z konfiguračního souboru. Služba je spouštěna s nastavením, které zvyšuje výřečnost služby, nicméně chybové výpisy jsou směrovány pouze do logovacího souboru.

Krom uvedených způsobů jak tuto službu spustit, je možné spustit přímo z příkazové řádky nástroj `snap_notifier(8)`. Při spuštění je možné použít stejné obecné parametry uvedené v předchozí části 3.4.1. Tento nástroj je také možné spustit s jedním specifickým parametrem:

–nodaemon Tento parametr způsobí, že služba se nerozdvojí a nezáhájí svůj běh na „pozadí“ systému. Tento způsob spouštění je zejména vhodný pro testovací účely.

3.11.2 Diagram tříd

Diagram tříd u tohoto nástroje je poněkud složitější. Je to způsobeno tím, že v nástroji je využívána další třída z externího balíku. Konkrétně se jedná o externí třídu, která zpracovává příchozí události na sledované soubory. Podrobnější popis příchozích událostí je popsán v implementační části 4.2.

Další třída zpracovává změny v souborovém systému a je v návrhu diagramu označena jako **ProcessEH**. Tato třída dědí metody pro zpracování událostí, které se spustí v případě jejího vyvolání některou ze sledovaných událostí. V tomto případě je poté voláno vytvoření snapshotu se vstupním parametrem okruhu souborů, z kterých změna vychází. Dále už vytvoření snapshotu probíhá stejně, jak je popsáno v části 3.10. Rozdíl je v předání vstupního argumentu do třídy snapshot. Třída je zodpovědná za vytvoření tohoto snapshotu. V tomto vstupu se nachází jméno záznamu z konfigurace a zároveň zpráva s údajem o nástroji, který spustil vytvoření snapshotu. Tedy částí aplikace, která sleduje změny v souborovém systému.

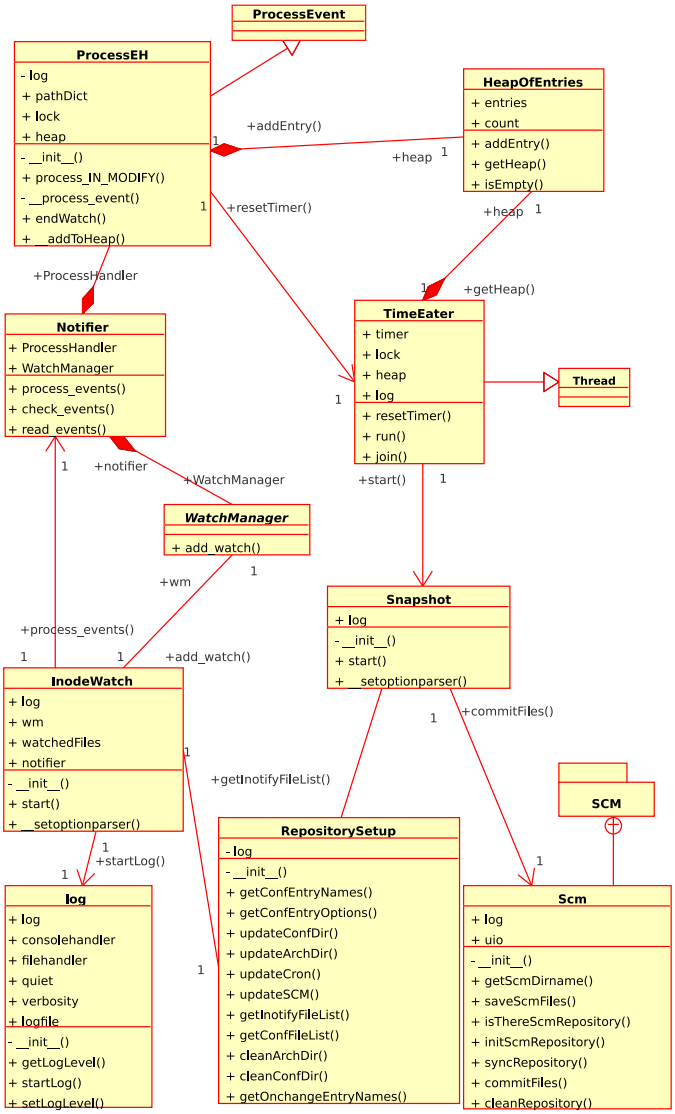
Následující diagram 3.8 zobrazuje třídy, které jsou využity nástrojem pro sledování změn v souborovém systému. Třídy vyskytující se v tomto diagramu tříd jsou popsány v následujícím seznamu.

InodeWatch Po spuštění uživatelem je vytvořena tato třída. Zpracuje vstupní parametry a nastaví chybový výstup pomocí třídy log. Nakonec inicializuje další třídy, které umožní práci sledování souborů. Hlavní funkcí této třídy je vytvoření třídy sledující změny. Další důležitou činností je přidání souborů, případně adresářů do sledování pomocí instance třídy notifier.

RepositorySetup Tato třída zprostředkuje základní vrstvu, která umožňuje synchronizaci obou adresářů, přístup ke konfiguračnímu souboru, případně dalším metodám, potřebným k zápisu do adresářů. Podrobnější popis této vrstvy se nachází v sekci 3.9.

Notifier Tato třída je externí třídou, která přijímá jako vstup třídu ProcessEH a WatchManager. Třídou je potom sledována aktivita odehrávající se na sledovaných souborech.

[9] Tento parametr nemusí být dostupný na všech distribucích GNU/Linux.



Obrázek 3.8: Diagram tříd zobrazující návrh služby pro sledování změn v souborovém systému.

WatchManager Třída spravující sledované soubory. Soubory jsou do této třídy předány jako seznam souborů určených pro sledování spolu s událostmi, které budou sledovány. V tomto případě se jedná především o modifikaci souboru. Jedná se o externí třídu použitou z přidaného modulu, který umožňuje sledování souborů.

ProcessEvent Třída z externího balíku umožňující sledování souborů. Tato třída je děděna třídou ProcessEH, která je součástí služby sledování souborů.

ProcessEH Tato třída dědí vlastnosti třídy ProcessEvent. Předefinováním akcí, volaných jako reakce na událost na souboru, je umožněno provádět vlastní akce. Vlastní metoda prováděná při některé z událostí je pojmenována **__process_event()**. V této akci je určen název záznamu, do kterého spadá soubor, na kterém se událost odehrála. Záznam je pak přidán do seznamu třídy **HeapOfEntries**.

Thread Externí třída, která je děděna následující třídou **TimeEater**. Tato třída umožňuje spustit tuto třídu nezávisle jako vlákno, které běží ve stejném adresovém prostoru. Pojem vlákno je už vysvětlen v předchozí části 3.1.21.

TimeEater Třída poskytující časovou prodlevu od každé příchozí události. Těchto událostí může být velké množství i v případě jednoho souboru. Třída čeká na další příchozí události. Po uběhnutí prodlevy po poslední události, tato třída převezme záznamy z třídy **HeapOfEntries** a předá záznamy třídě **Snapshot**, který vytvoří požadované zálohy.

HeapOfEntries Tato třída obsahuje seznam záznamů, které jsou přidávány po každé příchozí události.

Snapshot Třída pro vytvoření snapshotu. Do této třídy je předáno jméno záznamu z nastavení a zároveň zpráva obsahující informaci, který nástroj snapshot vytvořil. V tomto případě je to služba sledování událostí na souboru.

log Tato třída je inicializována z každého spustitelného souboru po spuštění, stejně tak u tohoto nástroje. Logování je nastaveno podle vstupních parametrů zadaných uživatelem. Tyto parametry jsou popsány v části 3.4.1.

Scm Tato třída slouží jako prostředek komunikace se SCM. V tomto případě poskytuje možnost uložit změněné soubory do repositáře SCM pomocí metody **commitFiles()**.

3.11.3 Sekvenční diagram

V této sekci je sekvenční diagram 3.9 s posloupností vykonávaných akcí. Diagram by měl napomoci vysvětlit, jak je provedeno přidávání souborů do sledování a samotné sledování. Dále vysvětluje vytváření snapshotů ze souborů, na kterých se událost udála. Popis metod, které jsou naznačeny v diagramu, následuje níže.

getOnchangeEntryNames() Metoda použitá pro získání názvů záznamů, které jsou nastaveny pro sledování událostí na změnu.

getInotifyFileList() Touto metodou je získán seznam souborů, které budou předány pro sledování třídou notifier. Seznam je vytvořen na základě názvu záznamu z konfiguračního souboru.

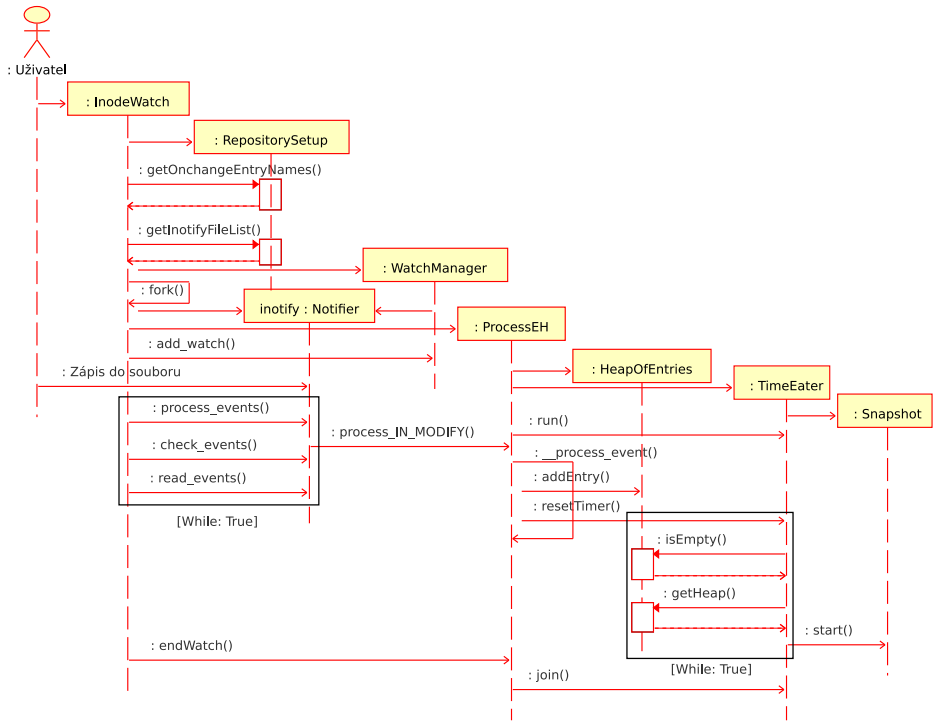
fork() Metoda umožňující přesunout běžící proces na pozadí. Tedy rozdvojení procesu a ukončení části běžící na popředí. Druhý proces pokračuje dál během na pozadí systému jako démon. Více je už popsáno v předchozí části 3.1.6.

add_watch() Touto metodou jsou přidány soubory a adresáře určené pro sledování do třídy **WatchManager**. Tato je následně předána jako vstupní parametr do třídy **Notifier** spolu s třídou **ProcessEH**.

Zápis do souboru Touto akcí je označen zápis uživatelem do sledovaného souboru. V případě, že k této akci dojde, je akce zaznamenána třídou **Notifier**.

process_events() Tato metoda náleží do externího balíku[10], který umožňuje sledování událostí nad soubory. Tato akce zpracuje frontu událostí voláním jejich přiřazené procesní funkce (Třída ProcessEvent zastoupená zde třídou ProcessEH).

[10]Popis následujících metod byl převzat z dokumentu, zabývající se zpracováním událostí [15].



Obrázek 3.9: Diagram tříd zobrazující návrh služby pro sledování změn v souborovém systému.

check_events() Tato akce kontroluje nové akce, které jsou k dispozici pro čtení.

read_events() Akce přečte události z podsystému jádra a přidá je do fronty pro zpracování.

run() Metoda spouštějící běh vlákna. Tato metoda obsahuje cyklus, který periodicky kontroluje seznam příchozích záznamů.

process_IN_MODIFY() Procesní akce, která je volána metodou **process_events()**, pokud se odehraje sledovaná událost. Tato akce přímo volá další **__process_event()**.

__process_event() Tato metoda provede zpracování události, která se odehrála nad změněným souborem. Činnost, kterou třída provede, je zjištění, který soubor byl změněn, a do kterého okruhu souborů patří. Tento záznam přidá pomocí další metody **addEntry()** do seznamu záznamů, u kterých byla zjištěna nějaká událost. Dále tato akce volá metodu **resetTimer()** třídy **TimeEater**.

addEntry() Touto metodou je předán záznam z okruhu souborů, na kterých se odehrála sledovaná událost. Přidání záznamu do seznamu se provádí při každé ze sledovaných událostí. Těchto událostí může být velké množství. Během přidávání záznamu je přístup k seznamu uzamčen pomocí zámku.

resetTimer() Metoda třídy **TimeEater**, která nastaví časovač třídy na přednastavenou hodnotu. Třída poté provádí odpočet, během kterého čeká na další příchozí události.

isEmpty() Metoda, kterou si třída zjišťuje, jestli jsou k dispozici nějaké záznamy, které je třeba zpracovat.

getHeap() Touto metodou třída TimeEater převezme všechny záznamy, které vyprodukovalo sledování událostí souborů. Po dobu vybírání seznamu záznamů je přístup k seznamu z třídy **HeapOfEntries** uzamčen pomocí zámku.

start() Metoda třídy snapshot, která provede zálohování souborů do úložiště SCM. Zálohování je provedeno pro odpovídající záznam.

endWatch() Touto metodou se ukončuje činnost celé služby. Metoda je volána jako reakce na signál, který je poslán hlavnímu procesu, běžícímu jako proces „na pozadí“. Tímto signálem je službě oznámeno, že se má ukončit.

join() Tato metoda je děděna z třídy **Thread**. Umožňuje počkat na vlákno, než ukončí svou činnost. V rámci tohoto ukončování vlákno předá třídě **Snapshot** záznamy, pokud nějaké čekají na zpracování.

3.12 Úložiště

Vytvořené snapshoty budou ukládány do systému správy verzí. Tento systém je v textu označován SCM, jak je již uvedeno v předchozí části vysvětlení pojmů 3.1.18. SCM používá pro uložení rozdílných verzí repositáře. Podrobnější popis se nachází v části 3.1.12.

V navržených nástrojích je využita funkčnost SCM pro ukládání a obnovování snapshotů. Dalšími využitými funkcemi jsou možnosti prohlížení uložených revizí, případně jejich obnova do stavu, v jakém byly snapshoty pořízeny. SCM poskytuje API, které tyto operace umožňuje. K zálohám je pak možné přistupovat buď pro obnovu, nebo případné zobrazení rozdílů.

Ukládání snapshotů se provádí spolu s textovým popisem a datem vytvoření. V textovém popisu je uveden údaj obsahující název záznamu z konfigurace, do kterého soubory uložené v tomto snapshotu spadají. Dále je v popisu uveden nástroj, který tento snapshot vytvořil. V případě, že byl snapshot vytvořen manuálně uživatelem, může obsahovat textovou zprávu uživatele.

Popis může obsahovat údaj, že se jedná o snapshot vytvořený během aktualizace archivního adresáře. V popisu je pak uveden údaj o přidání, případně odebrání souborů z okruhu souborů sledovaných pomocí SCM.

Dalším údajem jsou soubory obsažené v záloze. Tyto soubory jsou obsaženy v seznamu. Soubory jsou uloženy do snapshotu pouze pokud obsahují změny oproti předchozímu snapshotu. Aby SCM mohlo provádět toto rozlišení, potřebuje mít soubory uvedené v seznamu sledovaných souborů. Do tohoto seznamu sledovaných souborů jsou přidávány, případně odebrány během aktualizace archivního adresáře, která se provádí při spuštění většiny nástrojů.

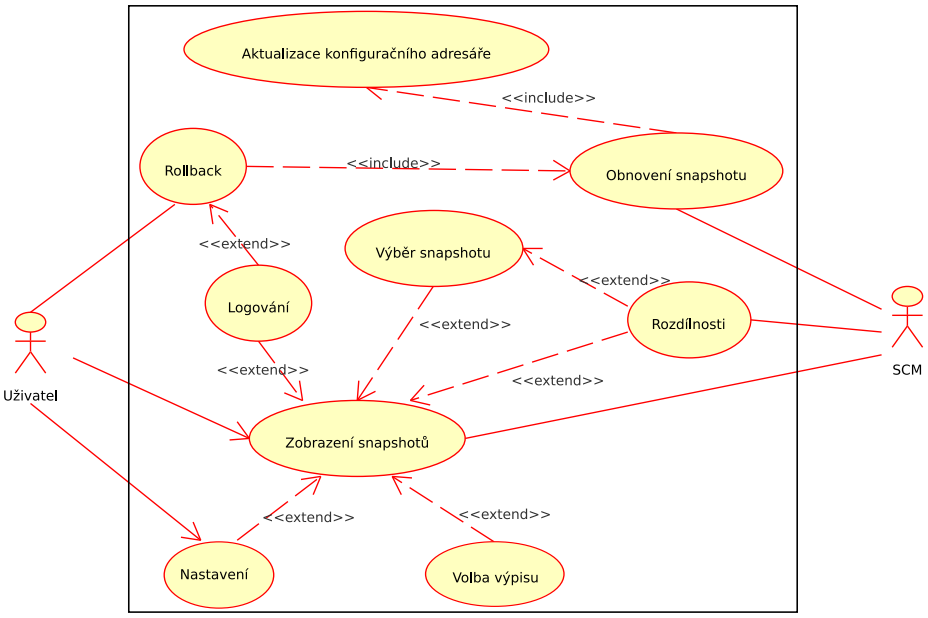
Všechny snapshoty v záloze jsou řazeny podle čísla revize. Každá revize má svůj jedinečný identifikátor. Revize společně s identifikátorem dávají dohromady dvojici, podle které je možné se odkázat na konkrétní snapshot. Identifikace umožňuje předat nástroji informaci, který snapshot má být obnoven, případně zobrazen.

Funkce, které poskytuje úložiště, jsou dostupné přes třídy z balíku SCM. Balík obsahuje několik modulů, které umožňují přistupovat k API systému správy verzí. Tento balík je součástí aplikace „snaproll“. V diagramech jsou tyto třídy pojmenovány tak, že součástí názvu je označení „Scm“. Dále je u těchto tříd naznačeno, že jsou součástí balíku SCM.

Balík SCM je navržen tak, aby bylo možné v případě potřeby tento balík nahradit dalším, který by přes stejně pojmenované metody poskytoval stejnou funkčnost na jiném systému správy verzí. V nahrazujícím balíku SCM by se změnily pouze odpovídající příkazy pro API, například by byl změněn princip uvedený v části 4.3.3.

3.13 Návrh nástrojů umožňujících prohlížení a obnovení snapshotů

V této sekci je popsán návrh požadavků na prohlížení a obnovu uložených snapshotů. Prohlížení dostupných snapshotů, uložených pomocí nástroje, vytvoření snapshotu popsaného v předchozí části textu 3.10. Pořípadě službu na sledování změn v souborech 3.11. Návrh těchto nástrojů je zobrazen v diagramu případů užití 3.10.



Obrázek 3.10: Diagram případů užití nástrojů, prohlížení a obnovy snapshotů.

Způsob, jak zobrazit obsah repositáře, verze uložených snapshotů, bude výpis do konzole. Tímto způsobem by mělo být možné získat informace o snapshotech v úložišti. Výsledný nástroj bude umožňovat spuštění s parametry pro nastavení požadovaného výstupu. Těmito parametry budou vlastnosti výstupu a označení snapshotů, které chce uživatel zobrazit. Vstupní parametry, které uživatel může použít při spuštění nástroje jsou popsány v další části 3.14.2.

Jednou z možností, kterou tento nástroj bude poskytovat, je zobrazení obsahu vybraného snapshotu. V tomto případě bude nutné, aby uživatel zadal, o který soubor a revizi má zájem. Nástroj s těmito parametry vypíše obsah vybraného souboru ve verzi, kterou uživatel zadal se vstupním parametrem.

Další možnost, kterou tento nástroj nabízí je zobrazení rozdílu stejného souboru mezi různými verzemi. V tomto případě uživatel zadá soubor, o který má zájem a revize snapshotů, z kterých mají zobrazené rozdíly vycházet. Toto porovnání se provede pomocí externího nástroje určeného pro zobrazování rozdílů. Více informací se nachází v části 3.14.

Rozšířením předchozí funkčnosti je zobrazení rozdílu mezi klientskými adresáři, které nástroj provede podobně, jako v předchozím případě. To znamená pomocí externího nástroje, který porovná soubor v různých verzích. V tomto případě budou verze souboru umístěné v rozdílném umístění.

Další nástroj umožňuje obnovu vybraného snapshotu. Tato obnova je v textu označována jako rollback. Pojem je už vysvětlen v předchozí části 3.1.13. Po obnově budou soubory propojeny pomocí základní vrstvy do konfiguračního adresáře. Základní vrstva provede propojení souborů z archivního adresáře do konfiguračního, pokud toto propojení ještě neexistuje. Propojení je umožněno

pomocí pevných odkazů, popsaných v sekci 3.1.10. Základní vrstva, která provádí propojení a čtení nastavení. Tato základní vrstva je popsána v jedné z předchozích částí 3.9.

Pokud při obnově dochází k přepsání aktuálního souboru, přesouvá se do souboru s koncovkou `.orig`[11]. Na místo aktuálního souboru je umístěna obnovená verze souboru, která je propojena z archivního adresáře. Uživatel má tak možnost, v případě nevydařené obnovy souborů, navrátit se do stavu před obnovou. Jedná se v podstatě o jednokrokovou funkci zpět[12].

Nastavení nástrojů pro prohlížení a obnovu se nachází stejně jako je popsáno v předchozí části 3.4. Podstatné pro orientaci mezi soubory určenými pro zálohování jsou názvy záznamů v konfiguračním souboru. Příklad tohoto záznamu je uveden v příkladu 3.1.

Obdobně jako u předchozích nástrojů, je využíván chybový výstup, který je popsán v části 3.5. Dále je možné vypisovat textový výstup nástrojů jak do příkazové řádky, z které jsou nástroje spuštěny, tak do souboru s těmito zprávami.

3.14 Návrh nástroje pro zobrazení snapshotů

Tento nástroj umožňuje zobrazení dostupných revizí snapshotů. Snapshoty jsou vypisovány ve formátu, jehož příklad je uveden na výpisu 3.14.1.

Nástroj pro prohlížení umožňuje zobrazit výpisy, které uživatel upřesní pomocí vstupních parametrů. Parametry jsou vypsány v části 3.14.2 spolu s podrobnějším výpisem toho, co popisovaný parametr zobrazuje. Další možností, zobrazení obsahu souboru, případně zobrazení rozdílů verzí souboru nebo jejich sloučení, je provedeno také tímto nástrojem.

Další možnosti, jak zobrazit obsah repositáře, je vypsat revize uložené v klientských adresářích. Konfigurační záznamy[13]. Hlavní úlohou tohoto nástroje je zobrazovat snapshoty, které jsou uloženy v repositáři, a jaké soubory jsou uloženy v tomto repositáři.

3.14.1 Revize snapshotu

Jedna revize snapshotu je uvedena v následujícím příkladu. V tomto případě se jedná o záznam, který vychází z nastavení uvedeného v předchozím příkladu 3.1.

```
changeset:  30: 040dcee77524
date:   Sun Apr 20 19:12:50 2008
description:  EntryName: adresar/ exports Text: snapshot_create
files:
 /etc/adresar/dirlink
 /etc/adresar/linkzroot
 /etc/adresar/make.conf
 /etc/exports
```

Výše uvedená revize obsahuje informace o snapshotu. Vypsaná revize obsahuje následující informace:

- Pořadové číslo zálohy na označeném řádku `changeset:` spolu s jedinečným označením tohoto snapshotu. Oba údaje je možné využít při odkazování na konkrétní snapshot, například při obnovování tohoto snapshotu.

[11] Jedná se o standardní chování při přepisování nějakého souboru. Podobné zálohy vytváří většina textových editorů. Obvyklé je přidání znaku „ “ nakonec názvu souboru.

[12] Obdoba příkazu zpět ve většině oblíbených textových editorů.

[13] Jsou to konfigurační záznamy, které jsou uloženy v repositáři. Například u klientských adresářů není jisté, jaké nastavení je v konfiguračním souboru.

- Dalším údajem je datum a čas vytvoření snapshotu. Datum je uveden ve formátu, který je popsán v dokumentu [8].
- Řádek označený `description:` obsahuje dva údaje. První z nich označený `EntryName:` uvádí, do kterého konfiguračního záznamu tento snapshot náleží. Druhý údaj `Text:` uvádí, že snapshot byl vytvořen za pomoci nástroje `snapshot_create` manuálně uživatelem z příkazové řádky.
- Posledním údajem jsou výpisy souborů, které byly uloženy v tomto snapshotu. Názvy souborů jsou vypsané za klíčovým slovem `files:`. Je dobré poznamenat, že nástroje pro vytváření snapshotů popsaných v části 3.10, případně služba vytvářející zálohy 4.2, ukládají pouze změněné soubory.

3.14.2 Vstupní parametry

Vstupními parametry je umožněno uživateli upřesnit, jak bude vypadat výsledný výpis. Následný výpis parametrů by měl upřesnit, co bude který parametr vypisovat.

–sites Tento parametr umožňuje zobrazit dostupné adresáře klientů. V každém z těchto adresářů je repositář odpovídající zálohám z klienta.

–site *klient* Tímto parametrem se upřesní, s kterým adresářem klienta se bude pracovat.

–entries Tímto parametrem je umožněno vypisovat, jaké konfigurační záznamy jsou dostupné. Pokud je upřesněno, o kterého klienta se jedná, jsou vypsány konfigurační záznamy obsažené v adresáři tohoto klienta. Výpis je získán přímo z repositáře, což je výhodné u klientů, kde není dostupný konfigurační soubor.

–cat [*revize*] -f *soubor* Jednou z možností, jak zobrazit obsah konkrétní verze souboru. Dále je nutné upřesnit, která revize a soubor jsou požadovány. Výpis obsahu se provede přímo do konzole. Tato varianta výpisu je popsána v části 3.14.3. Pokud není revize upřesněna, nástroj vypíše aktuální verzi souboru.

–diff *revize* [*revize*] -f *soubor* Tímto parametrem je zobrazení rozdílů mezi různými verzemi souboru. Dále je nutné upřesnit revize a název souboru, které mají být porovnány. Po upřesnění názvu souboru a pouze jedné revize, je možné zobrazit rozdíl mezi aktuální verzí a verzí z revize. Zobrazení rozdílu je provedeno pomocí externího nástroje. Tyto nástroje umožňují zároveň provedení sloučení rozdílů. Více lze nalézt v sekci 3.14.4.

–sitefiles *klient* [*klient*] Parametr, který umožňuje zobrazit rozdíl mezi dvěma klienty. Pokud je uveden pouze jeden klient, je jako druhý klient použit lokální počítač. Pak je možné porovnávat klienta a lokální počítač. Způsob, jakým je rozdíl zobrazen, je popsán v následující části 3.14.5.

–sitediff *klient* [*klient*] -f *soubor* Pomocí tohoto parametru je možné zobrazit rozdíl mezi dvěma soubory, které jsou umístěny v stejné adresářové cestě dvou vybraných klientů. Pokud je vybrán pouze jeden klient, je zobrazen rozdíl mezi lokálním počítačem a vybraným klientem.

Zobrazené rozdíly je možné editovat a tím zapsat do archivních adresářů klientů. V případě lokálního umístění je soubor zapsán do konfiguračního adresáře.

Další možností výpisu jsou jednotlivé revize a soubory obsažené v repositáři. Tyto výpisy je možné dále filtrovat, pokud je upřesněn konfigurační záznam, případně soubor, který chce uživatel sledovat.

–revs S tímto parametrem je možné zobrazit revize obsažené v repositáři. Příklad, jak tato revize vypadá, je uveden na výpisu 3.14.1. V případě, že není upřesněn některý z klientských adresářů, je vypisován obsah odpovídající lokálnímu umístění.

–files Po zadání tohoto parametru jsou zobrazeny pouze soubory obsažené v repositáři bez dalších informací.

Filtry, které je možné použít pro filtrování výpisů revizí a souborů, jsou uvedeny níže.

–entry *záznam* Tímto výpisem je možné upřesnit jeden konkrétní konfigurační záznam pro zobrazení. V uvedeném příkladu 3.14.1 by tímto záznamem byl `'adresar/ exports'`.

–file *soubor* Při upřesnění souboru jsou zobrazovány pouze snapshoty obsahující tento soubor.

–sync Tímto parametrem se zapíná možnost vypisovat aktualizace repositáře, to znamená revize, v kterých jsou soubory pouze přidány do sledování pomocí SCM. Při běžném použití nejsou tyto revize zobrazovány.

–rem Poslední parametr umožňuje zobrazit revize, v kterých byly odebrány soubory z archivního adresáře. Tyto revize nejsou standardně zobrazovány.

Další parametry, které je možné použít u tohoto nástroje jsou již popsány v části 3.4.1. Pomocí těchto parametrů je možné nastavit chování nástroje jako konkrétní soubor pro chybový výstup atd.

3.14.3 Zobrazení obsahu souboru

Obsah souboru v revizi, zadané uživatelem, je jednou z možností, jak zobrazit obsah souboru. Obsah se vypisuje přímo do konzole, z které byl příkaz spuštěn. Toto by mělo umožnit uživateli další práci s tímto výstupem. Například použít na výstup nástroj `grep(1)`.

3.14.4 Zobrazení a sloučení rozdílů

Zde je popsáno, jakým způsobem je provedeno prohlížení rozdílů v různých revizích souboru. V tomto zobrazení a prohlížení jsou využity externí nástroje. Externí nástroj je použit vzhledem k filozofii, která panuje jako konvence pro OS GNU/Linux nástroje. To znamená používat jednoduché nástroje, z nichž každý provádí svůj okruh činností [19]. Nástrojů, vykonávajících tuto činnost, je dostupných několik. Některé z nich jsou uvedeny v implementační části 4.4. Tyto nástroje umožní zobrazit rozdíl mezi různými verzemi jednoho souboru. V případě zobrazení rozdílů pro celý adresář by některé externí nástroje pro zobrazení rozdílů mohly způsobit obtíže. Tento nástroj je možné nastavit v konfiguraci 3.4.

Většina nástrojů pro zobrazení rozdílů umožňuje sloučení rozdílů, případně přímo editaci výstupního souboru. Tento soubor je po uložení umístěn jako aktuální verze souboru. Zálohování tohoto souboru není přímo řešeno. Zálohování se provede v případě, pokud je automaticky prováděno některým z popisovaných nástrojů v části 3.2. Případně může uživatel zálohování provést sám, manuálně.

Nastavení, jaký nástroj použít, se provádí v konfiguračním souboru `/etc/snaproll.conf`. Nastavení je provedeno pomocí parametrů „diffprogram“ a „diffoption“. Parametrem „diffoption“ je

možné předat programu pro zobrazování rozdílů další vstupní parametry externího nástroje, například:

```
#diffprogram kdiff3
diffprogram vimdiff
diffoptions ""
```

3.14.5 Porovnání souborů mezi dvěma klienty

Porovnání je zobrazeno jako strom souborů a adresářů, který obsahuje položky, shodné s umístěním a názvem souboru. Zobrazení je provedeno jako jeden adresářový strom, který obsahuje překrytí obou klientských adresářů.

U těchto souborů je pomocí označení `==` a `!=` označeno, jestli se soubory shodují i obsahem. Pokud se tyto soubory obsahem neshodují, je možné porovnat jejich obsah pomocí parametru `--sitediff`.

Soubory nacházející se pouze u jednoho klientského stromu, jsou označeny `<<` případně `>>`. Posledními značkami, které popisovaný výpis používá, jsou `D>` nebo `D>`. Tyto značky označují, podobně jako u souborů, že se položka nachází pouze v jednom adresářovém stromu. V tomto případě se jedná o adresáře. Pokud jsou adresáře pouze v jednom stromu jejich obsah není v stromu zobrazen.

3.14.6 Diagram tříd

Na následujícím diagramu tříd 3.11 jsou naznačeny třídy nástroje pro prohlížení snapshotů.

Následující výpis popisuje význam jednotlivých tříd z diagramu 3.11.

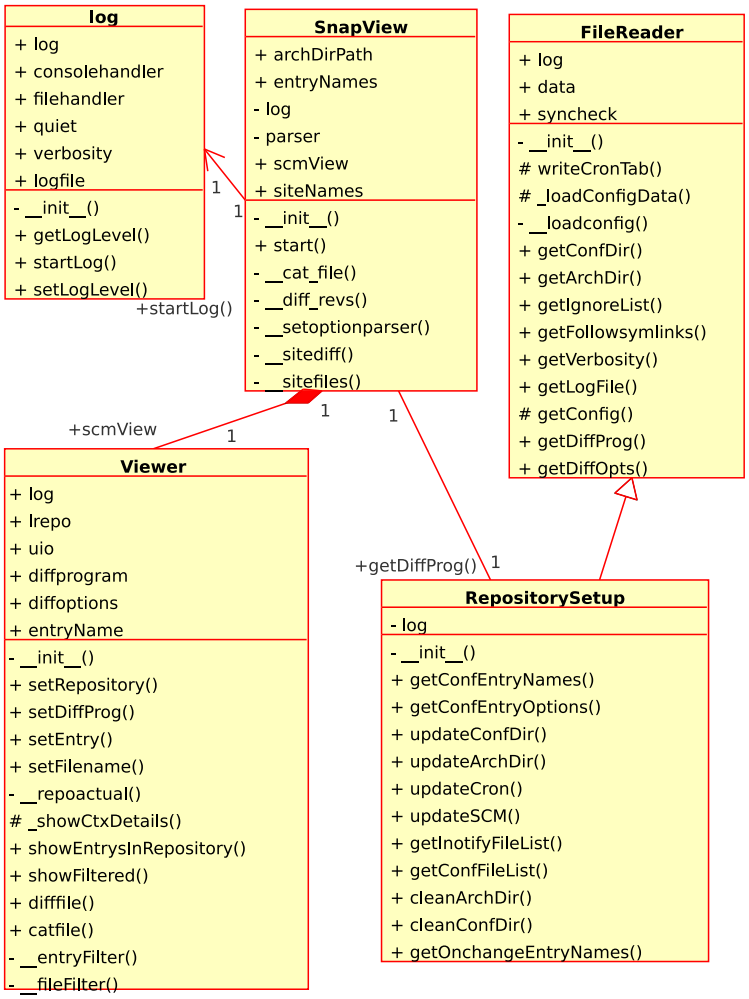
SnapView Je hlavní třídou, která provádí zobrazení obsahu úložiště. Tato třída provede spuštění a nastavení chybového výstupu. Provede vytvoření dalších potřebných tříd. Zpracuje vstup od uživatele a podle zadaných parametrů provede nastavení dalších tříd. Zpracovaný výstup je dále předán další třídě **Viewer**.

Viewer Tato třída provádí samotné čtení repositáře SCM. Z nastavení, která byla provedena třídou **SnapView**, případně dalších předaných parametrů provede výpis vyžádaných revizí, případně obsahů souborů. Dále umožňuje spuštění externího nástroje pro zobrazení a sloučení rozdílů. Toto je popsáno v části 3.14.4.

log Třída provádějící nastavení chybového výstupu. Toto nastavení přechází na všechny další třídy, které budou spuštěny při běhu zobrazování.

RepositorySetup Čtení nastavení je provedeno touto třídou. V tomto případě je získáno umístění repositáře a dále nastavení externího programu zobrazování rozdílů. Metody provádějící čtení jsou děděny z další třídy **FileHandler**. Na některých dalších diagramech tříd není tato třída pro přehlednost uvedena, nicméně zdědění třídy se provádí vždy.

FileHandler Tato třída poskytuje metody třídě **RepositorySetup** pro čtení konfiguračního souboru `/etc/snaproll.conf`.



Obrázek 3.11: Diagram tříd nástroje pro prohlížení snapshotů.

3.14.7 Sekvenční diagram

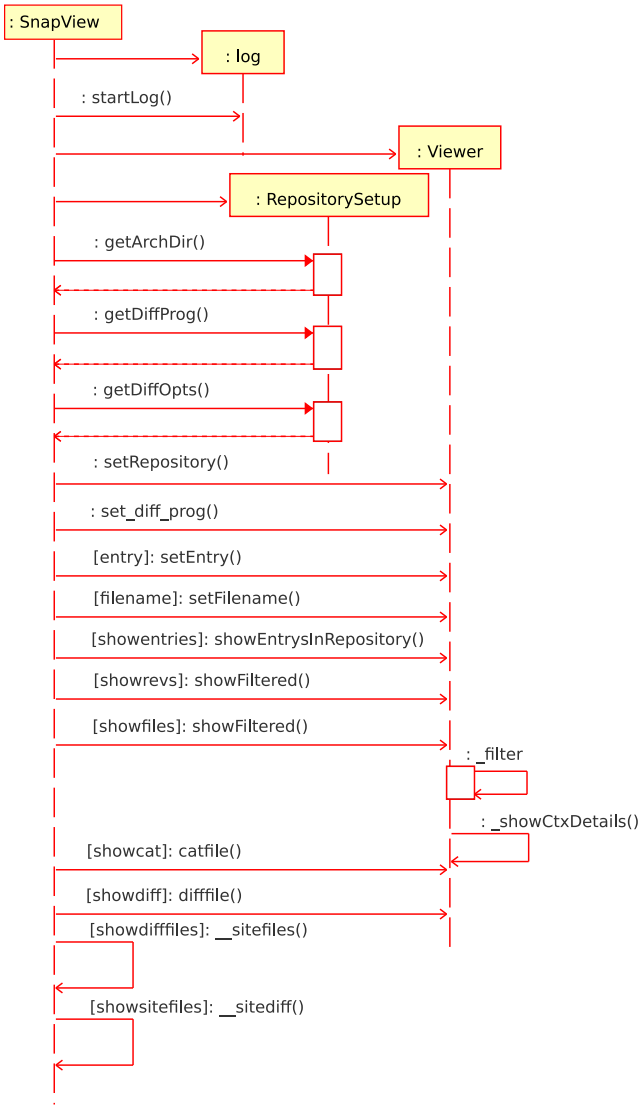
Posloupnost inicializací a volání jednotlivých metod pomáhá objasnit následující sekvenční diagram 3.12. V tomto diagramu je uvedeno, jakým způsobem je provedeno zobrazení filtrovaného výpisu. Následuje výpis uvedených metod. Průchod sekvenčním diagramem se podstatně neliší u jednotlivých výpisů. Konkrétně se jedná o volané funkce výpisu třídy **Viewer**.

Následující výpis objasňuje účel metod použitých v diagramu 3.12.

startLog() Tato metoda provádí nastavení chybového výstupu nástroje. Provedené nastavení přechází na další třídy v diagramu.

getArchDir() Touto metodou se získává umístění archivního adresáře. V umístění je zároveň uvedeno, o který klientský adresář se jedná. Pokud uživatel žádný z těchto adresářů neuvedl, jedná se o adresář s repositářem z lokálního počítače. Struktura těchto adresářů je popsána v části 3.2.

getDiffProg() Metoda, kterou je získán externí nástroj pro zobrazení rozdílů.



Obrázek 3.12: Sekvenční diagram nástroje pro prohlížení snapshotů.

getDiffOpts() Získání dodatečných parametrů pro externí nástroj.

setRepository() Nastavení třídy **Viewer** na požadovaný repositář, odpovídající klientskému adresáři s repositářem.

setDiffProg() Touto metodou je předán spouštěcí soubor externího nástroje a jeho nastavení třídě **Viewer**.

setEntry() Tato metoda nastavuje záznam, na který má být výstup filtrován. Volání je podmíněno zadáním tohoto záznamu uživatelem.

setFilename() Nastavení souboru pro filtrování, pokud uživatel tento soubor nastaví.

showEntrysInRepository() Zobrazí záznamy z konfigurace uložené v repositáři. Jedna z možností spustitelná uživatelem.

showFiltered() Zobrazí filtrovaný výpis. Vypisuje podle požadavků uživatele dostupné revize, případně pouze uložené soubory.

filter() Samotné provedení filtrování podle požadavků uživatele.

_showCtxDetails() Zobrazí detaily o uložených záznamech, které nebyly odfiltrovány. Záznamy jsou zobrazeny způsobem naznačeným v předchozím příkladu 3.14.1.

catFile() Umožňuje zobrazit obsah souboru, buď z některé revize, nebo z aktuálních souborů.

diffFile() Zobrazí rozdíl mezi dvěma verzemi souboru z repositáře. Případně umožňuje zobrazit a vytvořit sloučení rozdílů mezi aktuálním souborem a jeho vybranou revizí. Tato činnost je umožněna externím nástrojem pro zobrazení rozdílů.

__siteFiles() Jedná se o metodu, která porovnává stromy souborů mezi dvěma klientskými adresáři. Výstup je zobrazen pomocí značek, které rozlišují, jestli se soubor nachází v obou, případně pouze v jednom klientském adresáři. U souborů, které se nacházejí v obou adresářích, je porovnán jejich obsah. Pokud se obsah liší, je možné ho porovnat pomocí dalšího parametru, který využívá metodu **__siteDiff()**.

__siteDiff() Metoda umožňující zobrazit rozdíl mezi soubory se stejným jménem a umístěním v rozdílných klientských adresářích. Soubory použitelné pro tento výstup je možné získat z výstupu metody **__siteFiles()**.

3.15 Návrh nástroje umožňujícího obnovu snapshotů

V této části je uveden návrh na nástroj, který umožňuje obnovení snapshotu do stavu, v jakém byl pořízen. Tento nástroj umožní uživateli pomocí vstupních parametrů obnovit vybraný snapshot. Výběr požadovaného snapshotu bude umožňovat nástroj popsaný v předchozí sekci 3.14. Podle údajů zobrazených tímto nástrojem bude možné snapshoty obnovit. Zobrazované údaje jsou uvedeny v příkladu 3.14.1. Údaje, podle kterých bude možné specifikovat konkrétní revizi jsou:

- číslo revize
- jedinečné označení revize

Upřesnění, kterou revizi snapshotu obnovit a případně, které soubory budou v této obnově, je možné specifikovat pomocí vstupních parametrů. Je tedy možné, pokud je třeba obnovit nějaký soubor, který se v konfiguračním adresáři nenachází, specifikovat konkrétní revizi s tímto souborem. Dále je možné obnovit pouze požadovaný soubor, pokud je tento zadán, jako vstupní parametr. Vstupní parametry jsou popsány v části 3.15.1.

Pokud nástroj během obnovy přepíše některý ze souborů, které se v konfiguračním adresáři nacházejí, je tento soubor uložen do zálohy přímo v konfiguračním adresáři. Záložní soubory mají stejné jméno, za kterým je přípona `.orig`. Tyto soubory mají stejný obsah, jako soubor před provedením obnovy. Jedná se o možnost, jak zvrátit provedenou obnovu. Soubory s příponou `.orig` nebudou podle přednastavení ignorovány. Aplikace poskytuje seznam ignorovaných souborů v konfiguračním souboru. Proto je ponecháno na uživateli, jestli chce tyto soubory přidat do seznamu ignorovaných souborů.

3.15.1 Vstupní parametry

Vstupní parametry, pomocí kterých je možné nastavovat chování nástroje, jsou následující:

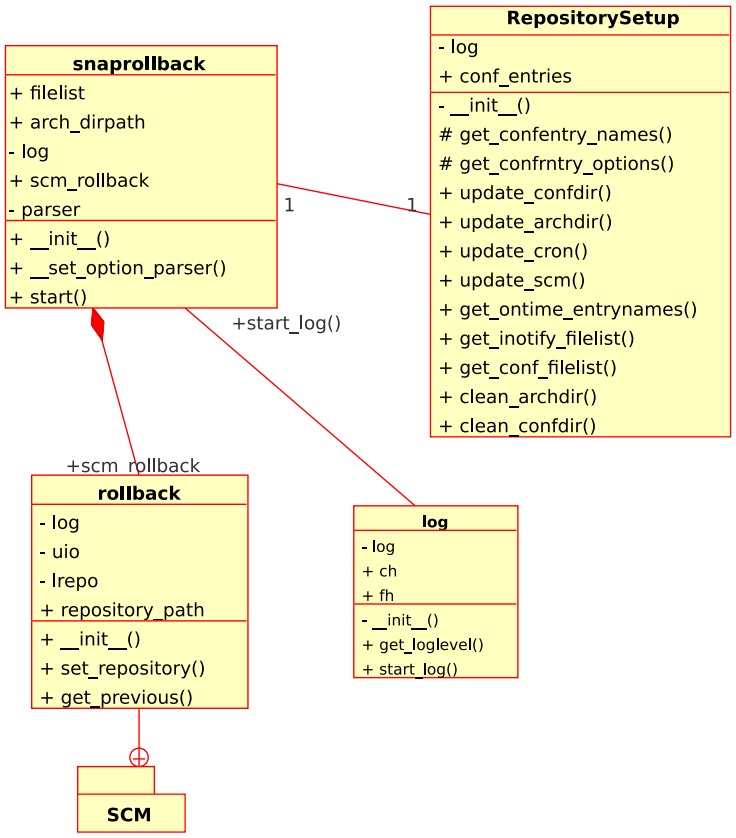
–rev ***revize*** **[-f** ***soubor*****]** Číslo *revize* snapshotu, který je určen pro obnovu. Pokud je upřesněn nějaký soubor, nebudou obnoveny všechny soubory, které jsou v této revizi snapshotu uloženy. Bude obnoven pouze tento specifikovaný soubor.

–site *klient* Při zadání konkrétního klienta je obnova provedena pouze v upřesněném klientském adresáři. Obnovené soubory nejsou poté přenášeny do konfiguračního adresáře.

Dále je možné použít vstupní parametry, které jsou běžně dostupné pro všechny nástroje. Tyto parametry jsou popsány v části 3.4.1. Pomocí dalších parametrů je možné přenastavit umístění konfiguračního souboru, změnit umístění logovacího souboru atd.

3.15.2 Diagram tříd

Na následujícím diagramu 3.13 je možné vidět třídy, z kterých je utvořen nástroj pro obnovu snapshotů, to znamená nástroj umožňující rollback. V následujícím výpisu je popis těchto tříd, z kterých se nástroj sestává.



Obrázek 3.13: Diagram tříd nástroje pro obnovu snapshotů.

SnapRollback Toto je hlavní třída nástroje. Provádí nastavení a spuštění chybového výstupu pro další třídy. Zpracovává vstup od uživatele, případně vytváří seznam souborů určených pro obnovu.

Rollback Třída provádějící samotnou obnovu souborů. Obnova je provedena přímo funkcí API, která tuto obnovu umožňuje. Třída je součástí balíku SCM, který obsahuje třídy pro práci se SCM.

RepositorySetup Třída reprezentující nastavení z konfiguračního souboru. Nastavení je čteno pomocí děděných funkcí z další třídy. Na diagramu tříd je toto zobrazeno na 3.11. Na diagramu je tato skutečnost pro přehlednost vynechána.

log Třída provádějící nastavení logovacího výstupu. Umožňuje nastavit výřečnost a umístění logovacího souboru. Dále zapíná, případně zakazuje výstup do příkazové řádky, z které byl nástroj spuštěn.

DirectoryWrite Je vytvořena jedna instance této třídy, která umožní zapsat obnovené revize snapshotů do konfiguračního adresáře a tímto způsobem aktualizovat konfigurační adresář. Zápis do konfiguračního adresáře se provádí pouze v případě, že se jedná o zálohy z lokálního počítače.

3.15.3 Sekvenční diagram

Posloupnost provádění jednotlivých metod je uvedena v sekvenčním digramu 3.14. V tomto diagramu je uveden průchod metodami, v případě že uživatel zadal záznam z konfigurace. V tomto případě se průchod liší od zadaného souboru metodou **get_conf_filelist()**. Při zadaném souboru není nutné získávat seznam souborů pro obnovu.

V následujícím výpisu jsou popsány jednotlivé metody uvedené v předchozím diagramu.

startLog() Touto metodou je nastaven chybový výstup. Nastavení výřečnosti, případně souboru pro chybový výstup.

getArchDir() Metoda vracející umístění archivního adresáře. Tento údaj je získán pomocí třídy **RepositorySetup**.

setRepository() Touto metodou je nastaven repositář, z kterého bude získána revize snapshotu. Repositář je určen umístěním archivního adresáře a vybraným klientským adresářem, respektive lokálním adresářem.

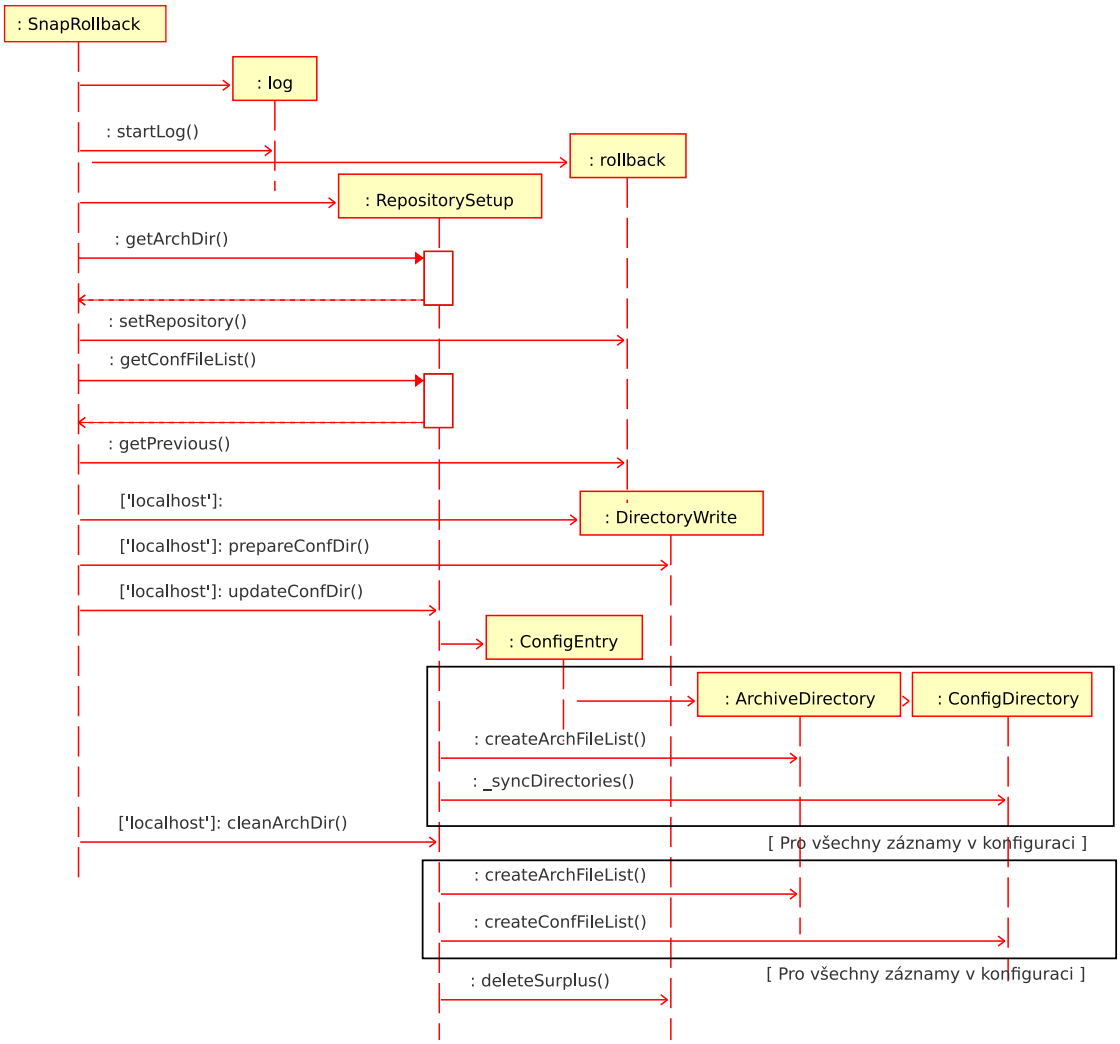
getConfFileList() Touto metodou je z třídy RepositorySetup získán seznam souborů, pokud je zadán záznam z konfigurace. Pokud uživatel zadá soubor pro obnovu, není nutné tento seznam získávat, je dodán uživatelem.

getPrevious() Touto metodou se provádí samotné obnovení zadané revize snapshotu.

prepareConfDir() Metoda, s pomocí které je provedena příprava konfiguračního adresáře na aktualizaci z archivního adresáře.

updateConfDir() Touto metodou je provedeno propojení do konfiguračního adresáře. Obnova se provádí pouze v případě, že se jedná o repositář na lokálním počítači.

createArchDirList() Metoda vytvářející seznam souborů v archivním adresáři.



Obrázek 3.14: Sekvenční diagram nástroje pro obnovu snapshotů.

_syncDirecotories() Touto metodou je provedena aktualizace konfiguračního adresáře. To znamená vytvoření pevných odkazů do konfiguračního adresáře.

createConfDirList() Vytvoření seznamu souborů, nacházejících se v konfiguračním adresáři.

deleteSurplus() Mazání souborů, které neodpovídají konfiguračnímu adresáři případně nastavení nástroje. Toto čištění archivního adresáře se provádí hlavně kvůli souborům s koncovkou `.orig`. Neboť v tímto způsobem je možné smazat tyto soubory pokud jsou uvedeny v seznamu ignorovaných souborů.

Kapitola 4

Implementace

V této kapitole popisuji, jakým způsobem jsem implementoval předchozí návrh. V následujícím textu 4.1 vysvětluji svoji volbu programovacího jazyka Python.

V další části je popis podsystému jádra OS inotify 4.2. Tento podsystém jádra se stará o sledování zápisů do vybraných souborů.

Dále popisuji řešení úložiště aplikace 4.3. V této části je také zdůvodněna volba systému spravující verze záloh.

4.1 Jazyk Python

Tento programovací jazyk jsem vybral na základě předchozích zkušeností s tímto programovacím jazykem. Většinu zkušeností jsem získal při práci na školních projektech. Tento jazyk vždy usnadňoval dosažení zadání. Proto jsem zvolil jazyk Python pro tento projekt.

Na tomto programovacím jazyku oceňuji vlastnosti, s kterými byl tento jazyk navržen. Na výsledném kódu je vidět jeho jednoduchost a čitelnost. Při čtení kódu je rychle zřejmé, jak úsek kódu pracuje. Čitelnost kódu značně usnadňuje samotné psaní kódu. Jedna z vlastností je absence složených či jiných závorek oddělujících jednotlivé bloky kódu. Oddělení těchto bloků je u tohoto jazyka řešeno odsazením celého bloku. Tímto odsazením se zvyšuje přehlednost a zároveň napomáhá tuto přehlednost dodržovat.

Další hezkou vlastností je modularita jazyka [27]. Pro tento jazyk je dostupných mnoho modulů umožňující širokou paletu činností. Některé z těchto modulů používám i v tomto projektu. Přehled použitých modulů je uveden v příloze A.2 spolu se stručným popisem funkcí, které poskytuje. V projektu používám i moduly, které nejsou součástí standardní instalace jazyka Python. Jedná se o modul umožňující práci s podsystémem jádra `inotify(7)`. Tento modul je popsán v další části 4.2.1.

4.1.1 Čekání na poslední událost

Modul threading umožňuje využití vláken v programovacím jazyce Python. V tomto projektu si použití vlákna vyžádalo sledování změn na souborech. Událostí, které jsou na souborech sledovány je několik, jejich seznam a popis je v následující sekci 4.2. Je třeba sledovat každou z uvedených událostí. Zároveň, pokud dojde k nějaké složitější události na sledovaném souboru, je těchto událostí vytvořeno hned několik. Pokud je sledován celý adresář je počet těchto událostí ještě znásoben. Na druhou stranu vytváření záloh pro každou tuto změnu by znamenalo velké množství po sobě jdoucích snapshotů ze stejného umístění.

Nástroj sledování změn na souborech `snap_notifier(8)` proto obsahuje jednu třídu s vláknem, které provádí čekání na další možnou událost. Čekání je provedeno pomocí modulu **threading**. Při vytváření této části kódu jsem vycházel z článku [13]. V následujícím příkladu 4.1 je naznačena funkce, která zmiňované čekání provádí. Princip vychází z předpokladu, že nástroj sleduje konfigurační soubory.

```
1  if not heap.isEmpty():
2      while timer > 0:
3          time.sleep(1)
4          timer -= 1

6          lock.acquire()
7          entries = heap.getHeap()
8          lock.release()

10         for entry in entries:
11                 Snapshot.start(entry)
```

Příklad 4.1: Část kódu provádějící čekání na poslední událost.

Na řádku 1 je provedena kontrola dostupnosti nějakého záznamu pro zpracování. Pokud je nějaký záznam dostupný, vlákno začne postupně snižovat hodnotu **timer**. Tato proměnná je nastavena na hodnotu[1], která je uvedena v souboru popisovaném v předchozí části o nastavení 3.4.5. Proměnné **timer** je znovu přiřazena hodnota pokud `inotify(7)` předá další událost. Nastavení této proměnné provádí hlavní proces spolu s přidáním záznamu do seznamu **heap**. Přidaný záznam odpovídá souboru nebo adresáři u kterého `inotify(7)` nahlásilo tuto událost. Záznam je do seznamu přidán pouze v případě, že už se v tomto seznamu nenachází. Po poslední příchozí události, po uběhnutí prodlevy na řádku 9, pokračuje vlákno dál získáním všech záznamů ze seznamu **heap** na řádku 7. Během získávání seznamu je přístup k tomuto seznamu chráněn pomocí zámku. Podobně je chráněno i přidávání záznamu hlavním procesem do seznamu **heap**. Nakonec jsou na řádku 11 ze získaného seznamu záznamů vytvořeny zálohy pomocí třídy **Snapshot**. Před koncem služby `snap_notifier(8)` jsou provedeny zálohy záznamů, pokud nějaké čekají na zpracování.

4.2 Inotify

Inotify pracuje na principu sledování i-uzlů[2]. Sledování i-uzlů poskytuje velice dobrou granularitu[3]. Umožňuje sledovat jak soubory, tak adresáře na předem definované události. Tyto události jsou vypsány v následující tabulce 4.1.

Při práci s adresářem je možné sledovat změnu v některém ze souborů, případně adresářů. Nahlášení změny se provádí pomocí souborového deskriptoru[4] Tento deskriptor slouží pro komunikaci s uživatelskými aplikacemi.

Inotify se používá pomocí sady systémových volání. Po změně některého ze sledovaných souborů inotify tuto událost oznámí. Zpracování této události je vysvětleno v další sekci 4.2.1. Události na souboru, které inotify oznamuje, jsou uvedeny v následující tabulce 4.1. Události jsou popsány

[1] Standardně nastavená doba čekání je 5 sekund.

[2] I-uzel (anglicky Inode) je datová struktura uchovávající všechny informace o souborech adresářích [24].

[3] Granularita je měřítko velikostí jednotlivých komponentů nebo popis těchto komponentů, které vytvářejí systém [23].

[4] Souborový deskriptor (anglicky File descriptor) je abstraktní klíč pro přístup k souboru. Přesněji je to číslo v tabulce otevřených souborů, která je umístěna v jádru operačního systému [22].

spolu s vysvětlením, při které operaci se souborem je toto hlášení oznámeno. Údaje v tabulce vycházejí z článku o podsystému jádra inotify [14].

IN_CREATE	soubor/adresář byl vytvořen v sledovaném adresáři
IN_DELETE	soubor/adresář byl smazán v sledovaném adresáři
IN_DELETE_SELF	sledovaný soubor nebo adresář byl smazán
IN_MODIFY	v sledovaném souboru byla uložena změna
IN_MOVED_SELF	sledovaná položka byla přesunuta
IN_MOVED_FROM	přesun souboru/adresáře ze sledovaného prostoru jinam
IN_MOVED_TO	přesun souboru/adresáře do sledovaného prostoru

Tabulka 4.1: Popis událostí použitých pro sledování konfiguračního adresáře.

Sledování souborů provádí služba `snap_notifier(1)`. Návrh této služby se nachází v předchozí části 3.11. Pokud služba obdrží oznámení o události, vytvoří snapshot, který bude obsahovat soubor nebo adresář, na kterém se tato událost odehrála.

Každá z těchto událostí je sledována, neboť může znamenat změnu v sledovaném prostoru konfiguračních souborů. Na druhou stranu je těchto událostí hlášeno vícero pro některé běžné operace se soubory nebo adresáři. V reálném použití je v případě, že byl do sledovaného prostoru zkopírován soubor, vyprodukováno větší množství událostí.

Služba proto obsahuje mechanismus čekání, který je po příchozí události nastaven na časovou prodlevu. Během této prodlevy služba čeká na další příchozí události. Na konci prodlevy jsou pak záznamy, do kterých sledované soubory patří, zálohovány.

Například při zkopírování jednoho souboru do sledovaného adresáře jsou postupně ohlášeny následující události:

```
IN_CREATE
IN_MODIFY
```

Pokud je nakopírován celý adresář souborů jsou tyto události hlášeny pro každý soubor. Všechny tyto události se budou pravděpodobně vztahovat k souborům spadajícím do jednoho okruhu záznamu z konfigurace. Je tedy žádoucí, aby bylo zálohování provedeno pouze jednou pro celý tento záznam.

Tato služba je implementována pomocí modulu, který je popsán v následující části 4.2.1.

4.2.1 Pyinotify

Pro rozšíření inotify je dostupný modul pyinotify, který obsahuje třídy pro sledování změn na souborech a jejich zpracování. Modul jazyka Python potřebuje ke své funkčnosti jádro OS GNU/Linux minimálně 2.6.13. Dalším požadavkem je balík dev-lang/python minimálně ve verzi 2.3. Pokud jsou tyto podmínky splněny, je možné používat tento modul.

Modul poskytuje třídy WatchManager a Notifier spolu s dalšími třídami, které jsou použity pro sledování souborů. Samotné sledování souborů je ve zkrácené podobě ukázáno v následujícím příkladu 4.2:

```
2 from pyinotify import WatchManager,
3                       Notifier,
4                       EventsCodes,
5                       ProcessEvent
```

```
7  WATCHED_EVENT= EventsCodes.IN_MODIFY | \
8                 EventsCodes.IN_CREATE

10 class ProcessEH(pyinotify.ProcessEvent):

12     def process_IN_MODIFY(self, event):
13         addEntryToHeap()

15 wm = WatchManager()

17 notifier = Notifier(wm, ProcessEH())

19 wm.add_watch(watched_files, WATCHED_EVENT)

21 while(True):
22     notifier.process_events()
23     if self.notifier.check_events():
24         self.notifier.read_events()
```

Příklad 4.2: Část kódu provádějící přidání a samotné sledování souborů

V předchozím příkladu 4.2 je naznačen princip zpracování událostí na souborech. Na řádku 2-4 jsou přidány moduly z externího balíku **pyinotify**. Vytvoření masky sledovaných událostí je provedeno na řádku 7. Vytvoření instance třídy spravující sledované soubory je provedeno na řádce 15. Vytvořeni instance třídy sledující změny na souborech a přiřazení tříd pro zpracování těchto událostí **ProcessEH()** a třídy pro správu sledovaných souborů je na řádce 17. Na řádce 19 jsou pak přidány sledované soubory. Od této chvíle jsou události odehrávající se na souborech sledovány.

Samotné sledování poté probíhá v cyklu, na řádku 21. Pokud funkce na řádku 23 zjistí, že k nějaké události došlo, je předána do fronty čekání na zpracování příkazem na řádce 24. Zpracování událostí provede v předchozím příkladu třída **ProcessEH()**, které jsou předány příchozí události při volání funkce na řádku 22. Tímto zpracováním je volání zděděné funkce na řádku 12, která odpovídá příchozí události. Ta provede přidání záznamu do seznamu pro vytvoření snapshotu. Po prodlevě, po poslední události, jsou vytvořeny snapshoty záznamů uvedených v tomto seznamu.

Tato ukázka vychází z informací získaných na stránce projektu pyinotify [15]. Postup provádění této služby je zobrazen i na návrhu v sekvenčním diagramu 3.9.

4.3 Úložiště

Všechny snapshoty jsou ukládány do jednoho konkrétního umístění. Umístění, nebo-li úložiště, je tvořeno systémem správy verzí[5]. Tato skutečnost byla již zmíněna v části 3.1.18. Částečně byl význam systému popsán v části 3.12. V této kapitole je popsán výběr úložiště a zároveň, která implementace úložiště bude použita.

4.3.1 Požadavky na SCM

Sledování konfiguračního adresáře bude probíhat na lokálním počítači. Vytvořený repositář bude opět využit hlavně na lokálním počítači. Z těchto dvou důvodů je zvolen systém s distribuovaným modelem repositáře. Tento model je vytvořen tak, že repositář je umístěn ve stejném adresáři jako zálohovaná data. Zároveň je možné tyto repositáře přesouvat přímo spolu se zálohovanými soubory. Tento model je výhodný pro zamýšlené použití.

[5] Systém správy verzí je běžně označován jako SCM. Tento text není výjimkou.

Další model Klient-server je výhodnější pro centralizovaný vývoj zdrojových kódů, do kterého zapisuje velký počet přispěvatelů. U tohoto modelu je repositář umístěn na jednom serveru. Klienti pak přistupují na server přes síť [21]. Zároveň je nutná relativně složitá příprava repositáře pro ukládání souborů. Repositář je zároveň centralizován do jednoho umístění. Není proto příliš vhodný pro zálohování konfigurace.

Pokud je samotné SCM implementováno v jazyce Python. Zároveň je velice výhodné pokud je dostupné API pro jazyk Python. Toto programovací rozhraní se označuje API[6]. Toto API by mělo umožňovat řešit okruh činností potřebných pro práci se SCM. Dostupnost tohoto API je tedy jednou z hlavních vlastností při výběru.

Další vlastností, ke které jsem přihlížel, je rychlost uvedeného SCM. Rychlost těchto SCM vycházejí částečně z programovacího jazyka, ve kterém je uvedené SCM implementováno. Pro rychlost SCM je implementace v jazyce Python spíše nevýhodou, neboť například implementace v jazyce C je mnohem rychlejší. Dobrou volbou je pravděpodobně kompromis mezi těmito programovacími jazyky.

4.3.2 Dostupná SCM

V úvahu připadají systémy Bazaar, Git, Mercurial [21]. Všechny systémy jsou založeny na systému distribuovaného modelu repositáře. Zároveň je možné označit tyto systémy jako jedny z nejvíce diskutovaných v současné době.

Situace u jednotlivých SCM je popsána v následujícím výpisu. Výpis obsahuje tři uvedené SCM s distribuovaným modelem repositáře.

Git Za nejrychlejší SCM lze z těchto tří označit Git [11]. Zároveň zaměření tohoto SCM na textové soubory lze označit jako vhodné. Git [2] je původně navržen pro správu zdrojových souborů linuxového jádra. Git se sestává z mnoha malých nástrojů napsaných v programovacím jazyce C a několika skriptů, které slouží k ovládání těchto nástrojů.

Velkou nevýhodou tohoto SCM je absence API pro jazyk Python. Ovládání tohoto SCM by tedy bylo nutné řešit problematicky, spouštěním externích nástrojů. Vzhledem k dostupnosti tohoto API u ostatních uvedených SCM je zřejmé, že Git není vhodná volba.

Bazaar Dobrou vlastností tohoto SCM je kompletní implementace v programovacím jazyce Python. K tomuto SCM je zároveň dostupné i API v jazyce Python, které umožňuje ovládat SCM přímo importováním modulů zprostředkovávajících funkčnost tohoto SCM [5].

Nevýhodou tohoto SCM je jeho poměrně pomalá činnost [11]. Tato „pomalost“ je pravděpodobně způsobena skutečností, že i pro výkon podstatné části SCM jsou napsané v jazyce Python.

Marcurial Je SCM implementované v jazyce Python, přičemž některé kritické části kódu jsou implementovány v programovacím jazyce C. Tento způsob implementace se dá označit jako velice výhodný. Splňuje totiž kompromis dvou hlavních požadavků, které jsem již uvedl. Implementace pro rychlost důležitých částí v jazyce C umožňuje tomuto SCM poměrně rychlé odezvy. V některých případech je toto SCM označováno jako rychlejší než Git [16].

Mercurial zároveň obsahuje API pro jazyk Python. Dokumentaci k tomuto API lze nalézt přímo v zdrojových souborech, případně na stránkách projektu [4]. Toto API je poměrně jednoduché. Hlavní část API poskytuje stejnou funkčnost, kterou poskytuje SCM při použití z příkazové řádky. V případě potřeby je možné přistupovat i k nižším funkcím tohoto SCM.

[6] Rozhraní pro programování aplikací (anglicky Application Programming Interface).

Vzhledem k uvedeným vlastnostem vypadá SCM Mercurial jako nejlepší volba. Tento systém se dá považovat za dobrou volbu i do budoucna, neboť mnoho projektů přechází na tento systém správy verzí. Dá se tedy předpokládat, že uvedené SCM se bude dále rozvíjet a používat.

V dalších fázích projektu mohou být využita i další SCM jako rozšíření. V tom případě by odpadla nutnost použít pouze vybrané SCM. Toto rozšíření bude možné provést pomocí změny tříd v balíku SCM, která zprostředkovává funkčnost úložiště.

4.3.3 Přístup k repositáři

Struktura archivního adresáře vyplývá z jedné z podmínek SCM. Ta vyžaduje, aby soubory, s kterými SCM pracuje, měly stejný počáteční adresář jako SCM. Tímto společným počátečním adresářem je `/var/db/snaproll/localhost`. Na obrázku 3.2 je adresář `.hg` repositářem SCM Mercurial. V tomto společném adresáři se spouštějí operace SCM, které potřebují pro svoji funkci právě tento repositář. V případě používání programového API SCM se tento adresář uvádí jako jeden ze vstupů volané metody. Například pro uložení snapshotu do SCM pomocí následující sekvence kódu:

```
1  import os
2  from mercurial import ui, localrepo, commands

4  uio = ui.ui(debug = options.debug,
5                  verbose = options.verbose)

7  lrepo = localrepo.instance(ui = uio,
8                      path = "/var/db/snaproll/localhost/.hg",
9                      create = False)

11 commitFiles = [os.path.join("/var/db/snaproll/localhost", filepath) \
12             for filepath in FileList]

14 options['message'] = "snapshot_create"
15 options['addremove'] = True

17 commands.commit(uio, lrepo,
18                  *commitFiles,
19                  **options)
```

Příklad 4.3: Část kódu zapisující snapshot do repositáře

Instance repositáře je vytvořena na řádku 7. Při vytváření instance jsou potřebné následující vstupní hodnoty. Prvním vstupem je uživatelské rozhraní `ui`, které je vytvořeno na řádce 4, to umožňuje vypsat zprávu uživateli. Například informovat o tom, které soubory byly do repositáře přidány. Nastavením tohoto rozhraní se určuje výřečnost operací s repositářem, případně další komunikace s uživatelem. Dalším vstupem je cesta k archivnímu adresáři `/var/db/snaproll/localhost/.hg/` repositářem. Hodnota uvedená na řádku 9 určuje, jestli má být repositář vytvořen, v případě že ještě neexistuje.

Tato instance reprezentující repositář je použita spolu s uživatelským rozhraním `ui` jako vstupní parametry pro příkaz ukládající soubory do repositáře. Tento příkaz je v příkladu 4.3 uveden na řádce 17.

Při uložení jsou dále použity jako vstup seznam souborů určených pro vytvoření snapshotu. Na řádce 11 je k seznamu souborů určených pro uložení do repositáře připojena cesta do archivního adresáře. Tento seznam je pak předán jako vstupní parametr.

Posledním vstupem je nastavení operace `commit`. Nastavení je provedeno pomocí proměnné `options`. Tento slovník obsahuje parametry odpovídající vstupním parametrům při běžném spuštění nástroje `hg commit --message "snapshot_create" --addremove`. Podobným způsobem je možné přistupovat i k dalším funkcím modulu `commands` [4].

V případě vytváření tohoto snapshotu po události je v tomto textu uvedeno o jakou událost se jedná. V tomto případě tedy manuální uložení snapshotu. Soubory odpovídají seznamu, který je vytvořen z názvu konfiguračního záznamu a předán této metodě. Příklad takového záznamu je v předchozí části 3.1.

4.4 Nástroje zobrazující rozdíly mezi verzemi

Nabízí se hned několik externích nástrojů, tedy nástrojů, které nejsou součástí aplikace. Jejich princip je většinou stejný. Rozdíly jsou zobrazeny ve dvou oknech, přičemž v každém je jedna verze souboru. Rozdíly v jednotlivých řádcích jsou zvýrazněny podle toho, o který nástroj se jedná.

Spuštění tohoto nástroje je umožněno pomocí jedné z funkcí API, které poskytuje SCM. Tato funkce je po vložení všech potřebných parametrů, mezi které patří požadovaná revize souboru a dále také nástroj pro zobrazení rozdílů a jeho nastavení.

Některé z externích nástrojů, které je možné použít pro zobrazení rozdílů, jsou v následujícím seznamu:

Kdiff3 Umožňuje barevně zobrazit rozdíl mezi dvěma i více soubory. Rozdíly jsou výrazně vyznačené. Nástroj zobrazuje i rozdíly mezi jednotlivými znaky na řádku. Rozdíly jsou zobrazeny přehledně do dvou oken, případně lze nastavení změnit.

Nástroj umožňuje editaci rozdílů a jejich případné sloučení. Sloučení rozdílů lze provést velice snadno pomocí nástrojové lišty a kurzorem myši. Jednou z možností je automatické sloučení rozdílů[7].

Jedná se o nástroj pro správce oken KDE[8]. Nástroj má jednoduché a na ovládání intuitivní GUI [6].

VimDiff Tento nástroj umožňuje barevně rozlišit rozdíly mezi jednotlivými verzemi souboru. Rozdíly jsou zvýrazněny pro každý celý řádek. Obě verze jsou zobrazeny, každá v jednom okně, která jsou vedle sebe. Případně při změně nastavení[9], tato okna mohou být horizontálně nad sebou.

Sloučení verzí je možné pomocí přímé editace souboru. U tohoto nástroje může být matoucí skutečnost, že není zvýrazněno, které z oken je tím aktuálním souborem. Respektive, z kterého okna se změny uloží do aktuálního souboru. V mém případě to bylo vždy pravé okno, bohužel tato skutečnost nemusí být vždy stejná. V dolním stavovém pruhu je zobrazeno, který soubor je v cílovém umístění, a který je v přechodném adresáři `/tmp`.

Tento nástroj má i grafickou alternativu `gvimdiff`. Tato alternativa poskytuje vylepšené ovládání a lepší grafický výstup. Jinak se tento nástroj výrazně neliší.

Prázdné nastavení v konfiguračním souboru u parametru `diffprogram`, je v případě zažádání o zobrazení rozdílů, spuštěno pouze jednoduché zobrazení rozdílů do konzole. Toto zobrazení

[7] Vzhledem k zaměření navrhované aplikace na konfigurační soubory se domnívám, že automatické sloučení není vhodné. Výsledný soubor může obsahovat i několik se navzájem překrývajících nastavení.

[8] Zkratka označující K desktopové prostředí (anglicky K Desktop Environment) Bližší informace je možné nalézt na stránkách projektu `http://www.kde.org/`.

[9] Do nastavení je nutné zadat diffoptions „-o“.

vypíše jednotlivé řádky obou verzí. V případě, že se řádky liší, je tato skutečnost zobrazena za pomoci znaků $+$ a $-$. V případě, že je řádek navíc oproti starší verzi, je před řádek umístěno $+$. V opačném případě je zobrazeno znaménko $-$. Tímto způsobem je možné zobrazit rozdíly mezi soubory. Nevýhodou je, že kromě zobrazení rozdílů jinou funkčnost neposkytuje.

Další nástroje zobrazení rozdílů je možné nalézt na internetu. Obsáhlý seznam těchto nástrojů je na stránkách [26]. Stránky obsahují i porovnání jednotlivých nástrojů.

Kapitola 5

Testovací případy

V této kapitole uvádím testy, kterými ověřuji funkčnost výsledného programu. Testy vycházejí z funkčních požadavků uvedených v části 2.3. Tímto způsobem by mělo být pokryta celá funkčnost systému.

U některých testovacích případů, u kterých se nabízí možnost provést několik testů najednou, je testováno i několik testovacích případů[1].

Testy byly provedeny na počítači s GNU/Linux distribucí Fedora 7. Konkrétně byla použita LiveCD distribuce[2]. Tato distribuce je dostupná na webových stránkách projektu `http://fedoraproject.org/get-fedora`. Z této LiveCD distribuce byla provedena nová instalace na pevný disk. Touto novou instalací jsem získal prostředí, v kterém se navrhovaná aplikace nenacházela. Na tuto čerstvou instalaci je nutné doinstalovat potřebné balíky softwaru, tedy pyinotify a mercurial.

Všechny příkazy uvedené v následujících testech jsou spuštěny uživatelem root. Tyto nástroje jsou určeny pro tohoto uživatele tedy je použit tento uživatel pro spouštění příkazů. Tato aplikace si neklade za cíl řešit přístupová práva uživatele jak už je uvedeno v úvodu 2.3.

5.1 Kontrola kódu pomocí nástroje pylint

Všechny skripty aplikace byly zkontrolovány nástrojem `pylint(1)`, který upozorňuje na možné chyby ve zdrojovém kódu. Kontrola byla provedena s průměrným hodnocením 8/10.

Pro skript nástroje `snap_notifier(8)`, použití modulu threading způsobilo zhoršení hodnocení kvality kódu. Modul je použit podobně tak jak je běžně uváděno jeho použití. Příklad zdrojových kódů z kterých jsem vycházel je uveden v článku [13].

5.2 Konfigurační soubor a odpovídající prostředí

Pro testování je nutné nejprve vytvořit konfigurační soubor, pomocí kterého budou prováděny samotné testy. Konfigurační soubory budou vytvořeny dva, jeden nacházející se v standardním umístění a další alternativní na nestandardním umístění.

K těmto dvěma konfiguračním souborům bude nutné vytvořit odpovídající prostředí, tedy soubory a adresáře, které budou sledovány, zálohovány a obnovovány.

[1] Většinou se jedná o vstupy konfigurace.

[2] Tento termín je vysvětlen v předchozí části 3.1.9.

5.2.1 Standardní konfigurační soubor pro testovací případy

Konfigurační soubor bude použit pro většinu testovacích případů. Tento konfigurační soubor bude umístěn v souboru `/etc/snaproll.conf`. Až na výjimky bude pro testy využit soubor z výpisu 5.1.

```
conf_dir /etc
arch_dir /var/db/snaproll

ignorelist * *bak
followsymlinks yes
diffprogram vimdiff
diffoptions ""

# Zaznam s~jednim souborem
soubor {
     manual
     }

# Zaznam s~glob syntaxi a jednim souborem
odkaz *release* {
        ontime */5 * * * *
        followsymlinks no
        }

# Zaznam s~celym adresarem
onchange.test adresar/ {
        onchange
        followsymlinks yes
        ignore *.orig
        }
```

Příklad 5.1: Výpis defaultního konfiguračního souboru

Pro tento konfigurační soubor je třeba vytvořit odpovídající prostředí, tedy soubory a adresáře, které odpovídají uvedeným nastavením v souboru. Vytvoření tohoto prostředí je možné vidět na výpisu 5.2.

```
echo test > /etc/soubor
mkdir /home/sruser
echo test > /home/sruser/symcil1.txt

ln -s /home/sruser/symcil1.txt /etc/odkaz

echo test > /etc/first.release
echo test > /etc/second.release

echo ignorovano > /etc/first.release

echo test > /etc/onchange.test
mkdir -p /etc/adresar/podadresar
echo test > /etc/adresar/soubor.test

echo ignorovano > /etc/adresar/soubor.test
echo ignorovano > /etc/adresar/soubor.orig
echo ignorovano > /etc/adresar/zaloha.bak

echo test > /etc/adresar/podadresar/podadresar.txt
echo test > /home/sruser/symcil2.txt
```

```
ln -s /home/sruser/symcil2.txt /etc/adresar/podadresar/odkaz2
```

Příklad 5.2: Prostředí odpovídající alternativnímu konfiguračnímu souboru

5.2.2 Alternativní konfigurační soubor pro testovací případy

Další konfigurační soubor 5.3 obsahuje konfiguraci určenou pro vstup, jako alternativní konfigurace. Tato alternativní konfigurace bude obsahovat nestandardní umístění konfiguračního a archivního adresáře. Samotný konfigurační soubor bude umístěna na jiném místě, než je běžné. Konkrétně toto místo odpovídá adresáři `/home/sruser/snaproll.alt`.

```
conf_dir /home/sruser
arch_dir /home/sruser/snaproll

# Zaznam obsahujici jeden soubor
Xdefaults {
      manual
      }

# Zaznam obsahujici cely adresar
mc/ {
      onchange
      followsymlinks yes
}
```

Příklad 5.3: Výpis alternativniho konfiguračního souboru

Konfigurační soubor 5.3 určený pro zálohování souborů se nachází v adresáři běžného uživatele. Tato aplikace není určena pro toto zaměření[3]. Nicméně, pro testovací účely tento adresář postačí. Pro konfigurační soubor je třeba vytvořit odpovídající prostředí, tedy soubory a adresáře, které odpovídají uvedeným nastavením v souboru. Vytvoření prostředí je možné vidět na výpisu 5.4.

```
mkdir -p /home/sruser/snaproll

echo test > /home/sruser/Xdefaults
echo test > /home/sruser/cilodkazu

mkdir /home/sruser/mc

echo test > /home/sruser/mc/ini

ln -s /home/sruser/cilodkazu /home/sruser/mc/odkaz
```

Příklad 5.4: Prostředí odpovídající alternativnímu konfiguračnímu souboru

5.3 Instalace nástroje

Jako posledním krokem nutným k testování je samotná instalace balíku. Je možné instalovat balík `snaproll-0.3-3.i386.rpm`, který je dostupný na stránkách projektu

[3] Je pravděpodobné, že v dalších verzích se aplikace změní.

http://sourceforge.net/projects/snaproll. Postup instalace je popsán v příloze B.1.

5.4 Testování nástroje pro aktualizace adresářů

V této části jsou testy nástroje `snap_update(8)`. Tento nástroj provádí aktualizaci archivního a konfiguračního adresáře.

5.4.1 Vstup konfiguračního souboru

Zavedení archivního adresáře do standardního umístění.

Požadavek: Použití standardního konfiguračního souboru. Konfigurační adresář obsahuje další podadresáře a symbolické odkazy do dalšího umístění.

Specifikace: Cílem je aktualizovat archivní adresář strukturou souborů a adresářů odpovídající konfiguračnímu adresáři. Soubory propojené do archivního adresáře dále odpovídají nastavení, které se nachází v konfiguračním souboru `/etc/snaproll.conf`. Obsah a nastavení odpovídají vytvořenému prostředí uvedenému v části 5.2.1.

Struktura obou adresářů by měla odpovídat uvedenému nastavení. Soubory, které mají být ignorovány, by neměly být propojeny do archivního adresáře. Dále by v adresáři měly být pouze upřesněné soubory a adresáře.

Provedení:

```
/usr/sbin/snap_update --archdir
ls /etc/*release*
cd /var/db/snaproll/localhost
ls -la .
ls -la etc/adresar/podadresar
/usr/sbin/snap_view --revs --sync
cat /etc/crontab
```

Výsledek: V archivním adresáři je vytvořen adresář `localhost`, který odpovídá lokálnímu počítači. Při spuštění prvního příkazu jsou vypsány soubory přidané do archivního adresáře. Již v tomto výpisu je patrná absence ignorovaných souborů. Dále je patrné, že symbolický odkaz odkazující na umístění `/home/sruser/symcil2.txt`, je přidáno do repositáře SCM. U druhého jména `odkaz` je vidět, že byl přidán, ale již bez cíle odkazu. Ten byl pomocí parametru `followsymlinks no` vyřazen z okruhu souborů určených pro zálohování.

Výpis souborů dále obsahuje několik souborů, které odpovídají druhému záznamu. To je docíleno pomocí glob syntaxe[4] `*release*` v druhém konfiguračním záznamu. Soubory z třetího konfiguračního záznamu jsou přidány, až na ignorované, všechny soubory.

Po zobrazení obsahu adresáře je vidět vytvořený repositář SCM v adresáři `/var/db/snaproll/.hg`. Je patrné, že struktura adresáře odpovídá konfiguračnímu adresáři podle nastavení. To znamená podle upřesněných okruhů. V adresáři je `/var/db/snaproll/localhost/etc` vidět symbolický odkaz, který ukazuje do prázdného umístění. V adresáři `/var/db/snaproll/localhost/etc/adresar/podadresar` se nachází odkaz, který ukazuje do existujícího umístění.

[4] Tento pojem je vysvětlen v předchozí části 3.1.7.

Při použití nástroje `snap_view(1)` s uvedenými parametry, je vidět záznam o provedení aktualizace repositáře, spolu s přidanými soubory. Posledním příkazem je získán obsah souboru, v kterém je uvedeno nastavení služby `cron(8)`. Na posledním řádku je následující záznam:

```
*/5 * * * * root python /usr/sbin/snap_create "odkaz *release*" \
            -m "crontab"
```

Tento záznam bude dále použit při testu vytváření záloh v časovém intervalu 5.5.2. Případně je aktualizace použita jako výchozí bod pro další testování.

5.4.2 Vstup alternativního konfiguračního souboru

Vlastností všech nástrojů je možnost použít jiný konfigurační soubor. Tento soubor je upřesněn pomocí vstupního parametru.

Požadavek: Použití alternativního konfiguračního souboru. Vytvoření symbolického odkazu v archivním adresáři. Zvýšení výřečnosti nástroje.

Specifikace: Cílem je aktualizovat archivní adresář pomocí nestandardního konfiguračního souboru uvedeného v sekci 5.2.2 v souboru `/home/sruser/snaproll.alt`. Na standardním umístění je uložen konfigurační soubor, který **nebude** použit. Při zadání nestandardního konfiguračního souboru bude vytvořen repositář v archivním adresáři, podle nastavení z tohoto souboru. Zároveň je zapnuta vyšší výřečnost nástroje, takže vypisuje postup provádění aktualizace. Dále bude v okruhu souborů v jednom z adresářů umístěn symbolický odkaz.

Provedení:

```
/usr/sbin/snap_update -A -c /home/sruser/snaproll.alt -v info

ls -lR  /home/sruser/snaproll/localhost/home/sruser
```

Výsledek: Nástroj vypíše vzhledem k nastavené výřečnosti informace o svojí činnosti. To znamená vytváření pevných a symbolických odkazů, případně adresářů. Zároveň jsou vidět soubory, které jsou přidány do repositáře. V následujícím výpisu je pomocí příkazu `ls(1)` možné vidět soubory, adresáře a symbolické odkazy, které byly vytvořeny na základě alternativního konfiguračního souboru.

5.4.3 Kontrola existence uvedeného nástroje pro zobrazení rozdílů

Požadavek: Nástroj by měl zakončit, pokud je v konfiguraci uvedený neexistující diff nástroj.

Specifikace: Kontrola syntaxe by měla zahlásit chybu, pokud je v nastavení uveden neexistující nástroj pro zobrazení rozdílů.

Provedení: Do konfiguračního souboru jsem doplnil u parametru diffprogram nespustitelný název. Například `diffprogram nespustitelne`. Dále jsem spustil následující příkaz.

```
/usr/sbin/snap_update --archdir
```

Výsledek: Nástroj skončí s chybovým výpisem, který oznamuje neexistenci souboru, který je uvedený jako `diffprogram`. Výpis obsahuje číslo řádku a text řádku.

5.4.4 Kontrola existence archivního, konfiguračního adresáře a tichý režim

Požadavek: Nástroj by měl zakončit, pokud je v konfiguraci uvedený neexistující archivní adresář, nebo konfigurační adresář. Dále je možné nastavit nástroj tak, aby nevypisoval žádné zprávy do příkazové řádky.

Specifikace: Nástroj by měl uživatele upozornit, že uvedený adresář u parametrů `arch_dir` a `conf_dir` neexistují. U tohoto upozornění je možné vypnout vypisování do příkazové řádky pomocí parametru `--quiet`.

Provedení: Do konfiguračního souboru jsem doplnil u parametru `archdir` neexistující adresář. Například `arch_dir /neexistuje1`. Dále jsem spustil následující příkaz.

```
/usr/sbin/snap_update -A
/usr/sbin/snap_update --archidir --quiet
```

Pokračoval jsem změnou údaje parametru `conf_dir /neexistuje2`. Poté spustím příkaz znovu. Jako poslední příkaz spouští aktualizaci archivního adresáře s parametrem `--quiet`.

Výsledek: V obou případech při spuštění nástroje vypíše chybové hlášení oznamující neexistenci uvedeného adresáře, spolu s číslem řádku a údajem, který chybu obsahuje. V obou případech `neexistuje1`, nebo `neexistuje2`. U dalšího příkazu, který má sice stejný výsledek, ale chybový výstup je směrován pouze do logovacího souboru. Proto nástroj po spuštění nevypíše žádné chybové hlášení.

5.4.5 Vytváření symbolických odkazů podle parametru –symlinks

Požadavek: Nástroj vytvoří cíl odkazu v archivním adresáři, pokud je nastaveno sledování symbolických odkazů. Jestliže se toto nastavení změní, cíl bude smazán z archivního adresáře.

Specifikace: Cílem je, aby nástroj `snap_update(1)` reagoval na změnu nastavení parametru, sledování symbolických odkazů. To znamená, pokud je v konfiguračním souboru povoleno sledování symbolických odkazů, nástroj aktualizace je spuštěn s parametrem, který sledování zakazuje, odstraní tento odkaz[5].

Provedení: Konfigurační soubor uvedený v 5.2.1 upravím následující záznam:

```
odkaz *release* {
   ontime */5 * * * *
   #followsymlinks no
   }
```

Zakomentováním parametru `followsymlinks` se docílí povolení sledování symbolických odkazů pro uvedený záznam. Následovně spustím příkazy.

```
/usr/sbin/snap_update --archdir
/usr/sbin/snap_update --archdir --symlinks no

ls -l /var/db/snaproll/localhost/etc

/usr/sbin/snap_view --revs --rem
```

[5]Toto nastavení parametrem se nevztahuje na upřesněné nastavení uvedené pro specifický záznam.

Výsledek: Po první aktualizaci je možné vidět, že je vytvořen cíl pro soubor `odkaz` odkazující na soubor `symcil1.txt`. Po spuštění druhé aktualizace bude cíl odkazu opět odebrán. Je možné zobrazit výsledky těchto změn pomocí posledního příkazu.

5.5 Vytvoření snapshotu

V této části jsou testy nástrojů pro vytváření snapshotů. Jedná se o tři hlavní způsoby, jak vytvořit snapshot.

5.5.1 Manuální vytvoření pomocí nástroje pro vytvoření snapshotu

Požadavek: Vytvořit snapshot a uložit jej do repositáře úložiště s přidanou zprávou od uživatele.

Specifikace: Po vytvoření snapshotu ze záznamu uvedeném v konfiguračním souboru `/etc/snaproll.conf` uvedeném v sekci 5.2.1. Pro vytvoření je použit záznam z konfigurace `soubor`[6]. Zobrazení je provedeno pomocí nástroje `snap_view(1)`.

Provedení: Před provedením testu je nutné provést aktualizaci adresáře, nejlépe tak, jak je uvedeno v části 5.4.1. Nejprve přidám text do souboru určeného pro zálohování. Následně spustím vytvoření snapshotu s přidanou zprávou.

```
echo manual >> /etc/soubor

/usr/sbin/snap_create 'soubor' -m 'manualne vytvoreny snapshot'
/usr/sbin/snap_view --revs
```

Výsledek: V repositáři je jeden záznam navíc z okruhu 'soubor' se souborem `/etc/soubor`. U záznamu je také uveden text `manualne vytvoreny snapshot`. Záznam dále obsahuje záznam o datu, kdy byl záznam vytvořen a verzi s unikátním označením.

5.5.2 Vytvoření snapshotu po časovém intervalu

Požadavek: Vytvoření snapshotu v pravidelném časovém intervalu.

Specifikace: Cílem je zahájení vytváření pravidelných snapshotů v časovém intervalu. Snapshoty jsou vytvářeny pouze při změněném souboru. Využívám tedy crontab i pro vytváření této pravidelné změny. Tyto změny se budu zapisovat do souborů `/etc/first.release` a `/etc/second.release` u každého v jiném intervalu. Sledování bude prováděno pro jeden konfigurační záznam, a to pro `odkaz *release*`. Tento záznam je uveden v sekci 5.2.1. Výsledkem bude sada záloh z okruhu souborů tohoto konfiguračního záznamu. Po provedení zápisu do `/etc/crontab` se začnou zapisovat do repositáře zálohy.

Provedení: Nejprve provedu změny do souboru `/etc/crontab`. V souboru už bude uveden záznam o provádění záloh po provedení aktualizace v předchozí části 5.4.1.

```
echo "*/1 * * * * root echo test >> /etc/first.release" >> /etc/crontab
echo "*/5 * * * * root echo test >> /etc/second.release" >> /etc/crontab
...
/usr/sbin/snap_view --revs
```

[6] Záznamem je označen celý okruh souborů. Nicméně, v tomto případě se jedná pouze o jeden soubor .

Výsledek: Po časovém intervalu pěti minut by mělo proběhnout první spuštění vytvoření snapshotu. Příkazem `snap_view(1)` je možné zobrazit uložené snapshoty. Snapshoty pořízené pomocí služby `cron(8)` mají přidaný text „crontab“. Další údaje uvedené u snapshotu jsou stejné jako v předchozím případě, to znamená datum a čas vytvoření snapshotu. Název záznamu, číslo revize a jedinečné označení.

Pokud je ponecháno toto nastavení v souboru `/etc/crontab` delší dobu, je po spuštění nástroje `snap_view(1)` vidět řada záloh v pravidelných rozestupech pěti minut.

5.5.3 Vytvoření snapshotu po zápisu do sledovaného souboru

Požadavek: Vytvoření snapshotu při zápisu do sledovaného souboru.

Specifikace: Cílem je, aby služba sledující události na souborech, vytvořila snapshot. Tento snapshot bude vytvořen po zapsání do některého ze sledovaných souborů. Zápisů do různých souborů bude několik. Tímto záznamem je `onchange.test adresar/` v konfiguraci uvedené v části 5.1. V tomto případě bude sledován jednotlivý soubor a celý adresář.

Provedení: Službu je možné spustit startovacím skriptem `/etc/init.d/snapwatch`, případně jiným způsobem, který distribuce Fedora poskytuje[7]. Před spuštěním služby je možné sledovat chybový výstup služby. Služba je defaultně spouštěna s výřečností nastavenou na úroveň `info`. Vypisuje všechny operace, které provádí.

```
tail -f /var/log/snaproll.log

/etc/init.d/snapwatch start
/etc/init.d/snapwatch status

echo zmena >> /etc/onchange.test

/usr/sbin/snap_view --revs

echo ignorovano >> /etc/adresar/ignorovano.bak
echo zmena >> /home/sruser/symcil2.txt

/usr/sbin/snap_view --revs

/etc/init.d/snapwatch stop
```

Výsledek: Prvním příkazem je zahájeno sledování souboru, do kterého služba vypisuje chybová hlášení. Provedením dalších dvou příkazů je možné službu sledování spustit a zkontrolovat, jestli služba běží. Dále po zapsání do souboru je možné zjistit pomocí nástroje `snap_view(1)` existenci nového snapshotu. Při dalším zápisu do souboru `/etc/adresar/ignorovano.bak` se vytvoření snapshotu neprovede, neboť tento soubor není v okruhu sledovaných souborů (je zařazen v seznamu ignorovaných souborů).

V posledním zápisu je zapisováno do souboru, který je sice mimo konfigurační adresář, nicméně je na něj odkazováno přes symbolický odkaz ze souboru `/etc/adresar/podadresar/odkaz`. Nastavení je uvedeno v záznamu z konfigurace spolu s povolením sledování symbolických odkazů. V posledním výpisu revizí je vidět, že byl

[7] Služba snapwatch je během instalace registrována jako služba systému. Je proto možné ji používat stejně, jako ostatní služby systému.

vytvořen snapshot se změnou v souboru `symcil2.txt`. Posledním příkazem je služba snapwatch ukončena.

5.5.4 Spuštění sledování souborů se změněným chybovým výstupem

Požadavek: Spuštění nástroje sledování souborů bez přechodu služby do běhu na pozadí s výpisem do alternativního logovacího souboru.

Specifikace: Cílem tohoto testu je spustit službu sledování souborů s parametrem, který vypne přechod služby do běhu na pozadí systému. Tento parametr způsobý, že se nástroj nepřejde do běhu na pozadí. Dalším parametrem bude zvýšena výřečnost nástroje, která způsobí, že služba bude vypisovat informace o sledovaných událostech. Poslední parametr způsobí, že chybový výstup bude přesměrován do souboru v jiném než standartním umístění.

Provedení: Spuštění služby se provede z příkazové řádky s uvedenými parametry. Pro zápis je možné použít záznam `onchange.test adresar/` z konfigurace uvedené v části 5.1. Nakonec provedení zápisu je možné provést textovým editorem.

```
/usr/sbin/snap_notifier -d -v info -l /home/sruser/snaplog.alt

tail -f /home/sruser/snaplog.alt

vim /etc/adresar/novy_soubor.txt

:wq

/usr/sbin/snap_view --revs
```

Výsledek: Prvním příkazem je spuštěn nástroj sledování událostí na souborech. Nástroj spuštěný tímto způsobem začne do konzole vypisovat sledované soubory a adresáře spolu s dalšími informativními výpisy o své činnosti. Spuštěním příkazu `tail` v jiné konzoly je možné sledovat výpisy směrované do alternativního logovacího souboru. Výpisy do tohoto souboru jsou stejné jako výpisy, které vypisuje nástroj do konzole.

Textovým editorem `vim(1)` je zahájeno editování nového souboru. Uložení souboru spustí řadu událostí, které jsou zachycené běžícím nástrojem pro sledování změn na souborech. Mezi tyto události patří vytvoření swapovacího souboru a jeho smazaní. Dále pak vytvoření nového souboru a jeho modifikace. Příkazem `snap_view(1)` je možné vypsat uložené revize. Nástroj sledování souborů je možné ukončit stiskem kláves `Ctrl+C`.

5.6 Zobrazení snapshotů

V této sekci budu testovat nástroj, který slouží k prohlížení uložených snapshotů. Nástroj prohlížení snapshotů umožňuje zobrazit, které snapshoty jsou uloženy v repositáři. Dále bude otestováno prohlížení obsahu souboru obsaženého v revizi. Další částí bude zobrazení rozdílu mezi dvěmi soubory, případně vytvoření jejich sloučení.

5.6.1 Zobrazení uložených revizí

Požadavek: Vypsání uložených revizí snapshotů.

Specifikace: Cílem je vypsat uložené snapshoty, které se nacházejí v repositáři. Tento výpis je dále možné filtrovat pomocí parametru, kterým je možné určit pouze některé konfigurační záznamy, případně soubory. Jako konfigurační soubor je použit soubor s obsahem stejným, jako je obsah uvedený v předchozím příkladu 5.1. V repositáři by měly být uloženy snapshoty z některého z předchozích testů. Nejlépe se hodí test 5.5.2, který poskytuje velké množství revizí, pokud byl záznam v souboru `/etc/crontab` dostatečně dlouho.

Provedení:

```
/usr/sbin/snap_view --entries
/usr/sbin/snap_view --revs
/usr/sbin/snap_view --revs -e "odkaz *release*"
/usr/sbin/snap_view --revs -f /etc/onchange.test
/usr/sbin/snap_view --files -e "odkaz *release*
```

Výsledek: První příkaz vypíše, které záznamy z konfigurace jsou uloženy v lokálním archivním adresáři. Výstupem dalšího příkazu jsou všechny revize snapshotů uložené v repositáři v lokálním archivním adresáři. Dalším příkazem získám všechny revize snapshotů, které spadají pod záznam `odkaz *release*`. Třetím příkazem je vypsání revizí, které obsahuje specifický soubor v tomto případě `/etc/onchange.test`. Zápis snapshotu s tímto souborem byl proveden v předchozím testu 5.5.3. Poslední příkaz používá jiný výstupní parametr `--files`. Pomocí tohoto parametru jsou vypsány soubory uložené ve všech revizích, které spadají do konfiguračního záznamu `"odkaz *release*`.

5.6.2 Zobrazení obsahu souboru z vybrané revize

Požadavek: Nástroj by měl zobrazit obsah souboru z předchozí revize snapshotu.

Specifikace: Vypsání obsahu je možné docílit pomocí parametru `--cat`, který umožňuje zobrazit obsah souboru uloženého v některém snapshotu. Je nutné předat do parametru konkrétní revizi a jméno požadovaného souboru. K vypsání obsahu je možné použít soubor `/etc/first.release`, který by po testu 5.5.2 měl obsahovat dostatečné množství revizí.

Provedení:

```
/usr/sbin/snap_view --revs -f /etc/first.release
/usr/sbin/snap_view --cat 0e14a8b2b05f -f /etc/first.release
```

Výsledek: Vypsáním dostupných revizí snapshotů, které obsahují vybraný soubor, je možné vybrat vhodnou revizi snapshotu. Vypsání obsahu snapshotu je provedeno následujícím příkazem. Revize je upřesněna pomocí jedinečného označení, které je v předchozím výpisu uvedeno hned za číslem revize. Výsledný výpis bude obsahovat text souboru z uvedené revize.

5.6.3 Použití nástroje pro zobrazení rozdílů verzí

Požadavek: Zobrazení rozdílu obsahu mezi vybranými verzemi souboru.

Specifikace: Cílem je vypsání rozdílů mezi dvěmi revizemi souboru zálohovaného v snapshotech. Dále je možné zobrazit rozdíl mezi aktuální verzí a některým ze zálohovaných snapshotů. V tomto případě je možné zapisovat do aktuální verze zálohy a tím vytvořit sloučenou verzi

souboru. Pro editaci je nutné, aby v konfiguračním souboru `snaproll.conf` byl uveden externí nástroj, který umožňuje toto editování.

Provedení: Po vypsání dostupných záloh vybraného souboru. Pro tyto účely mohu využít záznam `soubor`, který nemá příliš mnoho záloh z předchozího testu 5.5.1. Pro zobrazení rozdílů je vhodné nějaké rozdíly vytvořit spolu s některými snapshoty.

```
echo diff1 >> /etc/soubor
echo diff2 >> /etc/soubor

/usr/sbin/snap_create "soubor"

echo test1 >> /etc/soubor
echo test2 >> /etc/soubor

/usr/sbin/snap_create "soubor"

echo actual1 >> /etc/soubor
echo actual2 >> /etc/soubot

/usr/sbin/snap_view --revs -e "soubor"
/usr/sbin/snap_view --diff 64 65 -f /etc/soubor
/usr/sbin/snap_view --diff 64 -f /etc/soubor

cat /etc/soubor
```

Výsledek: V uvedené sekvenci příkazů je na začátku vytvoření změn v souboru určeném pro výpis změn.

Příkazem `snap_create(1)` jsou vytvořeny další revize. Příkaz `snap_view(1)` vypíše revize souboru určeného pro zobrazení rozdílů. Dalším příkazem `snap_view(1)` s uvedeným parametrem je možné vypsat, které revize obsahují požadující konfigurační záznam. V tomto případě záznam s požadovaným souborem.

Příkaz spuštěný s parametrem `--diff` zobrazí externí nástroj[8] pro zobrazování rozdílů s verzemi 64 a 65. Tyto revize jsou v mém případě poslední dvě uložené. Jsou to revize obsahující záznamy přidané na začátku testu. Je vhodné poznamenat, že nástroj zobrazení rozdílů je spuštěn pouze pokud v různých verzích souboru jsou nějaké rozdíly.

Při dalším použití tohoto nástroje je zadána pouze jedna revize. Jako druhou revizi použije nástroj aktuální verzi souboru, který je uvedený jako poslední vstupní parametr. Toto lze editovat[9] a zároveň uložit do aktuální verze v konfiguračním adresáři. Po provedení posledního příkazu je možné vidět změny uložené v `/etc/soubor`.

5.6.4 Výpis revizí snapshotů v klientském adresáři

Požadavek: Vypis revizí nacházejících se v klientském archivním adresáři

Specifikace: S pomocí nástroje `snap_view(1)` je možné vypsat revize dostupné z klientských repositářů. Dále je možné vypsat, které klientské adresáře jsou dostupné, případně, které konfigurační záznamy obsahují.

[8] Externí nástroj je zobrazen, pokud byl nastaven v konfiguračním souboru. Jestliže tento nástroj nastaven nebyl, je použito jednoduché zobrazení rozdílů pomocí nástroje diff(1).

[9] Ve většině případů je revize uvedena v pravém panelu nástroje. Pozice tohoto souboru se může měnit nástroj od nástroje.

Provedení: Nejprve vytvořím rozdílné zálohy do lokálního archivního adresáře. Poté tento adresář použiji jako klientský archivní adresář. V reálném prostředí bych získal tento klientský adresář z dalšího počítače. To mohu provést jednoduše zkopírováním pomocí nástroje `scp(1)`. V tomto případě je jednodušší zkopírovat archivní adresář z lokálního počítače, z `/var/db/snaproll/localhost` do adresáře, který odpovídá klientskému `/var/db/snaproll/klient`.

```
echo test > /etc/first.release
echo test > /etc/second.release

/usr/sbin/snap_create "odkaz *release*"

echo test klienta > /etc/first.release
echo test klienta > /etc/second.release

cp -r  /var/db/snaproll/localhost /var/db/snaproll/klient

/usr/sbin/snap_view --sites
/usr/sbin/snap_view --entries -s klient
/usr/sbin/snap_view --revs -f /etc/first.release -s klient

/usr/sbin/snap_view --cat 62 -f /etc/first.release -s klient

echo test local > /etc/first.release
echo test local > /etc/second.release

/usr/sbin/snap_view --sitefiles klient
/usr/sbin/snap_view --sitediff klient -f /etc/first.release

cat /etc/first.release
```

Výsledek: Po vytvoření klientského adresáře je možné zobrazit nástrojem `snap_view(1)` seznam dostupných klientských adresářů. Další parametr tohoto nástroje vypisuje všechny záznamy z konfigurace, které jsou uložené na tomto klientu. Dalším výpisem získám všechny revize souboru `/etc/first.release`, které jsou uloženy v klientském adresáři. Posledním nástrojem je možné vypsat obsah souboru v konkrétní revizi. Tato revize je v mém případě poslední, obsahuje tedy pouze text „test“.

V další části testu, na lokálním počítači, vytvářím změnu `test local` do souborů konfiguračního adresáře . Příkazy, které následují poté, provádějí zobrazení rozdílů mezi dvěma klientskými adresáři a další pak rozdíl v souboru mezi dvěma adresáři. V uvedených příkazech je zobrazen rozdíl mezi klientskými adresáři `klient` a `localhost`. Lokální adresář je standardně doplňován, pokud není uveden jiný klientský adresář. Rozdíl mezi dvěma klientskými adresáři je zobrazen částí adresářových stromů, které jsou v obou klientských adresářích stejné. Jsou vypsány soubory, které jsou na stejném umístění v adresářovém stromu a zároveň mají stejné jméno souboru. Dále pak soubory a adresáře, označené `<<,>>` případně `D<,D>`, které se nacházejí pouze v jednom adresářovém stromu klienta. U souborů se **stejným** jménem a umístěním je pomocí označení `==`, nebo `!=` tato skutečnost naznačena.

Soubory s označením `!=` mají jiný obsah. Obsah souboru je možné porovnat pomocí posledního příkazu. Rozdíl[10] je opět zobrazen pomocí externího nástroje uvedeného v konfigurač-

[10] Pokud nebyl uveden jako jeden z klientských adresářů localhost, je zobrazen na pravé straně.

ním souboru `snaproll.conf(5)`. V tomto externím nástroji je možné editovat oba soubory. Posledním příkazem je zobrazen obsah souboru, v kterém budou viditelné změny, pokud byly nějaké provedeny.

5.7 Obnovení snapshotu

Obnovení revize snapshotu se provádí nástrojem `snap_roll(1)`. Zde jsou uvedeny testy, které ověřují obnovování souborů do stavu, v kterém byly pořízeny.

Požadavek: Obnovení snapshotu podle číslem zadané revize.

Specifikace: Cílem testu je obnovení uloženého snapshotu do původního stavu, tedy do konfiguračního adresáře. Obnovený soubor by měl být v stavu, ve kterém byl pořízen. Aktuální soubor by měl být uložen do souboru s příponou `.orig`, pokud je tento soubor obnovou revize přepisován. Pro test bude použito prostředí uvedené v části 5.2.1.

Provedení: Před obnovením provedu zálohu, která umožní zviditelnit rozdíl mezi aktuálním souborem a obnovenou zálohou. Pro obnovení použiji záznam `soubor`, který je ukládán manuální cestou. Před vytvořením snapshotu přidám nejprve do souboru text. Poté vytvořím porovnávací soubor `soubor.zaloha`. Po provedení obnovy pomocí nástroje `snap_roll(1)` je obnovený soubor porovnán s porovnávací zálohou.

```
echo test >> /etc/soubor

/usr/sbin/snap_create 'soubor'

echo test2 >> /etc/soubor
cp /etc/soubor /etc/soubor.zaloha

/usr/sbin/snap_view --revs -f /etc/soubor

/usr/sbin/snap_roll -r 44

diff /etc/soubor.zaloha /etc/soubor.orig
diff /etc/soubor /etc/soubor.zaloha
```

Výsledek: Nástroj nejprve vypíše název změněného souboru jako potvrzení vytvoření snapshotu. Spuštěním příkazu `snap_view(1)` se zobrazí dostupné snapshoty s tímto souborem. V mém případě jsou snapshoty dostupné dvě následující revize s tímto souborem[11]:

```
...
changeset: 40: 390a5693eed1
...
changeset: 44: b27a038c54fc
```

Příkazem `snap_roll(1)` je obnovena poslední revize číslo 44. Po spuštění nástroje je vypsáno hlášení o obnovování souboru a jeho zálohování do souboru `.orig`. Z porovnání souborů je viditelné, že první dvojice `soubor.zaloha` a `soubor.orig`, neobsahují žádné rozdíly. U druhého porovnání je vidět rozdíl souborů, to znamená záznam navíc `test2`.

[11] Jsou uvedeny pouze sady změn bez dalších zobrazovaných údajů.

5.7.1 Obnovení snapshotu na klientu

Požadavek: Obnova některého snapshotu uloženého v klientském adresáři pomocí uvedeného záznamu.

Specifikace: Cílem je obnovit předchozí zálohu v klientském adresáři. Tato obnova se provede pouze v adresáři klienta a nepromítne se do konfiguračního adresáře v lokálním umístění. Klientský adresář je možné získat stejným způsobem jako v testu 5.6.4. Tedy zkopírováním adresáře se soubory z lokálního počítače do adresáře s jiným klientským názvem.

Provedení: Test zahájím vytvořením klientského adresáře.

```
cp -r /var/db/snaproll/localhost /var/db/snaproll/klient2
rm /etc/*release*
cat test klienta >>  /var/db/snaproll/klient2/etc/first.release

/usr/sbin/snap_view --sites
/usr/sbin/snap_view --entries -s klient2
/usr/sbin/snap_view --revs -s klient2 -e "odkaz *release*"

/usr/sbin/snap_roll -r 6 -s klient2

cd /var/db/snaproll/klient2/etc
cat first.release
cat first.release.orig
```

Výsledek: Výpisem pomocí nástroje `snap_view(1)` získám nejprve dostupné klientské adresáře. Dalším použitím tohoto nástroje získám seznam záznamů uložených v tomto klientském adresáři. Posledním získám revize uložené v klientském repositáři se záznamem, do kterého spadají i smazané soubory. Ze získaného výpisu obnovím některý z předchozích snapshotů, pomocí příkazu pro obnovu snapshotů `snap_roll(1)`. V mém případě se jedná o verzi 6. Po změně adresáře do umístění `/var/db/snaproll/klient2/etc` je po vypsání souboru vidět text odpovídající textu z obnovené revize. V souboru s příponou `.orig` je navíc text přidaný na začátku testu, tedy text `test klienta`.

Poslední příkaz listuje adresář `/etc`, do kterého se neobnovily klientské soubory. Obnovu do tohoto adresáře provádí nástroj, pouze pokud se jedná o lokální umístění.

5.7.2 Obnovení konfigurace z archivního adresáře

Požadavek: Obnova obsahu z archivního adresáře do konfiguračního.

Specifikace: Cílem je obnovit obsah konfiguračního adresáře z obsahu, který se nachází v archivním adresáři, pomocí nástroje `snap_update(8)`. Obnovení je možné provést, například na klientu při přenesení nové konfigurace, ze vzdáleného zdroje. Tuto obnovu je nutné provést před jakýmkoliv prohlížení, případně ukládáním, neboť tyto nástroje během své činnosti aktualizují archivní adresář obsahem z konfiguračního. Na druhé straně obnova vybraných souborů je prováděna jiným nástrojem, případně je možné zobrazit rozdíly a ty editovat. Aktualizace pomocí nástroje `snap_update(8)` pouze umožňuje kompletní přenos všeho, co je v archivním adresáři do konfiguračního.

Provedení: Z tohoto důvodu nástroj potřebuje ke své funkci správně nastavený konfigurační soubor `snaproll.conf(5)` zkopírovaný do souboru `snaproll.alt` v jiném umístění. Nástroj

spustím s parametrem, který použije soubor v alternativním umístění. Proto začnu s vytvořením prostředí pro obnovu obsahu. Smazáním částí konfiguračního adresáře.

```
rm -r /etc/adresar
/usr/sbin/snap_update -C -c snaproll.alt
```

Výsledek: Po spuštění nástroje bude obnoven smazaný `adresar` spolu s obsahem.

Kapitola 6

Závěr

V této práci jsem řešil problém zálohování konfiguračních souborů. Hlavní myšlenkou tohoto projektu bylo vytvoření nástrojů, které by usnadnily správu konfiguračních souborů v OS GNU/Linux. Nástroje umožňují správci OS vytvořit řadu záloh, které obsahují konfigurace, tak jak se v průběhu času měnily. Správci je umožněno vybrat při jaké události se mají tyto zálohy vytvářet.

Mezi události při kterých se vytvoření zálohy patří zápisu do souborů nebo adresářů, které byly nastaveny pro sledování na tyto události. Lze také vytvářet zálohy vybraných souborů nebo adresářů v přednastavených časových intervalech. Nakonec je možné ukládat zálohy i manuálně spuštěním z příkazové řádky. Zálohování je řešeno vytvářením snapshotů okruhů konfiguračních souborů. Obsah těchto okruhů je určen záznamy uvedenými v nastavení nástrojů. Záznamy se skládají z názvu obsahujícího soubory určené pro zálohovaní a upřesněného nastavení pro tyto soubory. Toto nastavení obsahuje například událost při které se provádí vytváření snapshotů.

Pomocí jednoho z nástrojů je možné zobrazit, které snapshoty byly vytvořeny. Tyto výpisy je možné filtrovat. Je také možné zobrazovat obsah souborů z pořízených snapshotů, případně tyto obsahy porovnávat. Současně je možné vytvořit sloučení mezi aktuálním a zálohovaným souborem. Nástroje umožňují i práci se zálohami pocházejícími z jiného než lokálního umístění. Tyto zálohy z jiného umístění je možné porovnat mezi sebou, případně vytvářet sloučení souborů. Jeden z nástrojů pak umožňuje obnovení snapshotu zpět do stavu, v kterém byl snapshot vytvořen.

Na začátku této práce jsem vymezil jaké činnosti by měly nástroje provádět. Okruh těchto činností vychází mimo jiné i z dalších projektů s podobným zaměřením. Některé požadavky také vycházejí z mých zkušeností s tímto OS. V hlavní částí této práce jsem vytvořil návrh nástrojů, které umožňují vykonávat činnosti uvedené v předchozích požadavcích. V tomto návrhu jsem popsal jak by měly vypadat jednotlivé části této aplikace. Z návrhu se povedlo vytvořit nástroje, které umožňují provádět požadované činnosti. Vytvořené nástroje byly nakonec otestovány, jestli splňují požadavky uvedené na začátku práce. Během testů se podařilo odstranit i některé chyby, které nebyly během návrhu a poté při implementaci zřejmé. Testy se nacházejí v jedné z kapitol tohoto textu.

Při práce na tomto projektu jsem vytvořil návrh s pomocí diagramů unifikovaného modelovacího jazyka. Konkrétně diagram případů užití pro upřesnění, které činnosti budou navrhované nástroje provádět. Dále jsem použil diagram tříd, pomocí kterého jsem rozdělil nástroje do funkčních celků, které dohromady umožňují požadovanou funkčnost. Nakonec jsem pomocí sekvenčních diagramů popsal posloupnost provádění jednotlivých funkcí.

Implementaci celého projektu jsem uskutečnil v programovacím jazyce Python. Tento jazyk umožňuje velice dobře vytvářet projekty jako je tento. Velkou výhodou tohoto programovacího jazyka byla dostupnost API pomocí modulů. Tyto moduly mi umožnily přistupovat k možnostem které nabízí podsystém jádra inotify nebo systém správy Mercurial.

Výsledná sada nástrojů je dostupná na stránkách projektu Sourceforge na webové adrese http://sourceforge.net/projects/snaproll. Na těchto stránkách je možné stáhnout balík obsahující vytvořené nástroje spolu s manuálovými stránkami. Balík je vytvořen pro distribuci Fedora, na kterou je možné provést snadnou instalaci pomocí nástroje `rpm`. Projekt je dostupný pod GNU všeobecnou veřejnou licencí verze 2.

Na projekt je možné navázat rozšiřováním funkčností nástrojů, případně způsobů nastavení. Způsob jakým je prováděno definování souborů určených pro zálohování by bylo možné rozšířit. Například definováním nepřímým způsobem uvedeným softwarového balíku. Zároveň rozšířením uživatelské přívětivosti by bylo možné dosáhnout vytvoření grafického rozhraní pro ovládání nástrojů. Jako jedno z možných rozšíření by bylo možné rozšířit práci v síťovém prostředí o hromadnou správu na klientských počítačích.

Literatura

[1] RollBack Rx, 2006, [online], [Cit. 15.5.2008].
<http://www.horizondatasys.com/>

[2] BAUDIŠ, P.: Git - Fast Version Control System. 2007, [online], [Cit. 15.5.2008].
<http://git.or.cz/index.html>

[3] BHE, T.; WALLACE, M.; PEREIRA, G.; aj.: *Event Management and Best Practices*. IBM International Technical Support Organization, první vydání, Červen 2004, ISBN SG24-6094-00, 458 s., [online], [Cit. 15.5.2008].
<http://ibm.com/redbooks>

[4] BUEHLMANN, A.: *The Mercurial API*. Duben 2008, ver. 1.0, [online], [Cit. 25.4.2008].
<http://www.selenic.com/mercurial/wiki/index.cgi/MercurialApi>

[5] Canonical Ltd.: *API Documentation for BzrLib*. Duben 2008, [online], [Cit. 25.4.2008].
<http://starship.python.net/crew/mwh/bzrlibapi/>

[6] EIBL, J.: KDiff3. Duben 2007, [online], [Cit. 15.5.2008].
<http://kdiff3.sourceforge.net/>

[7] KOCMAN, J.: Jak na démona Cron. *Interval.cz*, Duben 2002, ISSN 1212-8651, [online], [Cit. 23.4.2008].
<http://interval.cz/clanky/jak-na-demona-cron/>

[8] Kolektiv: Internet Message Format. Technická zpráva, QUALCOMM Incorporated, Duben 2001, [online], [Cit. 28.4.2008].
<http://www.ietf.org/rfc/rfc2822.txt>

[9] Kolektiv: Filesystem Hierarchy Standard. 2004, [online], [Cit. 15. 5. 2008].
<http://www.pathname.com/fhs/pub/fhs-2.3.html#PURPOSE31>

[10] Kolektiv: *PID Definition*. The Linux Information Project, Červenec 2005, [online], [Cit. 24.4.2008].
<http://www.linfo.org/pid.html>

[11] Kolektiv: GitBenchmarks. Listopad 2007, [online], [Cit. 15.5.2008].
<http://git.or.cz/gitwiki/GitBenchmarks>

[12] Kolektiv: *IBM® WebSphere® Extended Deployment*, kapitola Operations Optimization:Extended repository service. IBM Software Group, August 2007, str. 8, ver. 6.1 [online], [Cit. 15.5.2008].
<http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/index.jsp?

topic=/com.ibm.iea.wxd_v6/wxd/6.1/OperationsOptimization/XD61_Extended_Repository_Service/player.html>

[13] KRISHNA, P. G.: Understanding Threading in Python. *Linux Gazette*, Říjen 2004, [online], [Cit. 15. 5. 2008].
<http://linuxgazette.net/107/pai.html>

[14] LOVE, R.: Intro to inotify. *Linux Journal*, , č. 139, Září 2005, ISSN 1075-3583, [online], [cit. 2008-05-15].
<http://www.linuxjournal.com/article/8478>

[15] MARTINI, S.: *pyinotify: monitor filesystem events with Python under Linux*. Prosinec 2007, [online], [Cit. 15.5.2008].
<http://pyinotify.sourceforge.net>

[16] MURDOCK, B.: Cutting Edge Revision Control. *Cyclopedia Square*, Březen 2007, [online], [Cit. 15.5.2008].
<http://bryan-murdock.blogspot.com/2007/03/cutting-edge-revision-control.html>

[17] PŘIBIL, A.: Co je Fedora Linux. *Fedora český projekt*, Listopad 2007, [online], [Cit. 21.4.2008].
<http://wiki.fedora.cz/doku.php?id=fcz:o_fedore#hardwarove_pozadavky>

[18] van ROSSUM, G.: *Python Library Reference*. Python Software Foundation, Únor 2008, ver. 2.5.2, [online], [Cit. 15.5.2008].
<http://docs.python.org/lib/lib.html>

[19] STUTZ, M.: Linux and the Tools Philosophy. *O'Reilly linux dev center.com*, Srpen 2000, [online], [Cit. 25.4.2008].
<http://www.linuxdevcenter.com/pub/a/linux/2000/07/25/LivingLinux.html>

[20] TIBERI, G.: *Oracle Access Manager Configuration Manager Installation and Administration Guide*, kapitola Backup and Recovery Strategies. Oracle, 10 vydání, January 2007, ISBN B32392-01, str. 158.

[21] Wikipedia: Comparison of revision control software. *Wikipedia, the free encyclopedia*, Květen 2007, [online], [Cit. 15.5.2008].
<http://en.wikipedia.org/wiki/Comparison_of_revision_control_software>

[22] Wikipedia: File descriptor. *Wikipedia, the free encyclopedia*, November 2007, [online], [Cit. 15.5.2008].
<http://en.wikipedia.org/wiki/File_descriptor>

[23] Wikipedia: Granularity. *Wikipedia, the free encyclopedia*, December 2007, [online], [Cit. 15.5.2008].
<http://en.wikipedia.org/wiki/Granularity>

[24] Wikipedia: inode. *Wikipedia, the free encyclopedia*, December 2007, [online], [Cit. 15.5.2008].
<http://en.wikipedia.org/wiki/Inode>

[25] Wikipedia: Shadow Copy. *Wikipedia, the free encyclopedia*, November 2007, [online], [Cit. 15.5.2008].
<http://en.wikipedia.org/wiki/Shadow_Copy>

[26] Wikipedia: Comparison of file comparison tools. *Wikipedia, the free encyclopedia*, Duben 2008, [online], [Cit. 29.4.2008].
<http://en.wikipedia.org/wiki/Comparison_of_file_comparison_tools>

[27] Wikipedia: Comparison of programming languages. *Wikipedia, the free encyclopedia*, Duben 2008, [online], [Cit. 15.5.2008].
<http://en.wikipedia.org/wiki/Comparison_of_programming_languages>

[28] Wikipedia: Daemon (computer sortware). *Wikipedia, the free encyclopedia*, Duben 2008, [online], [Cit. 15.5.2008].
<http://en.wikipedia.org/wiki/Daemon_(computer_software)>

[29] Wikipedia: glob (programming). *Wikipedia, the free encyclopedia*, Duben 2008, [online], [Cit. 18.4.2008].
<http://en.wikipedia.org/wiki/Glob_(programming)>

[30] Wikipedia: Integer. *Wikipedia, the free encyclopedia*, Duben 2008, [online] , [Cit. 24.4.2008].
<http://en.wikipedia.org/wiki/Integer>

[31] Wikipedia: LiveCD. *Wikipedia, the free encyclopedia*, Duben 2008, [online] , [Cit. 1.5.2008].
<http://cs.wikipedia.org/wiki/LiveCD>

[32] Wikipedia: Regulární výraz. *Wikipedia, the free encyclopedia*, Březen 2008, [online], [Cit. 15.5.2008].
<http://cs.wikipedia.org/wiki/Regulární_výraz>

[33] Wikipedia: Snapshot (computer storage). *Wikipedia, the free encyclopedia*, Duben 2008, [online], [Cit. 15.5.2008].
<http://en.wikipedia.org/wiki/Snapshot_(computer_storage)>

[34] Wikipedia: Souborový systém. *Wikipedia, the free encyclopedia*, Duben 2008, [online], [Cit. 21.4.2008].
<http://cs.wikipedia.org/wiki/Souborový_systém>

[35] ŠVEC, J.: Létající cirkus (13). *Root.cz*, Červen 2002, ISSN 1212-8309, [online], [Cit. 26.4.2008].
<http://www.root.cz/clanky/letajici-cirkus/>

[36] ŠVEC, J.: *Učebnice jazyka Python (aneb Létající cirkus)*. honza@py.cz: PyCZ, Prosinec 2002, ver. 2.2, [online], [Cit. 14.5.2008].
<http://www.pythondocs.ic.cz/tut/tut.html>

Dodatek A

Zkratky v textu a moduly jazyka Python

V této příloze popisuji zkratky použité v předchozím textu. V další části jsou vypsány moduly jazyka Python, které jsem použil během implementace nástrojů.

A.1 Použité zkratky

Použité zkratky jsou uvedeny v této sekci. Pokud jsem někde v textu zkratku použil přímo na místě prvního použití, je vysvětlen její význam. Některé zkratky jsou používány v průběhu celého textu. Proto tento seznam může napomoci při čtení předchozího textu.

API Uživatelské programátorské rozhraní (anglicky Application Programming Interface), zpřístupňuje programové funkce aplikace.

GNU GNU není Unix (anglicky GNU‘s not Unix). Jedná se o rekurzivní zkratku, proto GNU ve vysvětlivce má stejný význam, jako popisovaná zkratka. Označuje projekt jehož smyslem je popularizace a tvorba svobodného operačního systému a sdílení informací v rámci komunity.

GPL Všeobecná veřejná licence (anglicky General Public License). Je součástí projektu GNU. Používá se tedy označení GNU GPL.

KDE K desktopové prostředí (anglicky K Desktop Environment).

LiveCD Označení CD, z kterého je možné spustit OS bez nutnosti jeho instalace na pevný disk.

OS Operační systém

PID Identifikační číslo procesu. Tímto číslem je možné identifikovat konkrétní proces.

SCM Systém správy verzí (anglicky Source Code Management), slouží k ukládání rozdílů mezi verzemi souborů.

A.2 Moduly jazyka Python

V této příloze jsou popsány všechny moduly jazyka Python, které jsem použil v projektu. Moduly jsou v seznamu spolu s popisem jejich činnosti. Popis pochází z dokumentace projektu programovacího jazyka Python [18].

glob Modul poskytuje metodu, která vrátí seznam souborů podle glob syntaxe předané jako vstup. Pro zpracování je použita zjednodušená syntaxe regulárních výrazů. Do výstupu jsou zařazeny soubory, které vrací například příkaz `ls(1)`. Glob syntaxe je uvedena v sekci 3.1.7.

logging Modul pro nastavování a používání jednotného logovacího výstupu.

mercurial Tento balík poskytuje funkce SCM Mercurial[1]. V tomto modulu je obsaženo několik dalších modulů, které umožňují práci s SCM.

- **commands** Třídy v tomto modulu odpovídají běžným příkazům Mercurial SCM. Jedná se o API SCM Mercurial.
- **localrepo** Tato třída reprezentuje samotný repositář.
- **hg** Třída poskytující některé další třídy balíku mercurial. Přes tuto třídu je možné přistupovat k nižším funkcím systému Mercurial.
- **ui** Tento modul poskytuje uživatelské rozhraní pro práci s uživatelskými výpisy.

optparse Modul poskytující funkce pro zpracování parametrů vstupu.

os Modul poskytující možnosti operačního systému, je použit hlavně pro určování zpracování cesty v adresářové struktuře. Jedná se především o třídu modulu *os.path*. Další funkce v tomto modulu jsou vytvoření pevného nebo symbolického odkazu, vytvoření souboru, adresáře a tak podobně.

pyinotify Modul jazyka Python pro subsystém jádra `inotify(7)`, který umožňuje sledovat události na souborech. Pyinotify poskytuje funkce pro nastavení tohoto subsystému a přiřazení funkce zpracovávající tyto události [15].

re Modul poskytující metody pro klasické regulární výrazy

sys Parametry a funkce specifické pro systém. Tento modul slouží jako vrstva pro přístup k objektům používaným nebo udržovaným přímo interpretem Python, případně silně spolupracují s interpretem [18].

threading Modul umožňující použití vláken a jejich synchronizaci.

time Modul poskytuje metody pro práci s časem. V této aplikaci jde především o převod formátu času na běžný, uživatelem čitelný formát.

[1] Použil jsem balík dev-util/mercurial verze 0.9.5-r1

Dodatek B

Instalace nástroje a manuálové stránky

Zde následuje popis jakým způsobem je možné nástroje nainstalovat. Instalace je popsána v následující sekci. V další sekci jsou vypsány názvy manuálových stránek jednotlivých nástrojů.

B.1 Instalace

Nástroj je dostupný na stránkách projektu http://sourceforge.net/projects/snaproll. Balík je dostupný v podobě balíku rpm. V tomto balíku jsou všechny potřebné soubory aplikace. Balík je možné nainstalovat pomocí příkazu `rpm(8)`. Instalace se provádí příkazem `rpm -ivh snaproll-0.3-3.i386.rpm`. Spuštěním příkazu se provede instalace souborů do systému. Tyto soubory jsou rozmístěny do následujících adresářů:

- Hlavní skupina souborů je umístěna v adresáři `/usr/share/snaproll`, kde se nacházejí všechny programové části nástrojů.
- Další soubory jsou umístěny do adresáře `/usr/share/man`, kde jsou umístěny manuálové stránky pro celý systém. Stejně tak jsou zde umístěny i manuálové stránky této aplikace. Výpis těchto stránek bude v následující příloze B.2.
- Do adresáře `/etc/rc.d/init.d` je umístěn startovací skript. Tento skript je použit pro spouštění a ukončování nástroje `snap_notifier(1)`. Během instalace je zároveň tento skript registrován, takže je možné jej spouštět stejně jako ostatní služby systému.
- Dále je také v adresáři `/etc` umístěn soubor `snaproll.conf(5)`, v kterém se nachází nastavení nástrojů.
- Nakonec jsou vytvořeny symbolické odkazy do adresáře `/usr/sbin`. Pomocí těchto odkazů je možné spouštět jednotlivé nástroje aplikace. Popis těchto nástrojů je možné získat například v manuálových stránkách v následující příloze B.2.

Odinstalace balíku ze systému se provádí příkazem `rpm -e snaproll`, který odstraní ze systému soubory tohoto balíku.

B.2 Manuálové stránky nástrojů

Pro uživatele jsou vytvořeny manuálové stránky obsahující popis použití jednotlivých nástrojů. Manuálové stránky jsou dostupné pomocí příkazu `man(1)`. Tento příkaz je použit pro formátování a

zobrazování manuálových stránek na OS GNU/Linux. Stránky jsou po instalaci dostupné v adresáři `/usr/share/man`. Příkaz `man(1)` sám po zadání názvu otevře manuálové stránky. Manuálové stránky k nástrojům z tohoto dokumentu jsou dostupné pod těmito názvy:

- `snap_update(8)`
- `snap_notifier(8)`
- `snap_view(1)`
- `snap_roll(1)`
- `snaproll.conf(5)`
- `snap_create(1)`

Manuálové stránky snaproll.conf(5) popisují formát konfiguračního souboru `/etc/snaproll.conf`. Manuál pro tento soubor je dostupný příkazem `man snaproll.conf`.