

Laboratorul 6

1. Sortarea fișierelor

Sortarea conținutului fișierelor text se face cu comanda `sort`, a cărei sintaxă este:

```
sort [opțiuni] nume_fisier
```

Prin sortarea unui fișier, se înțelege obținerea unui alt fișier din acesta, în care liniile sunt ordonate lexicografic după conținutul uneia sau mai multor chei de sortare.

O **cheie de sortare** poate fi o întreagă linie sau o secvență de câmpuri din liniile fișierului, considerând textul unei linii divizat în câmpuri delimitate prin separatori.

Separatorul implicit este spațiul. Se poate indica un alt separator prin opțiunea `-t`, urmată de un caracter `c`, unde `c` reprezintă caracterul folosit ca separator.

`nume_fisier` este singurul **argument obligatoriu**: fișierul (fișierele) de sortat.

Forma sortată este afișată implicit în fișierul standard de ieșire. Prin opțiunea `-o`, poate fi specificat un fișier care să conțină imaginea sortată a fișierului. Implicit, ordonarea liniilor se face crescător; cu opțiunea `-r` se inversează ordinea de sortare (descrescător).

Cheile se pot indica prin perechi de numere; primul este precedat de semnul `+` și indică câmpul de început al cheii; al doilea este precedat de semnul `-` și indică câmpul de sfârșit al liniei. Câmpurile se numerează de la stânga la dreapta, începând cu 0. Altă modalitate de a defini cheile de sortare este utilizarea opțiunii `-kpoz1, [poz2]`, prin care se indică câmpul de început, respectiv de sfârșit ale cheii de sortare; dacă `poz2` lipsește, atunci sfârșitul cheii de sortare va fi sfârșitul liniei respective. Dacă valoarea unui anumit câmp trebuie interpretată numeric și nu ca șir de caractere, se folosește opțiunea `-n`, care se atașează la numărul de ordine al câmpului.



Exemple. Sortarea fișierului de parole după valoarea numerică a identificatorului de utilizator (UID) (al treilea câmp al fiecărei linii), se face prin:

```
$sort -t: +2n -3 /etc/passwd
```

Cu opțiunea `-t`, s-a specificat faptul că separatorul de câmpuri în acest fișier este caracterul `:` și cu opțiune `n` că valoarea celui de-al treilea câmp se interpretează numeric. Dacă se tastează:

```
$sort -t: +2n -3 /etc/passwd -o fispass
```

Imaginea sortată a fișierului va fi fișierul `fispass`

Sortarea unei liste de numere de telefon după nume, iar în cazul aceluiași nume după prenume se face prin:

```
$sort +1 -2 +0 -1 listatelefon
```



1. Rescrieți exemplele anterioare folosind opțiunea `-k`, pentru definirea cheilor de sortare.
2. Rescrieți exemplele anterioare astfel încât ordonarea să se facă descrescător.
3. Presupunem că am creat un fișier text numit `studenti` care conține câmpurile: `nume`, reprezentat pe 10 caractere; `prenume`, reprezentat pe 9 caractere; `grupa`, reprezentată pe 4 caractere; `media`, numeric reprezentat pe 5 caractere. Scrieți o comandă care să sorteze crescător după `grupa`, descrescător după `media` declarată numeric și crescător după `nume` și `prenume`.

2. Numărarea liniilor, cuvintelor și caracterelor unui fișier

Determinarea numărului de linii, cuvinte și caractere dintr-un fișier se face cu comanda `wc` ("word count"), a cărei sintaxă este:

```
wc [-lwc] [nume_fis]
```



Exemple. Dacă se tastează:

```
$ls -l *.txt
total 2
-rw-r--r-- 1 ion 102 nov 9 12:57 fis1.txt
-rw-r--r-- 1 ion 201 nov 9 13:33 fis2.txt
adică în directorul curent avem două fișiere cu extensia .txt
$wc *.txt
2   6   102  fis1.txt
4  16   201  fis2.txt
6  22   303  total
```

Observăm că numărul de caractere afișate de comanda `wc` coincide lungimea fișierelor afișată de `ls`.

Se consideră cuvânt, un șir de caractere delimitat de spațiu, tab sau “newline”. Dacă `nume_fis` lipsește, fișierul standard de intrare este implicit. Prin opțiuni se specifică dacă se dorește numărul de linii(`l`), cuvinte(`w`) sau caractere (`c`). Fără specificarea nici unei opțiuni, se afișează toate. Dacă în comandă apar mai multe nume de fișiere se afișează și numărul total.



Exemplu. Pentru a găsi numărului de fișiere din directorul curent, se dă comanda:

```
$ls | wc -l
```

Dacă vrem să le numărăm și pe cele ascunse, tastăm:

```
$ls -A | wc -l
```



1. Folosind comenzile `find`, `ls` și `wc`, numărați fișierele dintr-o structură dată ca argument al comenzii `find`.

3. Filtrarea fișierelor cu comanda `cut`

Comanda `cut` este utilizată pentru a selecta anumite bucăți din liniile unuia sau mai multor fișiere. Aceste bucăți pot fi caractere, cuvinte sau unități definite de către utilizator. Sintaxa comenzii este :

```
cut optiuni [fișier(e)]
```

Principalele opțiuni ale acestei comenzi sunt:

- `c lista` extragere în mod caracter;
- `f lista` extragerea câmpurilor specificate în listă;
- `d c` permite specificarea caracterului utilizat ca și separator în fișierul text între diferitele entități (în mod implicit este considerat separator caracterul `tab`).

Formatul listei poate fi de forma:

- `n` a n-a unitate
- `n,m` doar a n-a unitate și a m-a unitate;
- `n-m` de la a n-a unitate la a m-a unitate;
- `n-` de la a n-a unitate până la sfârșit;
- `n` de la prima unitate la a n-a unitate.



Exemplu.

Presupunem că am creat un fișier text numit `studenti` care conține câmpurile:
 nume, reprezentat pe 10 caractere;
 prenume, reprezentat pe 9 caractere;
 grupa, reprezentată pe 4 caractere;
 numar matricol, reprezentat pe 5 caractere;
 adresa, reprezentată pe 9 caractere;
 numar de telefon, reprezentat pe 9 caractere;
 anul de studiu, reprezentat pe 2 caractere;
 notele obtinute la examene, reprezentate pe 4 caractere (abs=absent la examen) - fiecare nota reprezintă un câmp.

Pentru a selecta primul caracter de pe fiecare linie, se va folosi comanda :

```
$cut -c 1 studenti
```

Pentru a selecta numele și prenumele, se va folosi comanda :

```
$cut -c 1-20 studenti
```

Același lucru se poate obține prin:

```
$cut -d ' ' -f1 1,2
```



Exemplu. Trimiterea unui mesaj prin poșta electronică, tuturor utilizatorilor aflați momentan în sesiune, se face prin:

```
$mail `who | cut -c 1-8 | sort -u`
```

Prin `who` se obține lista sesiunilor deschise (un utilizator poate avea simultan mai multe sesiuni); prin `cut` se rețin primele 8 caractere din linie (numele utilizatorului), iar prin `sort -u` se face sortarea numelor cu eliminarea duplicatelor.



Exemplu. Extragerea numelui de utilizator, a numelui real și shell-ului de start din `/etc/passwd`, se face prin

```
$cut -d: -f1,5,7 /etc/passwd
```



1. Scrieți o legare în pipe a comenzilor `cut` și `sort`, care să extragă din fișierul `studenti` câmpurile: nume, prenume, grupa, numar matricol, notele obtinute la examene; liniile obținute să fie sortate după câmpul `numar matricol`.

2. Scrieți o legare în pipe a comenzilor `ps` și `cut` prin care să se afișeze următoarele informații despre procesele din sistem: PID-ul procesului, proprietarul și comanda executată.

4. Căutarea în fișiere cu comanda **grep**

Comanda `grep` are sintaxa:

```
grep [optiuni] sablon [nume_fis]
```

`sablon` poate fi interpretat și ca o expresie regulată. Dacă lipsește `nume_fis`, atunci liniile în care se face căutarea sunt citite de la intrarea standard, altfel sunt luate din fișierul sau fișierele date ca parametri. În funcție de opțiuni și de liniile care se potrivesc cu șablonul dat, se afișează anumite informații.

Opțiunile cele mai utilizate sunt:

- i nu face diferențele dintre literele mari și cele mici;
- v tipărește liniile cu care șablonul dat *nu* se potrivește;

- n tipărește numărul liniei, urmat de caracterul :, urmat de conținutul liniei;
- x tipărește doar liniile pentru care se potrivește întreaga linie cu tiparul dat, nu cu un subșir al acesteia;
- c tipărește doar numărul de linii în care s-au găsit potriviri;
- h nu returnează numele de fișiere;
- l listează numai numele de fișiere, nu și textul care se potrivește.
- q verifică numai codul de ieșire al comenzii;
- s suprimă mesajele de eroare.



Exemplu. `$grep root /etc/passwd`

Afișează linia corespunzătoare din `/etc/passwd` a utilizatorului cu dreptur de `root`

`$grep -n root /etc/passwd`

În plus, afișează și numărul de linie, la începutul acesteia.



Exemplu.

`$ grep -v bash /etc/passwd | grep -v nologin`

Se afișează utilizatorii care nu folosesc `bash` ca shell, dar conturile shell `Nologin` nu sunt afișate.



Exemplu. `$grep -c false /etc/passwd`

Afișează numărul utilizatorilor care au shell-ul `/bin/false`



Exemplu. Combinație între `grep` și `find`

`$find . -name "*.mp3"|grep -i andra| grep -vi „remix”`

5. Caractere speciale în shell

Unele caractere au semnificație specială pentru interpretorul shell, în diverse specificații, cum este cazul celor utilizate în contextul expresiilor regulate, în modul de evaluare al conținutului variabilelor etc. În cazul `Bash` caracterele speciale sunt:

`# & * ? [] () = | ^ ; < > ` $ " ' \`

Caracterele spațiu, tab și newline au rolul de separatori de cuvinte. Dezautorizarea unui caracter, înseamnă că el va fi interpretat într-un anumit text ca un simbol obișnuit, fără a mai avea o altă semnificație.

Dezautorizarea semnificației caracterelor speciale se face prin trei astfel de metode, indicate de caracterele speciale `'`, `"` și `\`, care operează astfel:

`'text'` Dezautorizează toate caracterele speciale din `text`.

`"text"` Dezautorizează caracterele speciale din `text` cu excepția `$`, ``` și `\`.

`\c` Dezautorizează caracterul special `c`. La sfârșit de linie elimină un newline.

În cele ce urmează, folosim comanda `echo`, care afișează în fișierul standard de ieșire argumentele date în linia de comandă.



Exemplu.

`$echo 99 \> 89`

`99 > 89`

Se anulează sensul special al caracterului `>`, deci el va fi interpretat ca un simplu caracter.

`$echo 99 > 89`

echo: syntax error.
Argumentul comenzii este o expresie eronată.



Exemplu.

```
$echo unu plus doi ' egal trei? ' sau patru # sau $$.  
unu plus doi egal trei? sau patru
```

Cele două apostrofuri sunt considerate caracter de citare, deci nu sunt tipărite. Caracterul ? dintre apostrofuri nu mai este de caracter special. Caracterul # marchează pentru shell începutul unui comentariu, deci restul liniei este ignorată.



Exemplu.

```
$echo "unu plus doi 'egal trei?'sau patru # sau $$. "  
unu plus doi 'egal trei?'sau patru # sau 1191.
```

Spațiile din textul dintre ghilimele sunt tratate ca și caractere obișnuite, ca și apostroful. \$\$ desemnează valoarea variabilei de mediu \$, care conține identificatorul de proces (PID) al shell-ului, care se înlocuiește în text.

6. Definirea și referențierea variabilelor

Sub Linux, variabilele păstrează informații necesare stabilirii mediului de execuție al comenzilor. Valorile lor sunt șiruri de caractere și sunt inițializate prin comenzi de atribuire de forma: `var=sir`

Referirile la valoarea unei variabile de mediu este de forma `$var`. Pentru ca inițializarea unei variabile să rămână valabilă pe întreaga durată a unei sesiuni de lucru a unui utilizator ea trebuie *exportată*. În acest caz, atribuirea are forma: `var=sir; export var`



Exemplu. Inițializarea unei variabile recunoscute numai pe durata unei sesiuni shell.

```
$VAR=text  
$echo $VAR  
text
```



Exemplu. Inițializarea unei variabile recunoscute pe durata unei întregi sesiuni de lucru a unui utilizator.

```
$VAR=text  
$echo $VAR  
text  
$export VAR  
$bash  
$echo $VAR  
Text
```

Prin execuția comenzii `bash` se lansează un nou proces shell; observăm că și acest nou proces recunoaște setarea variabilei exportate.

Variabilele de mediu se mai pot clasifica în variabile globale și variabile locale. **Variabilele locale** sunt vizibile numai în cadrul comenzii sau scriptului în care au fost create. **Variabilele globale** sunt vizibile în cadrul altor entități legate de un proces sau un script, cum ar fi o funcție din cadrul unui script sau un fiu al unui proces, noțiuni pe care le vom discuta mai târziu. Comanda `printenv` listează toate variabilele de mediu existente în momentul tastării; se pot observa diferențele dintre variabilele exportate și neexportate.



Exemplu. Un fragment din ieșirea comenzii `printenv`.

```
$printenv
HOSTNAME=statia1.info.unitbv.ro
TERM=xterm
SHELL=/bin/bash
HISTSIZE=1000
OLDPWD=/home/ilflorea/test/test1
SSH_TTY=/dev/pts/0
USER=ilflorea
PATH=../bin: /HOME/ilflorea/bin:/bin:/usr/bin
INPUTRC=/etc/inputrc
PWD=/home/ilflorea
HOME=/home/ilflorea
LOGNAME= ilflorea
```

Există variabile *definite de utilizator* în scripturile scrise de acesta și variabile *predefinite*, care pot fi utilizate în toate scripturile sau în toate comenzile. Valorile acestor variabile sunt stabilite de sistem la intrarea în sesiune, dar pot fi modificate de utilizator pe parcurs.

Substituirea valorii unei variabile shell se poate specifica printr-una dintre specificațiile:

```
$var
${var}
```

7. Variabile de sistem

Variabila `PATH` conține o listă a directorilor în care CPU caută numele de comenzi lansate în execuție. Există astfel posibilitatea lansării acestora fără specificarea căii către fișierele de comenzi sau executabile respective. Elementele listei sunt separate prin caracterul `:` și ordinea de căutare este dată de ordinea specificării în listă. Pentru a adăuga noi directoare în lista de căutare, se utilizează o comandă de atribuire: `PATH=$PATH:cale`



Exemplu. Atribuire de valori pentru această variabilă:

```
PATH=../bin:$HOME/bin:/bin:/usr/bin
```

Conform acestei specificații, numele comenzii lansate sau se va căuta mai întâi în directorul curent (`.` înaintea primului `:`), apoi în subdirectorul `bin` al directorului curent (`./bin`), apoi în subdirectorul `bin` al directorului lui gazdă al utilizatorului (conținut în variabila `HOME`), în directorul `/bin` și în final în directorul `/usr/bin`. Adăugarea căii de căutare `/usr/local/bin`:

```
PATH=$PATH:/usr/local/bin
```

Variabila `HOME` conține directorul gazdă al utilizatorului; observăm că în exemplul anterior am utilizat această variabilă.



Exemplu.

```
$echo $HOME
/home/ilflorea
```

Variabila `MANPATH` conține lista directorilor în care se caută paginile de manual on-line de către comanda `man`. O valoare tipică este

```
MANPATH=/usr/man:/usr/local/man
```

Variabila `TERM` conține tipul terminalului folosit în sesiunea curentă.

Variabila `MAIL` conține numele de cale pentru cutia poștală a utilizatorului.

Variabila `PS1` conține șirul folosit ca prompt. Implicit utilizatorii obișnuiți au promptul `$`, iar cel cu drepturi de root are promptul `#`. Ca și celelalte variabile de mediu, `PS1` poate fi redefinit de utilizator. Se pot utiliza mai multe argumente de setare, cele mai utilizate fiind:

`\h` numele calculatorului (host) la care s-a loginat utilizatorul;
`\s` numele shell-ului;
`\t` valoarea curentă a timpului exprimată în formatul `ora:min:sec`;
`\u` nume de login al utilizatorului(user);
`\v` versiunea shell-ului Bash utilizat;
`\w` calea absolută către directorul curent (working directory);
`\W` numele directorului curent;
`\[` începutul unei secvențe de codificare;
`\]` sfârșitul unei secvențe de codificare.

Exemplu. Să presupunem că terminalul bash afișează:



```
ilflorea@l[~]$
```

Dacă tastăm:

```
ilflorea@l[~]$echo $PS1
```

o să se afișeze:

```
\u@\l[\W]\$
```

Observăm cele trei componente ale prompt-ului:

- numele utilizatorului care a lansat shellul (`\u`);
- numărul terminalului virtual (`\l`);
- numele directorului curent (`\W`); în acest caz este directorul gazdă (semnul `tilda` afișat între paranteze drepte).



Exemplu. Să presupunem că terminalul bash afișează:

```
[ilflorea@statia1 ~]$
```

În acest caz, se folosește numele de gazdă, în locul numărului de consolă.



Exemplu. Pentru ca terminalul să afișeze un nou prompt, atribuim o nouă valoare lui `PS1`.

```
[ilflorea@statia1~]$PS1="[\t][\u]\$"
```

```
[21:50:12][ilflorea]$
```

Observăm că noul prompt afișează ora curentă, și numele de user, ambele între paranteze drepte. Dacă tastăm:

```
[ilflorea@statia1~]$PS1="\u@\h:\w\$"
```

O să apară prompt-ul:

```
ilflorea@statia1:\home\ilflorea$
```

Observăm că am folosit codurile: `\u` - user, `\h` - host, `\w` - working directory. De asemenea, am inserat caracterul `@` între numele utilizatorului și numele de gazdă, respectiv caracterul `:` între numele de gazdă și directorul gazdă al utilizatorului. Nu am mai utilizat parantezele drepte ca delimitatori.

Variabila `PS2` conține șirul prompt suplimentar, utilizat când o comandă se întinde pe mai multe linii (implicit este caracterul `>`).

Variabila `$` conține numărul procesului (PID) pentru acest shell. Frecvent valoarea acestei variabile este folosită pentru generarea de nume de fișiere temporare, deoarece PID-urile existente în sistem la un moment dat sunt unice.

Variabila # conține numărul de argumente din linia de comandă.

Variabila 0 (zero) conține numele comenzii executate. Se folosește când pentru aceeași comandă există mai multe nume.

Variabila n conține (pentru valori între 1 și 9) reprezintă al n-lea argument din linia de comandă.

8. Variabile vector

Un vector se declară și se inițializează specificând elementele sale între două paranteze rotunde, separate prin caracterul spațiu.



Exemplu:

```
$sir_num=(unu doi trei patru cini)  
Am declarat și inițializat un vector cu 5 elemente.
```

Valorile indicelui sunt 0, 1, 2,... Referința la un element al vectorului se face specificând numele vectorului și indicele între paranteze drepte.



Exemplu:

```
$echo ${sir[2]}  
trei
```

Pentru a afișa tot vectorul, în poziția indicelui trebuie specificat caracterul *.



Exemplu:

```
$echo $sir  
unu  
În schimb:  
$echo ${sir [*]}  
unu doi trei patru cini
```

Se poate modifica valoarea unui element al vectorului printr-o instrucțiune de atribuire.



Exemplu:

```
$sir[2]=sase  
$echo ${mytest[*]}  
unu doi sase patru cini
```

Pentru a șterge conținutul unui element al tabloului, putem folosi comanda unset.



Exemplu:

```
$unset sir[2]  
$echo ${sir[*]}  
unu doi patru cini  
$echo ${sir[2]}  
$echo ${sir[3]}  
patru
```

Se poate șterge un întreg vector, dacă acesta este argument al unei comenzi unset. Comanda poate fi folosită și pentru ștergerea unei variabile oarecare.



```
$unset sir  
$echo ${sir[*]}
```

9. Substituția comenzilor

Ieșirea standard produsă de o comandă poate fi substituită în locul în care apare comanda respectivă, dacă acea comandă este încadrată între accente inverse (`...`) sau este

inclusă între acolade care sunt precedate de semnul \$. Prin această construcție, ieșirea unei comenzi poate fi folosită ca argument al altei comenzi. De asemenea, există posibilitatea de a se asocia aliasuri unei comenzi.



Exemplu. Un utilizator dorește să trimită un mesaj prin poșta electronică tuturor utilizatorilor aflați momentan în sesiune cu el. Se tastează:

```
$mail `who | cut -c1-8 | sort -u`
```

Cu `who` se obține lista sesiunilor deschise (un utilizator poate avea simultan mai multe sesiuni); prin `cut` se rețin primele 8 coloane din linie (numele utilizatorului), iar prin `sort -u` se face sortarea numelor cu eliminarea duplicatelor.



Exemplu. Dacă directorul curent este `/home/ilflorea`, comanda:

```
$ echo Va aflatii in directorul `pwd`!
```

va afișa la ieșirea standard:

```
Va aflatii in directorul /home/ilflorea!
```

Dacă se lansează:

```
$echo Va aflatii in directorul pwd!
```

rezultatul execuției va fi:

```
Va aflatii in directorul pwd!
```

Redefinirea comenzilor(alias-uri). În multe situații este preferabil să se asocieze unei comenzi (eventual lansată cu un număr mare de opțiuni sau parametri) un identificator. Prin tastarea acestuia în linia de comandă, se obține același efect ca și când am lansa comanda respectivă. Sintaxa unei astfel de declarații este:

```
nume=`instructiune`
```



Exemplu. O opțiune utilă a comenzii `ls` este `-F`; efectul ei este de a pune un slash (/) după directoare și un asterisk (*) după fișiere executabile. Se poate defini:

```
$alias lf=`ls -F`
```



Exemplu. Se definește o asociere între un identificator și legarea în conductă a două instrucțiuni. Se tastează:

```
$alias llm=`ls -al | more`
```



Exemplu de utilizare greșită a alias-ului. Dacă se definește un alias pentru un nume de director, de genul:

```
alias Nume_dir=/dir1/dir2/dir3
```

și se tastează

```
$cd Nume_dir
```

bash va afișa un mesaj de eroare: **No such file or directory.**

Anularea aliasurilor definite anterior se face cu comanda `unalias`.