4. ADMINISTRARE CONTURI SI PERMISIUNI

4.1. Administrarea conturilor de utilizator....2
4.1.1.Generalitati....2
4.1.2.Autentificarea utilizatorilor....3
4.1.2.1.Descriere si posibilitati....3
4.1.2.2.Fisierul /etc/passwd....3
4.1.2.3.Fisierul /etc/shadow....4
4.1.2.4.Fisierul /etc/group....4
4.1.3.Comenzi pentru administrarea conturilor de utilizator....5
4.1.3.1.Lista de comenzi....5
4.1.3.2.Crearea unui cont de utilizator....5
4.1.3.3.Modificarea datelor unui cont de utilizator....6
4.1.3.4.Stergerea unui cont de utilizator....6
4.1.4.Obtinerea privilegiilor altui utilizator....6
4.1.4.1.Ratiuni....6
4.1.4.2.Privilegii complete: comanda su....7
4.1.4.3.Privilegii partiale: comanda sudo....7
4.1.5.Monitorizarea utilizatorilor prezenti in sistem....8
4.1.5.1.Comenzile who si w....8
4.1.5.2.Comanda last....9
4.1.5.3.lastlog....9
4.2. Permisiuni....9
4.2.1.Sistemul de permisiuni la nivel de fisier....9
4.2.2.Efectul permisiunilor asupra diferitelor tipuri de fisiere....9
4.2.3.Vizualizarea permisiunilor....11
4.2.4.Felul in care sistemul de operare determina permisiunile unui user pe un fisier....12
4.2.5.Comenzi pentru administrarea permisiunilor....12
4.2.5.1.Schimbarea proprietarului si a grupului....12
4.2.5.2.Schimbarea permisiunilor pe fisiere....13
4.2.6.Controlul permisiunilor pentru fisierele nou-create....15

4.1. Administrarea conturilor de utilizator

4.1.1. Generalitati

Administrarea conturilor de utilizator (user accounts) reprezinta una dintre atributiile de baza ale administratorului de sistem. Un cont presupune in general:

- **user name** – numele folosit de utilizator pentru a accesa sistemul, numit si *login name*. Permite sistemului de operare sa distinga intre mai multi utilizatori, astfel incat fiecare dintre ei sa aiba propriul set de setari si de permisiuni
- **parola** – combinatie secreta de caractere care împreuna cu user name-ul realizeaza *autentificarea*; sistemul trebuie sa se asigure ca utilizatorul este cel care se pretinde a fi
- **directorul personal (home directory)** – este acel director din arborele de fisiere in care utilizatorul isi poate stoca fisierele proprii, si in acelasi timp directorul in care utilizatorul este plasat automat dupa login
- **mediul de lucru (environment)** – reprezinta totalitatea configurarilor personalizate proprii userului in cauza. Fiecare utilizator are propriul sau set de fisiere de initializare care “construiesc” mediul de lucru, aflate in directorul sau personal sub forma unor fisiere/directoare ascunse

Din punct de vedere al sistemului de operare, fiecare utilizator este identificat printr-un numar, numit *user ID* (identificatorul utilizatorului), la fel cum fiecare persoana are un CNP (cod numeric personal). Desi in viata reala noi spunem “fiecare nume are asociat un CNP”, in sistemul de operare asocierea este inversa: utilizatorii sunt identificati nativ dupa UID, iar corespondenta UID – username exista doar pentru comoditatea noastra.

Utilizatorii sunt organizati in *grupuri*, ceea ce ofera administratorului avantajul de a putea configura permisiunile la nivel de grup (in loc sa le stabileasca individual, pentru fiecare utilizator in parte). Sistemul de operare identifica fiecare grup printr-un group ID (GID). Fiecare user aparține unui grup primar și poate face parte, suplimentar, din pana la 65536 grupuri secundare. După cum se va vedea în sectiunea dedicata permisiunilor, fiecare nou grup în care este introdus un utilizator nu face altceva decât să-i adauge acestuia noi permisiuni, iar acest mecanism este esential pentru accesul la o întreaga serie de servicii ale sistemului de operare.

Sistemul de operare Linux fiind multiuser, este necesar ca fiecare utilizator sa poata avea fisiere proprii si care sa poata fi protejate fata de alti utilizatori. De aceea UID-ul intervine in cel putin doua puncte cheie din sistemul de operare:

- fiecare fisier are un utilizator proprietar si un grup proprietar (identificate prin UID si GID, memorate în inode-ul fisierului) și care au fiecare câte un set de permisiuni pe fisier. Exista asadar permisiuni la nivel de fisier ce se aplica numai ownerului, permitand separarea permisiunilor acestuia de restul utilizatorilor
- fiecare proces ce ruleaza in sistemul de operare are asociata o anumita pereche (UID,GID), care determina contextul lui de securitate: in functie de relatia intre UID-ul/GID-ul său si cele ale fisierelor pe care incearca sa le acceseze, permisiunile de pe fisiere vor stabili dacă accesul ii va fi permis sau nu

UID-ul este un numar mai mic sau egal cu 2147483647, dar tipic este limitat la 60000 de către setarile sistemului de operare (oricum sunt rare situatiile când cineva ar dori sa creeze pe o singura mașină un numar atât de mare de conturi de utilizator). In functie de valoarea sa, conturile se impart in:

- contul de superuser (root) – este cel cu UID-ul 0. Acest cont are permisiuni totale in sistemul de operare
- conturi de sistem – sunt conturi destinate rularii de servicii (ex: servere). Aceste conturi nu au in general permisiune de login interactiv, ci sunt folosite numai pentru rularea de procese. UID-ul acestor conturi se afla intre 1 si o limita convențională, ce poate varia de la o distributie la alta (valori tipice sunt 100, 500 sau 1000).
- conturi de utilizator – sunt conturi destinate utilizatorilor umani si permit login-ul interactiv

UID	Cont	Descriere
0 – 100/500/1000	Root, daemon, bin, sys etc	Conturi de sistem
100/500/1000 - 2147483647	Useri obisnuiti	Conturi normale de utilizator

Nota: *de fapt, motivul pentru care utilizatorii se autentifica este pentru ca sistemul de operare sa stie cu UID/GID sa ruleze procesele acestora. Login-ul in sistem reprezinta un proces de autentificare urmat de rularea unuia sau mai multor procese cu UID/GID-ul autentificat.*

4.1.2. Autentificarea utilizatorilor

4.1.2.1. Descriere si posibilitati

Autentificarea reprezinta verificarea identitatii utilizatorilor. Ea presupune existenta unor baze de date ce depoziteaza informatiile de autentificare. Exista doua abordari ale autentificarii:

- **autentificare locala** – bazele de date folosite pentru verificarea identitatii utilizatorului se afla chiar pe statia pe care acesta se autentifica. Solutia este satisfacatoare pentru statii izolate, insa pentru retele in care utilizatorii migreaza de la o statie la alta (si au nevoie de drept de login pe mai multe statii) solutia nu scaleaza bine, deoarece administratorul retelei ar trebui sa creeze acelasi cont pe toate statiile implicate
- **autentificare in retea** – conturile nu mai sunt create pe fiecare statie in parte, ci pe unul sau mai multe servere centrale; atunci cand utilizatorul se autentifica pe o anumita statie, cererea de autentificare este trimisa catre un astfel de server. Aceasta solutie permite administrarea centralizata a conturilor de utilizator – contul se creeaza intr-un singur loc (pe server) iar utilizatorul in cauza se poate loga pe toate statiile configurate sa depinda de serverul respectiv.

  Nota: *ca exemple de implementare, amintim domeniile NIS (Network Information Service) in Unix/Linux, respectiv domeniile Microsoft, in lumea Windows*

Atunci cand informatiile de autentificare sunt memorate in fisiere text locale (cazul implicit), fisierele de configurare sunt:

- **/etc/passwd** – baza de date cu utilizatori
- **/etc/shadow** – baza de date cu parole
- **/etc/group** – baza de date cu grupuri
- **/etc/gshadow** – baza de date cu parole de grup

4.1.2.2. Fisierul /etc/passwd

Contine baza de date cu conturi si proprietatile acestora. Fiecare linie din fisier contine toate informatiile unui utilizator; pentru separarea informatiilor (campurilor) se foloseste caracterul **:** după cum urmează:

```
username:x:UID:GID:comment:home_directory:login_shell
```

Semnificatiile campurilor sunt urmatoarele:

username	eticheta text asociata userID-ului. Trebuie sa inceapa cu o litera, caracterele permise fiind litera mica sau mare, cifra, punct, minus şi underscore (ultimele trei nefiind însă permise în toate distributiile)
x	inainte in acest camp era pastrata parola utilizatorului; acum parolele sunt memorate separat, in */etc/shadow*
UID (User Identifier)	numărul asociat fiecărui utilizator înregistrat. Numerele între 0 şi 100/500/1000 sunt rezervate penru utilizatori privilegiaţi şi de sistem. Peste acest prag se afla conturile de utilizatori obişnuiţi. Este permis dar nu indicat ca doi utilizatori să aibă acealaşi UID
GID (Group Identifier)	numărul grupului primar din care face parte utilizatorul
Comment	conţine de obicei numele întreg al utilizatorului
home_directory	conţine calea către directorul personal al utilizatorului
login_shell	defineşte interpretorul de comenzi implicit al utilizatorului, care poate fi /bin/sh, /bin/ksh, /bin/csh, /bin/zsh, /bin/bash, /bin/tcsh sau orice alt shell prezent in sistem (si accesibil userului in cauza!)

Informatia din fisierul **/etc/passwd** se modifica in doua moduri:

- prin utilizarea comenzilor **useradd,usermod** si **userdel**
- prin editare manuala, folosind comanda **vipw**, care nu permite salvarea fisierului decat daca acesta este valid.

Nota: *verificarea validitatii fisierului /etc/passwd se poate efectua cu comanda* ***pwck****.*

4.1.2.3. Fisierul /etc/shadow

Contine baza de date cu parole si setari legate de acestea. Formatul unei linii este:

```
loginID:password:lastchange:min:max:warn:inactive:expire:
```

loginID	username-ul din passwd (reprezinta legatura intre cele doua baze de date, echivalentul unei „chei externe” in bazele de date relationale)
password	o succesiune de caractere reprezentând hash-ul parolei, sau ! pentru conturi care încă nu au parola setata. Hash-ul este o secventa de caractere de lungime fixa, calculata ireversibil pe baza parolei.
lastchg	reprezintă numărul de zile scurse între 1 ianuarie 1970 şi data ultimei schimbări a parolei
min	nr minim de zile între două schimbări successive ale parolei
max	nr maxim de zile între două schimbări successive ale parolei (altfel spus, nr de zile după care parola expira)
warn	cu câte zile înainte de expirarea parolei este avertizat userul
inactive	nr de zile de după expirarea parolei în care userului încă i se permite login-ul cu vechea parola; în acest interval userului i se propune schimbarea de parola la fiecare login. Dacă se depășește acest numar de zile, contul se blochează automat
expire	data la care contul expiră

Modificarea continutului acestui fisier se face prin intermediul comenzilor **passwd** (schimbarea parolei utilizatorilor) si **chage** (detaliile legate de expirarea parolelor).
Este posibila sincronizarea bazelor de date *passwd* si *shadow*: comanda **pwconv** creeaza sau modifica shadow pentru a corespunde cu passwd.

4.1.2.4. Fisierul /etc/group

Contine baza de date cu grupuri si specificarea apartenentei utilizatorilor la acestea. Fiecare user face parte dintr-un grup primar (cel declarat in /etc/passwd) si pana la 65536 de grupuri secundare. Fiecare intrare din /etc/group are urmatorul format:

```
groupname:group-password:GID:username-list
```

groupname	este numele grupului şi este format din maximum 8 caractere
group-password	este o relicvă din versiunile vechi de UNIX (nu conţine nimic sau un asterisc)
GID	identificatorul numeric de grup
username list	conţine lista userilor pentru care acest grup este grup **secundar**; grupul primar apare doar în passwd

Fisierul **/etc/group** nu se editeaza manual, ci se modifica cu ajutorul comenzilor **groupadd, groupmod** si **groupdel.**

4.1.3. Comenzi pentru administrarea conturilor de utilizator

4.1.3.1. Lista de comenzi

Sunt disponibile in linia de comanda urmatoarele utilitare:

- **useradd,userdel,usermod** – utilitare de adaugare/modificare/stergere de useri
- **groupadd, groupdel, groupmod** – folosite pentru administrarea grupurilor
- **passwd –** utilitar de modificare a parolei utilizatorilor
- **adduser** (pe unele variante de Linux) – utilitar de tip wizard, in mod text
- **chfn,chpasswd,chage,chsh –** pentru modificarea campurilor din passwd/shadow

Nota: *daca mediul grafic este instalat, majoritatea desktop environment-urilor ofera utilitare grafice de administrare a conturilor de utilizator. Acestea opereaza asupra acelorasi fisiere de configurare mentionate anterior.*

4.1.3.2. Crearea unui cont de utilizator

Comanda **useradd** primeste ca argument username-ul, iar toate atributele utilizatorului respectiv pot fi pasate sub forma de optiuni:

- **-d** – specifica calea catre directorul personal al utilizatorului
- **-m** – solicita crearea acestui director
- **-s** – specifica shell-ul default al utilizatorului
- **-c** – specifica continutul campului comment
- **-g** – specifica grupul primar al utilizatorului
- **-G** – specifica lista de grupuri secundare
- **-u** – stabileste UID-ul utilizatorului creat

Exemplu:

```
useradd –u 102 –d /home/student –m –s /bin/bash –g users student
```

In Linux, comanda *useradd* este guvernata de fisierul ***/etc/default/useradd***, unde pot fi stabilite valori din oficiu pentru optiunile comenzii ale caror valori lipsesc la apelare. Lista de setari default poate fi vizualizata si controlata cu optiunea **useradd –D** (vezi manpage).

Comanda *useradd* permite de asemenea specificarea unui „skeleton directory” – locul din care vor fi copiate fisierele de configurare default la crearea userului, si care este in general ***/etc/skel***.

4.1.3.3. Modificarea datelor unui cont de utilizator

Comanda **usermod** este folosita pentru schimbarea atributelor unui cont de utilizator. Optiunile sale sunt in buna parte aceleasi cu cele ale lui useradd. Cu usermod se pot efectua insa doua operatii suplimentare:

- **-L** (lock) - blocarea temporara a unui cont de utilizator. Efectul este plasarea unui ! in fata hash-ului parolei din /etc/shadow, in acest fel utilizatorul nemaiputandu-se autentifica, dar putand reveni la vechea parola odata ce contul ii este deblocat
- **-U** (unlock) – operatia inversa; elimina caracterul ! din fata hash-ului, contul fiind din nou disponibil pentru login

4.1.3.4. Stergerea unui cont de utilizator

Comanda **userdel** este cea folosita in acest scop. Folosita fara argumente, ea elimina doar contul din *passwd* si *shadow*, fara a sterge directorul personal al utilizatorului. Stergerea directorului poate fi fortata folosind optiunea **–r**:

```
userdel –r student
```

Stergerea unui cont de utilizator nu determina si stergerea tuturor fisierelor ce-l au ca proprietar (owner); acestea vor ramane in continuare in sistemul de fisiere si – mai mult – UID-ul lor proprietar va ramane fostul UID al contului sters, chiar daca contul nu mai exista! La listarea fisierelor, owner-ul va apărea în forma numerica:

```
root@server# ls –l f1
-rw-rw-r-- 1 user1 user1 849 sep 29 11:33 f1
root@server# userdel -r user1
root@server# ls -l f1
-rw-rw-r-- 1 1003 1003 849 sep 29 11:33 f1
```

4.1.4. Obtinerea privilegiilor altui utilizator

4.1.4.1. Ratiuni

Exista diferite situatii in care dorim sa executam comenzi cu drepturile altui cont decat cel cu care suntem logati:

- *nevoia de privilegii partiale sau totale de root*
  - *in conditiile in care suntem posesorii contului de root*. Chiar daca dispunem de parola de root, nu este indicat sa lucram logati cu acest user, ci cu unul neprivilegiat, si sa comutam in root numai atunci cand este cazul. Lucrul in linia de comanda presupune din cand in cand executia de comenzi ce solicita privilegii de root: instalare/dezinstalare soft, configurare de interfete de retea etc. Este incomod si nepractic sa trebuiasca sa ne delogam sau sa deschidem o alta consola virtuala pentru a obtine aceste privilegii
  - *in conditiile in care nu se doreste sa avem acces la contul de root*. Administratorul sistemului poate delega anumite operatii catre utilizatori neprivilegiati, acest lucru presupunand sa le confere acestora o parte din privilegiile sale fara a le divulga insa si parola sa
- *testare de permisiuni*. Atunci cand gandim un sistem de permisiuni sau un ansamblu de setari de securitate ce implica conturi de utilizator, cea mai sigura modalitate de verificare a corectei lor functionari este sa “intram in pielea” userului in cauza si sa verificam

4.1.4.2. Privilegii complete: comanda su

Comanda **su** (substitute user) poate fi folosita pentru a „deveni” alt user. Din punct de vedere al sistemului de operare, acest lucru presupune posibilitatea de a executa comenzi cu UID-ul si GID-ul asociate userului dorit, si chiar „imprumutarea” mediului de lucru al acestuia (variabilele de shell declarate în fisierele de configurare).

Dupa apelare este necesara tastarea parolei userului tinta, cu excepția cazului în care suntem logati ca root și dorim sa devenim un user neprivilegiat, situație în care nu se mai solicita parola.

Daca autentificarea se efectueaza corect, **su** porneste un shell nou ce ruleaza sub UID-ul/GID-ul userului specificat. De aceea, a parasi o sesiune su se reduce la a scrie *exit* sau a apasa *CTRL-d*.

Aceasta posibilitate de a migra temporar către o alta identitate a dus la aparitia urmatoarelor noi acronime:

- **RUID** (Real UID) - UID-ul initial, real, al utilizatorului care executa comanda su
- **EUID** (Effective UID) - este UID-ul temporar, cel pe care userul il capata cât timp este într-o sesiune su

Determinarea RUID se poate face cu ajutorul comenzii **who am i**, iar EUID-ul poate fi aflat cu **whoami** sau **id**.

Sintaxa comenzii *su* este:

```
su [ - ] [ username [ argumente ... ] ]
```

Daca *username* nu este prezent, userul tinta este considerat a fi **root.**

Specificarea lui „-” are ca efect „imprumutarea” mediului de lucru al userului dorit (se preiau și setarile din fisierele de configurare ale acestuia). Daca „-” lipseste este folosit în continuare environment-ul utilizatorului care executa comanda **su**.

Argumentele specificate suplimentar sunt pasate interpretorului de comenzi, care este cel specificat in */etc/passwd* pentru userul tinta. De exemplu, *-c comanda* va executa comanda respectiva in shell, iar *–r* va porni un shell restrictionat. Daca shell-ul lipseste, este folosit implicit /bin/bash.

Nota: *în unele distributii, comanda su "asculta" de fisierul de configurare /etc/suauth (daca exista), in care se specifica, in functie de userul curent si userul tinta, ce restrictii să fie impuse. Astfel, este posibil de exemplu ca unor utilizatori sa nu li se ceara deloc parola cand executa su, iar altora sa le fie interzisa complet preluarea identitatii altui user.*

4.1.4.3. Privilegii partiale: comanda sudo

Comanda su prezentata anterior are doua dezavantaje:

- pentru a executa comenzi ca root, utilizatorul neprivilegiat trebuie sa cunoasca parola de root
- odata aflat in posesia acestei parole, utilizatorul are privilegiile de root complete, chiar daca el are nevoie doar de un subset al acestora

Comanda **sudo** a fost creata pentru a oferi un mecanism de delegare de privilegii care sa nu sufere de aceste doua neajunsuri. Comanda se folosește de fișierul **/etc/sudoers**, în care administratorul sistemului poate configura cu exactitate cine ce are voie sa execute și în ce condiții, mergând pana la a specifica comanda exactă (inclusiv opțiuni și argumente) pe care un user are voie sa o execute folosind sudo.

Sintaxa comenzii **sudo** este următoarea:

```
sudo [ –u username ]  comanda
```

Efectul este executarea comenzii specificate cu privilegiile userului dorit, daca administratorul sistemului a permis acest lucru. Pentru a stabili ce utilizatori pot executa comenzi cu sudo si care sunt acestea, administratorul editeaza fisierul **/etc/sudoers**; fiecare linie din acest fisier are formatul urmator:

```
username statie=(user_tinta)comanda
```

Exemplu:

```
student         ALL=(root)ifconfig eth0
```

Aceasta linie ii va permite userului *student* sa execute ca root, pe orice statie, comanda *ifconfig eth0* (exact in aceasta forma! Daca userul incearca *ifconfig eth1* sau *ifconfig eth0 10.0.0.3*, nu i se va permite executia). Pentru a executa comanda, userul *student* va scrie:

```
student@server$ sudo ifconfig eth0
```

Nota: *la executarea acestei comenzi, userului curent i se va solicita parola **sa**! (nu cea de root)*

Constatam ca, folosind mecanismul sudo, putem stabili la modul exact cine ce poate executa, fara a divulga parola de root userilor neprivilegiati.

4.1.5. Monitorizarea utilizatorilor prezenti in sistem

Sarcina administratorului de sistem nu se opreste la a gestiona conturile de utilizator, ci presupune de asemenea monitorizarea logarilor, posibilitatea de a determina lista utilizatorilor logati in sistem la un moment dat (fie el prezent sau trecut) si supravegherea activitatii utilizatorilor.

În acest scop, sistemul de operare mentine doua baze de date cu informatii asupra activitatii utilizatorilor:

- */var/run/utmp* - conţine informatii legate de login-urile curente
- */var/log/wtmp* – contine istoria login-urilor in sistem.

Prezentam în continuare cateva comenzi ce ofera informaţii din aceste baze de date.

4.1.5.1. Comenzile who si w

Cu ajutorul comenzii **who** putem obtine lista de useri logati in sistem. Pentru fiecare user sunt afisate implicit numele, terminalul de login si ora login-ului, însă cu optiunile potrivite pot fi adaugate diferite alte detalii:

```
student@server:~$ who
student    :0           2015-09-28 12:00 (:0)
student    pts/3        2015-09-29 12:49 (:0)
```

Comanda **w** adauga informatii suplimentare fata de who – uptime, numar de useri logati, incarcarea sistemului in ultimele 1/5/15 minute, iar pentru fiecare user prezent in sistem aflam in plus ora logarii, timpul de procesor consumat de respectivul utilizator (proces curent sau cumulat pentru toate procesele pornite din terminalul sau) si linia de comanda a comenzii curente.

4.1.5.2. Comanda last

Comanda **last** afiseaza inregistrarile din *wtmp* in ordine cronologica inversa. Putem vizualiza inclusiv opririle sistemului (pseudo-userul *reboot*), iar pentru fiecare utilizator in parte aflam terminalul, data login-ului si intervalul in care a fost logat in sistem. Comanda poate primi ca parametru un nume de utilizator, pentru a afisa numai inregistrarile corespunzatoare acestuia:

```
last root
last reboot
```

Fisierul jurnal *wtmp* este rotit la atingerea unei anumite dimensiuni, si de aceea informatiile mai vechi pot fi gasite in fisierele /var/log/wtmp.1,/var/log/wtmp.2 etc. Vizualizarea lor se poate face specificand fisierul care trebuie folosit de catre comanda *last*:

```
last –f /var/log/wtmp.1
```

4.1.5.3. lastlog

Comanda afiseaza ultima inregistrare din *wtmp* pentru fiecare din userii declarati in sistem. Astfel putem repera login-urile ilegale (de exemplu, o logare in sistem a userului *daemon*, destinat doar rularii serviciilor).

4.2. Permisiuni

4.2.1. Sistemul de permisiuni la nivel de fisier

Fiecare fisier Linux dispune de permisiuni pentru 3 categorii de useri prestabilite:

- **owner** – este „proprietarul” fisierului, in general cel care l-a creat (desi ownerul poate fi modificat ulterior de catre root). Este cel care are dreptul de a schimba permisiunile pe fisier.
- **group** – grupul ce detine fisierul. Prin folosirea de grupuri se pot acorda drepturi asupra fisierului pentru mai multi utilizatori
- **other** – toti ceilalti useri ce nu sunt cuprinsi in primele doua categorii

Nota: *în Windows, fiecare fisier avea atașată o lista de useri, fiecare dintre ei având propriu-i set de permisiuni asupra fisierului. A acorda unui nou user privilegii se reducea la a-l adauga în lista cu permisiunile dorite. Prin contrast, în Linux nu putem seta permisiunile pe fisier pentru orice user dorim, ci doar pentru una dintre cele trei categorii de useri, ceea ce face ca unele scenarii simple în Windows sa devină dificil sau imposibil de implementat în Linux.*

Fiecare fisier dispune de doua seturi de permisiuni:

- permisiuni de baza: **read**, **write** și **execute**. Aceste permisiuni se definesc per categorie de useri - altfel spus, fiecare dintre cele 3 categorii de mai sus are propriu-i trio [read,write,execute]
- permisiuni speciale: **set UID on execution**, **set GID on execution** și **sticky**. Aceste permisiuni se definesc per fisier

Fiecare dintre permisiunile amintite are doua stari posibile - prezenta sau absenta.

4.2.2. Efectul permisiunilor asupra diferitelor tipuri de fisiere

In functie de tipul fisierului pe care sunt aplicate, permisiunile au efect diferit. Din acest punct de vedere distingem urmatoarele categorii de fisiere:

- *fisierele obisnuite, cele de tip dispozitiv, socket si named pipe* - aceste fisiere se comporta la fel din punct de vedere al efectului permisiunilor
- *directoarele* - data fiind natura speciala a unui fisier de tip director, unele permisiuni nu au sens pentru el, sau au o alta semnificatie decat in cazul unui fisier normal (vezi tabelul de mai jos)
- *symbolic link-urile* - un symlink are intotdeauna permisiuni complete pentru toate categoriile de useri, si ele nu pot fi modificate; altfel spus, nu poate fi interzis accesul la tinta unui symlink prin manipularea permisiunilor sale. Efectul este ca symlink-ul va fi intotdeauna transparent pentru utilizatorul sau

Nota: *in tabelul ce urmeaza, efectul permisiunilor de read si write asupra unui director se pot usor deduce daca ne amintim faptul ca un fisier de tip director este de fapt o lista de hard link-uri (asocieri nume-inode). Nu trebuie decat sa ii aplicam directorului efectul de la fisiere obisnuite.*

Permisiune	Efect asupra fisierelor obisnuite si speciale	Efect asupra directoarelor
Read	putem citi continutul fisierului	putem obtine lista numelor de fisiere din director (asadar am putea efectua listarea simpla (nu ls –l!) a directorului)
Write	putem modifica continutul fisierului (insa nu il putem neaparat si sterge sau redenumi!)	putem crea/sterge/redenumi fisiere inauntrul directorului
Execute	putem executa fisierul (cu conditia sa avem drept de read, pentru a-i putea citi continutul in memorie, deoarece executia aplicatiilor se realizeaza din RAM)	putem accesa fisierele din interiorul directorului
Set UID on execution	fisierul va fi executat cu UID-ul ownerului, nu al celui care il executa	*nu are efect pentru directoare*
Set GID on execution	Analog cu setUID, dar fisierul va rula cu GID-ul proprietar	fisierele nou create in director mostenesc grupul proprietar al directorului; in plus, subdirectoarele acestuia vor avea automat setGID setat
Sticky	*nu are efect pentru fisiere*	in directoarele cu acces public (permisiuni 777) un fisier nu mai poate fi sters-redenumit decat de catre ownerul său sau de catre ownerul directorului

Nota: *toate efectele din tabel sunt valabile doar in masura in care configuratia de permisiuni din sistemul de fisiere ne ofera acces la fisierul in cauza. Altfel spus, degeaba avem permisiuni de citire pe un fisier, daca nu putem ajunge pana la el!*

Se impun cateva observatii pe marginea informatiei din tabel:

- dreptul de stergere sau redenumire a unui fisier nu depinde in nici un fel de permisiunile pe acel fisier, ci de cele pe directorul parinte!
- pentru a putea executa cu succes ls –l (listare detaliata) pe un director, este nevoie atat de permisiune de read pe acesta (pentru a putea obtine lista de fisiere) cat si de execute (deoarece informatiile afisate de catre ls –l sunt preluate din inode-ul fiecarui fisier in parte)
- setUID poate fi folosit ca mecanism de acordare de privilegii de root partiale. Spre exemplu, in multe distributii executabilul *passwd* (cel folosit pentru schimbarea de parole) il are ca owner pe root si are setUID setat, astfel incat toti utilizatorii sa poată modifica fisierul /etc/passwd prin intermediul acestei comenzi (fisierul in cauza avand drept de scriere doar pentru root). *Atentie! Fisierele setuid reprezinta un risc de securitate, folosirea lor nefiind recomandata decat in cazuri de stricta necesitate*
- setGID pus pe directoare este folosit pentru a crea directoare partajate intre mai multi utilizatori din acelasi grup. De remarcat insa ca setGID nu se aplica retroactiv; daca setam setGID pe un director, fisierele si subdirectoarele existente nu vor fi modificate, grupul mostenindu-se numai la fisierele nou create!
- sticky (numit, de fapt, “restricted deletion flag”) este o modalitate de a crea directoare cu acces public in care care userii nu isi pot totusi sterge/redenumi fisiere unul altuia. Un exemplu tipic este */tmp*: directorul are permisiuni 777, deci orice user are drept de scriere pe director, si in consecinta ar avea posibilitatea de a sterge/redenumi orice fisier din cadrul directorului daca nu ar exista sticky.

4.2.3. Vizualizarea permisiunilor

Permisiunile unui fisier pot fi vizualizate folosind comenzile **ls -l** sau **stat**, dupa cum urmeaza:

- pentru fisiere obisnuite si speciale, putem aplica *ls -l* sau *stat* pe fisierul in cauza
- pentru directoare este necesar *ls -ld* sau *stat* (daca nu am folosi optiunea -d, am obtine o listare detaliata a fisierelor din interiorul directorului, nu a directorului insusi!)

Principalele diferente intre comenzile *ls -l* si *stat* sunt urmatoarele:

- *ls -l* arata permisiunile numai in modul simbolic, pe cand *stat* le prezinta si in mod octal (vezi mai jos sectiunea dedicata schimbarii permisiunilor)
- *stat* arata informatie suplimentara comparativ cu *ls -l* si formatata pe mai multe linii, pe cand *ls -l* produce implicit o singura linie per fisier.
- *stat* nu are nevoie de optiuni suplimentare pentru a lista caracteristicile unui director, pe cand *ls -l* necesita optiunea *-d*

```
student@server:~$ ls -l f1
-rwsr-Sr-x 1 user1 grup1 0 oct  1 11:31 f1

student@server:~$ stat f1
  File: ‘f1’
  Size: 0               Blocks: 0          IO Block: 4096   regular empty file
Device: 811h/2065d      Inode: 2513069     Links: 1
Access: (6745/-rwsr-Sr-x)  Uid: ( 1000/   user1)   Gid: ( 1000/   grup1)
Access: 2015-10-01 11:31:58.436894059 +0300
Modify: 2015-10-01 11:31:58.436894059 +0300
Change: 2015-10-01 11:32:17.392893266 +0300
 Birth: -
```

Nota: *caracterul - de dinaintea lui rwsr-Sr-x reprezinta tipul fisierului si nu face parte din setul de permisiuni!*

Prezentam in continuare modul de interpretare a permisiunilor exprimate simbolic; varianta in octal va fi abordata in sectiunea dedicata schimbarii permisiunilor. Dupa cum se observa in exemplul de mai sus, atat ls -cat si stat au acelasi mod de exprimare a permisiunilor simbolice.

Simbolurile folosite sunt urmatoarele:

- caracterul - (minus) indica lipsa permisiunii de pe o anumita pozitie
- literele r, w si x corespund permisiunilor de read, write si execute
- literele s sau S corespund fie permisiunii setUID fie setGID (explicatia mai jos)
- literele t sau T corespund permisiunii sticky

Setul de permisiuni este prezentat sub forma unei insiruiri de 9 caractere, structurata dupa cum urmeaza:

- primele 3 caractere reprezinta permisiunile pentru owner; primul caracter corespunde permisiunii read, al doilea lui write si al treilea lui execute
- urmatoarele 3 caractere reprezinta permisiunile pentru grup; caracterele corespund la fel ca mai sus celor 3 permisiuni
- ultimele 3 caractere sunt permisiunile pentru others, organizate in acelasi fel ca mai sus
- atunci cand exista permisiuni speciale, acestea sunt figurate tot in cadrul celor 9 caractere, dupa cum urmeaza:
  - setUID este evidentiat sub forma unui **s** plasat pe pozitia lui execute in setul de permisiuni al ownerului (caracterul al treilea). Daca dreptul de executie este prezent, litera **s** va fi mica; daca

execute lipseste, **S** va fi mare tocmai pentru a atrage atentia asupra acestei nepotriviri (set UID on execution, dar nu avem drept de executie?!?)
- setGID este evidentiat cu aceleasi conventii, dar pe pozitia lui execute din setul de permisiuni al grupului (caracterul al saselea din setul complet de permisiuni)
- sticky este evidentiat sub forma unui t sau T pe pozitia lui execute de la others

In lumina informatiilor prezentate, sa analizam efectul setului de permisiuni din exemplul de mai sus, **rwsr-Sr-x** :

- ownerul (user1) are drepturi depline asupra fisierului; litera **s** este mica, ceea ce inseamna ca este prezent si dreptul de executie
- grupul proprietar are drept de citire pe fisier; litera **S** este mare si deci permisiunea de executie lipseste
- ceilalti useri pot citi si executa fisierul
- orice user ar executa fisierul (dintre cei care au acest drept) se va porni un proces ce ruleaza cu UID-ul lui user1 si GID-ul lui grup1, datorita prezentei permisiunilor speciale setUID/setGID

4.2.4. Felul in care sistemul de operare determina permisiunile unui user pe un fisier

Atunci cand suntem logati cu un anumit user si incercam sa efectuam o operatie cu un fisier, sistemul de operare foloseste urmatoarea logica:

- daca userul cu care suntem logati este ownerul fisierului, ni se vor aplica permisiunile pentru owner si celelalte doua categorii de permisiuni nu vor mai conta
- daca userul nostru face parte din grupul proprietar al fisierului (fie pe post de grup primar, fie secundar) atunci ni se vor aplica permisiunile corespunzatoare grupului si celelalte doua categorii de permisiuni nu vor conta
- in caz ca nu suntem nici owner, nici nu facem parte din grupul proprietar, ni se aplica permisiunile pentru others

Facem urmatoarele observatii importante:

- sistemul de operare nu cumuleaza permisiunile de la mai multe categorii de useri - spre exemplu, daca suntem owner pe fisier, ni se vor aplica strict permisiunile de owner, chiar daca ele sunt mai limitate decat cele de grup sau de others!
- Permisiunile, de unele singure, nu ne spun nimic; conteaza cui i se aplica! Asadar, a cunoaste permisiunile unui user pe un fisier presupune a cunoaste trio-ul [owner, grup, permisiuni]

4.2.5. Comenzi pentru administrarea permisiunilor

4.2.5.1. Schimbarea proprietarului si a grupului

Comenzi introduse: **chown, chgrp**

Aceste comenzi permit modificarea proprietatii asupra fisierelor (schimbarea de owner sau/si grup). Comanda *chown* schimba ownerul sau ownerul+grupul fisierului primit ca argument si poate fi folosita numai de catre root; comanda *chgrp* schimba grupul proprietar al fisierului si poate fi utilizata si de catre ownerul acestuia, cu conditia ca grupul pe care il seteaza sa fie unul dintre cele din care face parte ownerul. Sintaxa comenzilor este:

```
chown [optiuni] [owner_nou[:grup_nou]] [cale_fisier]
chgrp [optiuni] [grup_nou] [cale_fisier]
```

Dispunem de urmatoarele optiuni:

-h specifica comportarea comenzii la intalnirea unui symlink; fara optiuni, chown actioneaza asupra tintei; cu optiunea *–h*, comanda va actiona asupra fisierului symlink insusi.

-R schimbarea de proprietate este aplicata recursiv tuturor fisierelor de sub calea specificata

Exemple:

chown student /tmp/poza1.jpg	schimbarea proprietarului fisierului poza1.jpg din directorul /tmp
chown student:users numere.txt	schimbarea simultana a ownerului+grupului pe fisierul numere.txt
chown –R student /poze	schimbarea ownerului pe directorul /poze si toate fisierele continute, in toata adancimea sa
chgrp -R wwwgroup /var/www	stabilirea lui wwwgroup ca grup proprietar pe /var/www si toate fisierele continute, recursiv

Nota: *ownerul unui fisier poate sa nu faca parte din grupul proprietar! Aceasta separare este chiar necesara atunci cand dorim sa acordam privilegii diferite pentru mai multi useri pe acelasi fisier.*

4.2.5.2. Schimbarea permisiunilor pe fisiere

4.2.5.2.1 Comanda folosita și moduri de specificare a permisiunilor

Comanda **chmod** este cea utilizata pentru modificarea drepturilor asupra fisierelor. Dispunem de doua moduri de specificare a permisiunilor:

- ***simbolic*** - numit astfel deoarece se folosesc simboluri pentru specificarea permisiunilor. Este un mod de lucru mai intuitiv dar uneori nu foarte eficient, și care permite operatii relative (adaugare/eliminare de permisiuni)
- ***octal*** - permisiunile sunt specificate folosind o forma numerica, mai compacta. Nu este la fel de intuitiv dar permite stabilirea mai rapidă a unui anumit set de permisiuni pe un fisier; în schimb, nu accepta operatii relative (adaugare/stergere permisiuni) ci doar suprascriere

Indiferent de modul de lucru folosit, sintaxa comenzii *chmod* este următoarea:

```
chmod [opţiuni] permisiuni fisier(e)
```

Formatul permisiunilor depinde de modul folosit (simbolic/octal).

Dintre optiuni amintim doar **-R** (recursiv), utila atunci cand dorim sa aplicam un set de permisiuni pe toate fisierele dintr-un director, in toata adancimea sa.

4.2.5.2.2 Modul simbolic

Modul simbolic presupune specificarea/ajustarea permisiunilor pe un fisier prin intermediul unei combinatii de simboluri. Permisiunile pasate ca prim parametru comenzii *chmod* contin trei informații:

- *ale cui drepturi se modifica*: **u** pentru user(owner), **g** pentru group, **o** pentru other, **a** pentru all (toate cele 3 categorii de useri simultan). ***Atenţie!*** *Litera* ***o*** *nu desemneaza ownerul, ci categoria* ***others****!*
- *operația dorita*: + pentru adaugare (relativ la setul de permisiuni existent), - pentru eliminare, = pentru setare (suprascrierea setului de permisiuni anterior)
- *despre ce drepturi este vorba*: **r** pentru read, **w** pentru write, **x** pentru execute, **s** pentru setuid/setgid, **t** pentru sticky
-

Iata mai jos un exemplu de fisier nou-creat trecut printr-o serie de modificari de permisiuni:

Comanda	Efect scontat/explicatii	Permisiuni rezultante
touch f1	Creare fisier cu permisiuni implicite (vezi mai jos sectiunea despre controlul acestor permisiuni)	rw-rw-r--
chmod g-w f1	Eliminare permisiune de scriere pentru grup	rw-r**-**-r--
chmod ug+x f1	Adaugare permisiune de executie pentru doua categorii diferite de useri, simultan	rw**x**r-**x**r--
chmod o+wx f1	Adaugare doua permisiuni diferite, simultan	rwxr-r**wx**
chmod g+w,o-x f1	Efectuarea a doua operatii diferite, simultan	rwxr**w**xrw**-**
chmod ug=rwx,o= f1	Suprascriere permisiuni; pt others se anuleaza toate permisiunile	**rwxrwx---**
chmod ug+s f1	Adaugare setUID si setGID	rw**s**rw**s**---
chmod u-x f1	Eliminare permisiune setUID; litera S devine mare pt owner	rw**S**rws---
mkdir d1 chmod a=rwx,o+t d1	Creare director cu acces public si sticky	**rwxrwxrwt**

Nota: *chmod suporta și permisiunea +X, care nu este o permisiune în sine ci mai degrabă un artificiu tehnic pentru acordarea selectiva a dreptului de execuție în toată adâncimea unui director. Aceasta optiune este gandita să fie folosită în conjunctie cu -R: chmod -R +X aplicat pe un director da drept de executie tuturor subdirectoarelor, pastreaza dreptul de executie pentru fisierele care îl au deja, și nu le afecteaza pe celelalte.*

4.2.5.2.3 Modul octal

În modul octal (denumit și „absolut”), specificarea permisiunilor se face numeric, folosind valorile pentru permisiuni așa cum sunt ele stocate în inode-ul fisierului. În acest mod de lucru permisiunile vechi se pierd - operatia nu mai este una relativa, ci suprascrie permisiunile curente.

Fiecare dintre permisiunile unui fisier poate fi prezenta sau absenta, fiind asadar memorata sub forma unui bit. In consecinta, permisiunile sunt stocate in inode sub forma unui ansamblu de 12 biti: 3 categorii de useri x 3 permisiuni de baza = 9 biti, plus inca 3 biti pentru cele speciale. Fiecare dintre aceste grupuri de 3 biti formeaza o cifra în octal (baza de numeratie opt), deoarece pe 3 biti se pot reprezenta $2^3 = 8$ valori. Ordinea grupurilor de câte 3 biti în inode este următoarea: permisiuni speciale, permisiuni pentru owner, permisiuni pentru grup, permisiuni pentru others.

Fie un fisier care are permisiunile următoare: *rwsr-Sr-x*. Permisiunile sale în octal se calculează astfel:

```
SUID  SGID  sticky     r  w  x          r  —  —          r  —  x
 1     1      0       1  1  1          1  0  0          1  0  1
 2²    2¹     2⁰      2² 2¹ 2⁰         2² 2¹ 2⁰         2² 2¹ 2⁰
 4+    2+     0       4+ 2+ 1          4+ 0+ 0          4+ 0+ 1
    6             7             4             5
```

Setarea acestor permisiuni pe un fisier se realizeaza astfel:

```
chmod –R 6745 /home/infoacad/file2
```

Explicatia de mai sus a fost oferita pe larg pentru o mai buna intelegere, dar in practica nu este nevoie sa efectuam de fiecare data aceste calcule aparent complicate. Permisiunea de read are intotdeauna valoarea 4, write are 2 iar execute are 1; in acelasi fel, setUID are4, setGID 2 si sticky 1. Asadar totul se reduce la simple sume de 1, 2 si 4.

Deseori modul octal este cea mai rapida/compacta modalitate de a stabili un anumit set de permisiuni pe un fisier sau un ansamblu de fisiere. Prin comparatie, pentru a seta permisiunile de mai sus folosind modul simbolic ar fi trebuit sa scriem: *chmod u=rwxs,g=rs,o=rx /home/infoacad/file2*.

4.2.6. Controlul permisiunilor pentru fisierele nou-create

Daca nu ar exista niciun mecanism de limitare, permisiunile implicite la crearea de fisiere si directoare ar fi urmatoarele:

- pentru directoare - 777 (rwxrwxrwx). Implicit, toti utilizatorii au dreptul de a lista si accesa fisierele continute si de a crea/sterge/redenumi fisiere in director
- pentru fisiere obisnuite si speciale - 666 (rw-rw-rw-). Ca urmare, toti utilizatorii au dreptul de a citi si modifica continutul fisierului. Observam insa ca lipseste permisiunea de executie, ceea ce are sens avand in vedere ca un fisier proaspat creat este un fisier text, gol - nimic nu il recomanda ca fiind un viitor executabil.

Este evident faptul ca permisiunile implicite sunt mult prea largi - daca un user creaza un fisier, orice alt user va avea dreptul de a-i modifica continutul. Acesta este si motivul pentru care a fost gandit un mecanism de limitare de permisiuni, prin care fiecare utilizator sa poata specifica ce permisiuni doreste sa fie *eliminate* din cele implicite, maximale. Mecanismul in cauza poarta denumirea de *user mask* (prescurtat umask).

Umask-ul este un filtru - o masca de biti suprapusa automat peste permisiunile implicite la fiecare creare de nou fisier/director si care poate elimina astfel o parte dintre permisiuni. Umask-ul are 12 biti, la fel ca permisiunile fisierului, insa functioneaza in oglinda: **contine 1 pe pozitiile ce se doresc eliminate** si 0 pe celelalte.

Exemplu: dorim ca pentru fisierele nou create sa nu aiba drept de scriere decat proprietarul, grupul sa poata doar citi aceste fisiere, iar ceilalti sa nu aiba deloc acces. Aceasta inseamna ca trebuie eliminat dreptul de scriere pentru grup si dreptul de citire si scriere pentru other:

```
rw- rw- rw-
000 010 110
 0   2   6
```

Aceasta masca este aplicata cu ajutorul comenzii **umask**, care accepta ambele forme (octal si simbolic):

- octal: umask 026
- simbolic: umask g-w,o=

Atunci cand se doreste ajustarea permisiunilor speciale poate fi folosita si cea de-a patra cifra din stanga, in acelasi fel ca la permisiuni (ex: umask-ul 2033 elimina setGID si, in plus, permisiunea de executie pentru group si others).

Nota: *in cazul fisierelor obisnuite si speciale, desi nu are sens sau efect, putem aplica si un umask care elimina permisiunea de executie (ex: 0077, care va avea acelasi efect ca 0066), fara a genera vreo eroare.*

Comanda **umask** data fara argumente afiseaza masca curenta:

```
# in mod octal
student@server$ umask
0026
```

```
# in mod simbolic
student@server$ umask -S
u=rwx,g=rx,o=x
```

Atentie la urmatoarele aspecte legate de efectul comenzii umask:

- schimbarile de umask vor afecta numai permisiunile fisierelelor create din momentul schimbarii incolo! Fisierele existente isi pastreaza permisiunile
- efectul comenzii se manifesta doar in shell-ul in care a fost executata! Daca stabilim o anumita masca in prima consola virtuala, ea nu va fi preluata automat si in celelalte sau in emulatoarele de terminal pornite in mediul grafic. Mai mult, daca ne delogam din prima consola virtuala, masca setata se va pierde, la urmatorul login revenindu-se la setarile implicite. Daca se doreste permanentizarea unui anumit umask este necesara editarea fisierelor de initializare ale shell-ului (vezi capitolul dedicat shell-urilor si scripting-ului).