Laboratorul 4

1. Comenzi de lucru cu directoare

Creearea directoarelor se face cu comanda `mkdir` ("make directory"). **Sintaxa** ei este**:**

```
mkdir [-p] nume_catalog
```

Dacă este prezentă opţiunea `-p` se creează şi cataloagele părinte care lipsesc din numele de cale dat pentru catalogul dorit.

Exemplu.
`$mkdir –p /home/ilflorea/so/Linux/exemple`
Înainte de crearea catalogului `exemple`, vor fi create cataloagele `ilflorea`, `so`, `Linux` dacă nu au fost încă create, în ordinea specificată.

Tipărirea directorului curent se face cu comanda `pwd` ("print working directory"). **Sintaxa** ei este**:** `$pwd`
Se utilizează pentru a se verifica poziţia momentană, într-o sesiune de lucru sub Linux, în ierarhia de cataloage.

Exemplu. Dacă numele de utilizator este `ilflorea` şi după loginare se tastează
`$pwd`
se va afişa:
`/home/ilflorea`

Comanda cd ("change directory") are sintaxa**:** `cd [nume_catalog]`
Are ca efect schimbarea **directorului curent** sau modificarea **poziţiei curente** în ierarhia de directoare. Directorul dat ca parametru devine noul director curent. Pentru ca această comandă să poată fi executată, utilizatorul trebuie să aibă drept de *căutare* (redat prin `x` în comanda `ls`) în catalogul dat ca argument. Utilizarea comenzii `cd` fără parametru determină revenirea în catalogul gazdă al utilizatorului.

Exemple.
`$cd /home/ilflorea/so/Linux/exemple`
directorul curent va fi `exemple`
`$cd ..` #directorul curent va fi `Linux`
`$cd /home/ilflorea` #directorul curent va fi `ilflorea`
Acelaşi efect se obţine, dacă se tastează :
`$cd ~`
sau
`$cd`

Ştergerea unui catalog vid se face cu comanda `rmdir` ("remove directory") care are **sintaxa**:

```
rmdir [opţiuni] nume_catalog
```

Cu opţiunea `–p` se şterg şi eventuale directoare părinte care au devenit vide.

Pentru ştergerea mai multor fişiere sau directoare se foloseşte comanda `rm`, care are sintaxa:

```
rm [optiuni] fisier(e)
```

Opţiunea `–i:` utilizatorul este întrebat înainte de ştergerea efectivă a intrării.
Opţiunea `–r` (sau `-R`): realizează ştergerea recursivă în catalogul la care se referă comanda. Nu este permisă ştergerea intrării "..", adică a directorului părinte al fişierului.

Opţiunea `-f` şterge fişiere fără mesaj de avertizare.
Opţiunea `-v` afişează numele fiecărui fişier înainte de ştergere.

Exemple. Presupunem că avem structura de directoare şi fişiere din figura 1.
Dacă tastăm comenzile:

```
$pwd
/home/ilflorea/so/exemple/Linux
$rmdir Grep
```

Se şterge directorul /`Grep`.
Dacă tastăm:

```
$pwd
/home/ilflorea
 $rd –r so
```

se vor şterge directoarele `/so, /exemple, /Linux, /Windows,/Awk, /Grep` şi fişierele `fis1` şi `fis2`.

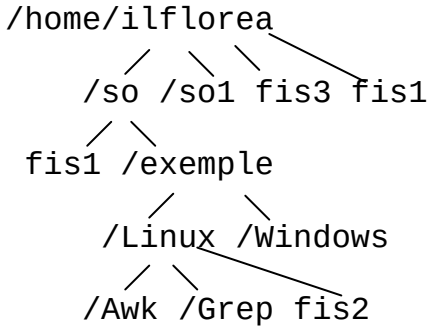

Figura 1. Structură de directoare şi fişiere

Comanda ls ("list") are sintaxa**:**

```
ls [opţiuni] [fişier_sau_director]
```

Are ca efect afişarea de informaţii despre un director sau fişier**.** Se utilizează pentru a se afişa numele fişierelor de diferite tipuri incluse într-un catalog şi, în funcţie de opţiuni alte informaţii despre aceste fişiere.

Exemplu. Utilizarea comenzii `ls` fără opţiuni. Relativ la structura din figura 1, dacă tastăm:

```
$ls /home/ilflorea
```

Se afişează:

```
so      so1
```

Cu opţiunea `-l` ("long") se afişează informaţii suplimentare (luate din nodul de index) despre fiecare fişier din catalogul argument al comenzii. Prima linie conţine numărul de blocuri ocupate de directorul respectiv; următoarele linii conţin următoarele informaţii:
- tipul fişierului (`d`: directory, `-`: fişier obişnuit `b`: fişier bloc; `c`: fişier special orientat pe caracter; `l`:legătură simbolică; `p`: fişier special FIFO (sau "pipe cu nume"); `s`: intrarea este un soclu ("socket") pentru comunicarea în reţea);
- conţinutul biţilor de protecţie/acces (r = read, w = write, x = execute), listaţi în ordinea: proprietar, grupul proprietarului, restul utilizatorilor (se va discuta la drepturi de acces);
- numărul de intrări în cataloage care punctează spre acel fişier fizic;
- numele de login al proprietarului fişierului;

- grupul proprietarului;
- lungimea fişierului (în octeţi);
- data ultimei modificări a fişierului;
- numele fişierului.

Exemplu. Utilizarea comenzii `ls` cu opţiunea `-l`. Relativ la structura din figura 1, dacă tastăm:

```
$ls –l /home/ilflorea
```

Se afişează:

```
total 22
drwxr-xr-x 1 ilflorea prof  4096 2010-08-24  22:04 so
drwxr-xr-x 1 ilflorea prof  2048 2010-08-24  23:04 so1
drwxr-xr-x 1 ilflorea prof 12048 2010-08-25  23:04 fis1
drwxr-xr-x 1 ilflorea prof 13048 2010-08-24  23:04 fis3
```

Cu opţiunea `-a` (“all”) se listează şi fişierele *ascunse* (fişierele al căror nume începe cu caracterul punct).

Exemplu. Utilizarea comenzii `ls` cu opţiunile `-al`. Relativ la structura din figura 1, dacă tastăm:

```
$ls –al /home/ilflorea
```

Se afişează:

```
total 22
drwxr-xr-x 1 ilflorea prof   512 2010-08-24  22:04 .
drwxr-xr-x 1 ilflorea prof  1024 2010-08-24  23:04 ..
drwxr-xr-x 1 ilflorea prof  4096 2010-08-24  22:04 so
drwxr-xr-x 1 ilflorea prof  2048 2010-08-24  23:04 so1
drwxr-xr-x 1 ilflorea prof 12048 2010-08-25  23:04 fis1
drwxr-xr-x 1 ilflorea prof 13048 2010-08-24  23:04 fis3
```

Cu opţiunea `-i` (“index”) se afişează şi numărul nodului index asociat fiecărei intrări din catalogul listat.
Cu opţiunea `-s` (“size”) se afişează dimensiunea în blocuri a fişierului.

Exemplu. Utilizarea comenzii `ls` cu opţiunile `-alis`. Relativ la structura din figura 1, dacă tastăm:

```
$ls –alis /home/ilflorea
```

Se afişează:

```
total 22
301860 8 drwxr-xr-x 1 ilflorea prof   512 2010-08-24  22:04  .
65473  8 drwxr-xr-x 1 ilflorea prof  1024 2010-08-24  23:04  ..
360621 8 drwxr-xr-x 1 ilflorea prof  4096 2010-08-24  22:04  so
301862 8 drwxr-xr-x 1 ilflorea prof  2048 2010-08-24  23:04  so1
361443 8 drwxr-xr-x 1 ilflorea prof 12048 2010-08-25  23:04  fis1
301879 8 drwxr-xr-x 1 ilflorea prof 13048 2010-08-24  23:04  fis3
```

Se observă că numărul nodului index este listat pe prima coloană.

Pentru a schimba ordinea de listare implicită (alfabetic, după numele fişierului) există mai multe **opţiuni:**
`-c:` listarea se face în ordinea ultimei modificări a modului de acces la fişier (cel mai recent modificat se listează primul);
`-t`: listarea se face în ordinea ultimei modificări a fişierului (cel mai recent modificat primul);
`-u`: listarea se face după timpul ultimului acces la fişier;

`-r`: inversează ordinea de sortare (invers alfabetic sau cel mai vechi modificat primul).
`-S` sorteaza fisierele in ordine descrescatoare dupa dimensiunea lor.

Exemplu. Pentru găsirea celor mai mari fisiere din directorul curent se listează fisierele sortate in ordine descrescatoare dupa dimensiunea lor, prin:
`$ls -lS`

Dacă argumentul unei comenzi `ls` este un fişier, în funcţie de opţiunile utilizate se afişează informaţii despre fişierul respectiv.

Creaţi o structură de directoare şi fişiere, similară celei din figura 1, în care să înlocuiţi `ilflorea`, cu numele vostru de utilizator. Adaptaţi exemplele prezentate, la structura creată.

2. Redirectarea intrărilor şi ieşirilor

Fiecare comandă Linux are un "fişier de intrare" şi un "fişier de ieşire". Implicit acestea sunt "intrarea standard" (tastatura de la care se introduce comanda) şi "ieşirea standard" (ecranul terminalului la care lucrează utilizatorul).

Interpretorul de comenzi stabileşte identitatea fişierelor standard de intrare şi standard de ieşire, pe care le ataşează comenzii. Atunci când se execută comanda respectivă, acestea primesc *descriptorii de fişier* `0` (intrare) şi `1` (ieşire). De asemenea, shell-ul stabileşte fişierul standard de erori, care are descriptorul `2` şi este transmis comenzii (de regulă este acelaşi cu fişierul standard de ieşire).

Shell-ul permite utilizatorului sa specifice explicit în linia de comandă o altă identitate pentru intrarea standard şi/sau ieşirea standard, prin caracterele speciale `<`, respectiv `>`.

Exemplu.
`$ ls -l fis1 >fis2`
Informaţiile despre fişierul `fis1` din catalogul curent, nu apar pe ecran, ele sunt memorate în fişierul `fis2` din directorul curent. Înainte de lansarea în execuţie a comenzii `ls`, shell-ul deschide pentru scriere `fis2` (cu descriptorul de fişier l). Dacă fişierul `fis2` există, prin operaţia de deschidere pentru scriere conţinutul său anterior se pierde; dacă fişierul nu există, el va fi creat de către shell şi va avea iniţial lungimea nulă.

Efectul unei specificări de forma:

```
$comanda <fisier
```

este acela că argumentele cu care va fi executată `comanda` nu vor fi luate de la intrarea standard, ci din fişierul cu numele `fisier` .

O comandă în care se redirectează atât intrarea cât şi ieşirea va fi de forma:

```
$comanda <input >output
```

Ordinea de specificare a redirectărilor nu este semnificativă.

Prim notaţia `n>`, în care `n` este un întreg, se specifică redirectarea pentru scriere a fişierului cu descriptorul `n`. Această facilitate se foloseşte pentru redirectarea fişierului standard de eroare. Sintaxa unei comenzi în care toate cele `3` fişiere standard sunt redirectate este:

```
$comanda <input >output 2>errors
```

Pentru ca fişierul în care se redirectează ieşirea să nu fie suprascris, ci să se adauge la sfârşitul lui noi date, se foloseşte notaţia `>>`. Pentru adăugarea la sfârşitul fişierului de erori sintaxa este `2>>`. Astfel:

```
$comanda >>logfile
```

va determina ca mesajele produse de comanda să se adauge la sfârşitul fişierului logfile.

Prin notaţia <<c se specifică preluarea informaţiilor necesare unei comenzi de la tastatură (în regim interactiv) sau din textul unei proceduri shell, până la întâlnirea lui c (caracter sau text). O astfel de constructie este de forma:

```
$comanda <<cuvant
.......
succesiune de linii
-----
Cuvant
```

Comanda cat este utilizată pentru a afişa, la ieşirea standard conţinutul unuia sau mai multor fişiere specificate în linia de comandă. Prin redirectarea ieşirii, folosind mecanismul prezentat anterior, cat se poate utiliza pentru a crea un fişier text.

Exemplu. Secvenţa:

```
$cat >fis <<EOF
Cursul de Linux
Este foarte important
Pentru mine
EOF
```

introdusă într-un fişier de comenzi, are ca efect crearea fişierului fis ce conţine trei linii ” Cursul de Linux”, ” Este foarte important”, ”Pentru mine”.

Altă modalitate de a crea fişiere cu cat, fără a folosi notaţia <<c, este de a folosi redirectarea ieşirii şi a tasta Ctrl-D, după ce s-a introdus în fişier textul respectiv.

Concatenarea fişierelor cu comanda cat. Dacă în linia de comandă a comenzii se specifică un număr de fişiere, dintre care ultimul este precedat de semnul >, se realizează concatenarea fişierelor specificate înaintea ultimului şi ultimul fişier va conţine rezultatul concatenării.

Exemplu. Secvenţa:

```
$cat fis1 fis2 fis3 >fis
```

Realizează concatenarea fişierelor fis1 fis2 fis3 în fişierul fis.

Dacă la redirectare se foloseşte notaţia >>fis, textul respectiv se va adăuga la sfârşitul fişierului, fără a distruge conţinutul lui.

Exemplu. Secvenţa:

```
$cat fi>fis
Cursul de Linux
Este foarte important
Pentru mine
Ctrl-D
```

se crează acelaşi fişier ca în exemplul anterior.

Legarea în pipe (“conductă”) are sintaxa:

```
$comanda1 | comanda2
```

Efectul este următorul: comenzile comanda1 şi comanda2 vor fi lansate în execuţie simultan, ieşirea lui comanda1 fiind utilizată ca intrare de comanda2. comanda2 poate începe efectiv execuţia numai după ce are un element de informaţie de la comanda1 (dar nu trebuie să aştepte până când comanda1 termină de produs tot ce va furniza ca ieşire). Ieşirea pentru comanda1 (descriptor de fişier 1) este un fişier (o structură numită **pipe)** şi acelaşi fişier va fi intrare pentru comanda2 (descriptor de fişier 0). Mecanismul *pipe* este frecvent utilizat în Linux pentru combinarea unor comenzi.

Exemplu. Secvenţa:
```
$ls -l|grep ilflorea
```
Va afişa fişierele din directorul curent ale căror proprietar este utilizatorul `ilflorea`. Comanda `grep` o vom studia într-un l următor. În acest exemplu, are ca efect căutarea cuvîntului `ilflorea` în ieşirea comenzii `ls -l`. Acelaşi efect se poate obţine şi prin secvenţa următoare:
```
$ls –l >fis
grep ilflorea fis
```

Se poate utiliza o legare în *pipe* a mai multor comenzi, de forma:
`$comanda_1|comanda_2 |...comanda_i|comanda_i+1...|comanda_n`
În această situaţie, ieşirea lui `comanda_i` este utilizată ca intrare de comanda_`i+1`.

Exemplu. Secvenţa:
```
$ls –l /home/ilflorea |grep ilflorea | sort +24
```
Va afişa fişierele din directorul `/home/ilflorea` ale căror proprietar este utilizatorul `ilflorea`, ordonate după dimensiune. Comanda `sort` o vom studia în această unitate de învăţare.

Transmiterea `stderr` în acelasi loc cu `stdout` foloseşte sintaxa `2>&1`.

Exemplu. Secvenţa:
```
$ls /home/ilflorea >outcom.txt 2>&1
```
Lucrează astfel:
`ls /usr/bin` listeaza continutul lui `/home/ilflorea`.
`>` redirecteaza iesirea comenzii `ls` (`stdout`) catre fisierul `outcom.txt`.
`2>&1` transmite fisierul cu descriptorul `2`(`stderr`), la aceeasi locaţie cu fişierul cu descriptorul `1` (stdout).
Deoarece `stdout` a fost redirectat, toate mesajele de eroare vor fi si ele trimise în fisierul `outcom.txt`.

Exemplu. Lansăm comanda `ls` pentru un director inexistent.
```
$ls /home/ilfloreax >outcom.txt 2>&1
$cat commands.txt
ls: /home/ilfloreax: No such file or directory
```
În acest exemplu, am presupus ca nu există directorul `/home/ilfloreax`.

Redirectarea în acelaşi timp a lui `stderr` şi `stdout` foloseşte sintaxa `&>`.

Exemplu.
```
$ls /home/ilfloreax &>outcom.txt
$cat outcom.txt
ls: /home/ilfloreax: No such file or directory
```
Lucrează astfel:
`&>outcom.txt` redirectează atât `stdout` cât şi `stderr` către fişierul `outcom.txt`.
Dacă nu se redirectează ambele fişiere, atunci mesajele de eroare vor fi transmise la terminal.

Exemplu.
```
$ ls /home/ilfloreax >outcom.txt
ls: /home/ilfloreax: No such file or directory
```
În acest caz, erorile vor fi afişate pe ecran, şi toate ieşirile vor fi transmise în fişierul `outcom.txt`.

Se mai pot utiliza construcţiile:
`i>&j` redirecteaza descriptorul de fisier `i` la `j`. Toate ieşirile, care în mod uzual sunt trimise fişierului cu descriptorul `i`, vor fi trimise catre cel cu descriptorul `j`.
`>&j` redirectează ieşirile către fişierul cu descriptorul `j`.

Execuţia unei comenzi fără transmiterea ieşirii. Există situaţii în care:
- O comandă crează o mulţime de date care nu sunt necesare.
- Dorim să vedem numai mesajele de eroare (dacă acestea există).
- Suntem interesaţi să vedem dacă comanda respectivă se execută cu succes sau nu.

În aceste cazuri, se face redirectarea iesirii catre fisierul nul, `/dev/null`.

Exemplu.
```
$ls /home/ilflorea >/dev/null
```
Se verifică dacă există directorul `/home/ilflorea.`

1. Creaţi cu comanda `cat` câte un fişier prin ambele metode. Adăugăţi text la fişierele create prin ambele metode.
2. Scrieţi o legare în pipe a două comenzi, prin care să se caute şi să se afişeze informaţii despre toate fişierele dintr-un director, al căror nume conţine şirul de caractere `fis`.
3. Scrieţi o secvenţă de concatenare a trei fişiere aflate în directoare diferite şi fişierul în care se face concatenarea să fie în directorul gazdă.

3. Copierea, mutarea şi redenumirea fişierelor

Copierea unui fişier se face cu comanda `cp` (“copy”). Prin copiere se crează un nou fişier cu acelaşi conţinut. Are sintaxa:

```
cp [opţiuni] nume1 nume2
```

Opţiunile comenzii sunt:
`-a` (“archive”): se foloseşte acestă opţiune pentru a se copia odată cu fişierul şi caracteristicile lui.
`-b` (“backup”): dacă `nume2` este numele unui fişier deja existent, se face o copie de siguranţă a cestuia.
`-f` („force”): se face copierea chiar dacă atributele lui `nume2` interzic acest lucru.
`-i` („interactive”): utilizatorul este întrebat atunci când se face suprascrierea.
`-p` („preserve”): se face copierea şi a drepturilor, concept care îl vom discuta în această unitate de învăţare.
`-r` („recursive”): se face copierea recursivă.
`-u` („update”): se face copierea numai dacă `nume2` nu există sau este mai vechi decât `nume1`.
Comanda are întotdeauna cel puţin două argumente.
• Dacă ambele argumente reprezintă nume de fişiere obişnuite, `nume2` va fi un fişier nou (i se alocă nod index şi blocuri de date proprii), cu conţinut identic cu cel al fişierului `nume1`.
• Dacă `nume1` este un fişier obişnuit, iar `nume2` desemnează un catalog, atunci în catalogul `nume2` se creează o intrare cu numele `nume1` pentru copia fişierului.

• Dacă atât `nume1` cât şi `nume2` reprezintă nume de cataloage, atunci comanda `cp` se poate executa doar dacă este prevăzută şi opţiunea `-R` sau `-r`, iar efectul este crearea în `nume2` a unui catalog `nume1` şi copierea recursivă a fişierelor din catalogul original `nume1` în catalogul nou creat.

Observaţie. Numele de fişier `nume1` poate fi definit şi ca un şablon, prin folosirea caracterelor `*` şi `?`. Dacă este prezent caracterul `*`, respectiv ?, acesta poate fi înlocuit cu orice şir de caractere, respectiv cu un caracter, cu condiţia ca numele obţinut să existe în structura respectivă.

Exemplu. Relativ la structura din figura 2; dacă se tastează:

```
$cd /home/ilflorea
$cp ./so/fis1 ./so/exem/linux
```

Se crează un nou fişier cu numele `fis1` în directorul `/home/ilflorea/so/exem/linux`, care are acelaşi conţinut cu fişierul cu numele `fis1` din directorul `/home/ilflorea/so`.

Exemple Dacă se tastează:

```
$cd /home/ilflorea
$cp ./so/fis1 ./so/exem/linux/fis12
```

Se crează un nou fişier cu numele `fis12` în directorul `/home/ilflorea/so/exem/linux`, care are acelaşi conţinut cu fişierul cu numele `fis1` din directorul `/home/ilflorea/so`.

Dacă se tastează:

```
$cd /home/ilflorea
$cp ./so/*fis? ./so/exem/linux1
```

Se crează noile fişiere cu numele `fis1,abfis2, afis2` în directorul `/home/ilflorea/so/exem/linux1`, care au aceleaşi conţinuturi cu cele din directorul `/home/ilflorea/so`.

Dacă se tastează:

```
$cd /home/ilflorea/so
$cp –r ./exem /home/ilflorea
```

Se creează o sub-structură de directoare şi fişiere în `/home/ilflorea`, care conţine sub-structura cu rădăcina în `/home/ilflorea/so/exem` (Figura 2).

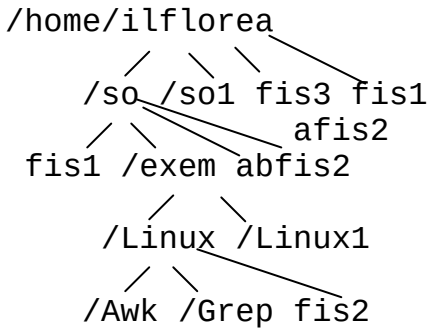

Figura 2. Structură de directoare şi fişiere

Schimbarea numelui (poziţiei) unui fişier se face cu comanda `mv` (“move”). Sintaxa comenzii este:

```
mv [opţiuni] sursa destinatie
```

• Ca şi comanda `cp`, `mv` are minimum 2 parametri. Deosebirea între cele două comenzi, este aceea că `mv` nu creează un fişier nou ci, pentru un fişier existent, doar schimbă numele său sau poziţia sa într-un sistem de fişiere. Din acest punct de vedere, se poate considera că `mv` este o operaţie asupra cataloagelor.
• Dacă `sursă` şi `destinatie` sunt nume de fişiere obişnuite, atunci intrarea din catalog cu numele `sursă` dispare şi se creează o intrare cu numele `destinatie`.
• Dacă `sursă` este nume de fişier obişnuit sau de catalog, iar `destinatie` este nume de catalog, dispare intrarea `sursa` şi se creează o intrare cu numele `sursa` în catalogul `destinatie`. În acest caz, trebuie să se folosească opţiune `–t`.
• În situaţia că `sursă` este un catalog, prin această comandă întreaga *sub-ierarhie* sursă îşi schimbă poziţia.
• Dacă intrarea destinaţie există, ea este suprascrisă. Se poate evita suprascrierea prin utilizarea opţiunii `-i`, la fel ca şi la comanda `cp`.

Observaţie: Comanda `mv` nu poate realiza traversarea sistemelor de fişiere în situaţia în care `sursă` este un catalog (deci cataloagele nu se pot muta dintr-un sistem de fişiere în altul). Pentru fişierele obişnuite, într-un astfel de caz se creează un alt fişier (se alocă nod index şi blocurile de date necesare) în noul sistem de fişiere.

Exemple. Relativ la structura din figura 2. ,dacă se tastează:

```
$cd /home/ilflorea/so
$mv fis1 fis2
```

Se va şterge intrarea `fis1` din directorul `/home/ilflorea/so` şi se va crea o noua intrare cu numele `fis2` în acelaşi director, care va corespunde unui fişier cu acelaşi conţinut cu vechiul `fis1.`

Dacă se tastează:

```
$cd /home/ilflorea
$mv ./so/fis1 –t ./so/exem/linux
```

Se crează o nouă intrare în directorul `/home/ilflorea/so/exem/linux` pentru fişierul `fis1` şi se distruge intrarea corespunzătoare din directorul `/home/ilflorea/so.` Fişierul va păstra vechiul conţinut.