

Universitatea “Politehnica” din București
Facultatea de Electronică, Telecomunicații și Tehnologia Informației
Universitatea Babeș-Bolyai
Facultatea de Matematică și Informatică

Sistem de fișiere cu un set extins de permisiuni

Proiect de diplomă

prezentat ca cerință parțială pentru obținerea titlului de
Inginer în domeniul *Calculatoare și tehnologia informației*
programul de studii de licență *Ingineria informației*

Conducător științific

Lect. Dr. Adrian STERCA

Absolvent

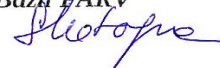
Nicolae NĂTRĂPEIU

2014

Universitatea "Politehnica" din București
Facultatea de Electronică, Telecomunicații și Tehnologia Informației
Universitatea Babeș-Bolyai
Departamentul * _____ Informatică _____

Aprobat Director de Departament *:

Dr. Bazil PÂRV



TEMA PROIECTULUI DE DIPLOMĂ
a studentului (nume, inițiala tatălui, prenume, grupă) NĂTRĂPEIU Gh. Nicolae 843

1. Titlul temei: **Sistem de fișiere cu un set extins de permisiuni**

2.

Lucrarea de față prezintă modul de implementare a unui sistem de fișiere în contextul sistemelor de operare de tip Unix, mai specific a kernelului de Linux.

Pentru aceasta s-a folosit modelul de dezvoltare bazat pe FUSE care permite dezvoltarea sistemelor de fișiere în spațiul utilizatorului și nu în cel al nucleului.

Principalele avantaje oferite de această abordare sunt: un nivel ridicat al flexibilității prin posibilitatea utilizării în proiectare a unor concepte extrem de avansate, specifice limbajelor de programare de nivel înalt, o cuplare cât mai mică cu nivelul hardware și totodată o oarecare independență față de sistemul de operare.

Proiectul propune o metodă de extindere a configurabilității sistemului de acces prin specificarea mult mai precisă și concretă a acestor drepturi chiar din interiorul sistemului de fișiere. Dintre posibilitățile oferite menționăm: configurarea permisiunilor având în vedere protocolul de acces (SSH, tty ș.a.), adresa IP a utilizatorului (sau a unui set de adrese IP), anumite intervale orare și mai ales definirea de grupuri imbricate de utilizatori (cea din urmă dovedindu-se a fi o îmbunătățire considerabilă în special în cazul unor sisteme de fișiere complexe cu un număr ridicat de utilizatori)

3. Proiectul se bazează pe cunoștințe dobândite în principal la următoarele 3-4 discipline:
Sisteme de operare, Sisteme de operare distribuite, Rețele de calculatoare

4. Realizarea practică/ proiectul rămân în proprietatea: **Nicolae Nătrăpeiu**

5. Proprietatea intelectuală asupra proiectului aparține: **Nicolae Nătrăpeiu**

6. Locul de desfășurare a activității: **Universitatea Babeș-Bolyai Cluj-Napoca / Universitatea Politehnica București**

7. Data eliberării temei: **Octombrie 2013**

CONDUCĂTOR LUCRARE:

STUDENT:

Lect. Dr. Adrian STERCA

Nicolae NĂTRĂPEIU



Copyright © 2014 , *Nicolae NĂTRĂPEIU*

Toate drepturile rezervate

Autorul acordă UPB dreptul de a reproduce și de a distribui public copii pe hârtie sau electronice ale acestei lucrări, în formă integrală sau parțială.

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or any later version.

You should have received a copy of the GNU General Public License along with this program. If not, see <<http://www.gnu.org/licenses/>>

For the sake of accessibility, the terms are reproduced inside the section called "Anexa1 Termenii de distribuire a conținutului lucrării"

Declarație de onestitate academică

Prin prezenta declar că lucrarea cu titlul “Sistem de fișiere cu un set extins de permisiuni”, prezentată în cadrul Facultății de Electronică, Telecomunicații și Tehnologia Informației a Universității “Politehnica” din București ca cerință parțială pentru obținerea titlului de *Inginer* în domeniul *domeniul Calculatoare și tehnologia informației*, programul de studii de licență *Ingineria informației* este scrisă de mine și nu a mai fost prezentată niciodată la o facultate sau instituție de învățământ superior din țară sau străinătate.

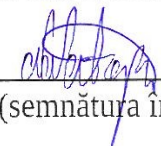
Declar că toate sursele utilizate, inclusiv cele de pe Internet, sunt indicate în lucrare, ca referințe bibliografice. Fragmentele de text din alte surse, reproduse exact, chiar și în traducere proprie din altă limbă, sunt scrise între ghilimele și fac referință la sursă. Reformularea în cuvinte proprii a textelor scrise de către alți autori face referință la sursă. Înțeleg că plagiatul constituie infracțiune și se sancționează conform legilor în vigoare.

Declar că toate rezultatele simulărilor, experimentelor și măsurărilor pe care le prezint ca fiind făcute de mine, precum și metodele prin care au fost obținute, sunt reale și provin din respectivele simulări, experimente și măsurători. Înțeleg că falsificarea datelor și rezultatelor constituie fraudă și se sancționează conform regulamentelor în vigoare.

București, data

20.06.2014

Absolvent Nicolae NĂTRĂPEIU


(semnătura în original)

Cuprins

Introducere	15
1. Sisteme de fişiere	19
1.1 Privire de ansablu asupra sistemelor de fişiere	19
1.2 Tipuri de fişiere	19
1.3 Protecţia fişierelor	20
1.4 Organizarea internă	21
1.5 Accesarea fişierelor în contextul proceselor	23
1.6 Ierarhia sistemului de fişiere	24
1.7 Exemplificări	25
1.8 EXT4 – Al patrulea sistem extins de fişiere.....	26
3. Sisteme de fişiere userspace	29
2.1 Abordarea FUSE – File System în User Space	29
2.2 Structura mediului de lucru FUSE	30
2.3 Sistemele virtuale de fişiere în context linux	31
2.4 Exemplificări	32
2.5 Interfeţe pentru limbaje de programare	33
4. EpsFS – Sistemul de fişiere cu un set extins de permisiuni	35
Concluziile proiectului	39
Bibliografie	41
Anexe suplimentare	43
Anexa1 Termenii de distribuire a conţinutului lucrării	43

Lista figurilor

Figura 0.1 Sisteme de operare de tip Unix	15
Figura 1.1 Vizualizarea din terminal a tipurilor de fişiere	20
Figura 1.2 Structura logică a unui sistem de fişiere pe disc	21
Figura 1.3 Structura internă a unui inod	22
Figura 1.4 Referinţele unui inod la blocurile de date	22
Figura 1.5 Procesul de autentificare a utilizatorilor	24
Figura 2.1 Structura unui sistem de operare de tip Unix	29
Figura 2.2 Structura apelurilor modulului FUSE	30

Lista acronimelor

ACL	Access Control List
Amazon S3	Amazon Simple Storage Service
API	Application Portable Interface
AWS	Amazon Web Services
BFFS	(Berkeley)Fast File System
BIOS	Basic Input/Output System
C	The C Programming Language
DE	Desktop Environment
EpsFS	Extended permissions set File System
ext/ext2/ext3/ext4	The [second, third, fourth] extended file system
FIFO	First In First Out
FSF	Free Software Foundation
FTP	File Transfer Protocol
FUSE	File System in User Space
GNU	Gnu is Not Unix
GVFS	Gnome Virtual File System
HDFS	Hadoop distributed File System
HTTP	Hypertext Transfer Protocol
I/O	Input/Output
IDE	Integrated Development Environment
IP	Internet Protocol
IPC	Inter-Process Communication
MTP	Media Transfer Protocol
NTFS	New Technology File System
OBEX	Object exchange
POSIX	Portable Operating Sistem Interface
RAID	Redundant Array of Independent Disks
RAM	Random Access Memory
REST	Representational State Transfer
ș.a.	și alții/ și altele
SFTP	SSH File Transfer Protocol
SMB	Samba
SSH	Secure Shell
SSHFS	SSH File System
trad.	în traducere
UFS	Unix File System
VFS	Virtual File System
WebDAV	Web Distributed Authoring and Versioning

Introducere

Sistemul de operare este componenta care joacă rolul de intermediar între utilizatorul calculatorului și componentele fizice ale acestuia. Scopul sistemului de operare este de a crea un mediu în care utilizatorul să poată executa programe și rutine într-un mod eficient, fără a fi nevoit să administreze resursele de care dispune.

Unix este unul dintre cele mai populare sisteme de operare și a fost creat de către Ken Thompson și Dennis Ritchie la centrul de cercetare și dezvoltare *AT&T Bell Labs* din Murray Hill, New Jersey în 1973. Popularitatea acestui sistem de operare se datorează în principal portabilității și flexibilității acestuia, fiind disponibil într-o multitudine de variante. La sfârșitul anilor '80 erau disponibile nu mai puțin de 100 de variante, cele mai notabile fiind Sun OS, Solaris (Oracle), HP-UX (Hewlett-Packard), AIX (IBM), BSD (Berkeley Software Design) iar mai recent, 2001, OS X (Apple Inc.); majoritatea având o licență de tip proprietar.

Principalele caracteristici ale sistemului de operare Unix sunt: *interactivitate* (comenzile sunt primite de la terminale într-un regim conversațional), *multi-utilizator* (mai mulți utilizatori pot accesa resursele sistemului în același timp), *multi-tasking* (utilizatorii pot rula mai multe programe simultan), *time-sharing* (procesorul alocă secvențial cuante fiecărui proces activ). Dintr-un alt punct de vedere, sistemul de operare este structurat în două părți¹:

- *userspace*: (trad. spațiul de utilizator) aici rulează de obicei servicii și programe
- *kernel*: (trad. nucleu) gestionează resursele hardware, memoria, conține drivere și implementarea sistemelor de fișiere

În prezent termenii “Unix” și “Unix-like”(trad. asemănător cu Unix) sunt folosiți informal pentru a caracteriza sistemele de operare care respectă standardele și principiile stabilite prin design-ul nucleului Unix dar nu presupune prezenta de certificări sau patente. Figura 0.1¹ exemplifică aceste sisteme de operare și modul lor general de înrudire.

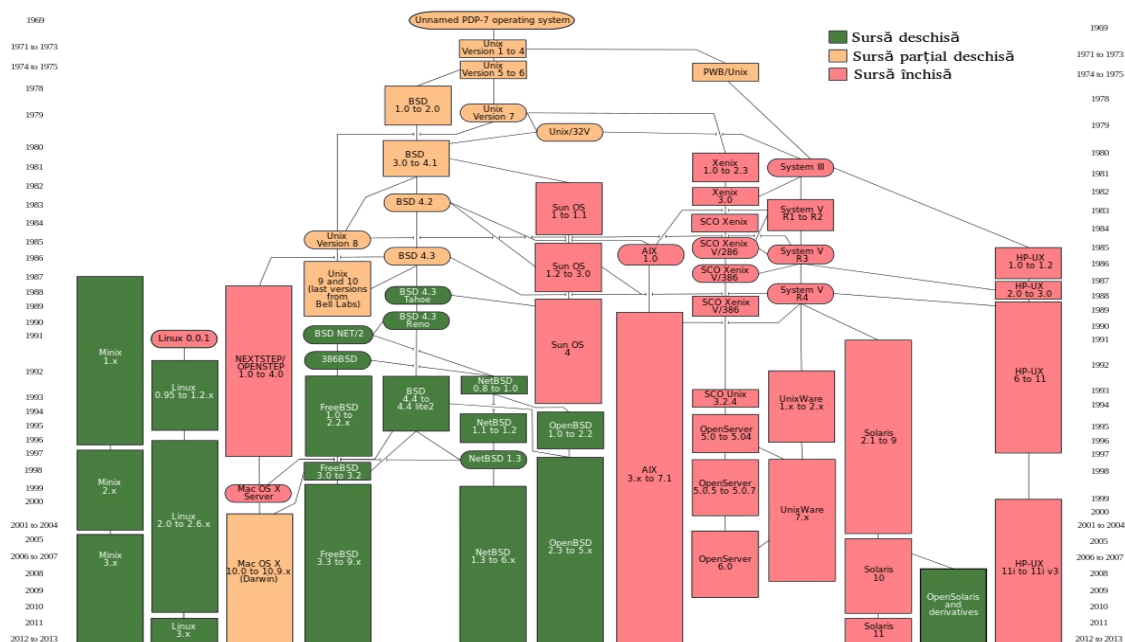


Figura 0.1: Sisteme de operare de tip Unix (variantă compactă)

¹ [1] Capitolul 3.2.1

¹ [9] distribuită conform CC BY-SA 3.0 <http://creativecommons.org/licenses/by-sa/3.0/>

Lucrarea de față va aborda subiectul sistemelor de fișiere din perspectiva sistemelor de operare de tip Unix, mai precis Linux. Sistemul de operare Linux a fost dezvoltat de către Linus Torvalds în 1991 cu scopul de a fi rulat pe calculatoarele personale de tip IBM cu microprocesoare Intel 80386 dar a ajuns de-a lungul timpului să suporte o mulțime de alte arhitecturi printre care Intel Itanium, HP Alpha AMD64 ș.a. .

Popularitatea sistemului de operare Linux, în special în mediul enterprise și al cercetării, se datorează în principal publicării acestuia sub formă de program liber, cu sursa deschisă, fiind distribuit conform reglementărilor licenței GPLv2 (deși inițial a fost publicat cu o licență proprietară), permițând astfel programatorilor să studieze și sugereze direcții de dezvoltare.

După cum se poate observa din *Figura 0.1*, Linux-ul nu este o derivată a Unix-ului, dar din punct de vedere tehnic este complet compatibil cu acesta deoarece deciziile tehnice au avut ca scop minimizarea diferențelor și păstrarea conceptelor și șabloanelor arhitecturale.²

Pentru a înțelege magnitudinea eforturilor depuse pentru dezvoltarea nucleului de Linux, este suficient să analizăm câteva dintre datele statistice publicate de *The Linux Foundation* în 2008³, de unde aflăm că peste 1000 de ingineri software din cel puțin 100 de companii diferite contribuie la fiecare lansare a unei noi versiuni (care are loc o dată la 2-3 luni). Ca și exemplu, pentru dezvoltarea versiunii de kernel 2.x, de-a lungul a 2-3 ani, au contribuit aproximativ 3200 de ingineri software din peste 200 de companii. De asemenea, aceștia estimează că pentru realizarea kernel-ului 2.6.25.i686, costurile dezvoltării (în 2008) ar atinge valoarea de 1.3 miliarde de dolari.

Arhitectura sistemelor de operare bazate pe Unix gravitează în jurul propriului sistem de fișiere care are un rol crucial în stocarea, accesarea și administrarea datelor. În lipsa acestuia, toată informația gestionată de sistemul de calcul ar fi o simplă înșiruire de biți pe un suport fizic.

Unul dintre principiile definitorii pentru sistemul de fișiere este *Principiul structurii arborescente* care stabilește că orice director sau fișier are un singur părinte iar astfel există o cale unică de la acesta la rădăcină.

Un alt concept extrem de important este dreptul de acces asupra conținutului (sau permisiunile). În raport cu acestea se definesc următoarele categorii de utilizatori: owner (trad. proprietar), group (trad. grup), others (trad. alții).

Pentru fiecare dintre acestea sunt stabilite permisiunile de rwx: read write execute (trad. citire scriere execuție). Astfel se realizează într-un mod cât se poate de simplu și eficient o mapare între utilizatorii sistemului de operare și datele pe care aceștia le pot accesa.

În ciuda eficacității acestei implementări am observat necesitatea de a oferi un spectru mai larg de configurare a acestor permisiuni. Una din primele opțiuni ar fi folosirea de ACL-uri (Access Control List) însă și acestea au o configurabilitate destul de limitată.

Să considerăm următoarea configurație de grupuri de utilizatori:

1. g1 cu utilizatorii u1, u2, u3
2. g2 cu utilizatorii u3, u4, u6

și dorim să aplicăm o regulă de acces care să cuprindă utilizatorii din ambele grupuri va trebui să creăm un nou grup g3 în care să-i adăugăm pe fiecare. În situația dată putem considera că acest neajuns duce la o ineficiență nedorită prin duplicarea informațiilor de acces, mai ales în cazul în care numărul de utilizatori este ridicat. Mai mult de atât, de fiecare dată când adăugăm un nou utilizator în oricare dintre cele două grupuri inițiale trebuie să ne asigurăm că va fi adăugat și în al treilea. Într-un sistem cu un număr ridicat de utilizatori și cu o configurație de acces complexă, un

² [4] Capitolul 1

³ [8]

utilizator va avea o listă destul de numeroasă cu toate grupurile în care este inclus. În același timp, prin simpla analiză a acestor grupuri utilizatorul poate deduce anumite tipare despre structura și ierarhizarea drepturilor în sistemul de fișiere.

Creșterea numărului de grupuri de utilizatori determină inevitabil o intensificare (poate chiar exponențială) a eforturilor administratorului de sistem de a configura și a preveni eventualele erori de permisiuni.

La momentul actual, în implementarea sistemelor de fișiere sub Linux, nu există conceptul de grupuri imbricate de utilizatori. Sistemul de fișiere NTFS (New Technology File System) este un sistem proprietar de fișiere dezvoltat de către Microsoft din 1993 și care permite stabilirea de reguli în raport cu grupurile de utilizatori, și chiar imbricarea acestora. NTFS a devenit sistemul de fișiere standard pentru toate sistemele din familia Windows NT, începând cu Windows NT 3.1 înlocuind astfel vechiul sistem de fișiere FAT care avea o implementare destul de rudimentară.

Deoarece sistemul de operare Windows nu are un nucleu de tip Unix și nu respecta patentele și principiile de arhitectură stabilite de acesta, nu va putea fi folosit pentru rezolvarea problemei menționate anterior.

Dintr-un alt punct de vedere, să considerăm că dorim să permitem unui anumit grup de utilizatori să acceseze fișierele numai dacă sunt autentificați local pe mașina de calcul iar altora să aibă numai acces de citire dacă sunt conectați la sistem folosind un anumit protocol (de exemplu SSH).

Soluția disponibilă în acest caz este utilizarea ACL-urilor de rețea (Networking ACLs) și a celor de sistem de fișiere (Filesystem ACLs) și configurare acestora pentru obținerea efectului dorit. Am considerat această metodă ca fiind eficientă însă nu suficient de flexibilă deoarece necesită din nou mai mult efort pentru mentenanță.

Lucrarea de față va prezenta modul de implementare a unui sistem de fișiere care să permită un spectru cât mai larg de configurare a drepturilor de acces oferind soluții pentru problemele menționate anterior, eficientizând astfel performanța sistemului de operare.

Principala direcție care va fi urmată este cea de a oferi administratorului sistemului de fișiere (care poate fi utilizatorul root (trad. rădăcină, cu sensul de superutilizatorul sistemului de tip Unix) sau un alt utilizator privilegiat) următoarele categorii de drepturi:

- permiterea configurării de grupuri imbricate de utilizatori
- permiterea accesului la date în funcție de protocolul prin care utilizatorul este conectat (spre exemplu SSH)
- permiterea accesului la date în funcție de adresa IP a utilizatorului (sau a unui set de adrese IP)
- accesul la datele sistemului în funcție de diferite intervale orare

Alte însușiri pe care le-am considerat extrem de utile sunt posibilitatea de conectare a acestor categorii primare folosind operatori logici (și, sau ș.a.) dar și crearea unei interfețe care să ușureze activitatea de administrare.

Deși majoritatea sistemelor de fișiere sunt implementate ca și module în nucleul sistemului de operare, am decis folosirea modului de kernel FUSE, care oferă o interfață pentru crearea de sisteme de fișiere în userspace care totodată păstrează rădăcinile și regulile de compatibilitate impuse de Unix. Principalul motiv a fost acela de a putea avea un model dinamic și flexibil, care să

nu fie strâns legat de mașina fizică pe care rulează. Datorită acestei decizii creăm premisele unei ulterioare dezvoltări în direcția cloud storage-ului și a infrastructurii puternic virtualizate.

1. Sisteme de fișiere

În acest capitol vom prezenta principalele caracteristici ale unui sistemelor de fișiere din perspectiva standardelor Unix și le vom exemplifica pe cele mai cunoscute și folosite la momentul actual.

1.1 Privire de ansamblu asupra sistemelor de fișiere

Sistemul de operare Unix se află într-o strânsă legatura cu sistemul propriu de fișiere care are un rol extrem de important în stocarea, accesarea și administrarea datelor.

Un sistem de fișiere de tip Unix este o colecție de fișiere (vom vedea ulterior, conform clasificării, directoarele sunt un tip de fișier) cu următoarele proprietăți¹:

- prezintă o entitate numită *root* (trad. Rădăcină) notat “/” care este un director special deoarece acesta nu are un părinte și conține alte fișiere.
- fiecare fișier din sistem este identificat printr-un nume și un identificator unic numit *inode*.
- principiul structurii arborescente: fiecare fișier are un singur părinte (excepție făcând *root*-ul). De aici putem deduce că numele fiecărui fișier este de asemenea unic deoarece acesta este reprezentat de path-ul (trad. calea) directoarelor părinte până la rădăcină
- are implementată o metodă de protecție prin maparea utilizatorilor și a fișierelor gestionate de sistem, restricționând astfel accesul la informații.

1.2 Tipuri de fișiere

În cadrul unui sistem de fișiere de tip Unix sunt definite următoarele tipuri de fișiere împreună cu *caracterul de reprezentare* a acestora ²:

- regular file (trad. fișier obișnuit): -
- directory (trad. Director): **d**
- hard link (trad. legătură hard)
- symbolic link (trad. legătură simbolică): **l**
- socket: **s**
- named pipe/FIFO (trad. pipe cu nume): **p**
- character device (trad. periferic caracter): **c**
- block device (trad. periferic bloc): **b**
- door (trad. ușă): **D**

Vom detalia în cele ce urmează tipurile de fișiere care sunt relevante pentru acest studiu de caz.

Fișierele obișnuite sunt containere de informații reprezentate ca șiruri de octeți.

Fișierele director se deosebesc de cele normale numai prin informațiile stocate. Directoarele conțin lista numelor și adreselor fișierelor pentru care va juca rolul de părinte. De asemenea mai conține două entități de intrare speciale notate cu “.”- care reprezintă însuși directorul, respectiv “..” - care reprezintă părintele directorului curent. După cum sesizam anterior, datorită structurii de arbore a

¹ [7] Capitolul 5

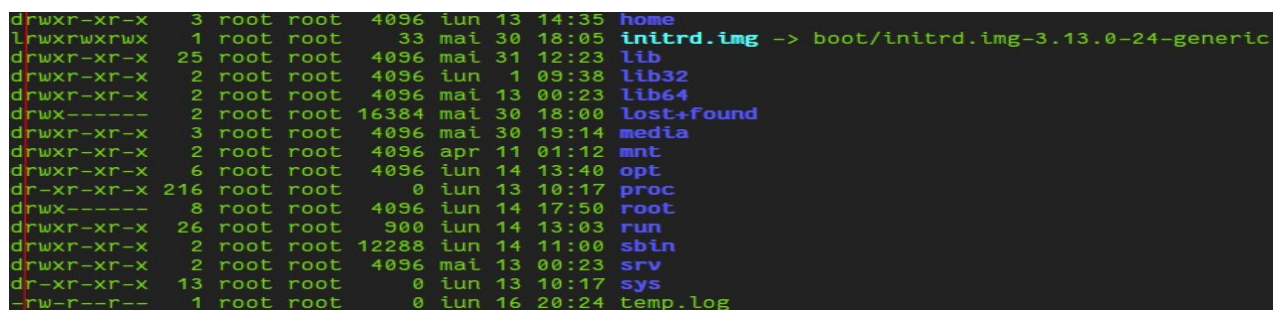
² [1] Capitolul 4.1, [7] Capitolul 4

sistemului de fișiere, rădăcina(care este un director) nu va avea un părinte, deci “..” va referi o adresă nulă (sau pe sine).

Legăturile hard sunt identice cu cele naturale și reprezintă un punct de intrare într-un director care referă spre o entitate din sistemul de fișiere, aceasta având două directoare părinte. Acest tip de legături se pot face numai de către superuser și pot referi numai la entități din cadrul aceluiasi sistem de fișiere.

Legăturile simbolice sunt puncte de intrare care referă un fișier oarecare. Acest tip de legături se poate face și de către utilizatorii obișnuiți și nu sunt constrânse la a referi numai fișiere din cadrul aceluiasi sistem de fișiere.

Tipul fișierului poate fi accesat de utilizator folosind comanda de terminal `ls -l`, fiind poziționat înaintea reprezentării permisiilor (concept despre care o să discutăm în prealabil) după cum se poate observa și în Figura 1.1 (caracterele aflate în stânga liniei verticale roșii).



```
drwxr-xr-x  3 root root  4096 iun 13 14:35 home
lrwxrwxrwx  1 root root    33 mai 30 18:05 initrd.img -> boot/initrd.img-3.13.0-24-generic
drwxr-xr-x 25 root root  4096 mai 31 12:23 lib
drwxr-xr-x  2 root root  4096 iun  1 09:38 lib32
drwxr-xr-x  2 root root  4096 mai 13 00:23 lib64
drwx----- 2 root root 16384 mai 30 18:00 lost+found
drwxr-xr-x  3 root root  4096 mai 30 19:14 media
drwxr-xr-x  2 root root  4096 apr 11 01:12 mnt
drwxr-xr-x  6 root root  4096 iun 14 13:40 opt
dr-xr-xr-x 216 root root    0 iun 13 10:17 proc
drwx----- 8 root root  4096 iun 14 17:50 root
drwxr-xr-x 26 root root   900 iun 14 13:03 run
drwxr-xr-x  2 root root 12288 iun 14 11:00 sbin
drwxr-xr-x  2 root root  4096 mai 13 00:23 srv
dr-xr-xr-x 13 root root    0 iun 13 10:17 sys
-rw-r--r--  1 root root    0 iun 16 20:24 temp.log
```

Figura 1.1 Vizualizarea din terminal a tipurilor de fișiere

Putem observa că fișierul `initrd.img` este o legătură simbolică către `boot/initrd...` fiind semnalizat corespunzător prin caracterul “l”. La fel, `temp.log` este un fișier obișnuit, tipul acestuia fiind semnalizat prin caracterul “-”

1.3 Protecția fișierelor

Sistemele de fișiere de tip Unix prezintă conceptul de acces asupra conținutului (sau permisiunile), în raport cu acestea definindu-se următoarele categorii de utilizatori:

- owner (trad. proprietar): reprezintă proprietarul fișierului
- group (trad. grup): reprezintă grupul de utilizatori care au acces asupra fișierului respectiv;
- others (trad. alții): orice utilizator care nu se încadrează în categoriile precedente

Pentru fiecare dintre aceste categorii sunt definite următoarele drepturi:

- r: read (trad. citire) prin setarea acestuia se permite accesul la citirea fișierului/directorului
- w: write (trad. scriere) prin setarea acestuia se permite accesul de scriere a fișierului/directorului;
- x: execute (trad. execuție) prin setarea acestuia se permite accesul pentru executarea fișierului;

Pentru a sublinia importanța acestor drepturi și faptul că sunt în totalitate independente, facem precizarea că dreptul de *write* nu implică automat și pe cel de *read*. Cu alte cuvinte un utilizator care poate să scrie(sau să șteargă) un fișier nu va putea să citească conținutul acestuia dacă nu are

efectiv setat bitul de r . Aceeași logică se aplică și pentru un folder care are setat bitul de x : utilizatorul îl va putea accesa (folosind comanda `cd`) dar nu va putea vedea conținutul acestuia (folosind comanda `ls`).

Pentru a specifica drepturile de citire, scriere și tipărire pentru cele trei tipuri de utilizatori sunt folosiți nouă biți.

Spre exemplu, un fișier care are toți biții setați este stocat ca și:

111	111	111	(reprezentarea binară)
rw	rw	rw	(reprezentarea externă/simbolică; vezi <i>Figura 1.1</i> , în dreapta liniei verticale)
7	7	7	(reprezentarea octală – fiecare cifră reprezentând grupuri de câte trei biți)

Pentru modificarea permisiilor și a categoriilor de utilizatori avem definite următoarele comenzi de terminal:

- `chown`: change owner (trad. schimbare proprietar)
- `chgrp`: change group (tr. schimbare grup)
- `chmod`: change mod (tr. schimbare mod cu sensul de permisiuni de acces)

Utilizarea acestora se va face folosind argumentele descrise astfel (manualul linux oferă mai multe de detalii despre utilizări mai avansate a acestor comenzi):

```
chown <nume_utilizator> <nume_fișier>
chgrp <nume_grup> <nume_fișier>
chmod <repr_octal> <nume_fișier>
```

Prin conceptul de drepturi implicite se stabilește un set de valori pentru permisiuni numit `umask` (trad. mască) și care reprezintă scăzătorul din diferența:

`<drepturi_absolute> - <umask> = <drepturi_inițiale>`

Pentru a vedea valoarea pentru sesiunea curentă se va rula în terminal comanda `umask` rezultatul fiind de obicei 022. Înlocuind în formula descrisă anterior `777-022=755`, aceasta fiind valoarea inițială pentru orice fișier nou adăugat.

1.4 Organizarea internă

Sistemul de fișiere va fi stocat pe un suport de memorie extern (hard-disk ș.a.) care trebuie să comunice cu memoria internă, gestionată de sistemul de operare, în unități numite blocks (trad. blocuri).

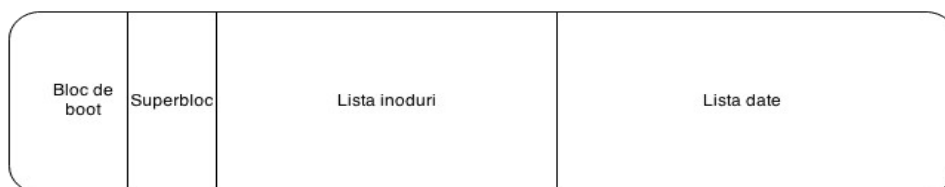


Figura 1.2: Structura logică a unui sistem de fișiere pe disc

În figura 1.2 se poate observa aranjarea logică a sistemului de fișiere în care se evidențiază cele patru categorii de blocuri⁴:

4 [1] Capitolul 4.2

1. Blocul de boot: conține bootloader-ul (trad. programul care încarcă sistemul de operare) care va fi citit și executat de către BIOS cu scopul de a porni progresiv întreg sistemul de operare
2. Superblocul: conține informații despre structura sistemului de fișiere salvat (numărul de inoduri, număr de zone de date definite ș.a)
3. Lista de inoduri: conține un număr fix de inoduri, stabilit odată cu formatarea discului. Fiecare inod este reprezentat de un număr care va face legătura între acesta și programele care îl utilizează. Numărul inodului care corespunde fiecărui fișier poate fi accesat executând comanda `ls -li`
4. Lista de date: este cea mai mare grupare de blocuri de pe disc și conține informațiile indexate

Conceptul de *inode* are un rol central în definirea unui sistem de fișiere ca fiind de tip Unix. Acesta este o structură cu o dimensiune cuprinsă între 64 și 128 de octeți, care are următoarea structură internă:

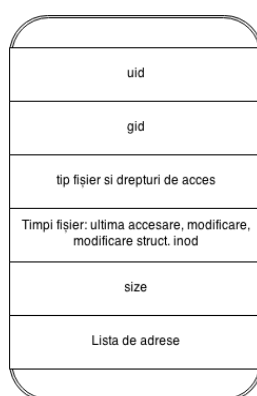


Figura 1.3 Structura internă a unui inod

După cum se poate observa acesta prezintă toate atributele amintite anterior, id-ul proprietarului, și al grupului, permisiunile, tipul de fișier, dimensiunea, timpii de acces/modificare și lista de blocuri de date.

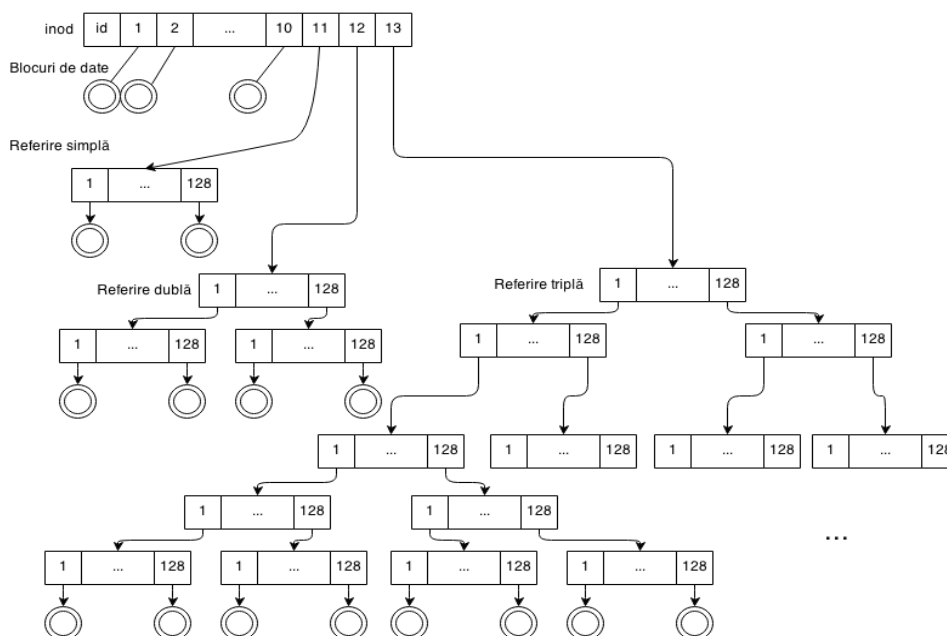


Figura 1.4 Referințele unui inod la blocurile de date

În Figura 1.4 putem observa modul în care inodul face referirea la blocurile de date care îi aparțin considerând că o adresă pe disc este reprezentată pe 4 octeți (astfel un bloc având 128 de adrese). Lista de adrese a inodului conține 13 adrese care referă direct sau indirect blocurile fizice de date. Astfel primele 10 reprezintă adresele pentru primele blocuri de date pentru fișier, blocul de date având o dimensiune constantă predefinită. Adresa 11 referă la adresa blocului de referire simplă care conține adresele următoarelor 128 de blocuri de date. Adresa 12 referă la adresa blocului de referire dublă care conține adresele a 128 de blocuri de referire simplă, fiecare dintre acestea, la rândul lor referind la blocuri de date. Adresa 13 conține adresa blocului de referire simplă, care conține referințe către 128 de blocuri de referire dublă, fiecare dintre acestea referind 128 de blocuri de referire simplă, fiecare având la rândul lor 128 de adrese de blocuri de date.⁵

Se poate observa că în funcție de dimensiunea blocului de adrese, capacitatea unui inod crește proporțional.

Un alt concept important de menționat este cel de montare. Montarea este operația prin care un sistem de fișiere este conectat la un director din sistemul de fișiere implicit. În cazul sistemelor de fișiere clasice aceasta se face folosind comanda `mount` (și `unmount` pentru demontare) care are următoarele argumente principale:

```
mount [opțiuni] <sistem_de_fișiere> <director_de_montare>
```

1.5 Accesul fișierelor în contextul proceselor

După cum aminteam anterior, există trei nivele principale de abstractizare: nivelul hardware, nivelul kernelului și nivelul de utilizator, ultimile două reprezentând sistemul de operare. Programele rulate de utilizatori în userspace comunică prin librăria de sistem (care are anumite apeluri de sistem pe care le vom detalia în prealabil) cu subsistemul de control al proceselor care comunică cu sistemul de fișiere. Sistemul de fișiere este de asemenea administrat de *nucleu* făcând legătura cu subsistemul care administrează perifericele și efectiv suportul fizic adică discul.

Are o importanță majoră pentru lucrarea de față înțelegerea conceptului de sistem de fișiere clasic care rulează în nucleu, deoarece după cum vom vedea în capitolul următor, modulul FUSE va face posibilă crearea de sisteme de fișiere dependente de spațiul utilizatorului.

Avem mai jos o listă cu principalele apeluri de sistem pentru sistemul de fișiere (care vor fi detaliate de asemenea pe parcursul capitolului următor) și categoria de operație la care se referă⁶:

- `open`, `creat`, `dup`, `pipe`, `close` (descriptori de fișier)
- `chown`, `chmod`, `stat` (atribute de fișier)
- `read`, `write`, `lseek` (operații I/O fișier)
- `creat`, `mknod`, `link`, `unlink` (operații inode)
- `mount`, `umount` (structura sistemului de fișiere)
- `chdir`, `chown` (manipularea structurii ierarhice)
- `getblk`, `brelse`, `breada`, `bwrite` (algoritmi alocare bloc)

În momentul rulării de către kernel a unui proces, nucleul asignează acestuia două tipuri de id-uri: id real de utilizator, respectiv de grup notate `uid` și `gid`, al doilea tip fiind id efectiv de utilizator, respectiv de grup notate `euid` și `egid`, folosit pentru verificarea permisiunilor de acces, pentru atribuirea fișierelor nou create și pentru trimiterea de semnale de sistem.

5 [1] Capitolul 4.2.3

6 [7] Capitolul 5

Kernelul permite modificarea acestor parametri prin folosirea apelurilor de sistem: `setuid`, `setgid`, `seteuid`, `setegid`. Putem analiza utilizarea acestor apeluri folosind Figura 1.5 care exemplifică modul de autentificare a unui utilizator.

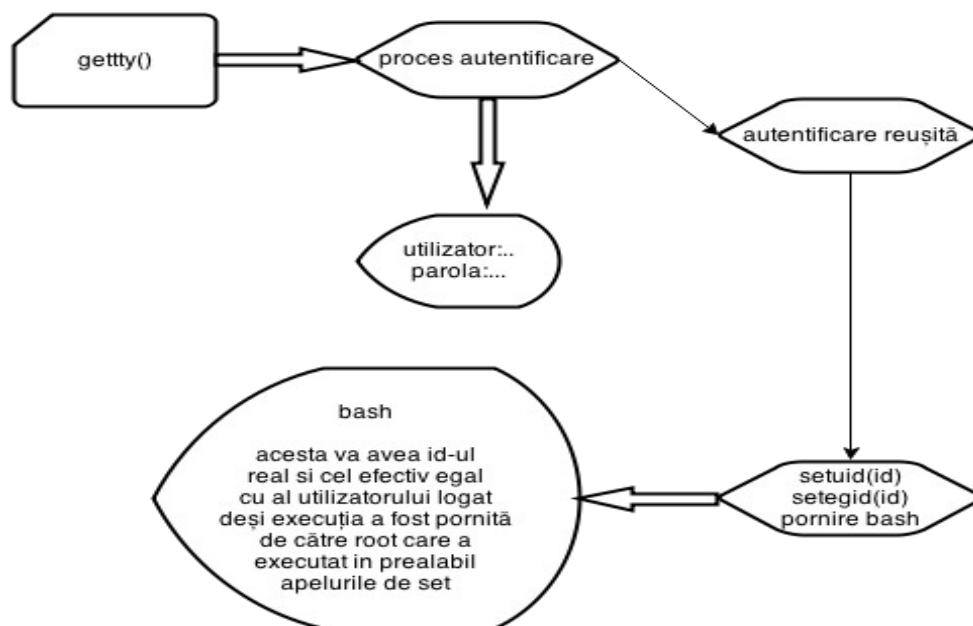


Figura 1.5: Procesul de autentificare al utilizatorilor

1.6 Ierarhia sistemului de fișiere

Principalele directoare dintr-un sistem de fișiere Unix au următoarea formă standardizată (care poate totuși suferi mici variații în funcție de varianta de Unix)⁷:

- `/usr`: reprezintă principala locație în care se află fișiere binare
- `/usr/bin`: conține fișierele binare specifice utilizatorilor
- `/usr/sbin`: conține fișiere binare folosite în administrarea sistemului
- `/usr/local`: conține programe care sunt instalate cu un oarecare grad de independență față de sistemul de operare, având fișierele binare stocate în `/usr/local/bin`
- `/usr/lib`: conține librăriile partajate dinamic între programe
- `/usr/include`: conține fișiere header pentru utilizator
- `/usr/include/sys`: conține fișiere header folosite de către kernel
- `/usr/src`: conține textele sursă scrise în limbajul C ale nucleului instalat pe mașina respectivă
- `/bin`: conține programe pentru principalele comenzi Unix, compilatoare, asamblatoare, mai nou constituind o referință către `/usr/bin`
- `/dev`: folosit pentru memorarea fișierelor speciale de tip device (trad. dispozitiv)
- `/etc`: conține informații specifice sistemului și necesare întreținerii; conține și script-uri necesare sistemului de operare pentru pornire
- `/var/log`: conține fișierele care stochează log-uri (trad. jurnale) de sistem
- `/var/tmp`: conține fișiere temporare folosite în timpul rulării sistemului
- `/home` (sau pe vechile implementări Unix `/u`): conține directoarele de home (trad. acasă folosit cu sensul de directoare primare) ale utilizatorilor.
- `/opt`: Folosit pentru pachetele și fișierele binare opționale

7 [1] Capitolul 4.1.5

1.7 Exemplificări

UFS cunoscut inițial sub numele de Berkeley Fast File System sau FFS datorită rădăcinilor acestuia în varianta BSD. UFS a fost unul dintre cele mai studiate sisteme de fișiere, fiind comprehensibil și ajungând să fie portat pe majoritatea variațiilor de Unix. Conține numeroase principii care stau la baza sistemelor de fișiere extinse (ext, ext2, ext4). Performanța acestui sistem de fișiere este considerabilă iar modelul de aranjare a inodurilor în clustere s-a dovedit a fi eficientă. Prin acest sistem de fișiere s-au introdus următoarele concepte: legături simbolice (inițial existau doar cele hard), nume cu lungime arbitrară pentru fișiere (inițial numele erau limitate la numai 15 caractere), redenumirea fișierelor, blocări (partajate sau exclusive) ale fișierelor⁸.

Solaris UFS a introdus un concept extrem de important de journaling (trad. Jurnalizare) la care ne vom referi ca și UFS logging. Deși avem puține informații în documentația oficială oferită de Sun despre modul în care acesta funcționează, știm că pentru a porni această opțiune trebuie să adăugăm la montare parametrul “logging”. Spațiul de jurnalizare este proporțional cu capacitatea discului, fiecărui GB de date corespunzându-i 1MB de log. Rolul jurnalizării este de a putea readuce sistemul la stadiul de funcționare în cazul unei erori.

Ext2, al doilea sistem de fișiere extins, este specific kernelului de linux și e strâns legat de conceptele stabilite de UFS. Principalele caracteristici ale acestuia sunt:

- sisteme de fișiere cu o capacitate de 4TB
- directoarele au o dimensiune variabilă cu o capacitate maximă de 255 de octeți
- la crearea sistemului de fișiere se pot specifica diferite dimensiuni pentru blocul de date: 1024, 2048, 4096
- rezervarea spațiului: până la 5% din sistemul de fișiere poate fi rezervat pentru fișierele care aparțin superutilizatorului
- verificări ale sistemului de fișiere care se execută periodic în funcție de timp sau după un anumit număr de cicluri de citire/scriere.
- atribute specifice fișierelor care pot desemna posibilitatea de recuperare după ștergere, faptul că un fișier este sau nu comprimat, metadata pentru fișiere ș.a.

Ext3, al treilea sistem de fișiere extins, a fost creat cu scopul de a rezolva o problemă de performanță și anume timpul destul de lung necesar verificării și revenirii la o stare de consistență a sistemului de fișiere după o blocare a acestuia. Inițial acest timp putea dura câteva ore pentru sisteme de fișiere complexe fiind datorat în special structurii inodului și nu neapărat al numărului de fișiere din sistem. Un alt scop a fost de a face aceste îmbunătățiri cu un număr cât mai mic de modificări pentru a păstra simplitatea și ușurința de folosire făcându-l totodată complet compatibil cu parintele său. Îmbunătățirea s-a făcut prin implementarea unui mecanism de tranzacții.

ReiserFS a fost dezvoltat de o echipă de programatori conduși de Hans Reiser. A fost primul sistem de fișiere jurnalizat care a fost inclus în kernelul de linux și a fost o perioadă standard pentru importanta distribuție SUSE Linux Enterprise, produsă de Novell. La momentul apariției a făcut disponibilă jurnalizarea sistemului de fișiere pentru linux bazată numai pe metadata (iar mai târziu și pe blocuri). S-a dovedit a fi mai rapid decât ext2/ext3 în cazul fișierelor de dimensiuni reduse (mai mici de 4KiB) pentru varianta 2.4 a kernelului de Linux. De asemenea a făcut posibilă schimbarea dimensiunii în timp real a sistemului de fișiere și a oferit soluții pentru reducerea fragmentării fără a avea un impact masiv asupra performanței. Proiectul a fost oprit în urmă cu câțiva ani datorită unor probleme de natură juridică.

ZFS este un sistem de fișiere dezvoltat de către Sun Microsystems la începutul anilor 2000, fiind bazat pe principiile sistemelor extinse de fișiere. Principalele caracteristici sunt protecția împotriva corupției datelor, suport pentru stocări masive de date (dimensiunea unui fișier poate fi de până la 16 exbiocteti (2^{64}), a unui volum de 256 zebiocteti (2^{78}) și un număr de până la 2^{48} de fișiere)

Tux, ajuns în prezent la versiunea 3, este un sistem de fișiere demn de menționat și care promite performanțe impresionante este dezvoltat în special pentru kernelul de Linux de către Daniel Phillips. Principalele scopuri sunt de a oferi o implementare mai eficientă a versionării, o mai bună replicare și să o variantă liberă (și predând autorii: mai bună) pentru ZFS. Se află deocamdată însă în stadiul de dezvoltare.

EXT4 – Al patrulea sistem extins de fișiere

Este cel mai nou sistem de fișiere din familia extended fiind dezvoltat începând cu anul 2006 și aduce îmbunătățiri majore de performanță, stabilitate și scalabilitate comparativ cu predecesorii săi cu scopul de a crește limitele de stocare existente.

Dintre principalele caracteristici amintim⁹:

1. Suport pentru sisteme de fișiere masive, putând suporta volume de până la 1EiB (16TiB fiind valoarea recomandată), fișiere cu dimensiunea maximă de 16TiB, și un număr de maxim 3 miliarde de fișiere (dimensiune fixă, stabilită la creare)
2. Introduce conceptul de *extent* care ar presupune să înlocuiască clasicele scheme de mapari de blocuri deoarece s-au dovedit a fi ineficiente în special pentru operațiile de ștergere și trunchiere producând ca și efect secundar fragmentarea. Un extent este un grup continuu de blocuri fizice încurajând așezarea continuă pe disc în special pentru fișierele mari care au nevoie de stocare pe mai mulți extenți.
3. Alocarea de multiblocuri care permite într-un singur apel obținerea blocurilor multiple de date comparativ cu ext3 care face câte un apel pentru fiecare bloc (spre exemplu pentru un bloc de 4KiB sunt nevoie de câteva mii de apeluri pentru a alocă un fișier de o dimensiune medie de câțiva GB). Această proprietate scade considerant starea de fragmentare din sistem.
4. Alocarea întârziată este o trăsătură de performanță (lăsând neschimbată structura internă a sistemului de fișiere) și este întâlnită și în alte sisteme de fișiere moderne cum ar fi Reiser4, XFS, ZFS ș.a.. Aceasta presupune întârzierea pe cât posibil a operației de alocare a blocurilor (spre deosebire de ext3 care le alocă cât de repede posibil), păstrând datele în cache (trad. zona tampon) dovedindu-se obținerea unei performanțe crescute și o reducere a fragmentării.
5. Complet compatibil cu versiunile precedente. Astfel sisteme de fișiere de tip ext2 și ext3 pot fi montate ca și ext4 (evident însă nu și invers, un sistemul de fișiere ext4 nu poate fi montat ca și ext3 sau 2)
6. Montarea fără opțiunea de jurnalizare, permite utilizatorilor specializați care au nevoie de o putere de calcul și o viteză sporită să renunțe conștient, într-o anumită măsură la consistența datelor.
7. Îmbunătățirea atributelor specifice *inode*-urilor prin folosirea de nanosecunde (spre deosebire de ext3 care folosește secunde) pentru stocarea timpilor de acces/modificare a fișierelor, rezervarea în avans a inodurilor care constă în reținerea câtorva inode-uri în momentul creării fiecărui director care vor fi folosiți în momentul adăugării de conținut în acesta. De asemenea s-a crescut dimensiunea standard a inodului de la 128 la 256 de octeți,

9 [11] New ext4 features

fiind necesară mai ales pentru stocarea informațiilor adiționale care vor permite astfel un acces mai rapid, îmbunătățind performanța aplicațiilor care folosesc atribute extinse de până la 3-7 ori.

8. Stocarea unei liste cu inode-uri nefolosite permite ca eficiența operația de fsck să crească de două până la 20 de ori.

Pentru a putea vedea configurarea unui sistem de fișiere de ext4, majoritatea sistemelor bazate pe Linux oferă utilitarul tune2fs, acesta putând fi folosit și pentru a configura aceste setări. Ieșirile de mai jos exemplifică rezultatul rulării acestei comenzi în următoarea formă. Deși timpurile -urile :

```
tune2fs -l /dev/sda1 | more
```

```
tune2fs 1.42.9 (4-Feb-2014)
Filesystem volume name:   400GB
Last mounted on:         /media/nicu/400GB
Filesystem UUID:         57f02c9b-bda4-43c4-bdd4-9ceae44fbc5a
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal ext_attr resize_inode dir_index filetype n
eeds_recovery extent flex_bg sparse_super large_file huge_file uninit_bg dir_nli
nk extra_isize
Filesystem flags:         signed_directory_hash
Default mount options:    user_xattr acl
Filesystem state:         clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              30212096
Block count:              120816384
Reserved block count:     6040819
Free blocks:              111502640
Free inodes:              30150436
First block:              0
Block size:               4096
Fragment size:            4096
Reserved GDT blocks:      995
Blocks per group:         32768
Fragments per group:      32768
Inodes per group:         8192
Inode blocks per group:   512
Flex block group size:    16
Filesystem created:       Sat Jun  7 09:24:22 2014
Last mount time:          Sat Jun 14 09:48:29 2014
Last write time:          Sat Jun 14 09:48:29 2014
Mount count:              4
Maximum mount count:      -1
Last checked:             Sat Jun  7 09:24:22 2014
Check interval:           0 (<none>)
Lifetime writes:          117 GB
Reserved blocks uid:      0 (user root)
Reserved blocks gid:      0 (group root)
First inode:              11
Inode size:               256
Required extra isize:     28
Desired extra isize:      28
Journal inode:            8
Default directory hash:   half_md4
Directory Hash Seed:      e8825992-118e-4705-b75c-df6abc564f58
Journal backup:           inode blocks
```

Deși a fost amânată acceptarea acestuia de către dezvoltatorii kernel-ului de Linux, la începutul anului 2010 Google a făcut public faptul că își va migra infrastructura de stocare de la ext2 la ext4. Din urmă cu câțiva ani este sistemul de fișiere standard folosit de distribuțiile majore de Linux fiind marcat ca și stabil începând cu kernelul 2.6.28.

2. Sisteme de fișiere în userspace

În acest capitol vom prezenta metoda alternativă de creare a sistemelor de fișiere și anume în spațiul utilizatorului, prin folosirea modului de kernel FUSE (File System in User Space). De asemenea vom prezenta caracteristicile generale ale acestei abordări din prisma faptului că acesta stă la baza sistemului de fișiere care face subiectul acestei lucrări.

2.1 Abordarea FUSE – File System in User Space

Conform standardelor trasate de către sistemele de operare de tip Unix, acestea sunt formate din două componente, amintite anterior, și anume: spațiul utilizatorului și spațiul nucleului, fiecare având anumite sarcini, după cum se poate vedea și în figura de mai jos.

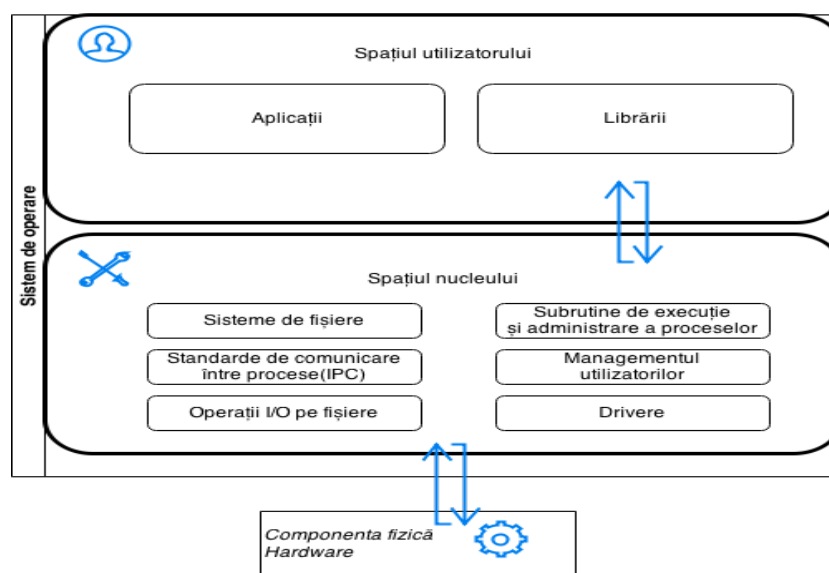


Figura 2.1: Structura unui sistem de operare de tip Unix

După cum se poate observa, pentru managementul fișierelor și a sistemelor de fișiere răspunderea cade pe seama nucleului. Crearea unui sistem de fișiere presupune cunoștințe interne despre structura acestuia, procese și constrânge programatorul la folosirea unui număr limitat de unelte. În mediul de dezvoltare a al kernelului chiar și cele mai triviale funcții din limbajele de nivel înalt lipsesc (spre exemplu protecția memoriei lipsește și apar așa numitele situații de *kernel panic* (trad. panică de nucleu) care reprezintă stări fatale de eroare în sistemul de operare din care nu se poate reveni, este nevoie de atenție sporită pentru folosirea primitivelor de sincronizare, codul programelor trebuie scris neapărat în limbajul C, o mică breșă în securitatea modului customizat poate compromite întreg sistemul). În același timp, implementarea devine foarte dependentă de arhitectura și versiunea kernelului, fiind necesare teste suplimentare chiar pentru portarea între diferitele distribuții de Linux.¹

O alternativă este folosirea modului de dezvoltare a sistemelor de fișiere în spațiul utilizatorului. Urmarea acestei direcții este posibilă prin folosirea modului de kernel FUSE, și a suitei de utilități aferente acestuia, principalul avantaj oferit fiind ușurința de dezvoltare având la îndemână toate trăsăturile și librăriile limbajelor de programare la nivel înalt.

¹ [6] Introducere

FUSE nu își propune să înlocuiască implementările sistemelor celebre din kernel (ca Ext, ReiserFs, XFS ș.a.) în spațiul utilizatorului deoarece este evident o oarecare restrângere la nivelul performanței datorată abstractizării și îndepărtării de nivelul fizic.

FUSE oferă un mediu prin care se poate construi un sistem de fișiere complet funcțional dar în același timp independent de mașină (evident ne referim aici la sisteme de fișiere bazate pe filozofia Unix și nu la cele cu arhitectură total diferită) cu o interfață sigură și foarte eficient implementată care s-a dovedit a fi stabilă de-a lungul timpului.

Un alt aspect care crește flexibilitatea sistemelor de fișiere în userspace este posibilitatea oferită utilizatorilor obișnuiți de a efectua operația de montare/demontare, dacă aceștia sunt agreați de superutilizator (amintim faptul că în implementările tradiționale de Unix, singurul utilizator capabil să execute aceste operații este root-ul; totuși unele distribuții de linux permit mai nou și altor utilizatori din sistem această capacitate).

Fiind un proiect relativ mare, distribuit sub o licență de tip liber, în jurul acestuia s-a format o comunitate de programatori care asigură compatibilitatea între platforme și ajută la evoluția acestuia în același timp cu kernelul de Linux.

Beneficiile enumerate mai sus sunt extrem de importante mai ales în contextul zilelor noastre, când folosirea cloud computing-ului și a infrastructurii puternic virtualizate sunt cotidene.

2.2 Structura mediului de lucru FUSE

Mediul de lucru FUSE are din următoarele componente principale:

- un modul de sistem de fișiere în kernel
- o librărie în mediul utilizatorului (libfuse)
- un utilitar care permite montarea și demontarea sistemelor de fișiere (fusermount)

Primele două componente sunt exemplificate în Figura 2.2¹⁰, unde vedem ruta unui apel care se adresează unui sistem de fișiere de tip fuse. Astfel apelul comenzii `ls` de terminal va fi preluată de sistemul virtual de fișiere (pe desen VFS – despre care o să discutăm ulterior), care îl direcțează în modulul de kernel fuse, care prin intermediul interfeței libfuse va accesa sistemul de fișiere

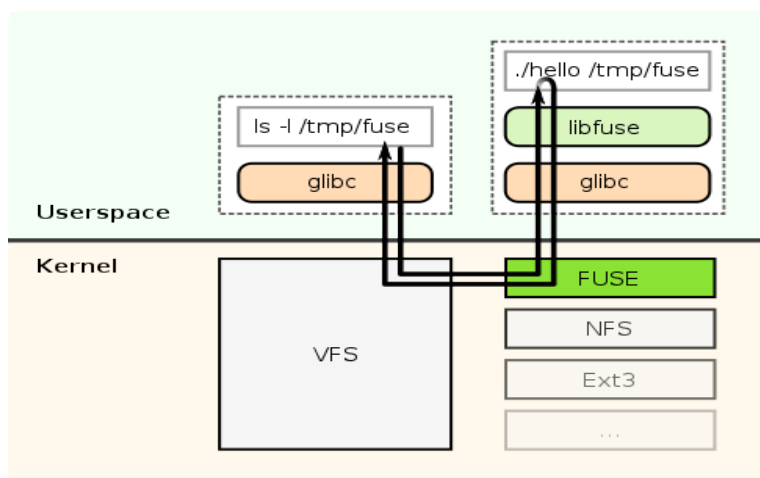


Figura 2.2: Structura apelurilor modulului FUSE

personalizat și va returna răspunsul definit de acesta. Putem observa celelalte tipuri de sisteme de fișiere (NFS, Ext3) disponibile în nucleu și care vor fi folosite fiecare în funcție de contextul adecvat.

În multe distribuții de Linux, modulul de FUSE vine deja compilat și activat în kernel fiind folosit pentru sarcini interne ale sistemului de operare. Pentru a verifica disponibilitatea acestuia este suficient să facem o căutare în configurația de boot a sistemului de operare.

Spre exemplu, pentru o mașină oarecare, pe care rulează kernelul de Linux 3.13.0-24, rulând comanda `grep FUSE_FS /boot/config-3.13.0-24-generic` obținem:

```
CONFIG_FUSE_FS=y
```

y(yes trad. da) care înseamnă că acesta este compilat în kernel

Modulul FUSE este în prezent compatibil cu următoarele sisteme de operare¹¹:

- Linux (începând cu versiunea 2.4)
- NetBSD (acesta avea dezvoltată cu câțiva ani înainte o implementare de sistem de fișiere în userspace numit PUFFS având același scop și aceeași structură cu FUSE dar o librărie cu standarde diferite. Progresiv s-a urmărit interconectarea acestor două implementări iar începând cu NetBSD 6.0 PERFUSE oferă un canal de comunicare între acestea chiar din interiorul kernelului)
- Mac OS X (începând cu versiunea 10.5 – Darwin, este disponibilă o variație numită Fuse4X care este total compatibilă cu FUSE oferind suplimentar și un mediu de lucru pentru limbajul Objective C și o extensie pentru kernel)
- OpenSolaris
- GNU/Hurd (kernelul oferit ca alternativă pentru Linux, dezvoltat de FSF (Free Software Foundation) trad. Fundația programelor libere; A jucat un rol extrem de important în dezvoltarea FUSE, observând necesitatea unui sistem de fișiere în spațiul utilizatorului încă din 1995)
- Câteva opțiuni pentru Windows: Dokan, fuse4win, Universal Fuse, FUSE-NT, WinFUSE, Cooperative Linux(împreună cu Samba)

2.3. Sistemele virtuale de fișiere în context Linux

Un sistem virtual de fișiere (VFS) sau Virtual File Switch(trad. Comutator virtual de fișiere) este un nivel de abstractizare peste un sistem de fișiere, având rolul de a permite aplicațiilor client să acceseze diferite tipuri de sisteme concrete de fișiere într-un mod uniform (de aici și termenul de comutator – are rol în normalizare).¹²

Un VFS este o interfață între kernel și sistemul de fișiere, din acest motiv orice sistem de fișiere care respectă restricțiilor acestuia poate fi atașat sistemului și folosit de către kernel și aplicațiile utilizator.

Cel mai trivial exemplu este modul în care sistemul de operare linux poate monta partiții de tip NTFS iar utilizatorul nu va simți aproape deloc diferența. Un alt exemplu din domeniul rețelelor este protocolul samba – SMB, care are aceleași principii și scop.

¹¹ [11]

¹² [12]

VFS-ul trebuie să administreze toate tipurile de sisteme de fișiere care sunt montate la un anumit moment iar pentru aceasta stochează structuri de date care descriu starea întregului sistem virtual și a fiecărui sistem concret care este montat.¹³

Fișierele de sistem sunt descrise în termeni similari cu FFS și ext2, împărțindu-le în blocuri și inoduri. De asemenea, toate sistemele de fișiere Linux folosesc o zonă de cache (trad. tampon) pentru a stoca într-o poziție mai accesibilă datele din sistemele montate. Această zonă este independentă de sistemele de fișiere dar este administrată de către kernelul de Linux folosind VFS-ul care are într-un mod similar, o zonă de cache astfel încât inode-urile folosite frecvent să poată fi accesate cât mai eficient.

2.4 Exemplificări

Pentru a putea înțelege mai bine avantajele folosirii sistemelor de fișiere în userspace și a posibilităților de virtualizare extreme oferite de acesta, vom prezenta în cele ce urmează o listă cât mai variată de cazuri de utilizare.¹⁴

FTPFS – Reprezintă un sistem de fișiere care suportă accesul la un protocol de transfer de date, și anume FTP, folosind API-uri standard. În sistemele de operare bazate pe Linux, acesta a fost inițial implementat ca și un modul de kernel observând-u-se de la bun început neeficiența acestuia, la începutul anilor 2000 începându-se trecerea progresivă la abordarea FUSE. Se numește în prezent CurlFtpFS deoarece este folosește conceptele din librăria libcurl. Există soluții similare și pentru celelalte sisteme de operare populare cum ar fi Mac OS X și Windows.

SSHFS – Este un sistem de fișiere menit să interacționeze cu fișiere remote (trad. care nu se află local) pe servere sau stații de lucru. Acesta se bazează pe SFTP, un protocol de rețele care permite transmiterea sigură (până la Heartbleed – vulnerabilitate descoperită recent în SSH) de date între un client și un server. SSHFS a fost creat de Miklos Szeredi (care este și creatorul FUSE) aflându-se în prezent într-o stare activă de dezvoltare.

Principalul avantaj care îl oferă SSHFS comparativ cu alte sisteme de fișiere peste rețea este faptul că are nevoie de o configurare minimală. Dat fiind faptul că stația are deja acces prin SSH la mașina remote, nu va fi necesară implementarea unui alt model de autentificare și de asigurare a encriptării informației interschimbate și nici de excepții în firewall.

GDriveFS (Google Drive File System) – Google Drive este serviciul de stocare de date în cloud oferit de către Google Inc. GDriveFs permite montarea acestui serviciu web (prin intermediul API-ului disponibil) ca și un sistem de fișiere oferind astfel posibilități vaste de utilizare ale acestuia. GDriveFS este extrem de util mai ales că deși serviciul Google Drive a fost lansat de câțiva ani, nu are o interfață care să poată fi rulată local pe orice sistem de operare (spre exemplu nu există deocamdată o aplicație nativă de Linux) fiind folosită una web care limitează utilizatorul.

WikipediaFS – Este un sistem de fișiere care permite utilizatorilor vizualizarea și modificarea articolelor de tip Wikipedia ca și pe niște fișiere reale din sistemul local. Acesta se bazează pe comunicarea cu o interfață REST-ful peste HTTP. Această abordare este extrem de eficientă deoarece permite crearea de articole prin folosirea utilităților aflate local pe sistem (cum ar fi un editor de text sau un IDE). De asemenea permite migrări masive de date/ articole între diferite servere.

¹³ [13] Capitolul 9.2 The filesystem

¹⁴ [10]

HDFS (Hadoop Distributed File System) – Este un sistem de fișiere distribuit, scalabil și portabil scris în limbajul Java, fiind folosit de platforma Hadoop care oferă un mediu de stocare și procesare de seturi de date organizate în clustere, fiind una dintre cele mai importante proiecte ale Apache Software Foundation. HDFS stochează date de dimensiuni mari și foarte mari (de ordinul GB și TB) peste mai multe sisteme făcând datele mai accesibile, asigurând persistența și eliminând necesitatea folosirii tehnologiei RAID (care se cunoaște că duplică datele). HDFS a fost proiectat în special pentru date cu caracter imobil și poate determina scăderi de performanță dacă este folosit pentru sisteme cu un număr mare de scrieri (de cele mai multe ori concurente).

S3FS – Este un sistem de fișiere care permite montarea transparentă a bucket-urilor (trad. găleată – cu sensul de grupare de date) de stocare S3 oferite de Amazon prin AWS. Din acest motiv utilizatorul va putea rula comenzi de sistem ca și asupra oricărui alt suport fizic. Din nefericire însă, deși utilitatea acestuia este extremă, dezvoltarea a fost blocată de mai multe ori de către Amazon din motive de copyright .

CloudFusion – Se folosește de aceleași principii ca și GDriveFs și S3FS, dar permite accesul într-un mod uniform la mai multe servicii de stocare în cloud cum ar fi: Sugarsync, Dropbox, Google Storage, Amazon S3, WebDAV(prin furnizorii T-Online, 4shared, GMX, OneDrive, yandex). De asemenea acest proiect a dus la câteva divergențe de copyright și afiliere cu furnizorii acestor servicii.

GVFS (Gnome Virtual File System) – Este un sistem virtual de fișiere extrem de important pentru un sistem de operare bazat pe Linux deoarece de acesta depinde DE-ul (Desktop Environment-ul trad. mediul de lucru). DE-ul este una dintre cele mai importante componente din spațiul de utilizator al sistemului de operare, reprezentând vertebra principală a mediului grafic de lucru. Cu alte cuvinte, lipsa acestuia va avea ca efect constrângerea utilizării exclusive a terminalului pentru interacțiunea cu sistemul – aceasta fiind o perspectivă destul de tristă pentru utilizatorul obișuit. GVFS permite o accesare uniformă a datelor externe (prin protocoale ca: SFTP, FTP, WebDAV, SMB ș.a.) și a celor locale (prin Udev, OBEX, MTP). Accesarea acestora se face printr-un apel cu un format asemănător URI-urilor (Unique Resource Identifier), cum ar fi: `smb://server/data`

2.5 Interfețe pentru limbaje de programare

Concentrându-se pe dezvoltarea în mediul utilizatorului, FUSE a încurajat crearea de API-uri care să permită accesul oricărui limbaj de nivel înalt la funcționalitățile oferite de acesta, ținând cont că atât libfuse cât și modulul de kernel sunt scrise în limbajul C. Aceste API-uri, numite în documentația oficială¹⁵ language binding-uri (trad. legături pentru limbaje) au reușit de-a lungul timpului să reducă deficitul de performanță, iar pentru unele limbaje (cum ar fi Python) au apărut mai multe binding-uri, fiecare fiind axată pe dezvoltarea de concepte de nivel înalt respectiv care să mimeze cât mai fidel lucrul în kernel.

În prezent FUSE are suport pentru majoritatea limbajelor de programare, atât imperative cât și funcționale. Dintre acestea enumerăm (împreună cu numărul de API-uri disponibile): C++(4), Java, C#(2), Go, Haskell, TCL, Python(3), Perl, Sh, Ruby(3), Lua, Erlang, PHP, Lisp – Common, Javascript.

Pentru sistemului de fișiere care face subiectul acestei lucrări am ales ca și limbaj principal de dezvoltare Python, mai specific API-ul fusepy, motivația fiind prezentată în capitolul următor.

15 [10] Language Bindings

4.EpsFS – Sistemul de fişiere cu un set extins de permisiuni

În acest capitol vom prezenta structura şi funcţionalitatea **Sistemului de fişiere cu set extins de permisiuni** (la care vom face referire în continuare ca **EpsFS**) acesta reprezentând scopul acestei lucrări. După cum se va putea observa şi pe parcurs, înţelegerea modului de funcţionare a acestuia este condiţionată de parcurgerea conceptelor prezentate anterior.

Principalul obiectiv luat în calcul a fost cel de a oferi posibilitatea configurării cât mai flexibile şi dinamice a drepturilor de acces a utilizatorilor la un sistem de fişiere. Principalele elemente de noutate aduse de implementarea acestui produs sunt:

- permiterea configurării de grupuri imbricate de utilizatori
- permiterea accesului la date în funcţie de protocolul prin care utilizatorul este conectat (spre exemplu SSH)
- permiterea accesului la date în funcţie de adresa IP a utilizatorului (sau a unui set de adrese IP)
- accesul la datele sistemului în funcţie de diferite intervale orare

Aceste inovaţii se bazează pe permisiunile specifice pentru sistemele de fişiere de tip Unix (rwx – citire/scriere/execuţie) corespunzătoare categoriilor reprezentative de utilizatori (owner – group – others) interconectate cu ajutorul operanzilor logici AND (trad. şi) şi OR (trad. sau).

Pentru implementare s-a folosit limbajul de programare Python care este un limbaj de nivel înalt, orientat pe obiecte, imperativ şi dinamic care permite un ritm alert de dezvoltare (comparativ cu cele statice ca: C, Java, C++). Python vine instalat ca şi standard pe majoritatea sistemelor bazate pe Linux, iar pentru a conştientiza dependenţa userspace-ului de acest limbaj trebuie menţionat că încercarea de ştergere a acestuia va lăsa sistemul într-o stare (poate chiar şi iremediabilă) de nefuncţionare – afirmaţie nefiind valabilă pentru multe alte limbaje mai populare (cum ar fi Java).

Dintre wrapper-urile(trad. librării de învelire) disponibile am ales fusepy, fiind cea mai adecvată soluţie pentru, oferind tratarea la nivelul optim (celelalte librării fiind prea low-level sau având o documentaţie necorespunzătoare).

Configurarea noilor permisiuni stabilite mai sus va fi stocată în interiorul fiecărui director din punctul de montare al sistemului de fişiere, într-un regular file (folosim termenul din engleză pentru a sublinia tipul acestuia conform clasificării făcute în primul capitol) denumit ,epsfs , prefixul , fiind folosit pentru a indica utilitatea specială a acestuia. Acest fişier nu va putea fi citit, scris sau executat de către utilizatorii obişnuiţi ai sistemului de fişiere, ci doar de utilizatorul privilegiat cărui îi este permisă operaţia de montare (mai specific fusermount)

,epsfs va conţine, pentru fiecare *fişier* din directorul curent (*fişier* cu sensul Unix – adică unul dintre regular file, directory ş.a.), intrări de forma următoare:

categorie		serviciu		ip		dată		interv_timp		dr_efective
u user		ssh		adrIP		yyyy-mm-dd		h:m:s,h:m:s		r
g group		*		adrIP,adrIP		yyyy-mm-dd,yyyy-mm-dd		*		w
o others				*		*				x

Operatori:

 reprezintă operatorul logic care poate (deocamdată) să fie *unul* dintre:

<and>(sensul de AND logic)

<or>(sensul de OR logic)

Specificăm că pentru evaluarea expresiilor logice, AND va avea întâietate, urmând ca pe viitor să definim un mod de specificare a ordinii efectuării operațiilor prin paranteze.

Operanzi:

* sau lipsa unei valori între oricare doi are sensul de all (trad. tot) cu alte cuvinte, toate posibilitățile din acea categorie. Spre exemplu lipsa unei valori pentru **interv_timp** va permite accesul fișierului în orice interval orar.

categorie reprezintă unul dintre cele trei tipuri de utilizatori primari

serviciu având ca posibile valori în prezent ssh sau *

ip având ca posibile valori o adresă ip, un interval de adrese ip - specificând capetele acestuia despărțite prin "," sau *

dată având ca posibile valori o dată de forma yyyy-mm-dd, sau un interval de date - specificând capetele acestuia despărțite prin "," sau *

dr_efective unul sau mai multe opțiuni dintre drepturile primare: r, w, x, -

Pentru exemplificare vom prezenta configurația de mai jos:

```
/dir1
|____ /dir2
|____ /file1
|____ /file2
|____ /,epsfs ----->va conține:
dir2: u<and><and><or><and><or>rw
dir2: g<or>ssh<and>192.168.1.15<and><and><and>rw
dir2: o<and><or><and><or><and>-
file1: u<and><and><or><and><or>rw
file1: g<or>ssh<and><and><and><and>r
file1: o<and><or><and><or><and>r
file2: u<and><and><or><and><or>rw
file2: g<or>ssh<and>127.0.0.1<and><and><and>r
file2: o<and><or><and><or><and>-
```

Se poate observa de exemplu că accesul la dir2 se va face de u (owner) în orice situații, g(grup) va avea citire/scriere doar dacă este conectat prin ssh cu ip-ul 192.168.1.15 iar o (others) nu au acces de niciun tip.

După cum putem deduce din structurarea acestui fișier special, posibilitățile de extindere sunt nelimitate, prin adăugarea unei noi categorii de operanzi.

Pentru configurarea grupurilor imbricate de utilizatori, în rădăcina punctului de montare va exista un fișier numit groups.,epsfs care va conține regulile de interconectare a grupurilor, și relațiile ierarhice între acestea sub formă de intrări grup:grup_părinte acest fișier fiind consultat ori de câte ori este necesar.

Pentru a simplifica modul de specificare a acestor permisi, utilizatorul va avea acces la o interfață web, nefiind astfel nevoit să modifice fișierele ,epsfs manual, cu un editor de text. Această interfață

va fi bazată pe cele mai recente tehnologii aderente limbajului Python dintre care amintim mediul de dezvoltare Django.

Configurarea sistemului de fişiere pentru ca operaţiile de bază să aibă efectul dorit se face prin moştenirea comportamentului clasei Operations din API-ul fusepy. În cele ce urmează vom prezenta semnatura operaţiilor oferite de către libfuse şi fusepy, acestea având corespondent direct în implementările sistemelor de fişiere scrise în kernel (după cum am exemplificat şi anterior). Pentru a face acest model cât mai comprehensibil, am ales prezentarea variantei de specificare în limbajul C, API-ul de Python reprezentând o simplă interfaţă cu rol de abstractizare. Considerăm numele funcţiei ca fiind auto-explicativ (manualul de Linux putând oferi mai multe detalii unde este cazul)

```
int(* getattr )(const char *, struct stat *)
int(* readlink )(const char *, char *, size_t)
int(* mknod )(const char *, mode_t, dev_t)
int(* mkdir )(const char *, mode_t)
int(* unlink )(const char *)
int(* rmdir )(const char *)
int(* symlink )(const char *, const char *)
int(* rename )(const char *, const char *)
int(* link )(const char *, const char *)
int(* chmod )(const char *, mode_t)
int(* chown )(const char *, uid_t, gid_t)
int(* truncate )(const char *, off_t)
int(* open )(const char *, struct fuse_file_info *)
int(* read )(const char *, char *, size_t, off_t, struct fuse_file_info *)
int(* write )(const char *, const char *, size_t, off_t, struct fuse_file_info *)
int(* statfs )(const char *, struct statvfs *)
int(* flush )(const char *, struct fuse_file_info *)
int(* release )(const char *, struct fuse_file_info *)
int(* fsync )(const char *, int, struct fuse_file_info *)
int(* setxattr )(const char *, const char *, const char *, size_t, int)
int(* getxattr )(const char *, const char *, char *, size_t)
int(* listxattr )(const char *, char *, size_t)
int(* removexattr )(const char *, const char *)
int(* opendir )(const char *, struct fuse_file_info *)
int(* readdir )(const char *, void *, fuse_fill_dir_t, off_t, struct fuse_file_info *)
int(* releasedir )(const char *, struct fuse_file_info *)
int(* fsyncdir )(const char *, int, struct fuse_file_info *)
void*( * init )(struct fuse_conn_info *conn)
void( * destroy )(void *)
int(* access )(const char *, int)
int(* create )(const char *, mode_t, struct fuse_file_info *)
int(* ftruncate )(const char *, off_t, struct fuse_file_info *)
int(* fgetattr )(const char *, struct stat *, struct fuse_file_info *)
int(* lock )(const char *, struct fuse_file_info *, int cmd, struct flock *)
int(* utimens )(const char *, const struct timespec tv[2])
int(* bmap )(const char *, size_t blocksize, uint64_t *idx)
int(* ioctl )(const char *, int cmd, void *arg, struct fuse_file_info *, unsigned int flags, void *data)
int(* poll )(const char *, struct fuse_file_info *, struct fuse_pollhandle *ph, unsigned *reventsp)
int(* write_buf )(const char *, struct fuse_bufvec *buf, off_t off, struct fuse_file_info *)
int(* read_buf )(const char *, struct fuse_bufvec **bufp, size_t size, off_t off, struct fuse_file_info *)
int(* flock )(const char *, struct fuse_file_info *, int op)
int(* fallocate )(const char *, int, off_t, off_t, struct fuse_file_info *)
```


Concluziile proiectului

Proiectul de față – EpsFS – prezintă implementarea unui sistem de fișiere cu o flexibilitate avansată de administrare a permisiilor. Opțiunea de specificare a grupurilor imbricate de utilizatori și modul de autorizare care se extinde până la protocolul de acces și adresa IP a utilizatorului sunt inovații considerabile, ținând cont că au loc chiar în interiorul sistemului de fișiere.

FUSE este o abordare revoluționară pentru dezvoltarea sistemelor de fișiere și oferă o flexibilitate extremă prin plasarea în userspace; iar după cum se poate observa chiar din exemplele oferite în a doua parte a capitolului secund, VFS-urile realizate prin metodologia FUSE sunt extrem de utile în contextul cloud computing-ului.

Dintr-o altă perspectivă, putem observa prejudiciul adus inovației de către companiile (ca și Google, Amazon, ș.a. prezentate în exemplele din al doilea capitol) care interzic utilizarea propriilor servicii de către dezvoltatorii care nu vor să se afilieze sau a oricărui mod automatizat de accesare, altul decât cel oferit de interfețele componente. Lucrarea de față încurajează dezvoltarea colaborativă și sugerează utilizarea standardelor și a licențelor libere pentru progresul tehnologiei prin publicarea acestora sub termenii unei licențe care respectă drepturile utilizatorilor.

Bibliografie

- [1] Boian, M.F., Vancea A., Boian R., Bufnea D., Sterca A., Cobârzan C., Cojocar D., **“Sisteme de operare”**, Editura Risoprint, 2006
- [2] Eggert P. R., Parker S.D., **“File Systems in User Space”**, Articol UCLA Computer Science Department
- [3] Mauerer W., **“Professional Linux Kernel Architecture”**, Editura Wiley Publishing, Inc., 2008
- [4] Bovet D.P., Cesati M., **“Understanding the Linux Kernel, Third Edition”**, O'Reilly Media, Inc., 2006
- [5] Rajgarhia A., Gehani A., **“Performance and Extension of User Space File Systems”**, Conferința ACM Symposium on Applied Computing, 2010
- [6] Kanaujia V., Giridhar C., **“FUSE'ing Python for Development of Storage Efficient Filesystem”**, articol <http://ojs.pythonpapers.org/index.php/tpp/article/view/244>, 2012
- [7] Pate S.D., **“UNIX Filesystems. Evolution, Design, and Implementation”**, Editura Wiley Publishing, Inc., 2003
- [8] McPherson A., Proffitt B., Hale-Evans R., **“Estimating the Total Development Cost of a Linux Distribution”**,
<http://www.linuxfoundation.org/sites/main/files/publications/estimatinglinux.html>, 2008
- [9] Matthäi K., http://upload.wikimedia.org/wikipedia/commons/thumb/7/77/Unix_history-simple.svg/3079px-Unix_history-simple.svg.png, 2008
- [10] Documentatia oficiala a modulului FUSE <http://fuse.sourceforge.net/>
- [11] Documentația oficială a sistemului EXT4 https://ext4.wiki.kernel.org/index.php/Main_Page,
<http://kernelnewbies.org/Ext4>
- [12] Braam P.J., **'Linux virtual file system'**, <http://www.coda.cs.cmu.edu/doc/talks/linuxvfs/>
- [13] Rusling D.A., **'The linux kernel documentation page'**, <http://www.tldp.org/LDP/tlk/tlk.html>
- [13] Sisteme de operare care suportă modulul fuse: <http://sourceforge.net/apps/mediawiki/fuse/index.php?title=OperatingSystems>

Anexe

Anexa1 Termenii de distribuire a conținutului lucrării

GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright © 2007 Free Software Foundation, Inc. <<http://fsf.org/>>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the

work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial

commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS