

Laboratorul 3

Utilizatori, fişiere, procese

1 Gestiunea conturilor utilizator

În mediul Unix, pentru a crea, modifica şi şterge utilizatori din sistem se folosesc comenzile `useradd(8)`, `usermod(8)` şi, respectiv, `userdel(8)`. Pentru a crea şi modifica parola de acces la sistem se foloseşte comanda `passwd(1)`.

```
# useradd -m -g users alex
# passwd alex
# usermod -G wheel alex
# userdel -r alex
```

În exemplul de mai sus, prima comandă creează utilizatorul `alex`, înscris în grupul `users` (`-g`), împreună cu directorul `/home/alex` (`-m`). A doua adaugă o parolă cu care noul utilizator poate intra în sistem. A treia comandă modifică datele utilizatorului `alex`, adăugându-l într-un grup suplimentar `wheel`. La final, `alex` este şters din sistem împreună cu directorul său `$HOME` şi tot ce conţinea acesta (`-r`).

Un mod mai prietenos de a adăuga un utilizator este cu ajutorul comenzii `adduser(8)`.

Pentru a crea şi şterge grupuri folosiţi comenzile `groupadd(8)` şi, respectiv, `groupdel(8)`.

```
# groupadd mygroup
# groupdel mygroup
```

În general, fişierele şi directoarele noi sunt deţinute de utilizatorul care le-a creat şi primul grup din care face parte. În exemplul de mai sus, dacă `alex` ar crea un fişier nou, acesta ar aparţine lui `alex` şi grupului `users`. Pentru a modifica proprietarul se foloseşte comanda `chown(8)`. Fie următoarele fişiere create de `root`:

```
# touch foo bar baz
# ls -l foo bar baz

-rw-r--r-- 1 root wheel 0 Mar 21 23:04 bar
-rw-r--r-- 1 root wheel 0 Mar 21 23:04 baz
-rw-r--r-- 1 root wheel 0 Mar 21 23:04 foo
```

Pe coloana a 3-a şi a 4-a vedem utilizatorul şi grupul care deţin aceste fişiere nou create. Comanda `chown(8)` aşteaptă noii proprietari în format `user:group`, urmat de fişierele pe care să le modifice:

```
# chown alex:users foo
# chown alex bar
# chown :users baz
```

Comenzile de mai sus modifică utilizatorul şi grupul, doar utilizatorul şi, respectiv, doar grupul celor trei fişiere create mai sus.

```
# ls -l foo bar baz

-rw-r--r-- 1 alex wheel 0 Mar 21 23:04 bar
-rw-r--r-- 1 root users 0 Mar 21 23:04 baz
-rw-r--r-- 1 alex users 0 Mar 21 23:04 foo
```

În exemplul de mai sus vedem pe prima coloană drepturile de acces asociate utilizatorului, grupului şi celorlalţi din sistem (vezi Cursul 4). Pentru a modifica aceste drepturi se foloseşte comanda `chmod(8)`. Reamintim notaţiile:

- acces: citire (`r`), scriere (`w`), executare (`x`);

- categorii: utilizator (u), grup (g), restul (o); pentru toate categoriile se folosește a (de la *all*).

Aceste simboluri pot fi folosite în orice combinație pentru a specifica adăugarea sau eliminarea de drepturi asupra fișierelor sau directoarelor. Formatul este *categorii:op:acces*, unde *op* este =, + sau -, pentru a seta, adăuga sau, respectiv, elimina permisiuni.

```
# chmod u-w bar
# chmod g+w baz
# chmod a+wx foo
# ls -l foo bar baz
-r--r--r-- 1 alex wheel 0 Mar 21 23:04 bar
-rw-rw-r-- 1 root users 0 Mar 21 23:04 baz
-rwxrwxrwx 1 alex users 0 Mar 21 23:04 foo
```

Într-un terminal se poate trece de la un utilizator la altul cu comanda `su(8)`. Pentru a executa o singură comandă drept alt utilizator se folosește `sudo(8)`.

Pentru `su(8)` este nevoie să cunoaștem parola utilizatorului în contul căruia vrem să intrăm.

Pentru `sudo(8)` e nevoie doar de propria parolă, care ne va da acces la comenzile specificate în fișierul `/etc/sudoers`. Acest fișier respectă un format fix ce permite comenzi și configurații complexe.

Menționăm aici cazul cel mai frecvent în care vrem să adăugăm permisiuni pentru un utilizator sau grup. Forma cea mai simplă este

```
user host = cmd1, cmd2
%group host = cmd1, cmd2
```

unde comenzile sunt separate prin virgulă. Pentru orice valoare se poate folosi cuvântul cheie `ALL` ce permite acces oricui (fie utilizator, host sau comandă). Grupurile trebuie prefixate cu %.

```
root ALL=(ALL) SETENV: ALL
%users ALL=/sbin/mount /cdrom, /sbin/umount /cdrom
%users localhost=/sbin/shutdown -h now
```

În exemplul de sus, utilizatorul `root` poate executa orice comandă de pe orice mașină (*host*) și, în plus, i se păstrează variabilele globale (ex. `$PATH`) cu ajutorul specificatorului `SETENV: ALL`.

A doua linie permite tuturor din grupul `users` să monteze și demonteze în directorul `/cdrom`. Atenție: asta înseamnă că este permisă doar comanda cu argumentele exact cum sunt trecute. Comanda `/sbin/mount /stick` nu ar fi permisă.

Ultima linie permite celor din grupul `users` să închidă calculatorul, dar doar dacă sunt autentificați de pe mașina curentă (`localhost`). Se poate și de la distanță (ex. prin `ssh`), acest subiect fiind tratat în laboratoarele viitoare.

2 Comenzi de monitorizare a utilizatorilor

Odată ce un utilizator s-a logat în sistem, e important să existe comenzi de monitorizare a activităților sale, a timpului petrecut în sistem, a locului de unde s-a logat (un terminal local sau o adresă a unei mașini de la distanță), a timpului utilizării active a CPU, respectiv a timpului `IDLE` (fără activitate) șamd. În acest sens, există o serie de comenzi care pot fi folosite, dintre cele mai cunoscute fiind `w`, `who` și `last/lastb`. Sesiunea de mai jos evidențiază unele dintre modalitățile cele mai comune de folosire a acestor comenzi. Pentru o mai bună înțelegere a lor, folosiți comanda `man` pentru a consulta paginile de manual corespunzătoare.

```
$ w
21:10:29 up 3:39, 3 users, load average: 0.08, 0.14, 0.09

USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT
guest tty7 :0 17:31 3:39m 2:16 0.20s /sbin/upstart -
guest pts/5 :0 18:10 4:48 0.09s 0.09s bash
guest pts/14 :0 18:32 0.00s 0.10s 0.00s w

$ who -a
system boot 2024-10-13 17:31
guest + tty7 2024-10-13 17:31 03:39 1420 (:0)
guest + pts/5 2024-10-13 18:10 00:04 6481 (:0)
guest + pts/14 2024-10-13 18:32 . 7806 (:0)
pts/15 2024-10-13 18:37 8276 id=s/15 term=0 exit=0
pts/21 2024-10-13 18:37 8296 id=s/21 term=0 exit=0

$ last -5
guest pts/21 :0 Sun Oct 13 18:36 - 18:37 (00:00)
guest pts/15 :0 Sun Oct 13 18:36 - 18:37 (00:00)
guest pts/14 :0 Sun Oct 13 18:32 gone - no logout
guest pts/14 :0 Sun Oct 13 18:11 - 18:20 (00:09)
guest pts/5 :0 Sun Oct 13 18:10 gone - no logout
wtmp begins Tue Oct 1 10:02:00 2024

$ last -s 17:35 -t 18:20
guest pts/14 :0 Sun Oct 13 18:11 still logged in
guest pts/5 :0 Sun Oct 13 18:10 gone - no logout
guest pts/14 :0 Sun Oct 13 17:50 - 17:59 (00:08)
guest pts/5 :0 Sun Oct 13 17:49 - 18:10 (00:20)
wtmp begins Tue Oct 1 10:02:00 2024
```

3 Procese

Procesele în cadrul sistemului de operare au un ID unic numit *process ID* (pid). Pentru a afișa procesele existente la un moment dat se folosește comanda `ps(1)`. Asemănătoare cu *task manager*-ul din Windows este comanda `top(1)`, care afișează în timp real schimbările aferente proceselor din sistem.

Implicit, comanda `ps(1)` afișează procesele utilizatorului curent:

```
$ ps
PID TT STAT TIME COMMAND
5714 p1 I s+p 0:00.02 bc -l
86860 p1 I+p 0:00.00 bc -l
24963 p2 Isp 0:00.02 -ksh (ksh)
74549 p2 Ip 0:00.02 man userdel
932 p2 I+p 0:00.01 more -s -T /tmp/man.f231wq9zSv
83124 p3 I s+p 0:00.01 tail -f /var/log/messages
61541 p4 Isp 0:00.02 -ksh (ksh)
24637 p4 S+ 0:25.60 vim --servername VIM
50096 p4 S+p 0:03.56 mupdf /home/paul/wrk/ub
65276 p5 Ssp 0:00.04 -ksh (ksh)
```

Pentru a specifica ce detalii să afișeze, folosiți argumentul `-o`. Lista completă de opțiuni o găsiți în manual. De exemplu, următoarea comandă afișează `pid`-ul, grupul și comanda cu care a fost lansat procesul pentru utilizatorul curent.

```
$ ps -o pid,group,command
```

PID	GROUP	COMMAND
5714	paul	bc -l
86860	paul	bc -l
24963	paul	-ksh (ksh)
74549	paul	man userdel
932	paul	more -s -T /tmp/man.f231wq9zSv
83124	paul	tail -f /var/log/messages
61541	paul	-ksh (ksh)
24637	paul	vim --servername VIM
50096	paul	mupdf /home/paul/wrk/ub
65276	paul	-ksh (ksh)

Pentru a vedea procesele tuturor utilizatorilor puteți folosi argumentele -A sau -a.

```
$ ps -a -o pid,user,command
```

Mai sus am înlocuit informația despre grup cu cea despre utilizator.

O altă comandă utilă este `ps tree`, care afișează ierarhia (arborele) de procese creat de un anumit proces (specificat prin PID) sau de un anumit utilizator (specificat prin nume) furnizată ca parametru al comenzii. În absența oricărui parametru se afișează ierarhia tuturor proceselor din sistem cu rădăcina în `/sbin/init`.

```
$ ps
PID    TTY      TIME CMD
19310  pts/14   00:00:00 bash
20274  pts/14   00:00:00 ps
```

```
$ ps tree 19310
bash---ps tree
```

```
$ sleep 60 &
[1] 20292
$ sleep 70 &
[2] 20293
```

```
$ ps tree 19310
bash---ps tree
|
+-2*[sleep]
```

Comanda `kill(1)` trimite un semnal unui proces identificat prin `pid`. Implicit acesta este semnalul `ce-i cere procesului să-și termine execuția`, dar pot fi trimise și alte semnale, precum cel de a reporni (util, de exemplu, când am schimbat un fișier de configurație și vrem ca modificările să intre în acțiune).

```
$ kill 5714
```

De multe ori nu cunoaștem `pid`-ul procesului dorit și trebuie să apelăm la `ps(1)` sau `top(1)` pentru a-l afla. Când sistemul execută multe procese acest lucru poate fi anevoios. O comandă mult mai utilă este `pgrep(1)`, care se comportă ca utilitarul `grep(1)` dar caută între procese.

```
$ pgrep bc
86860
5714
```

```
$ pgrep -lf bc
86860 bc -l
5714 bc -l
51775 xterm -e bc -l
```

Observați că, în cazul acesta, `pgrep(1)` a identificat tiparul `bc` într-un proces în plus, numit `xterm`, care a primit `bc` în lista de argumente.

Similar comenzii `pgrep(1)`, există comanda `pkill(1)`, care funcționează la fel, doar că la sfârșit trimite un semnal listei de procese găsită. Ca și în cazul `kill(1)`, semnalul implicit este cel de oprire a procesului.

4 Job control

Comenzile din `shell` care se execută cu un *ampersand* `&` la sfârșit instruiesc `shell`-ul să nu aștepte terminarea comenzii respective. Comanda este lansată în execuție, dar `shell`-ul întoarce imediat promptul și așteaptă noi comenzi de la utilizator. Se spune că respectiva comandă a fost lansată în `background`.

În exemplul de mai jos se folosește comanda `sleep` pentru a simula două comenzi de durată variabilă care sunt lansate în `background`. Cu ajutorul comenzii interne `jobs` se pot lista comenzile care se află în grupul de procese din `background`. Comanda `kill` se poate folosi pentru a termina aceste procese folosind o sintaxă specială, ca mai jos, în vreme ce comenzile interne `fg` și, respectiv, `bg` se pot folosi pentru a muta procese din `background` în `foreground` și, respectiv, procese suspendate (de pildă cu semnalul `SIGTSTP`, corespunzător combinației de taste `Ctrl-z`) din `foreground` în `background`.

```
$ sleep 60 &
[1] 19267

$ jobs
[1]+  Running      sleep 60 &

$ sleep 45 &
[2] 19268

$ jobs
[1]-  Running      sleep 60 &
[2]+  Running      sleep 45 &

$ kill %2

$ jobs
[1]-  Running      sleep 60 &
[2]+  Terminated sleep 45

$ fg %1
sleep 60
^Z
[1]+  Stopped      sleep 60

$ jobs
[1]+  Stopped      sleep 60

$ bg %1
[1]+  sleep 60 &

$ jobs
[1]+  Running      sleep 60 &

$ fg %1
sleep 60
```

^C

5 Legătura dintre procese și fișiere

Există situații în care e util să aflați care sunt fișierele folosite de către un anumit proces. De pildă, o asemenea situație apare când un program lansat în execuție folosește fișiere de pe un anumit sistem de fișiere care se dorește a fi deinstalat din sistem (i.e. supus operației de `umount`, operația inversă celei de `mount`). Operația de `umount` va eșua atâta vreme cât există încă procese care folosesc fișiere și directoare din acel sistem de fișiere. Aceste procese trebuie terminate pentru a elibera resursele pe care le folosesc, dar adesea e greu de aflat care sunt aceste procese (mai exact, PID-ul lor).

În acest scop se pot folosi comenzile `fuser` sau `lsuf`. Ele primesc ca argument un nume de fișier și returnează informații despre procesele care îl folosesc la momentul respectiv. În cele ce urmează puteți vedea și reproduce o sesiune de lucru care evidențiază modul de lucru cu cele două comenzi.

```
$ mkdir tmpdir
$ cd tmpdir/

$ fuser .
/tmp/tmpdir: 6586c
```

```
$ ps
PID   TTY      TIME    CMD
6586  pts/14   00:00:00 bash
6600  pts/14   00:00:00 ps
```

```
$ lsuf .
COMMAND PID  USER  FD  TYPE  DEVICE  SIZE/OFF  NODE    NAME
bash    6586 guest  cwd DIR   8,6    4096     147523  .
lsuf    6601 guest  cwd DIR   8,6    4096     147523  .
lsuf    6602 guest  cwd DIR   8,6    4096     147523  .
```

```
$ fuser -k .
```

În secvența de comenzi de mai sus se creează un subdirector al `/tmp` care devine utilizat de către `shell`-ul curent atunci când se schimbă directorul. Acest lucru este evidențiat de comanda `fuser`, care specifică PID-ul procesului care folosește subdirectorul și îl marchează cu caracterul `c`, care desemnează faptul că procesul (`shell`-ul) folosește directorul curent (pentru detalii, `man fuser`).

Comanda `ps` subsecvența certifică faptul că, într-adevăr, `shell`-ul curent are PID-ul afișat de comanda `fuser`. Cu comanda `fuser -k` se termină toate procesele care folosesc fișierul specificat ca parametru (în cazul nostru, directorul curent). Efectul net, în cazul nostru, va fi dispariția `shell`-ului. Suplimentar, secvența de comenzi de mai sus evidențiază și folosirea comenzii `lsuf` în același scop de a identifica procesul care folosește directorul curent.

Dacă folosiți comanda de mai jos, ce se întâmplă? Cum vă explicați?

```
$ cd ; fuser .
```

6 Sarcini de laborator

1. Creați utilizatorii `admin`, `prof`, `stud131`, `stud132`, `stud133`, `stud134` și grupurile: `seria13`, `gr131`, `gr132`, `gr133`, `gr134`.

2. Adăugați utilizatorul `admin` în grupul `wheel` și modificați `/etc/sudoers` cu comanda `visudo(8)` pentru a-i permite executarea oricărui program din sistem.
3. Creați următoarea arborescență de directoare și fișiere:

```
seria13
|-- 131
|   |-- discutii.txt
|   `-- laborator.txt
|-- 132
|   |-- discutii.txt
|   `-- laborator.txt
|-- 133
|   |-- discutii.txt
|   `-- laborator.txt
|-- 134
|   |-- discutii.txt
|   `-- laborator.txt
|-- catalog
|   |-- note131.txt
|   |-- note132.txt
|   |-- note133.txt
|   `-- note134.txt
`-- subiecte
    |-- examen.txt
    `-- restanta.txt
```

Unde:

- doar profesorul scrie în directorul `catalog` și doar cei din `seria13` pot accesa și citi;
 - fiecare grupă are acces de scriere și citire la propriul director;
 - toți studenții pot citi conținutul din fiecare director de grupă;
 - doar profesorul poate citi, scrie și accesa directorul `subiecte`.
4. Porniți trei procese terminal. Folosiți `kill(1)` pentru a opri aceste procese și doar pe acestea. Indiciu: folosiți `pgrep(1)` întâi pentru a vă asigura că lista de procese ce urmează a fi oprită este cea corectă.
 5. Afișați cu `ps(1)` toate procesele din sistem cu următoarele informații:
 - proprietar: utilizator și grup;
 - identificare: proces și părintele procesului;
 - altele: spațiu ocupat în memorie și comanda executată.