VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

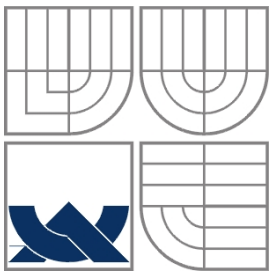
INTELIGENTNÍ HLÍDAČ SOUBOROVÉHO SYSTÉMU

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

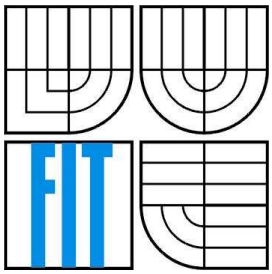
AUTOR PRÁCE
AUTHOR

JAROSLAV MATĚJÍČEK

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

INTELIGENTNÍ HLÍDAČ SOUBOROVÉHO SYSTÉMU

INTELIGENT FILE SYSTEM WATCHER

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

JAROSLAV MATĚJÍČEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. ALEŠ SMRČKA

BRNO 2008

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2007/2008

Zadání bakalářské práce

Řešitel: **Matějíček Jaroslav**
Obor: Informační technologie
Téma: **Inteligentní hlídač souborového systému**
Kategorie: Operační systémy

Pokyny:

1. Prostudujte problematiku nejznámějších souborových systémů, seznamte se s objektově orientovaným přístupem tvorby aplikací.
2. Navrhněte inteligentní systém pro hlídáni dat a zaznamenávání změn v souborovém systému.
3. Implementujte hlídací systém v jazyku C/C++ pro Linux
4. Zhodnoďte dosažené výsledky a navrhněte možné pokračování projektu.

Literatura:

- dle doporučení vedoucího projektu

Při obhajobě semestrální části projektu je požadováno:

- Bez požadavku

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Smrčka Aleš, Ing.**, UITS FIT VUT
Datum zadání: 1. listopadu 2007
Datum odevzdání: 23. ledna 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav inteligentních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. **Pan**

Jméno a příjmení: **Jaroslav Matějíček**
Id studenta: 84244
Bytem: Talichova 428/16, 623 00 Brno
Narozen: 19. 12. 1984, Brno
(dále jen "autor")

a

2. **Vysoké učení technické v Brně**

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
bakalářská práce

Název VŠKP: Inteligentní hlídač souborového systému
Vedoucí/školitel VŠKP: Smrčka Aleš, Ing.
Ústav: Ústav inteligentních systémů
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:
tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☐ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

....

Nabyvatel Autor

Abstrakt

Dokument osahuje stručný úvod do problematiky sledování změn na souborových systémemch, Návrh a implementaci programu určeného pro tento účel.

Klíčová slova

Souborový systém, soubor, adresář, dnotify, inotify, gamin, D-Bus

Abstract

This thesis contains introduction to file system monitoring. Alaylisys and implementation of application designed to do this.

Keywords

File system, file, directory, dnotify, inotify, gamin, D-Bus

Citace

Matějíček Jaroslav: Inteligentní hlídač souborového systému. Brno, 2008, bakalářská práce, FIT VUT v Brně.

Inteligentní hlídač souborového systému

Prohlášení

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

……
Jméno Příjmení
Datum

Poděkování

Rád bych poděkoval Ing. Aleši Smrčkovi za jeho vedení, věcné připomínky a užitečné rady, které mi pomohly při vytváření této práce.

Obsah

1 Úvod

S rostoucí složitostí počítačových systémů přestává být v lidských silách sledovat veškeré důležité probíhající události, proto se problematika automatického sledování dějů v systému, vytváření stručných seznamů a případné automatické reakce na ně stává s postupem času důležitější a důležitější.

Cílem této práce je vytvořit jednoduchou aplikaci, umožňující uživateli sledovat změny v jeho souborech, případně definovat možnosti automatické reakce na tyto události.

Struktura dokumentace

Tato dokunentace je rozčleněna do kapitol, které pojednávají o jednotlivých aspektech bakalářské práce.

Kapitola 1 je kapitola úvodní a má za úkol seznámit čtenáře s tématem a cílem bakalářské práce.

Kapitola 2 obsahuje stručný úvod do použitých technologií. Snahou nebylo poskytnout vyčerpávající informace o technických detailech, ale spíše shrnout jejich charakteristiky a poukázat na přednosti a případné nedostatky.

Kapitola 3 obsahuje souhrn požadavků na aplikaci, jejíž vývoj je cílem této práce a seznam omezení, vyplývajících z použitých technologií.

Kapitola 4 se věnuje návrhu této aplikace a definici formátů vstupů a výstupů.

Kapitola 5 popisuje impementaci návrhu z předchozí kapitoly a detailněji rozebírá princip funkce podstatných částí vytvořené aplikace.

Kapitola 6 tvoří závěr práce a přináší stručný návrh několika možných cest, kterými by se bylo možné ubírat při dalším pokračování této práce.

2 Současné technologie

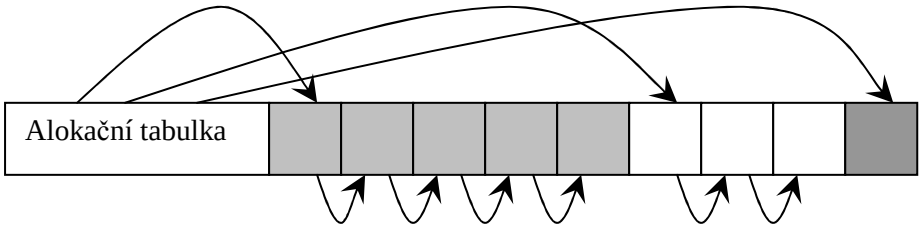
2.1 Souborové systémy

Souborové systémy (file systems) slouží k organizování a vyhledávání souborů dat na discích (pevných, pružných, CD apod.). Cílem je zajistit rychlé a spolehlivé ukládání informací i jejich opětné načítání. Souborový systém slouží k organizování dat uložených na disku. Zaznamenává si informace o fyzickém umístění všech datových položek a zároveň umožňuje uživateli rychlý a spolehlivý přístup k informacím.

Souborové systémy lze dělit podle mnoha různých kritérií. My se je pokusíme rozdělit podle způsobu, jakým si uchovávají informace o uložených datech.

2.1.1 Souborové systémy s alokační tabulkou

Souborové systémy s alokační tabulkou využívají, jak už z názvu vyplývá, pro uložení informací o datech jednu centrální tabulku. V této tabulce jsou uloženy informace o fysickém umístění každého souboru a adresáře a zpravidla další, doplňující informace, jako například datum a čas poslední modifikace souboru, jeho systémové atributy a další.



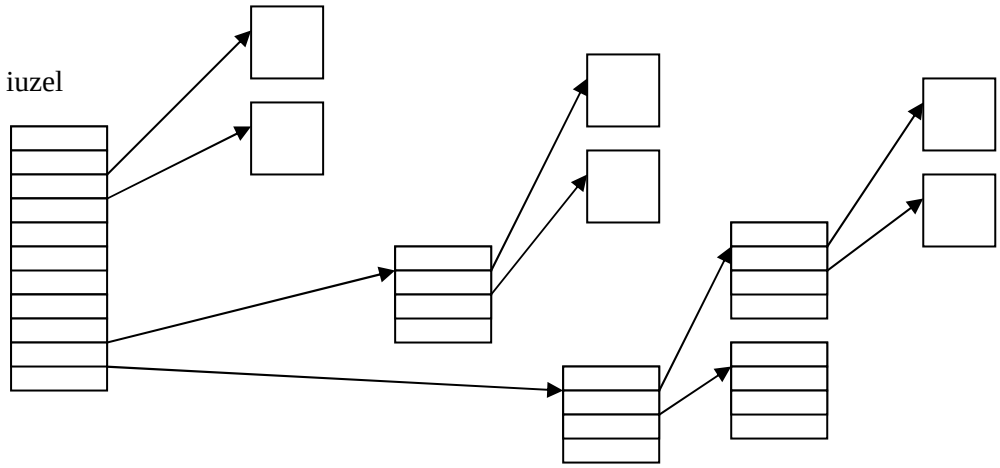
Obrázek 1: Souborový systém s alokační tabulkou.

Na obrázku 1 je naznačeno rozmístění dat v souborovém systému s alokační tabulkou. Na začátku zpravidla bývá alokační tabulka, většinou ve více kopiích, následovaná samotnou datovou oblastí. Kvůli snažšímu adresování dat je tato je rozdělena do logických bloků pevné délky.

Tyto souborové systémy jsou v porovnání s ostatními jednoduché, ale z jejich principu funkce vyplývají některá omezení, například maximální počet uložených souborů.
Nejznámější souborové systémy tohoto typu jsou Fat, Fat32 z dílny firmy Microsoft.

2.1.2 Iuzlové souborové systémy

Tento typ souborových systémů, používaný především na operačních systémech unixového typu, používá k adresaci dat hierarchické uspořádání datových struktur, nazývaných iuzly (inode). Každý soubor a adresář má minimálně jeden iuzel, obsahující informace důležité pro práci s ním, jako je jméno, systémové atributy, v případě, že se jedná o adresář, tak adresy všech uzlů náležících k jeho podadresářům a další. Velké soubory mohou obsahovat tzv. nepřímo adresované bloky, to znamená že mezi uzlem popisujícím soubor a hledaným blokem dat, se může nacházet ještě několik dalších uzlů.



Obrázek 2: Iuzlový souborový systém.

2.1.3 Ostatní souborové systémy

Síťové souborové systémy

Síťové souborové systémy jsou souborové systémy, které se chovají jako klient pro protokol vzdáleného přístupu, nabízející přístup k souborům na serveru - například NFS a SMB.

Databázové souborové systémy

Nový koncept pro práci se soubory je koncept databázově orientovaného souborového systému. Namísto, nebo navíc, k hierarchické struktuře uložení dat jsou soubory identifikovány jejich charakteristickými rysy, jako např. typ souboru nebo autor. Příkladem takovéhoto souborového systému je dbfs.

Ploché souborové systémy

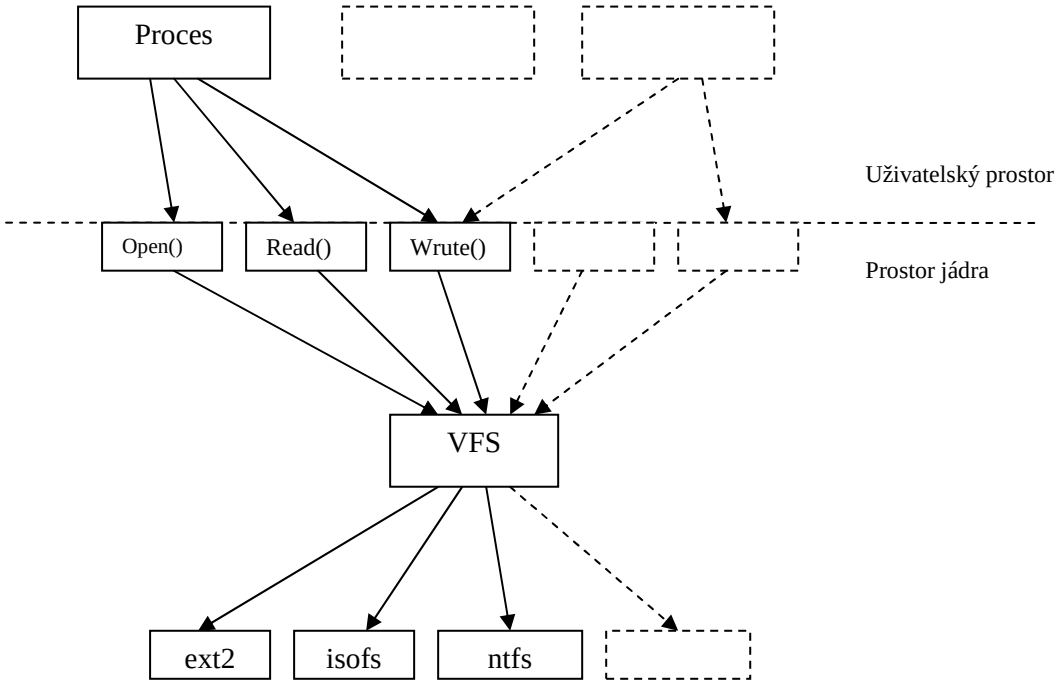
V plochém souborovém systému neexistuje hierarchická struktura a všechny data jsou uložena na stejné úrovni. Tento systém je sice jednoduchý, ale pro práci s větším počtem souborů nepříliš vhodný.

Jako mnoho malých systémů před ním, měl původní Apple Macitosh plochý souborový systém, nazývaný MFS. Verze Mac OS, která s tímto souborovým systémem pracovala, vytvářela iluzi částečně hierarchického systému nad MFS. To ale znamenalo, že každý soubor na disku musel mít jedinečné jméno, přestože se soubory zobrazovaly uživateli v různých adresářích.

2.2 Virtuální souborový systém

Virtuální souborový systém (VFS) je abstrakční vrstva obalující konkrétní implementace jednotlivých souborových systémů. Účelem VFS je umožnit klientským aplikacím přístupovat k různým druhům souborových systémů jednotnou cestou.

VFS specifikuje rozhraní mezi jádrem operačního systému a konkrétním souborovým systémem. Díky tomu je snadné přidat podporu pro nový souborový systém, stačí, aby tento splňoval dané rozhraní.



Obrázek 3: Virtuální souborový systém.

2.3 Sledování změn na souborových systémech

V průběhu vývoje linuxového jádra byly vytvořeny dva systémy umožňující informovat klientské aplikace o událostech, k nimž došlo na souborovém systému.

2.3.1 Dnotify

Dnotify je součástí systémového volání fcntl() představeného ve verzi jádra 2.4. Hlavním důvodem vytvoření tohoto systému bylo dát uživatelským aplikacím možnost reagovat na změny dat bez nutnosti je stále dokola načítat. Jedná se o adresářově orientovaný systém, umožňující sledovat pouze celé adresáře a nikoli jednotlivé soubory. Dále je možno rozhodnout, o kterých událostech chce být aplikace informována.

Definované události jsou:

- DN_ACCESS Soubor v adresáři byl otevřen pro čtení
- DN_MODIFY Soubor v adresáři byl modifikován
- DN_CREATE V adresáři byl vytvořen nový soubor
- DN_DELETE V adresáři byl smazán soubor
- DN_RENAME V adresáři byl přejmenován soubor
- DN_ATTRIB Souboru v adresáři byly změněny atributy

Dnotify potřebuje mít pro každý sledovaný adresář otevřen jeden popisovač souboru (file descriptor), proto se ukázal být nevhodný pro sledování většího počtu adresářů zaráz. V současnosti byl nahrazen systémem inotify a v linuxovém jádře je udržován pouze z důvodu kompatibility.

2.3.2 Inotify

Inotify vznikl jako náhrada za starší systém Dnotify, byl vyvíjen s důrazem na odstranění nedostatků starého systému. Do linuxového jádra byl přidán ve verzi 2.6.13, do starších jader mohl být doplněn pomocí vydaného patche.

Inotify má mnoho výhod oproti staršímu dnotify. U staršího systému bylo pro každý sledovaný adresář nutné mít otevřen vlastní popisovač souboru. Tím byl dán strop maximálního možného počtu sledovaných adresářů, protože linuxové jádro umožňuje procesu mít otevřen pouze omezený počet popisovačů souboru. Dále se ukázal problém při použití zařízení s výměnnými médii, která nemohla být odpojena, protože v systému byly stále přítomny otevřené popisovače souborů. U inotify stačí mít otevřen jediný popisovač souboru pro příjem dat a blokování zařízení již nehrozí.

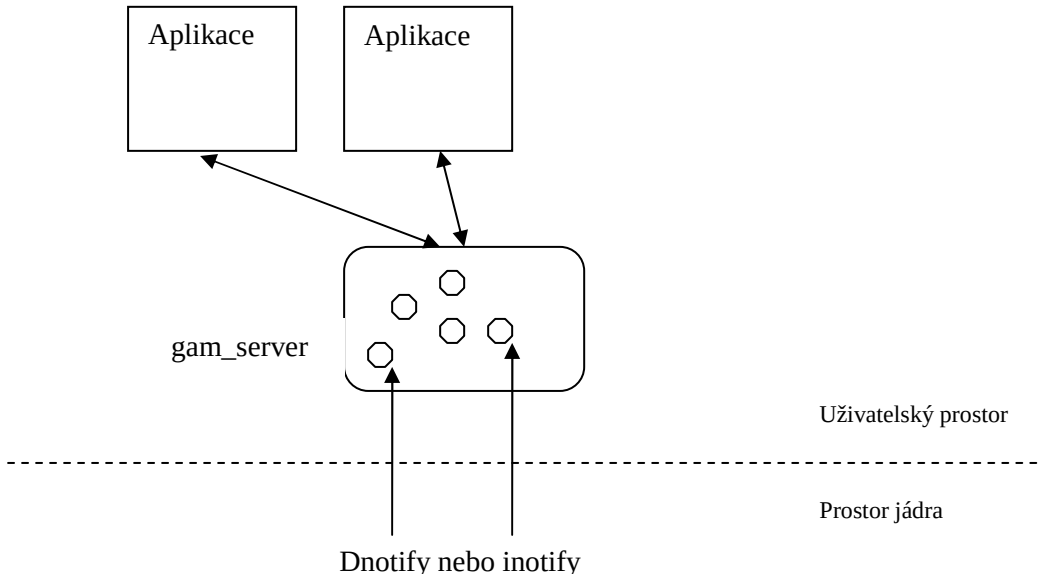
Dalším nedostatkem dnotify bylo omezené rozlišení, protože bylo možno zaznamenávat změny pouze na úrovni celých adresářů. Pro získání podrobnějších informací bylo potřeba držet v paměti stavové informace a při zachycení události porovnat nový stav s dříve uloženým. Tato nutnost při použití inotify odpadá, protože systém posílá mnohem podrobnější informace o proběhlých změnách.

Události inotify:

- IN_ACCESS Soubor byl otevřen pro čtení
- IN_ATTRIB Souboru byly změněny atributy
- IN_CLOSE_WRITE Soubor otevřený pro zápis byl zavřen
- IN_CLOSE_NOWRITE Soubor, který nebyl otevřen pro zápis, byl zavřen
- IN_CREATE Ve sledovaném adresáři byl vytvořen soubor/adresář
- IN_DELETE Ve sledovaném adresáři byl smazán soubor/adresář
- IN_DELETE_SELF Sledovaný soubor/adresář byl smazán
- IN_MODIFY Soubor byl modifikován
- IN_MOVE_SELF Sledovaný soubor/adresář byl přesunut
- IN_MOVED_FROM Soubor byl přesunut ze sledovaného adresáře
- IN_MOVED_TO Do sledovaného adresáře byl přesunut soubor
- IN_OPEN Soubor byl otevřen

2.3.3 Knihovna Gamin

Gamin je knihovna zastřešující jaderné rozhraní inotify a dnotify a zpřístupňující aplikacím jednotnou cestu k získávání informací o dění na souborovém systému.



Obrázek 4: Gamin. [1]

Vnitřně obsahuje gam_server různé datové struktury jako například:

- Strom monitorovaných uzlů složený z datových struktur typu GamNode obsahující informace jako je cesta, seznam odbíratelů událostí a další
- Data spojení pro každý otevřený socket. V této struktuře jsou obsaženy všechny potřebné informace pro dané spojení (file descriptor, pid aplikace etc.) a fronta případných nezpracovaných událostí.
- Pro každý sledovaný soubor nebo adresář strukturu GamSubscriptions se seznamem odebiratelů událostí.

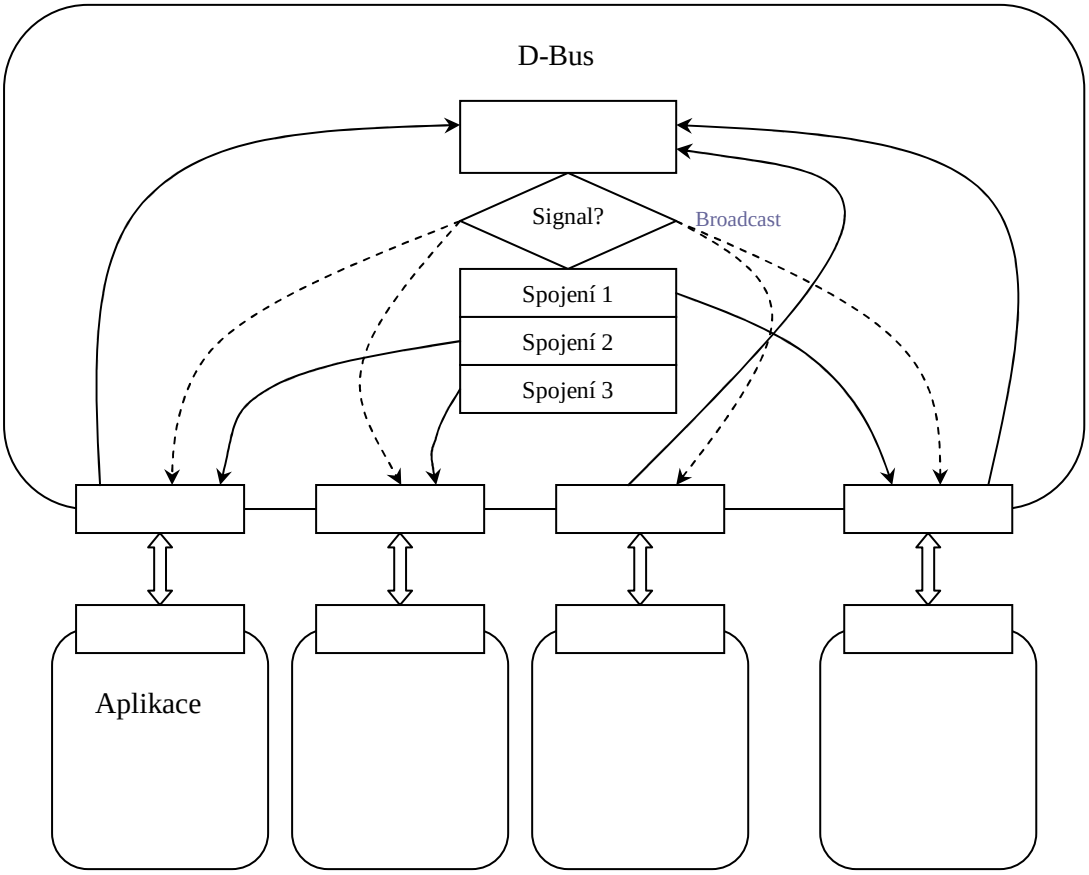
2.4 Sběrnice D-Bus

D-Bus je softwarový projekt, který nabízí jednoduchou cestu ke komunikaci mezi aplikacemi. Je vyvíjen jako součást freedesktop.org projektu.

D-Bus umožnuje aplikacím se zaregistrovat a nabízet své služby ostatním, stejně tak umožňuje klientským programům možnost vyhledat, které služby jsou dostupné a využívat je. Dále je možno se přes tento systém přihlásit k odběru některých zpráv jádra jako např. informace o připojení, odpojení hardwaru.

D-Bus je implementován jako démon, který může běžet v několika instancích. Každá z těchto instancí se nazývá kanál, obvykle je přítomna jedna privilegovaná instance nazývaná systémový kanál a soukromá instance pro každého přihlášeného uživatele. Instance pro jednotlivé uživatele budou potřebné hlavně do budoucna, protože při vývoji sběrnice D-Bus se počítá se zavedením různých omezení pro využívání systémového kanálu.

Hlavním úkolem systémového kanálu je doručovat zprávy systémových démonů jednotlivým aplikacím, které je požadují. Úkolem soukromých kanálů je nabízet aplikacím možnost komunikace bez jakýchkoli omezení.[2]



Obrázek 5: architektura sběrnice D-Bus.

D-Bus umožnuje aplikacím komunikovat pomocí tří různých vrstev:

- Knihovna *libdbus*, umožnující spojení a výměnu zpráv mezi aplikacemi.
- Démona postaveného na knihoně *libdbus*, který umožnuje propojení více aplikací.
- Obalující knihovny založené na zvléštních aplikačních rámcích.

3 Specifikace požadavků

V této kapitole se budeme věnovat specifikaci požadavků na jednoduchou aplikaci umožňující zaznamenávat proběhlé změny v souborovém systému a reagovat na ně.

Požadavky:

- Vytvořit systém umožňující sledování změn na souborovém systému.
- Systém bude sledovat jednotlivé soubory a adresáře a zaznamenávat události
- V reakci na událost bude možno spustit externí program
- Systém bude využívat služeb knihovny gamin
- Systém bude možno nastavit pomoci konfiguračního souboru
- V konfiguračním souboru bude možno zapnout rekurzivní sledování adresáře
- V konfiguračním souboru budou definovány metaznačky umožňujíci při spouštění externího programu předat jako parametry základní informace o události (jméno souboru a definovaný text označující k jaké události došlo)
- V konfiguračním souboru bude možno u cest k jednotlivým sledovaným souborům nebo adresářům použít zástupný znak '*' reprezentující libovolný adresář
- Program bude umožňovat odesílání informací o zachycených událostech přes sběrnici D-Bus

Známá omezení:

Z použitých technologií vyplývají jistá omezení

- Na jádrech starších než 2.6.13 bude omezen maximální počet zároveň sledovaných souborů
- Vzheldem ke změnám v api knihovny python-dbus je pro funkčnost systému zasílání zpráv přes sběrnici D-Bus vyžadována minimálně verze 0.82
- Na síťových souborových systémech se dá předpokládat, že nebudou zachyceny všechny události, toto však záleží na konkrétní implementaci síťového souborového systému.

4 Návrh řešení

Program bude obsahovat pro každý sledovaný soubor (adresář) samostatnou třídu obsahující informace o konfiguračním nastavení pro tento soubor (na které události reagovat, co při příchodu události udělat…).
Tyto třídy budou uloženy ve stromové struktuře odpovídající struktuře uložení na disku.
Dále bude obsahovat statickou třídu zajišťující komunikaci s knihovnou gamin. Tato třída bude dále rozdělovat jednotlivé události příslušným instancím tříd pro jednotlivé soubory nebo adresáře.

Při spuštění programu dojde k načtení konfiguračního souboru, pokud je konfigurace chybná, dojde k vypsání chyby na standartní výstup ukončení programu.
V případě, že se povedlo načíst konfigurační soubor, dojde k vytvoření tříd pro jednotlivé sledované soubory a adresáře a inicializaci vlastního sledovacího systému.

Pří příchodu události se provolá příslušná metoda ve třídě patřící k souboru, u něhož došlo k nějaké akci. V této metodě se událost zpracuje následujícím postupem:

1. kontrola, zda se na událost má vůbec reagovat
2. pokud je nastaveno, tak uložení informace do systémového záznamu
3. pokud je nastaveno, tak odeslání události do sběrnice D-Bus
4. pokud je nastaveno, tak spuštění externího programu s příslušnými parametry

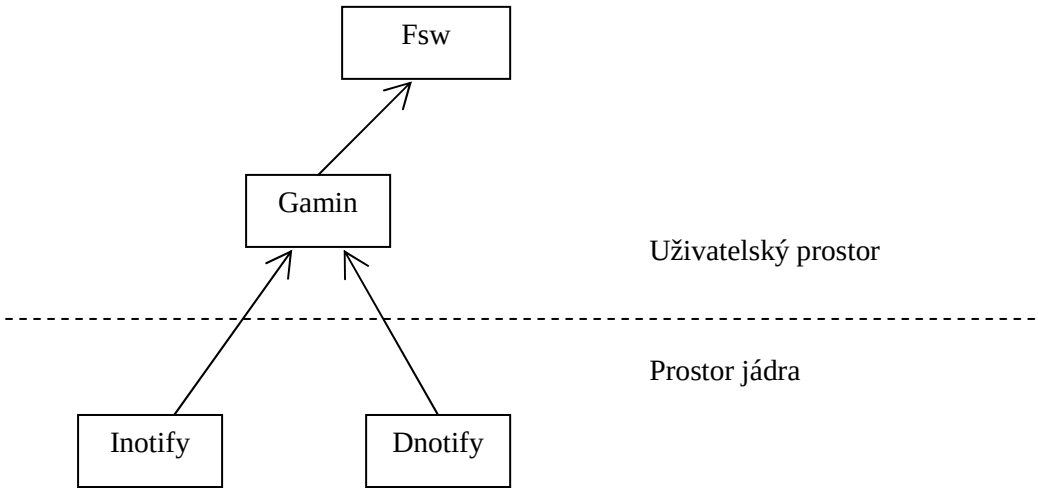
Dále bude systém reagovat na následující signály operačního systému:

- SIGHUP znovu načte konfiguraci.
- SIGTERM uvolní veškerou používanou paměť, uzavře otevřené popisovače souboru a ukončí běh programu.

Ostatní signály budou ignorovány.

4.1 Architektura řešení

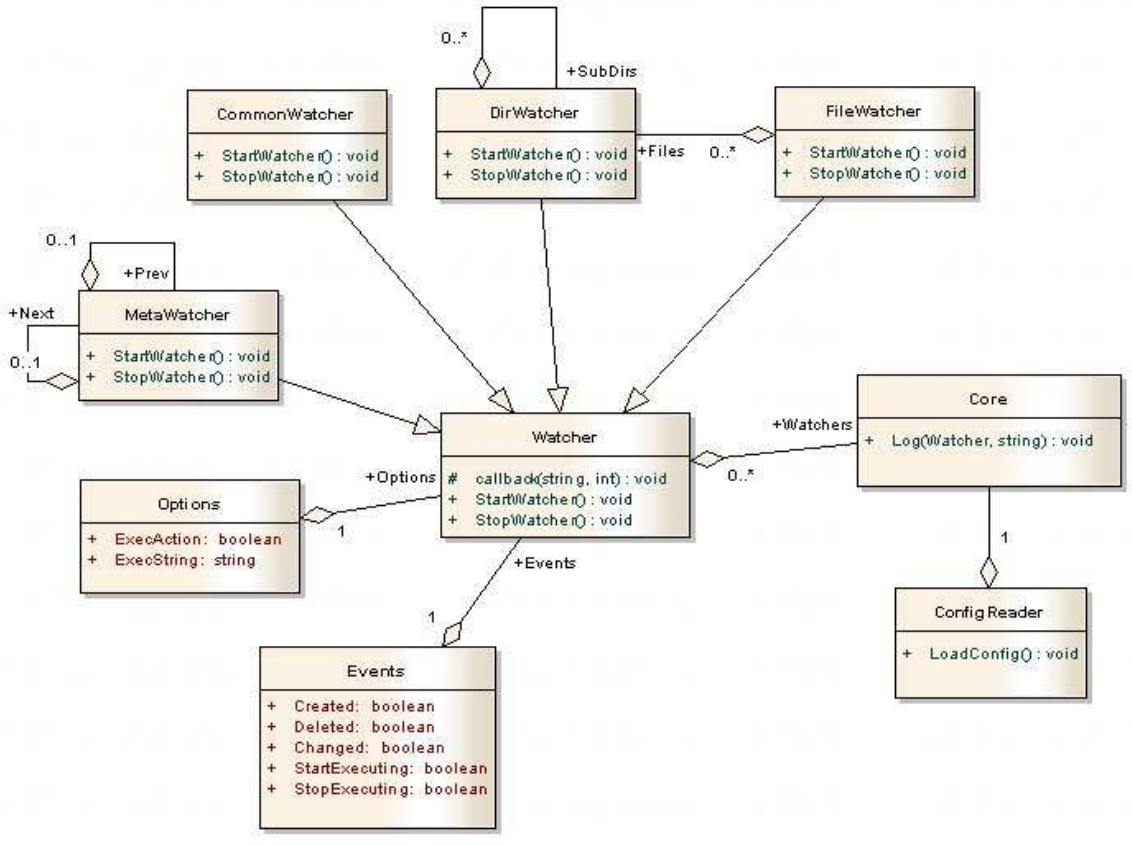
Program poběží v uživatelském prostoru systému a s jádrem nebude komunikovat přímo, ale bude využívat knihovnu gamin jako mezivrstvu sjednocující přístup k systému inotify a dnotify, jak je naznačeno na obrázku 6.



Obrázek 6: Architektura řešení.

4.2 Návrh tříd

Program bude vytvořen pomoci metod objektově orientovaného programování. Na následující obrázku jsou znázorněny vztahy mezi jednotlivými třídami programu. Z obrázku jsou pro větší přehlednost vynechány některé metody.



Obrázek 7: Zjednodušený diagram tříd

Core

Třída vyskytující se v programu pouze v jedinné instanci, uchovává základní nastavení…

Watcher

Bázová třída pro všechny třídy jednotlivých sledovačů.

Instance této třídy se v systému nevyskytují, pouze instance jejích potomků.

FileWatcher

Třída obsahující informace o konkrétním sledovaném souboru.

Při zachycení události DELETED vytvoří instanci třídy CommonWatcher a svou činnost ukončí.

DirWatcher

Třída obsahující informace o konkrétním sledovaném adresáři.

Při zachycení události DELETED vytvoří instanci třídy CommonWatcher a svou činnost ukončí.

CommonWatcher

Speciální třída pro ošetření případu, kdy je v konfiguračním souboru zadefinována cesta k neexistujícímu souboru/adresáři. Má speciální reakci na událost CREATED, po příchodu této události zjistí, zda byl vytvořen soubor nebo adresář, k němu vytoří příslušnou sledovací třdu (FileWatcher, DirWatcher) a sama sebe odstraní.

MetaWatcher

Třída určená k zajištění funkčnosti sledování souborů jejichž umístění, zadané v konfiguračním souboru, obsahuje zástupné znaky.

Events

Třída sloužící pro uložení filtru událostí, na které se má reagovat.

Options

Třída pro uchování informací o ostatních nastaveních pro jednotlivé sledované soubory/adresáře.

4.3 Formát konfiguračního souboru

Konfigurační soubor je standartní textový soubor obsahující informace nutné pro běh programu. Jeho defaultní umístění je /etc/fsw.conf. Programu může být spouštěcím parametrem předána jiná cesta.

Příklad konfiguračního souboru:

```
#komentar
[GLOBAL]
Logfile = /var/log/filelog.log
DBus = True

[DEFAULT]
```

```
Events = all

[/home/*/.bash_history]
Events = deleted, created

[/root/]
Recursive = True

[/etc/passwd]
Run = echo “%p %e” | mail my@adress.com
```

Všechny znaky na řádku následující po znaku # jsou považovány za komentáře a na nastavení programu nemaji vliv.

4.3.1 Sekce GLOBAL

Povinná část konfiguračního souboru obsahující informace o základním nastavení.

Volby:

LogFile = <cesta k systémovému záznamu>

Touto volbou je specifikována cesta k souboru, do kterého budou ukládány záznamy. Pokud není volba uvedena, informace se nezaznamenávají do žádného souboru.

DBus = [True/False]

Touto volbou je specifikováno, zda se informace o zachycených událostech mají posílat přes sběrnici DBus. Pokud není volba uvedena, informace se do sběrnice DBus neposílají.

4.3.2 Sekce DEFAULT

Volitelná část konfiguračního souboru. Může obsahovat stejné hodnoty jako části příslušející k jednotlivým souborům/adresářům. Definuje jejich základní hodnoty, které mohou, ale nemusí, být přepsány u jednotlivých souborů. Vhodná obzvláště v případě, že u většího množství sledovaných souborů/adresářů chceme používat stejné nastavení.

Volby:

Events = <nastaveni filtru zaznamenávaných událostí>

Touto volbou se specifikují události, na které má být reagováno.

all – bude se reagovat na všechy události

nebo libovolná kombinace následujících oddělených čárkou

created – soubor byl vytvořen

deleted – soubor byl smazán

changed – soubor byl změněn

Pokud není nastaveno reaguje se na všechny události.

Recursive = [True/False]

Určuje, zda se má adresář sledovat včetně všech jeho podadresářů a souborů v nich. U nastavení jednotlivých souborů se tato volba ignoruje. Pokud není volba uvedena, rekurzivní sledováni je vypnuto.

Run = <příkaz>

Touto volbou lze specifikovat příkaz, který má být při zachycení události spuštěn. Pokud se v těle příkazu nachází řetězce %p nebo %e, jsou před vykonáním příkazu nahrazeny podle následujícího vzoru:

%p bude nahrazeno úplnou cestou k souboru nebo adresáři u něhož k události došlo.

%e bude nahrazeno textovým řetězcem reprezentujícím událost.

Changed – soubor/adresář byl modifikován

Created – soubor/adresář byl vytvořen

Deleted – soubor/adresář byl smazán

4.3.3 Sekce jednotlivých sledovaných souborů/adresářů

Následují sekce pro nastavení jednotlivých sledovaných souborů a adresářů. Cesta k danému souboru nebo adresáři musí být uvedena v hranatých závorkách jak je vidět na příkladu, cesta může obsahovat jeden zástupný znak ‘*’ (nikoli však úplně na konci), který bude reprezentovat libovolný existující nebo v budoucnu vytvořený adresář.

Všechny volby, které mohou být uvedeny v této sekci jsou stejné jako u sekce DEFAULT..

4.4 Spouštěcí parametry

Program bude možno spustit s parametry, kterými lze specifikovat cestu ke konfiguračnímu souboru, spustit program na pozadí, nebo si nechat vypsat základní informace.

Seznam parametrů:

-c –configfile <cesta ke konfiguračnímu souboru>

V případě, že bude program spuštěn s tímto parametrem, použije se konfigurační soubor ze zadaného umístění misto implicitního.

-d –daemon

Pokud bude zadán tento parametr, program bude běžet na pozadí jako démon.

-v –version

Pokud bude zadán tento parametr, dojde k vypsání verze a program se ukončí.

-h –help

Pokud bude zadán tento parametr, dojde k vypsání krátké nápovědy a program se ukončí.

4.5 Formát systémového záznamu

Systémový záznam slouží k ukládání informací o zachycených událostech, informací o běhu programu a o případných chybách. Každý záznam bude obsahovat informaci o čase jeho vytvoření, úplnou cestu a textový řetězec reprezentující událost.

Ukázka systémového záznamu:

```
Tue Jan 22 14:05:04 2008    /home/matej/fsw/a       Created
Tue Jan 22 14:05:22 2008    /home/matej/fsw/a       Changed
Tue Jan 22 14:05:47 2008    /home/matej/fsw/a       Deleted
Tue Jan 22 14:05:04 2008    /home/matej/Desktop/a   DirCreated
```

4.6 Rozhraní pro sběrnici D-Bus

Každá aplikace, která chce odebírat zprávy musí na sběrnici D-Bus nabízet rozhraní specifikované následujícim xml.

```
<!DOCTYPE node PUBLIC "-//freedesktop//DTD D-BUS Object Introspection
1.0//EN" "http://www.freedesktop.org/standards/dbus/1.0/introspect.dtd">
<node name="/FswObject">
  <interface name="org.freedesktop.DBus.Introspectable">
    <method name="Introspect">
      <arg direction="out" type="s" />
    </method>
  </interface>
  <interface name="cz.vutbr.fit.fsw">
    <method name="FswEvent">
      <arg direction="in"  type="s" name="path" />
      <arg direction="in"  type="s" name="event" />
    </method>
  </interface>
</node>
```

5 Implementace

Pro implementaci dané aplikace byl zvolen jazyk python, jedná se o moderní programovací jazyk vyvinutý Guido van Rossumem. Tento jazyk díky vyšší abstrakci umožňuje vytvářet aplikace mnohem rychleji než tradiční programovací jazyky jako např. jazyk C.

5.1 Postup při načítaní konfiguračního souboru

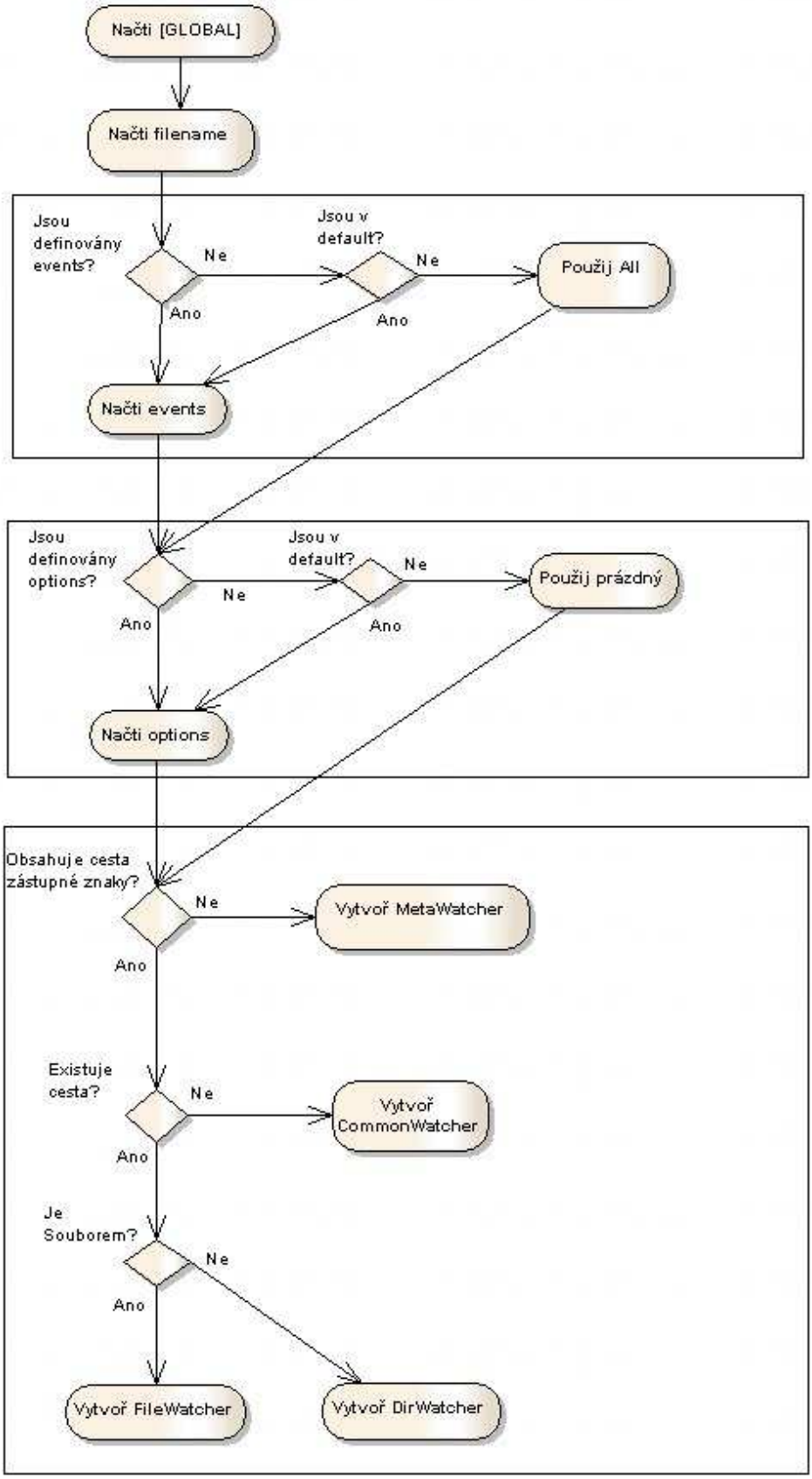
Po spuštění programu dojde nejdříve k načtení konfiguračního souboru následujícím způsobem:
načte se sekce [GLOBAL]

Postup načítání se dá logicky rozdělit na několik kroků, které jsou naznačeny na obr. 8.

- V části 1 se načítá nastavení filtru událostí pro daný sledovač. Nejdříve se jej pokusíme najít přímo v sekci daného souboru (adresáře), pokud v této sekci není definován, pokusíme se jej najít v sekci DEFAULT; pokud není definován ani tam, použije se defaultní nastavení (tzn. reakce na všechny možné příchozí události).

- V části 2 se načítají ostatní nastavení pro daný soubor nebo adresář postupem vpodstatě totožným s načítáním filtru událostí.

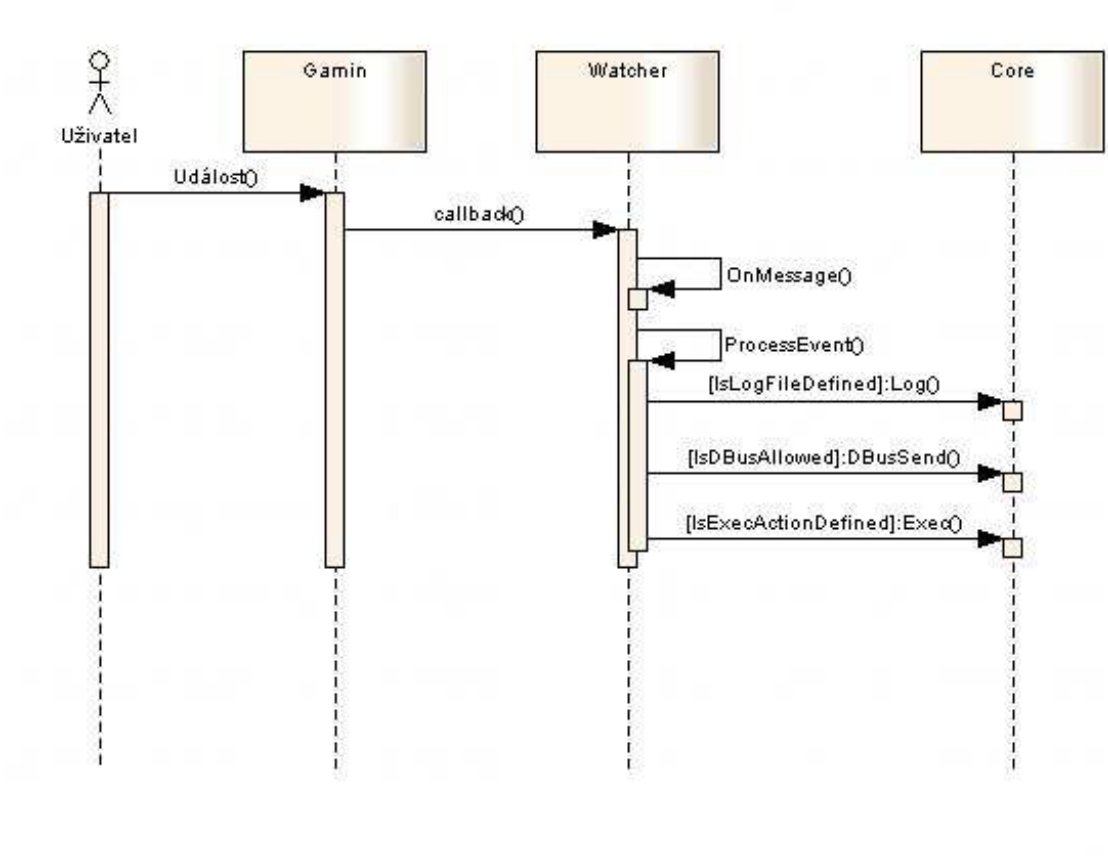
- V části 3 již máme z konfiguračního souboru načteny všechny potřebné informace a můžeme začít s inicializací vlastních tříd. Nejprve otestujeme, zda daný soubor nebo adresář vůbec existuje, pokud ne, vytvoříme instanci třídy CommonWatcher. Pokud daná cesta existuje, otestujeme, zda se jedná o soubor nebo adresář a vytvoříme instanci příslušné třídy.



Obrázek 8: Diagram aktivit při načítání konfiguračního souboru.

5.2 Postup při zpracování události

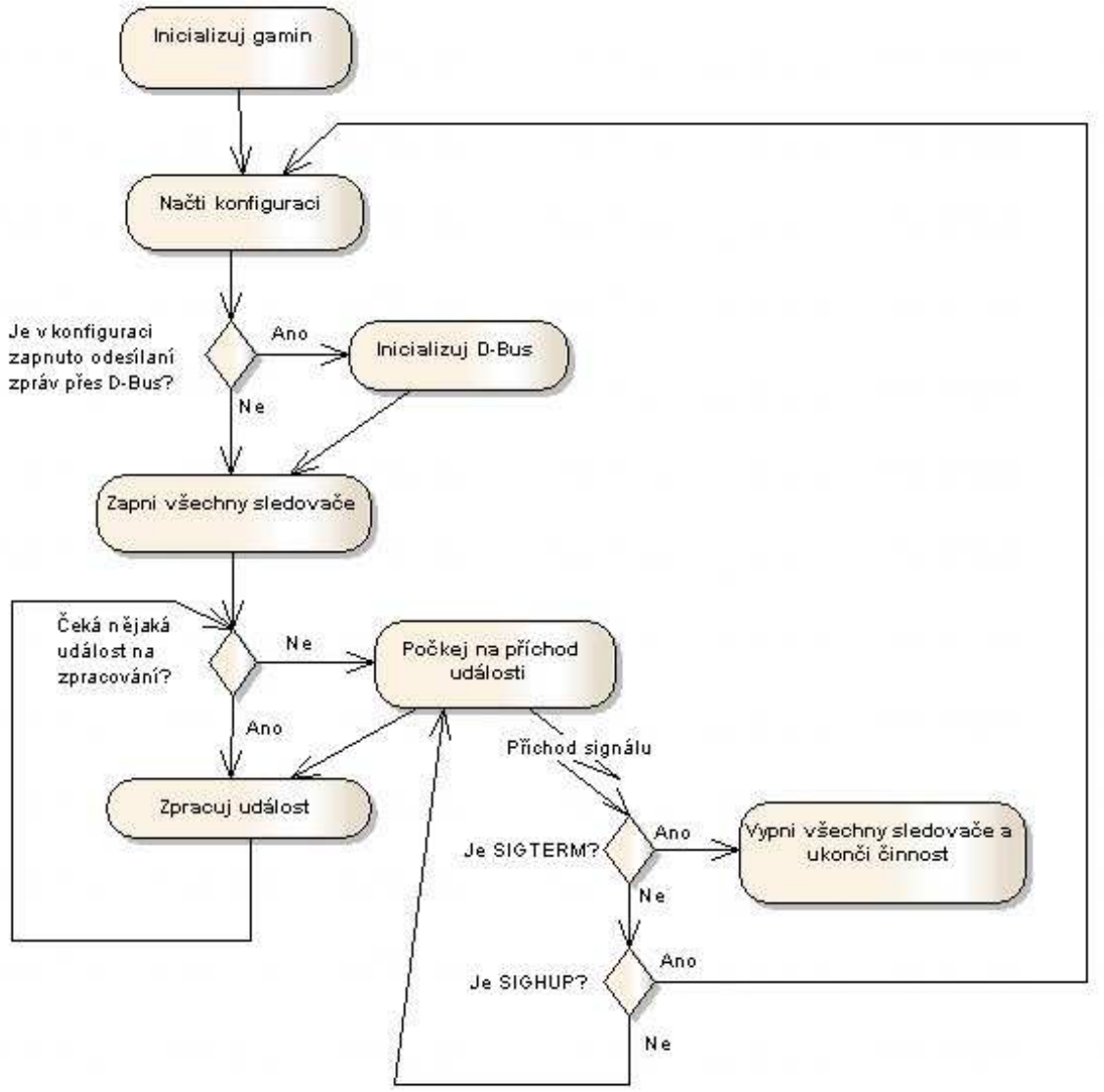
Na obrázku 9 je znázorněn postup činností programu při příchodu události. Průchod přes jádro operačního systému pro nás není důležitý a kvůli přehlednosti je z diagramu vynechán.



Obrázek 9: Diagram postupu při zpracování příchozí události.

1. Při příchodu události od jádra operačního systému a jejím zachycením knihovnou gamin je provolána metoda callback() třídy Watcher
2. Poté třída Watcher zavolá svoji vlastní metodu OnMessage(), díky polymorfizmu dojde k provolání metody definované u potomka této třídy a zpracuje se systémová reakce na událost nezávislá na uživatelském nastavení.
3. Zkontoluje se, zda je uživatelem nastaveno, že se na danou událost má reagovat. Pokud ne, obsluha události v tomto bodě končí. Pokud ano, pokračuje se k dalšímu bodu.
4. Zkontroluje, se zda je nastaveno ukládání informací do systémového záznamu a pokud ano, dojde k zapsání události.
5. Zkontroluje se, zda je nasaveno odesílání událostí přes sběrnici D-Bus a pokud ano, dojde k odeslání informací o proběhlé události.
6. Zkontroluje se, zda je nastaveno spuštění externího programu v reakci na událost a pokud ano, dojde k nahrazení zástupných znaků ve spouštěcím příkazu a jeho spuštění.

5.3 Hlavní smyčka programu



Obrázek 10: Diagram aktivit hlavní smyčky programu.

Po spuštění programu je vytvořena instance třídy Core a provolána její metoda Main()
Další postup se dá vyjádřit následující sekvencí kroků.

1. Dojde k inicializaci knihovny gamin, v případě, že inicializace z nějakého důvodu selže (např. knihovna není v systému naistalována), dojde k vypsání chyby a ukončení činnosti.
2. Program otestuje existenci konfiguračního souboru, pokud tento neexistuje, vypíše chybu. Pokud existuje, dojde k načtení konfigurace.

3. Zjistí se, zda je v konfiguračním souboru nastaveno odesílání informací o zachycených událostech přes sběrnici D-Bus a pokud ano, tato se inicializje.
4. Projde se seznam všech sledovačů a provolá se na nich metoda StartWatch()
5. Čeká se na příchod události a tato se zpracuje.

Při příchodu signálu SIGTERM dojde k zastavení činnosti všech sledovačů a ukončení programu.
Při příchodu signálu SIGHUP dojde k návratu k bodu 2.
Z důvodu problémů se zachycením signálů při blokujícím čtení je v hlavní smyčce použita funkce select(), která příchozí signály neblokuje.

6 Závěr

Tato práce se snažila poskytnout náhled na možnosti sledování změn na souborových systémech. Jejím cílem bylo navrhout jednoduchou aplikaci umožňující uživateli bez hlubších znalostí systému získávat přehled o dění na jeho disku, případně pokročilejším uživatelům dát možnost vytvářet automatické reakce na proběhlé změny.

6.1 Možné pokračování projektu

V této kapitole navrhneme možné pokračování vývoje inteligentního hlídače souborového systému, se záměrem usnadnit jeho používání a rozšířit jeho možnosti.

Rozšíření kódu

- Regulární výrazy v cestách k souborům

  Umožnit uživateli při konfiguraci používat v cestách k jednotlivým sledovaným souborům a adresářům použití regulárních výrazů.

Další rozšíření

- Sada skriptů pro interakci s verzovacím systémem

  Vytvořit sadu skriptů schopných v reakci na události uložit změněný soubor do verzovacího systému, nahradit změněný nebo smazaný soubor původní verzí.

- Sada apletů pro notifikaci uživatele

Vytvořit aplety pro různá grafická prostředí, které by umožnovaly přímou notifikaci uživatele o změnách v důležitých souborech. Ke komunikaci s aplikací fsw by se používala sběrnice D-Bus.

Použitá literatura

[1] WWW Stránky – Gamin home page
http://www.gnome.org/~veillard/gamin/internals.html

[2] WWW Stránky – D-Bus Specification
http://dbus.freedesktop.org/doc/dbus-specification.html

Seznam příloh

Příloha 1: Seznam chybových kódů

Obsahem této přílohy je seznam všech chybových hlašení které může program vypsat, stručný popis jejich přícin a postup řešení.

101: cannot load gamin
Nepodařilo se načíst knihovnu gamin.
Řešení: zkontrolujte zda je knihovna gamin správně nainstalována a zda jsou správně nastaveny vyhledávací cesty.

102: no config file
Nelze nalézt konfigurační soubor.
Řešení: zkontrolujte, zda je konfigurační soubor na správném místě.

103: cannot load config file
Nelze načíst konfigurační soubor.
Řešení: zkontrolujte, zda máte dostatečná práva ke čtení konfiguračního souboru.

201: no GLOBAL section in config file
Konfigurační soubor neobsahuje sekci [GLOBAL].
Řešení : do konfiguračního souboru doplňte sekci [GLOBAL].

Příloha 2: Manuálová stránka programu

fsw.py(1) fsw.py(1)

JMÉNO

fsw – inteligentní hlídač souborového systému

POUŽITÍ

fsw.py [-c=FILE] [--configfile=FILE] [-d] [--daemon] [-v] [—version] [-h] [--help]

POPIS

Program slouží ke sledování změn v soubrech a adresářích na disku, umožnuje ukádat informace o těchto změnách do souboru, odesílat informace přes sběrnici D-Bus a v reakci na změny spouštět externí programy

VOLBY

-c=FILE --configfile=FILE

Použije zadaný konfigurační soubor

-d –daemon

Program bude spuštěn jako démon

-v --version

Vypíše verzi a skončí

-h –help

Vypíše nápovědu a skončí

Příloha 3: Ukázka propojení přes sběrnici D-Bus

Pro demonstaci propojení přes sběrnici D-Bus jsem vytvořil jednoduchého démona který poskytuje patřičné rozhraní pro příjem událostí z programu fsw a předává je dále grafickému prostředí GNOME přes jeho notifikační systém. Události se pak zbrazují v notifikační oblasti pracovní plochy.

Démon se nachází na přiloženém CD v adresáři /dbusfswd

Pro jeho běh je nutná verze knihovny python-dbus 0.82 nebo vyšší.