3. LUCRUL CU LINIA DE COMANDA SI SISTEMUL DE FISIERE

3.1. Căi in sistemul de fisiere

O cale in sistemul de fisiere reprezinta "adresa" in arborele de fisiere a unui anumit fisier (oricare ar fi tipul lui). Ea este formata dintr-o succesiune de nume de fisiere (majoritatea fiind de tip director) separate de asemenea prin / (“slash”).

Element	Windows	Linux
Arbori de fisiere	mai multi	unul
Radacina sistem(e) de fisiere	C:\, D:\, ...	/
Separator elemente cale	\	/
Exemplu cale absoluta	C:\temp\test.txt	/tmp/test.txt
Exemplu cale relativa	..\d1\f1	../d1/f1

Caile pot fi de doua feluri:

- **absolute** – au ca referinta (punct de pornire) radacina sistemului de fisiere. O cale absoluta incepe cu simbolul pentru radacina (/).

  Nota: *in Windows o cale absoluta era obligata sa contina si identificatorul partitiei (C:, D:) si abia apoi simbolul pentru radacina sistemului de fisiere (\). In Linux arborele de fisiere este unic, de aceea caile absolute incep direct cu /*

- **relative** – au ca referinta directorul curent. Pentru a construi o cale relativa la acest director, pot fi folosite simboluri precum:
  - **.** – desemneaza directorul curent. Exemplu: comanda *./configure* executa fisierul *configure* din directorul curent
  - **..** – desemneaza directorul parinte al directorului curent. Exemplu: ../f1 se refera la un fisier aflat pe acelasi nivel din arborele de fisiere cu directorul curent

Directorul curent este un artificiu creat pentru ca utilizatorul sa poata scrie cai mai scurte in linia de comanda. Atunci cand "ne aflam intr-un director", acest lucru nu inseamna decat ca undeva in memorie este stocata calea catre acel director, ea fiind folosita apoi pe post de prefix pentru caile care nu incep cu caracterul /. Exemplu: daca directorul curent este /var/www, iar noi aplicam o comanda fisierului *localhost/index.html*, atunci calea completa catre fisierul dorit este */var/www/localhost/index.html*.

Nota: *directorul curent este stabilit per shell - doua terminale sau console virtuale diferite pot avea directoare curente diferite.*

In afara de . si .. exista si un alt simbol ce scurteaza caile scrise in linia de comanda Linux: ~ (tilda), care este automat inlocuita cu calea catre directorul personal al utilizatorului curent. Daca un utilizator este logat in sistem cu contul *student*, iar acest cont are directorul personal in */home/student*, atunci:

- ~ – se refera la directorul */home/student*
- *~/poze* – se refera la */home/student/poze*
- *~info* – se refera la directorul personal al utilizatorului info
- *~info/comenzi.txt* – se refera la fisierul *comenzi.txt* aflat in directorul personal al utilizatorului info

3.2. Navigare

Comenzi introduse: **cd, pwd**

Comanda **pwd** ("print working directory") afiseaza calea catre directorul curent. Existenta ei se justifica prin faptul ca nu intotdeauna directorul curent este afisat in linia de comanda; shell-ul prezinta utilizatorului, in stanga cursorului, asa-numitul "prompt string", care este insa configurabil, putand contine sau nu directorul curent.

prompt string

```
student@server1 $ pwd
/var/www
```

In acest caz, shell-ul a fost configurat ca prompt string-ul sa contina username-ul (student) si numele statiei (server1), insa nu si calea catre directorul curent.

Nota: *modalitatea de configurare a diferitelor aspecte ale shell-ului – inclusiv prompt string – va fi abordata intr-o lectie viitoare.*

Comanda *cd* ("change directory") este folosita pentru schimbarea directorului curent. Din momentul schimbarii, toate caile relative vor avea ca punct de pornire noul director curent.

Argumente particulare:

- comanda *cd* rulata fara argumente este echivalenta cu *cd ~* - noul director curent va fi directorul personal al utilizatorului logat
- argumentul – (minus) pasat lui *cd* va determina schimbarea directorului curent cu cel imediat anterior (o optiune echivalenta cu "back" din Windows Explorer)

```
student@server1 $ cd /var/www
student@server1 $ pwd
/var/www
student@server1 $ cd ..
student@server1 $ pwd
/var
student@server1 $ cd log
student@server1 $ pwd
/var/log
student@server1 $ cd -
student@server1 $ pwd
/var
student@server1 $ cd
student@server1 $ pwd
/home/student
```

3.3. Dimensiuni si spatiu liber

Comenzi introduse: **du, df**

Comanda **df** ("disk free") prezinta spatiul ocupat/disponibil pentru toate sistemele de fisiere montate. Este folosita de obicei in combinatie cu optiunea –h ("human-readable"), care afiseaza toate dimensiunile in KB/MB/GB (fara –h, ele ar fi afisate in kocteti):

```
student@server1 $ df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/hda1              2931732   1964720    967012  68% /
/dev/hda5              9767184   4310200   5456984  45% /usr/local
/dev/hda6              5855476   4911040    944436  84% /home
student@server1 $ df –h
```

```
Filesystem            Size  Used Avail Use% Mounted on
/dev/hda1             2.8G  1.9G  945M  68% /
/dev/hda5             9.4G  4.2G  5.3G  45% /usr/local
/dev/hda6             5.6G  4.7G  923M  84% /home
```

Comanda **du** ("disk usage") afiseaza dimensiunea directorului primit ca argument si a tuturor subdirectoarelor directe. Exista si aici optiunea –h, cu acelasi efect ca in cazul comenzii *df*. In plus, optiunea –s (sum) va determina afisarea doar a dimensiunii argumentului, nu si a fisierelor/subdirectoarelor componente:

```
student@server1 $ du -h /var/log/
169K    /var/log/cups
512     /var/log/samba
6.1M    /var/log/packages
1.5M    /var/log/scripts
512     /var/log/iptraf
71M     /var/log
student@server1 $ du –hs /var/log
71M     /var/log
```

3.4. Listare directoare si fisiere

Comanda introdusa: **ls**

Comanda **ls** este folosita pentru a lista in diferite moduri continutul directoarelor si detaliile despre fisierele continute (dimensiune, permisiuni, data crearii/modificarii etc).

Folosita fara argumente, comanda prezinta o lista a fisierelor din directorul curent. **ls** poate primi ca argument una sau mai multe cai, comanda aplicandu-se pe rand fiecaruia dintre argumente.

```
student@server1 $ pwd
/home/student/Desktop
student@server1 $ ls
poze referat.pdf    todo.txt
student@server1 $ ls poze
prietena1.jpg  prietena2.jpg
student@server1 $ ls  .  poze
.:
poze referat.pdf    todo.txt

poze:
prietena1.jpg  prietena2.jpg
```

Iata cateva optiuni uzuale ale comenzii ls:

- **-a** (all) – folosit pentru a afisa si fisierele "ascunse" – cele al caror nume incepe cu punct (ex: .ssh, .xinit). Proprietatea unui fisier de a fi ascuns nu este o setare de securitate – masura a fost luata pentru a nu polua output-ul comenzii ls cu fisiere de configurare. Un exemplu clasic este directorul personal al unui utilizator, unde multe aplicatii isi stocheaza setari sub forma unor fisiere si directoare ascunse.

```
student@server1 $ ls poze
sotie.jpg
student@server1 $ ls –a poze
.    ..    .amanta1.jpg     .amanta2.jpg     sotie.jpg
```

- **-R** (recursiv) – efectueaza listarea in profunzime a directoarelor pe care ls le primeste ca argument (fiecare subdirector intalnit este listat la randul sau)

```
student@server1 $ ls /home/student/Desktop
Home            Trash/
student@server1 $ ls -R /home/student/Desktop
/home/student/Desktop/:
Home            Trash/

/home/student/Desktop/Trash:
manea1.mp3      manea2.mp3
```

Nota: *o alta solutie pentru listarea recursiva este comanda* ***tree****.*

- **-l** (long listing) – efectueaza o listare detaliata. Pentru fiecare fisier va fi afisata o linie asemanatoare cu urmatoarea:

```
-rw-r--r--    2 catalin     users          80322 oct 16 2014 pic1.jpg
```

Semnificatia elementelor este urmatoarea:

- \- (primul caracter de pe linie) indica tipul fisierului: - pentru fisiere obisnuite, d pentru directoare, l pentru symlink-uri, b pentru block device, c pentru character device etc
- **rw-r--r--** reprezinta permisiunile pe fisier (lucrul cu sistemul de permisiuni va fi discutat pe larg intr-o lectie viitoare)
- **2** reprezinta numarul de hard link-uri (cate nume are per total acel fisier). In Linux exista posibilitatea ca continutul aceluiasi fisier sa fie vizibil in mai multe puncte din arborele de fisiere, sub diverse nume
- **catalin** este owner-ul (proprietarul) fisierului. In general owner-ul unui fisier este cel care l-a creat, insa pot exista si exceptii (de exemplu, root poate schimba proprietarul fisierelor)
- **users** este grupul proprietar. In Linux, fiecare fisier are un user si un grup proprietar.
- **80** este dimensiunea in octeti a fisierului. Daca se doreste exprimarea ei in KB/MB/GB se poate folosi optiunea –h a comenzii ls

  Atentie! *In cazul unui director, aceasta nu este dimensiunea cumulata a fisierelor continute, ci doar marimea fisierului de tip director! Daca se intentioneaza obtinerea dimensiunii cumulate, se va folosi comanda du -hs*

- **oct 16 2014** – reprezinta data ultimei modificari. Aceasta este numai una dintre cele 3 date existente pentru un fisier: data crearii, data ultimei modificari si data ultimei accesari
- **pic1.jpg** – numele fisierului

- **-d** (director) – atunci cand unul dintre argumentele lui ls este un director, in loc sa se afiseze continutul acestuia va fi afisat chiar directorul. Optiunea este foarte utila in tandem cu –l, permitand astfel vizualizarea proprietatilor unui director:

```
student@server1 $ ls –l poze
-rw-r--r--    1 catalin     users          56854 Jun  2  2008 pic1.jpg
-rw-r--r--    1 catalin     users          67899 Jun  3  2008 pic2.jpg
student@server1 $ ls –ld poze
drwxr-xr-x    2 catalin     users             96 Jun  1  2008 poze/
```

3.5. Metacaractere in linia de comanda

Deseori apare nevoia de a ne referi, din linia de comanda, la un intreg grup de fisiere sau directoare ce impartasesc o caracteristica comuna. De exemplu, dorim sa stergem toate fisierele cu extensia .mp3 dintr-un director, vrem sa listam numai fisierele al caror nume incepe cu *pic* si se continua cu doua cifre etc. In acest scop exista mici constructii ce permit utilizatorului descrierea formatului numelui de fisier in locul numelui exact.

Constructiile amintite sunt create folosind tot caracterele obisnuite, prezente pe tastatura si afisabile pe ecran; in acest scop, unora dintre caractere li se confera semnificatii speciale, astfel incat interpretorul de comenzi sa recunoasca, pe baza acestor caractere distinctive, intentia utilizatorului de a descrie o familie de nume de fisiere si nu un nume exact. Caracterele de acest fel poarta denumirea de *metacaractere*.

Iata in continuare cateva metacaractere si constructiile ce se pot realiza cu ajutorul lor:

- ***** - inlocuieste 0 sau mai multe caractere

ls –d /muzica/a*	listeaza fisierele din /muzica al caror nume incepe cu a si are cel putin 1 caracter
ls –l /home/*/Desktop	listeaza detaliat continutul desktop-urilor tuturor utilizatorilor din sistem
ls –ld /*	listarea detaliata a tuturor fisierelor/directoarelor din radacina

- **?** – inlocuieste un singur caracter.

ls *.???	listeaza toate fisierele din directorul curent care au extensie de 3 caractere
ls -d /??r	afiseaza fisierele din radacina al caror nume are 3 caractere si se termina cu r (usr si var)

- **[** - indica debutul unei constructii de tipul […], ce specifica un set de caractere permise. Constructia […] inlocuieste un singur caracter, insa numai unul dintre cele cuprinse intre parantezele drepte:

ls –l f[12].txt	afiseaza detaliile fisierelor f1.txt si f2.txt din directorul curent
ls –l /poze/poza1[135].jpg	afiseaza detaliile fisierelor poza11.jpg, poza13.jpg si poza15.jpg din directorul /poze

Inauntrul parantezelor drepte pot fi folosite de asemenea:

- intervale de caractere – atunci cand se doreste specificarea unei succesiuni de caractere permise. Exemplu: [bcd] se poate scrie [b-d]
- negari - caracterele ! sau ^, daca apar imediat dupa paranteza dreapta deschisa, au semnificatie de interzicere. Exemplu: [^a-d] sau [!a-d] inseamna "orice alt caracter decat a,b,c sau d".

Atentie! *"Orice alt caracter" inseamna nu doar celelalte litere, ci si cifre, semne de punctuatie etc.!*

- clase de caractere predefinite – sunt de forma [[:nume_clasa:]] si descriu seturi de caractere uzuale. Iata cateva exemple:
  - [[:alpha:]] – litere ale alfabetului

- [[:alnum:]] – caractere alfanumerice
- [[:digit:]] – cifre
- [[:blank:]] – SPACE sau TAB

ls –l /dev/[hs]da[1-4]	afiseaza detaliile fisierelor corespunzatoare partitiilor primare
ls –l /poze/poza[0-9].jpg ls –l /poze/poza[[:digit:]].jpg	listeaza toate fisierele din directorul */poze* al caror nume este *poza* urmat de o cifra si care au extensia jpg
ls –l [[:alpha:]][[:digit:]]??	toate fisierele al caror nume are 4 caractere, primele doua fiind o litera si o cifra

Nota: *metacaracterele descrise mai sus sunt repezinta o facilitate generala a shell-ului si pot fi folosite nu doar pentru ls, ci in cazul oricarei comenzi ce primeste ca argument o cale in sistemul de fisiere.*

3.6. Redirectionarea output-ului comenzilor

Fiecare comanda Linux/Unix dispune de 3 canale de comunicatie cu terminalul din care a fost pornita:

- **input** – conectat din oficiu la tastatura terminalului din care a fost pornita comanda
- **output** – conectat la ecranul terminalului din care a pornit comanda
- **error** – conectat tot la ecranul terminalului

Output-ul normal al comenzii si erorile sunt transmise in Linux/Unix pe canale diferite. In mod normal utilizatorul nu percepe acest lucru, deoarece implicit amandoua canalele sunt trimise catre ecranul terminalului; ele pot fi insa redirectionate in mod independent catre fisiere, permitand capturarea sau eliminarea separata a output-ului sau a erorilor.

Shell-ul Unix ofera urmatoarele posibilitati:

- directionarea output-ului unei comenzi catre un fisier, folosind simbolurile > sau >>:
  - in cazul lui >, daca fisierul nu exista va fi creat, iar daca exista va fi suprascris:

```
ls –lR /etc > fisier1
```

  - in cazul lui >>, daca fisierul nu exista el va fi creat, iar daca exista informatia va fi adaugata la sfarsitul sau, fara a afecta datele deja existente (“append”):

```
ls –lR /etc >> fisier2
```

- directionarea erorilor unei comenzi catre un fisier – se realizeaza folosind aceleasi simboluri si dupa aceleasi reguli, precedend insa simbolurile de redirectionare cu cifra 2 (ce identifica canalul de erori):

```
ls –lR /etc 2> fisier3
ls –lR /etc 2>> fisier4
```

Atunci cand se doreste suprimarea output-ului sau erorilor unei comenzi, respectivul canal poate fi trimis catre fisierul special **/dev/null**:

```
# este eliminat output-ul; eventualele erori raman insa pe ecran
ls /root > /dev/null
# este pastrat pe ecran doar output-ul comenzii; erorile sunt suprimate
ls /root 2>/dev/null
# suprimarea tuturor informatiilor furnizate de comanda in cauza
ls /root >/dev/null 2>&1
# 2>&1 inseamna ca sunt trimise erorile in acelasi fisier ca si output-ul
```

3.7. Inlantuirea mai multor comenzi: caracterul pipe “|”

Principiul liniei de comanda UNIX este acela al comenzilor foarte specializate care conlucreaza usor. Utilizatorul poate prelucra succesiv date, combinand diverse comenzi, prin crearea unor insiruiri de instructiuni separate prin caracterul | ("pipe"):

```
comanda 1 | comanda 2 | comanda 3…
```

Efectul caracterului | consta in conectarea output-ului unei comenzi la input-ul celei imediat urmatoare. In acest fel, ceea ce comanda 1 de mai sus ar fi scris pe ecran este acum pasat ca input comenzii 2; aceasta la randul sau prelucreaza datele si, in loc sa le afiseze pe ecran, le va oferi ca input comenzii 3 etc.

Exemplu: avem un fisier ce contine o lista de nume, si dorim sa obtinem numele care este ultimul din punct de vedere alfabetic. In acest scop, ordonam continutul fisierului si extragem ultima linie:

```
cat fisier_nume | sort | tail -1
```

In exemplul de mai sus, comanda *cat* – care in mod normal ar afisa pe ecran continutul fisierului – il paseaza de aceasta data lui *sort*, care ordoneaza datele alfabetic si apoi trimite lista de nume catre comanda *tail*. Aceasta din urma va afisa pe ecran ultima linie din datele ordonate.

Exista posibilitatea de a trimite simultan output-ul unei comenzi atat catre ecran, cat si catre un fisier, folosind comanda **tee**:

```
ls –l /etc | tee continut_etc
```

Comanda de mai sus listeaza directorul *etc* aflat in radacina sistemului de fisiere, afisand pe ecran rezultatul si in acelasi timp memorandu-l in fisierul *continut_etc*.

3.8. Creare fisiere/directoare

Comenzi introduse: **touch, mkdir**

Comanda *touch* este folosita pentru crearea de fisiere. Ea accepta unul sau mai multe argumente; pentru fiecare dintre ele, comanda are doua efecte posibile:

- daca fisierul nu exista, este creat un fisier gol cu numele specificat
- daca fisierul exista, este modificata doar data ultimei sale accesari

touch file1	creeaza fisierul file1 in directorul curent
touch ../file2	creeaza fisierul file2 in directorul parinte
touch /tmp/file3	creeaza fisierul file3 in directorul /tmp
touch file1 ../file2 /tmp/file3	creeaza dintr-o singura comanda toate cele 3 fisiere de mai sus

Comanda *mkdir* este folosita pentru crearea de directoare. La fel ca si *touch*, poate accepta mai multe argumente:

mkdir dir1	creeaza subdirectorul dir1 in directorul curent
mkdir dir2 ../dir3	creeaza subdirectorul dir2 in directorul curent si subdirectorul dir3 in directorul parinte

Atunci cand se incearca crearea unui director intr-o locatie inexistenta (ex: mkdir /dir1/dir2/dir3, iar dir1 si dir2 nu exista), mkdir va afisa o eroare. Pentru a forta crearea tuturor directoarelor componente ale caii se foloseste optiunea –p ("parent"):

```
mkdir –p /dir1/dir2/dir3
```

3.9. Stergere fisiere/directoare

Comanda introdusa: **rm**

Comanda **rm** este folosita pentru stergerea oricarui tip de fisier (inclusiv directoare); in cazul directoarelor insa este necesara folosirea optiunii –r (recursiv). Atunci cand utilizatorul nu are drept de scriere in fisier, rm prezinta o cerere de confirmare, care poate fi inhibata cu optiunea –f.

Nota: *in Linux, posibilitatea de a sterge un fisier rezulta nu din permisiunile pe acel fisier, ci din permisiunile pe directorul parinte. De aceea, este posibil ca utiliatorul, desi nu are drept de scriere pe un fisier, sa-l poata totusi sterge.*

rm f1	sterge fisierul f1 din directorul curent
rm /tmp/f[23]	sterge fisierele f2 si f3 din directorul /tmp
rm –r /tmp/dir2	sterge directorul dir2 aflat in /tmp si intregul sau continut

Optiuni de interes:

- **-r** (recursiv) – folosita pentru stergerea recursiva a directoarelor
- **-f** (force) – pentru a forta stergerea fisierelor fara afisare de avertizari sau mesaje de confirmare. A se folosi cu precautie!
- **-i** (interactiv) – determina solicitarea unei confirmari de la utilizator inaintea fiecarei stergeri

3.10. Copiere fisiere/directoare

Comanda introdusa: **cp**

Comanda **cp** este folosita pentru copierea fisierelor si directoarelor. Sintaxa sa generala este:

```
cp   optiuni   fisier(e)_sursa   destinatie
```

unde sursa si destinatia pot fi fisiere sau directoare. Distingem urmatoarele cazuri:

- sursa si destinatia sunt fisiere. In acest caz, cp creeaza o copie a fisierului sursa, ale carei locatie si nume sunt cele specificate in cel de-al doilea argument. Atunci cand destinatia exista deja, ea va fi suprascrisa.

```
cp f1.txt f2.txt
cp poza1.jpg /poze/poza4.jpg
```

- sursa este formata dintr-unul sau mai multe fisiere iar destinatia este un director. In acest caz, cp va copia fisierele dedesubtul directorului tinta:

```
cp poza1.jpg poza2.jpg poza3.jpg /poze
```

- sursa este un director. In acest caz, pentru a copia directorul impreuna cu intregul sau continut, este necesara folosirea optiunii –r (recursiv):

```
cp –r /media/cdrom/Pictures /poze/concedii/vara2015
```

Exemplul de mai sus copiaza directorul *Pictures* de pe CD in */poze/concedii*, sub noul nume *vara2015*.

3.11. Mutare/redenumire

Comanda introdusa: **mv**

Comanda *mv* realizeaza mutarea sau redenumirea unuia sau mai multor fisiere. La modul general, mutarea inseamna plasarea unui fisier intr-un alt director, sub un alt nume; redenumirea reprezinta un caz particular de mutare, in care directorul sursa si cel destinatie coincid.

Sintaxa generala a comenzii este:

```
mv sursa destinatie
```

Modul de functionare este identic cu cel al comenzii *cp*, cu exceptia faptului ca nu mai exista optiunea **–r**.

3.12. Vizualizarea completa sau partiala a continutului unui fisier

Comenzi introduse: **cat, tac, less, head, tail**

3.12.1. cat si *tac*

Comanda **cat** primeste ca argument unul sau mai multe fisiere si le afiseaza continutul pe ecran:

```
student@server1 $ cat /etc/hosts
127.0.0.1                localhost
192.168.1.2              server1
```

Nota: *comanda* ***cat*** *nu este practica atunci cand continutul fisierului nu incape pe un singur ecran (textul va "curge in sus" pana la epuizarea sa, pe ecran ramanand in final numai ultima parte).*

Comanda **tac** afiseaza continutul fisierelor primite ca argument, insa in ordinea inversa a liniilor:

```
student@server1 $ tac /etc/hosts
192.168.1.2                server1
127.0.0.1                  localhost
```

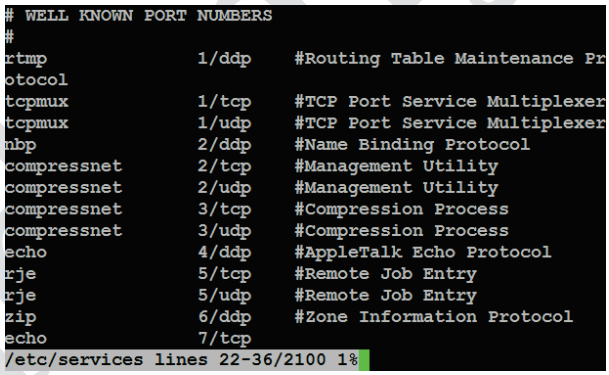
3.12.2. less

Comanda **less** se foloseste in scopul afisarii controlate a unui continut prea extins pentru a incapea pe ecran. O prima forma a sa primeste ca argument fisierul al carui continut se doreste a fi vizualizat, ca mai jos:

```
less /etc/services
```

Comanda less permite parcurgerea continutului in orice directie dorita, cautare de text etc. Cateva taste utile:

- sagetile sus-jos pot fi folosite pentru a parcurge linie cu linie continutul afisat
- **SPACE** afiseaza ecranul urmator
- ***/sir*** deplaseaza pozitia curenta la prima aparitie a sirului de caractere specificat. Daca se doreste gasirea urmatoarei aparitii se poate folosi **n** (next), iar aparitia precedenta este accesata apasand **N**
- **q** este folosit pentru a parasi less ("quit")



O a doua forma de folosire a lui less (poate mai frecventa decat prima) este in capatul unui pipe ce produce un output prea bogat pentru a putea fi afisat pe ecran:

```
ls -l /etc | less
```

In acest exemplu, comanda ls –l /etc produce o lista lunga de fisiere si directoare care, in loc sa fie afisata pe ecran, va fi folosita ca input de catre comanda less, utilizatorul putand astfel parcurge lista dupa voie.

3.12.3. head si tail

Comenzile **head** si **tail** sunt folosite pentru a extrage in mod controlat primele, respectiv ultimele linii ale unui fisier sau output de comanda. Ca si less, pot fi folosite in doua forme: una in care se aplica direct unui fisier, si alta in care sunt folosite ca parte a unui pipe.

Sintaxa pentru operarea pe continutul unui fisier:

```
head optiuni fisier
tail optiuni fisier
```

Sintaxa pentru citirea din standard input:

```
comanda1 | comanda2 | head optiuni | comanda 3 |…
comanda1 | comanda2 | tail optiuni | comanda 3 |…
```

In cea de-a doua forma, head/tail preia output-ul comenzii 2, pastreaza doar primele/ultimele linii si le paseaza comenzii 3 ca input.

Optiuni de interes pentru head:

- **-n** NR – afiseaza primele NR linii din fisier sau output-ul comenzii anterioare
- **-n** -NR – afiseaza toate liniile din fisier/input cu exceptia ultimelor NR linii

Optiuni de interes pentru tail:

- **-n** NR – afiseaza ultimele NR linii din fisier sau output-ul comenzii anterioare
- **-n** +NR – afiseaza ultimele linii din fisier/input incepand cu linia NR
- **-f** – nu se opreste cand atinge sfarsitul fisierului, ci ramane conectat la acesta, afisand noile linii adaugate in timp real. Optiune utila pentru urmarirea fisierelor log (jurnal)

Exemple:

```
student@server1 $ cat nume
Mihai
Elena
Andrei
Ioana
George
student@server1 $ head –n 2 nume
Mihai
Elena
student@server1 $ head –n -2 nume
Mihai
Elena
Andrei
student@server1 $ tail –n 1 nume
George
student@server1 $ tail -n +3 nume | head -n 2
Andrei
Ioana
```

3.13. Analiza continut fisier

Comenzi introduse: **file, wc**

3.13.1. file

Comanda *file* afiseaza informatii despre fisierul primit ca argument, incepand cu tipul acestuia (fisier obisnuit, director, fisier special etc) si continuand cu tipul de continut acolo unde este cazul.

```
student@server1 $ file /etc
/etc: directory
student@server1 $ file /etc/hosts
/etc/hosts: ASCII English text
student@server1 $ file /bin/ls
/bin/ls: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically
linked (uses shared libs), stripped
```

```
student@server1 $ file /usr/share/pixmaps/gnome-gimp.png
/usr/share/pixmaps/gnome-gimp.png: PNG image data, 48 x 48, 8-bit/color RGBA,
non-interlaced
```

3.13.2. wc

Comanda wc ("word count") poate afisa numarul de caractere, cuvinte sau linii dintr-un fisier sau dintr-un output al unei comenzi.

Sintaxa:

```
wc optiuni fisier
comanda1 | wc optiuni
```

Optiuni de interes:

- -w – determina afisarea numarului de cuvinte
- -m - determina afisarea numarului de caractere
- -l – determina afisarea numarului de linii

Exemple:

wc –m /etc/termcap	afiseaza numarul de caractere al fisierului /etc/termcap
ls /etc | wc -l	afiseaza numarul de fisiere din /etc
cat nume1 nume2 | wc -l	afiseaza numarul total de linii din fisierele nume1 si nume2

3.14. Link-uri

3.14.1. Tipuri de link-uri

Prin intermediul link-urilor putem face ca un acelasi fisier (mai exact continutul sau) sa fie vizibil simultan in mai multe locuri din sistemul de fisiere, fara insa a-i multiplica informatia pe hard-disk. Motivele pentru care am dori acest lucru sunt diverse - iata cateva exemple:

- pentru aplicatii instalate in directoare nestandard, putem face fisierele lor de configurare vizibile in /etc, pentru comoditate si o mai buna organizare a configurarii sistemului
- putem crea pe desktop echivalentul shortcut-urilor din Windows - fisiere care ne redirectioneaza catre continut aflat in alte directoare din sistemul de fisiere
- putem face continut de pe o partitie vizibil in cadrul alteia, in cazul in care cea de-a doua nu dispune de spatiul necesar pentru copierea efectiva a informatiei

In Linux exista doua tipuri de link-uri:

- **hard link** - o asociere intre un nume de fisier si un inode. Prin intermediul hard link-urilor, un acelasi continut poate capata mai multe nume, fara a multiplica informatia si fara a crea niciun fisier in plus
- **soft link (symbolic link)** – un fisier care redirectioneaza catre un altul (pe principiul shortcut-urilor din Windows, dar cu functionalitate extinsa). In acest caz avem doua fisiere distincte - fisierul symlink si fisierul ce constituie tinta sa.

Desi cele doua tipuri de link par a realiza acelasi lucru, se va vedea in cele ce urmeaza ca “dedesubturile” lor sunt total diferite: conceptul de hard link este de fapt o veriga vitala pentru functionarea sistemului de fisiere, pe cand cel de symbolic link ofera doar comoditate, nefiind critic.

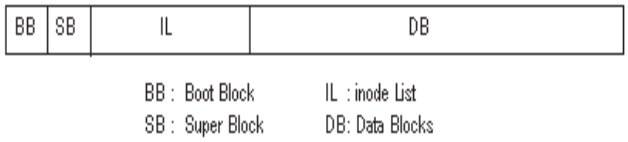
3.14.2. Notiunea de inode

Un fisier nu inseamna numai informatia continuta (text, imagine, sunet etc); sistemul de fisiere trebuie sa memoreze, pe langa datele efective, detalii esentiale despre fisier, cum ar fi:

- tipul fisierului – normal, director, symbolic link etc
- owner-ul său – userID-ul "proprietarului" fisierului
- grupID-ul asociat cu fisierul
- permisiunile - drepturile de citire/scriere/executie asociate owner-ului, grupului si restului lumii
- numarul de hard-link-uri catre fisier
- lungimea sa
- data crearii
- data ultimei modificari
- data ultimei accesari
- lungimea fisierului (numarul de octeti)

Structura care memoreaza aceste informatii se numeste *index node* (pe scurt i-node). Un inode memoreaza toate detaliile unui fisier in afara de numele acestuia. Pe langa lista de mai sus, inode-ul dispune si de o lista a blocurilor de date de pe hard-disk in care este stocat continutul fisierului. Fiecare inode este identificat printr-un numar.

In figura alaturata este prezentata structura bloc a sistemului de fisiere aflat pe o partitie Linux. Elementele ce intervin in figura sunt:



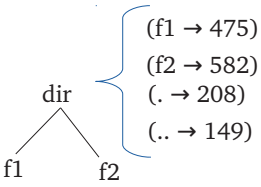
- *Boot Block* – zona in care se afla bootloader-ul (programul raspunzator cu incarcarea sistemului de operare) daca acesta a fost instalat in partitia respectiva
- *Super Block* – contine informatii despre sistemul de fisiere in ansamblul sau (numarul de blocuri de date si dimensiunea acestora, mount point-ul, validitatea sistemului de fisiere etc)
- *Inode List* – lista de inode-uri. Este creata in momentul crearii sistemului de fisiere de pe partitie si contine un numar limitat de structuri de tip inode
- *Data Blocks* – zona in care este stocat continutul fisierelor. Fiecare inode face referire la unul sau mai multe blocuri de date din aceasta zona

Fiecare fisier nou creat consuma un inode din lista. Numarul de inode-uri fiind limitat, aceasta inseamna ca, daca am crea suficient de multe fisiere, am epuiza inode-urile disponibile, moment in care am fi in imposibilitatea de a mai crea fisiere. In practica insa, numarul de inode-uri creat la formatarea partitiei este acoperitor pentru nevoile uzuale si chiar si cele mai putin uzuale.

3.14.3. Hard link-uri

S-a mentionat anterior faptul ca inode-ul unui fisier contine toate informatiile despre acel fisier cu exceptia numelui. Daca stam sa ne gandim, numele nu ar fi un element necesar pentru sistemul de operare; daca acestuia i-am indica inode-ul 254, el ar avea toate informatiile necesare despre fisier, inclusiv continutul acestuia! In schimb, conceptul de nume al unui fisier ii este indispensabil utilizatorului uman - si deci infrastructura de nume este practic o adaugire la acest sistem de evidenta a fisierelor bazat pe identificatori numerici care este cel de inode-uri.

Numele fisierelor sunt memorate in Linux astfel: fiecare fisier de tip director contine lista de nume ale fisierelor din directorul in cauza, fiecare nume fiind pus in corespondenta cu un inode. O astfel de asociere nume-inode poarta denumirea de **hard link**; asadar, un fisier de tip director reprezinta o lista de hard-link-uri corespunzatoare fisierelor continute.



Dupa cum se observa in figura alaturata, directorul *dir* contine doua fisiere, avand insa un total de 4 hard link-uri: doua corespund fisierelor *f1* si *f2*, al treilea directorului curent iar ultimul directorului parinte. Aceste doua ultime hard-link-uri sunt prezente in toate directoarele si sunt cele care ne permit sa folosim simbolurile . si .. in cadrul cailor relative.

Utilizatorul Linux poate crea multiple hard-link-uri pentru acelasi fisier - practic, mai multe nume asociate cu acelasi inode. In acest fel, acelasi continut de pe hard-disk va putea fi accesat prin mai multe nume - fara a consuma spatiu suplimentar sau inode-uri in plus. Spre exemplu, daca un fisier are numele */dir1/fisier1* si cream un al doilea hard link */dir2/nume2*, cele doua nume vor fi perfect echivalente - continutul fisierului va exista intr-un singur exemplar pe hard-disk si va putea fi modificat prin intermediul oricaruia dintre cele doua nume; nici nu mai conteaza care dintre ele a existat primul.

Comanda folosita pentru crearea unui hard link este **ln**, ce are urmatoarea sintaxa**:**

```
ln nume_deja_existent nume_nou
```

Exemplu:

```
ln /etc/hosts /tmp/statii
```

Numarul de hard-linkuri ale unui fisier poate fi evidentiat cu ajutorul comenzii **ls –l:**

```
student@server:$ ls -l /tmp/statii
-rw-r--r-- 2 root root 226 apr 23  2014 /tmp/statii
```

Atentie! *Hard link-urile nu pot depasi granitele de partitie - toate numele unui fisier trebuie sa se afle in aceeasi partitie! Aceasta limitare provine din faptul ca numerele de inode se refolosesc de la o partitie la alta.*

3.14.4. Symbolic link-uri

Un **soft link** (numit si *symbolic link*, pe scurt *symlink*) este un fisier care redirectioneaza in mod transparent catre altul; el ocupa un inode propriu, diferit de al fisierului tinta. Este o versiune imbunatatita a shortcut-urilor din Windows. Se creaza tot cu comanda **ln** dar cu optiunea **–s**:

```
ln –s fisier_deja_existent fisier_shortcut
```

Exemplu:

```
ln –s /usr/local/apache/bin/apachectl /bin/apachectl
```

Atentie! Un symlink poate fi creat si cu o cale relativa, insa trebuie tinut cont de faptul ca acea cale va fi considerata relativa la directorul in care se afla symlink-ul, nu la cel curent din momentul crearii lui! Exemplu:

```
student@server$ pwd
/home/student
student@server$ ln -s Desktop/Carte.txt /tmp
student@server$ ls -l /tmp/Carte.txt
lrwxrwxrwx 1 u1 g1 14 iun 25 09:20 /tmp/Carte.txt -> Desktop/Carte.txt
student@server$ cat /tmp/Carte.txt
cat: /tmp/Carte.txt: No such file or directory
#...deoarece fisierul este cautat in /tmp/Desktop/Carte.txt !
```

3.15. Cautarea fisierelor

3.15.1. Abordari

Exista doua abordari ale cautarii fisierelor in sistemele de operare moderne:

- cautarea efectuata direct in sistemul de fisiere. Are avantajul acuratetii maxime, insa poate consuma mult timp in cazul cautarii intr-un director foarte bogat in informatie. In Linux, aceasta abordare este folosita de comanda **find**
- mentinerea unei baze de date (actualizata periodic sau la cerere) cu resursele continute in sistemul de fisiere. Cautarea se efectueaza in aceasta baza de date, avand avantajul unei viteze sporite, insa neoferind acuratete maxima (spre exemplu, nu vor fi gasite fisierele create dupa ultima actualizare a bazei de date). Aceasta abordare este cea oferita de comenzile Linux **locate** sau **slocate**

3.15.2. Comanda find

Comanda *find* cauta direct in sistemul de fisiere, in toata adancimea directorului primit ca argument (cautare recursiva). Ea permite specificarea unuia sau mai multor criterii de cautare sub forma unei expresii. Setul de criterii posibile este foarte bogat. Sintaxa comenzii este:

```
find  cale  expresie
```

Iata cateva exemple de criterii elementare:

- **-name** nume – permite filtrarea rezultatelor dupa numele fisierului. Se admit si wildcard-uri (ex: -name “*jpg”)
- **-type** tip – cautare dupa tipul de fisier. *Tip* poate fi *f* pentru fisier obisnuit, *d* pentru director, *l* pentru symlink etc.
- **-size** dimensiune – filtrare dupa marimea fisierului. Dimensiunea poate fi exprimata in octeti sau in multipli (ex: 10M, 5G). Daca se doreste o dimensiune mai mare decat un anumit prag, marimea fisierului va fi precedata de caracterul +; pentru o dimensiune mai mica, se va folosi – (exemplu: +3M, -200k). *Atentie! Pentru kilobytes se foloseste k (litera mica) iar pentru megabytes M (litera mare)!*
- **-user** user – cautare dupa proprietarul fisierului (owner)

Criteriile pot fi asociate folosind SI logic (–a - implicit) si SAU logic (–o), putandu-se astfel forma astfel expresii complexe pentru filtrarea rezultatelor produse de find.

Exemple:

```
# cautarea tuturor fisierelor obisnuite cu dimensiune intre 100k si 2MB
find / -type f –size +100k -size -2M

# cautarea fisierelor mp3 din /MUZICA care il au ca owner pe andrei
find /MUZICA –type f –name “*mp3” –user andrei
```

3.15.3. Comenzile locate/slocate

Comenzile *locate/slocate* cauta intr-o baza de date actualizata periodic sau la cerere, care contine lista de fisiere gasite in sistem. In comparatie cu find, cautarea in aceasta baza de date este mult mai rapida, insa acuratetea poate avea de suferit, iar setul de criterii de cautare este mult mai redus. Sintaxa comenzii este:

```
locate fragmentnume
```

Exemplu:

```
student@server$ locate updatedb
/etc/updatedb.conf
/etc/alternatives/updatedb
/usr/bin/updatedb
/usr/bin/updatedb.mlocate
/usr/share/man/man5/updatedb.conf.5.gz
/usr/share/man/man8/updatedb.8.gz
```

In cazul lui slocate, baza de date este de asa natura alcatuita incat cautarea sa fie sigura - ea este criptata incremental (format binar) si contine permisiuni/ownership, astfel incat userii sa nu poata gasi fisiere la care nu ar avea acces in mod normal.

Baza de date este de obicei actualizata periodic in distributiile Linux (perioada default este de o zi sau mai putin) insa poate fi adusa la zi la cerere ruland una dintre urmatoarele comenzi:

```
slocate –u
...sau...
updatedb
(ultimul este valabil atat pentru locate cat si pentru slocate)
```

Baza de date se gaseste in */var/lib/slocate/slocate.db* sau */var/lib/mlocate/mlocate.db.* Crearea sa este guvernata de fisierul /etc/updatedb.conf care specifica directoarele si tipurile de sisteme de fisiere ce trebuie excluse de la indexare.

3.16. ANEXA 1: Filtrare si validare de text folosind regular expressions

3.16.1. Concepte

Regular expressions (regex, pe scurt) reprezinta o solutie de a descrie formatul unei familii de siruri de caractere. Sa luam, spre exemplu, cautarea intr-un fisier: daca dorim sa gasim sirul "InfoAcademy", orice viewer sau editor de text ne va oferi aceasta facilitate; surpriza nu va fi ceea ce vom gasi (caci stim de la bun inceput sirul exact cautat) ci doar numarul de aparitii si locul acestora. Prin contrast, daca vrem sa gasim toate adresele de mail dintr-un fisier, de aceasta data nu mai cunoastem sirul cautat ci doar formatul lui (o succesiune de litere, cifre si punct, urmata de caracterul @ si de o alta succesiune de litere, cifre si punct etc); odata ce avem o solutie pentru specificarea formatului, noutatea va consta nu numai in numarul si locul aparitiilor adreselor de mail, ci si in continutul efectiv al sirurilor gasite - nu stim de la bun inceput *ce* vom gasi ci doar *cum arata* (in linii mari) sirurile cautate. In orice caz, nu mai vorbim neaparat de un sigur sir, ci de o intreaga familie ce se incadreaza in acel format.

Regex-urile au diferite utilizari, atat in Linux cat si in multe alte domenii; iata cateva:

- cautarea intr-un fisier a unor informatii carora le cunoastem doar formatul sau compozitia aproximativa (ex: adrese IP, nume DNS)
- filtrarea output-ului bogat al unei comenzi Linux, pastrand doar informatiile de interes
- filtre definite pe servere Linux (ex: filtre de spam pe serverele de mail, filtre pentru site-uri/continut nepotrivit pe serverele web sau proy web etc)
- editoarele de text avansate (indiferent de sistemul de operare) dispun de o facilitate de search&replace bazata pe regex-uri
- in programare, validarea informatiilor primite de la utilizator se realizeaza in foarte multe cazuri folosind regex-uri

Nota: *in traducere mot-a-mot, "regular expressions" inseamna "expresii uzuale", intelesul fiind de fapt acela de siruri de caractere uzuale, cu format binecunoscut (adresa IP, numar de masina, cod IBAN etc). Asadar traducerea dupa ureche "expresii regulate", folosita din pacate pe scara larga, este nu numai hilara ci si incorecta.*

3.16.2. Clase de caractere

Fie cazul unui sir de caractere binecunoscut - seria si numarul de buletin din Romania. Formatul sau poate fi descris in cuvinte astfel: "doua litere mari urmate de sase cifre", sau, mai exact:

- primul caracter este de tip litera mare
- al doilea caracter este de tip litera mare
- al treilea caracter este de tip cifra
- etc

Observam ca una dintre necesitatile de baza in a specifica formatul unui sir este aceea de a putea preciza, la nivel de *fiecare caracter in parte*, care este setul de caractere permis pe acea pozitie. Practic, in cazul extrem, pe fiecare pozitie din sirul cautat putem avea o multime permisa de caractere. O astfel de multime poarta denumirea de **clasa de caractere.**

Clasele de caractere se specifica cu ajutorul unor constructii speciale, ce presupun metacaractere. Conceptul nu este nou; sa ne amintim ca in linia de comanda Linux putem deja specifica familii de siruri de caractere: a*.txt, img[0-9].jpg etc. Trebuie facute insa doua mentiuni: in primul rand, regex-urile sunt mult mai bogate in

posibilitati decat facilitatile de shell prezentate in materialele anterioare; in al doilea rand, in regex-uri exista metacaractere fie in plus fata de shell, fie identice dar cu semnificatie diferita. A nu se confunda facilitatea de globbing din shell cu regex-urile!

Iata modalitatile de a specifica caracterul/caracterele dorite pe o anumita pozitie dintr-o familie de siruri:

- un caracter obisnuit, altul decat metacaracterele **^.[$()|*+?{** , se reprezinta pe el insusi (altfel spus, cautand sirurile corespunzatoare expresiei *Infoacademy* vom gasi chiar acest sir de caractere, deoarece nu exista metacaractere in cadrul regex-ului)
- constructii speciale si metacaractere:
  - simbolul . (punct) corespunde unui singur caracter (oricare, dar unul singur!). Spre exemplu, expresiei *an.a* ii corespund in particular sirurile *anca* si *anda*, dar si *an#a*, *an>a* etc.
  - caracterul \ joaca rolul de *escape character*; pus in fata unui meta-caracter el ii elimina acestuia semnificatia speciala. Exemplu: expresia *30\.15* va descrie sirul *30.15*
  - un sir nevid de caractere inclus in paranteze drepte (“[ ]”) tine loc de un singur caracter si reprezinta o multime de caractere permise. Aceasta constructie este o forma de a face “sau” intre caracterele continute (ex: [bf]an corespunde sirurilor *ban* si *fan*). In cadrul constructiei:
    - caracterul “-” (minus”) poate fi folosit pentru a indica un interval de caractere (ex: *[b-d]ar* selecteaza sirurile *bar*, *car*, *dar*).
    - caracterul “^” se reprezinta pe sine (nu este metacaracter), cu exceptia cazului cand se gaseste imediat dupa [caz in care are semnificatia de negare (ex: *[^ab]* inseamna orice alt caracter in afara de *a* sau *b*)
    - pot fi folosite clase de caractere predefinite (vezi manpage) folosind constructia [[:nume_clasa:]]; exemplu: expresia *[[:upper:]][[:lower:]]* selecteaza formatiunile de doua litere in care prima este litera mare si a doua mica

Asa cum un sir este format din caractere, un regex corespunzator unei familii de siruri poate fi format dintr-o succesiune de regex-uri elementare de tip clasa de caracter. Iata mai jos exemple:

Regex	Siruri ce corespund	Comentarii, explicatii
[flp]in	fin, lin, pin	constructia [flp] corespunde unui singur caracter, si anume unuia dintre cele specificate in interiorul parantezelor
[r-t]ara	rara, sara, tara	constructia [r-t] corespunde unui singur caracter, mai exact unuia dintre cele aflate intre r si t (r,s,t)
[Aa]n[cd]a	anca,anda,Anca si Anda	constructia cu paranteze drepte poate interveni in mai multe locuri
[br-tv]	bara, rara, sara, tara, vara	pot fi combinate primele doua modalitati. Indiferent de cate caractere se afla inauntrul parantezelor, constructia tine loc de un singur caracter (in acest caz, acela poate fi b,r,s,t sau v)
[A-Z][a-z]	orice cuvant de doua litere care incepe cu litera mare	constructia [] poate fi folosita de oricate ori este nevoie in cadrul unui regex
[A-Z][^a-z]	orice sir de doua litere care incepe cu litera mare si are pe pozitia a doua orice altceva decat litera mica (ex: C#, A4 etc)	[^a-z] corespunde unui singur caracter, care nu poate fi litera mica. **Atentie!** Negarea unei litere sau a unui set de litere inseamna ca pe pozitia respectiva se pot gasi cifre, semne de punctuatie etc
.in	*fin, lin, pin* dar si *#in, %in* etc	. inseamna orice caracter, inclusiv semne de punctuatie, cifre etc
[r-t]a.a	*rara, sara, tara* dar si *rata, raba, tata, ta(a, sa@a etc*	punctul poate fi combinat cu oricare dintre celelalte constructii
[A-Z].	o litera mare urmata de orice caracter (ex: *Am, Nu, F&, H** etc)	
[[:digit:]]-[[:digit:]]	un scor la fotbal: *3-2, 1-0* etc	cifra, minus, cifra. Regex-ul nu ar functiona pentru scoruri ce depasesc 9 (ex: 10-2, 15-40)
B [[:digit:]][[:digit:]] [A-Z][A-Z][A-Z]	un numar de automobil de Bucuresti (ex: B 13 RTG)	caracterul B, spatiu, doua cifre, spatiu, 3 litere mari.

3.16.3. Repetitii

Putem specifica repetitia controlata a unui caracter, clase de caractere sau a unei intregi subexpresii folosind doua constructii:

- constructia cu acolade:

```
caracter{min,max} – caracterul (sau clasa de caractere) se repeta intre min si
max ori
(regex){min,max} – intreaga expresie se repeta intre min si max ori
```

Sunt permise variatiuni ale acestei sintaxe:

```
(regex){n} – repetitie de exact n ori
(regex){min,} – repetitie de minim n ori, fara limita superioara a numarului de
repetitii
```

Observatie: *atunci cand dorim sa punem conditia de repetare a unei parti a regexului ce contine mai multe caractere, acea subexpresie trebuie inclusa intre paranteze rotunde; daca e vorba de un singur caracter, parantezele sunt optionale*

Exemple:

Regex	Siruri ce corespund	Comentarii, explicatii
Mwa(ha){2,3}	Mwahaha, Mwahahaha	literele *Mwa* urmate de doua sau trei repetitii ale grupului *ha*
07\d{8}	un numar de mobil (ex: *0720123456*)	caracterele 0 si 7 urmate de 8 cifre
[A-Z]{2}[0-9]{6}	serie si numar de buletin (ex: *VF735245)*	o litera mare care se repeta de exact 2 ori, urmata de o cifra care se repeta de exact 6 ori
[A-Z][a-z]{0,1}	un element chimic (ex: *H, Na* etc)	o litera mare, urmata de o litera mica ce apare o data sau deloc
[A-Z]{1,2} [0-9]{2,3} [A-Z]{3}	un numar de automobil (ex: *B 35 EDX* sau *MM 67 WSL*)	una sau doua litere mari, un spatiu, doua sau trei cifre, un spatiu si apoi trei litere mari
[A-Z][a-z]{1,}	un cuvant care incepe cu litera mare (ex: Marius)	o litera mare urmata de una sau mai multe litere mici

- metacaractere folosite pentru cazuri de repetitii particulare

```
(regex)+ - echivalent cu (regex){1,} (minim o repetitie a lui regex)
(regex)* - echivalent cu (regex){0,} (0 sau mai multe repetitii ale lui regex)
(regex)? – echivalent cu (regex){0,1} (regex apare o data sau deloc)
```

Exemple:

Regex	Siruri ce corespund	Comentarii, explicatii
[A-Z][a-z]+\.	o propozitie (ex: Am un mar.)	o litera mare urmata de una sau mai multe litere mici sau spatii si terminandu-se cu un punct
[a-z]+\.[a-z]+@[a-z]\.[a-z]{2-4}	o adresa de mail de forma victor.manu@gmail.com	o litera mica ce se repeta minim o data, un caracter . (observati \-ul), o litera mica care se repeta cel putin o data, caracterul @, din nou o succesiune de litere mici (cel putin una) apoi punct si numele domeniului radacina (com, net, org, info, tv etc)

3.16.4. Ancorare

Exista posibilitatea ca, pe langa specificarea formatului sirurilor de caractere cautate/validate, sa le fie impusa acestora pozitia in cadrul liniei dupa cum urmeaza:

- *^regex* – impune ca sirurile ce corespund regex-ului sa se afle la inceput de linie
- *regex$* - impune ca sirurile ce corespund regex-ului sa se afle la final de linie
- *^regex$* - sirurile descrise de regex trebuie sa ocupe toate linia

Fie urmatorul text pe mai multe linii:

```
mere
pere
prune
capere
```

In aceste conditii:

- a cauta in acest text folosind regex-ul *e* (simpla litera e) va gasi toate aparitiile acestei litere, 7 la numar
- cautand sirurile ce corespund regex-ului *e$* vor fi gasite 4 e-uri – cele aflate la final de linie
- cautand dupa regex-ul *p* vom gasi 3 aparitii
- regex-ului ^*p* ii vor corespunde doar 2 aparitii – literele *p* aflate la inceput de linie

3.16.5. Formate alternative

Uneori, familia de siruri pe care incercam sa o descriem are, pe unele portiuni sau in integralitatea sa, formate atat de diferite incat nu pot fi “prinse” cu un singur regex. In aceste conditii este necesara specificarea unor formate posibile, indicand faptul ca oricare dintre ele este permis.

In cadrul unui regex, caracterul special | permite specificarea unor formate alternative pentru sirul de caractere cautat/validat sau pentru o portiune a sa:

```
regex1 | regex2   caracterul | are rol de "sau"
inceput_regex (regex1|regex2) sfarsit_regex   portiunea de mijloc poate avea doua
formate
```

Exemple:

Regex	Siruri ce corespund	Comentarii, explicatii
Mari(e|oara)	Marie, Marioara	finalul sirului cautat are doua formate posibile
(021|07[0-9])[0-9]{7}	un numar de Bucuresti (fix) sau de mobil	021 urmat de 7 cifre, sau 07 urmat de 8 cifre

3.16.6. Comenzi care folosesc regex-uri

3.16.6.1. Familia de comenzi grep

Familia de comenzi grep (GNU regular expression parser) este formata din urmatorii membri:

- **grep** - comanda “originara”, care implicit interpreteaza regex-ul primit ca pe un Basic Regular Expression. Aceste tipuri de expresii reprezinta o forma mai veche, rudimentara, fata de ceea ce a fost prezentat in acest material, suferind prin comparatie de o intreaga serie de limitari (spre exemplu, caracterele *+?| nu vor functiona ca metacaractere, iar toate parantezele/acoladele vor trebui precedate de caracterul escape “\”)
- **egrep** (echivalent cu **grep -E**) - comanda ce interpreteaza regex-ul primit ca fiind un Extended Regular Expression. Acesta este tipul de expresii prezentat in materialul de fata
- **fgrep** (echivalent cu grep -F) - "fast grep". Comanda realizeaza cautarea clasica, exacta, fara regex-uri - sirul de caractere primit ca argument este cautat ca atare; niciunul dintre caracterele componente nu are semnificatie speciala
- **rgrep** (echivalent cu grep -r) - realizeaza cautare recursiva (vezi mai jos optiuni)

Prezentam in continuare comanda egrep (din care putem obtine fgrep cu egrep -F, sau rgrep cu egrep -r).

Comanda **egrep** realizeaza filtrarea input-ului primit folosind regex-uri si afiseaza **liniile complete** pe care este gasit un sir de formatul specificat, chiar daca acesta nu ocupa toata linia. Input-ul poate proveni:

- dintr-un fisier pasat ca argument comenzii
- din input-ul primit de la comanda anterioara din cadrul unui pipe. Cel mai adesea comanda *egrep* este utilizata ca parte a unui pipe, pentru restrangerea output-ului altor comenzi sau pentru a gasi anumite siruri in acel output.

Sintaxa comenzii este:

```
# pentru input preluat din fisier
egrep     [optiuni]  regex fisier

# pentru input preluat de la comanda anterioara
comanda1 | comanda 2 | egrep    [optiuni]  regex
```

Iata cateva optiuni utile ale comenzii *egrep*:

-i determina o procesare case-insensitive a input-ului (literele mici vor fi considerate egale cu cele mari)
-n fiecare linie afisata este precedata de numarul sau din fisier (prima linie este 1)
-v inversiune; sunt afisate doar liniile care NU corespund regex-ului
-r realizeaza o cautare recursiva in toate fisierele din directorul primit ca argument (echivalent cu rgrep)
-o afiseaza numai sirurile ce au corespuns regex-ului, nu liniile integrale pe care acestea se gasesc

Tabelul alaturat contine cateva exemple de aplicare a comenzilor grep. Exemplele se folosesc de fisierul *f1* ce are urmatorul continut:

f1

Democratia,
dictatura majoritatii

Comanda	Output
egrep “d” f1	dictatura majoritatii
egrep -ni “d” f1	1:Democratia, 2:dictatura majoritatii
egrep -v “d” f1	Democratia,
egrep -v “m” f1	*nu produce output*
egrep -o “e.” f1	em
fgrep “e.” f1	*nu produce output*

3.16.6.2. Comanda sed

Comanda **sed** este una complexa, cu capabilitati multiple; dintre ele vom aminti aici inlocuirea de text pe baza de regex-uri. Comanda poate actiona asupra unui fisier, sau ca parte a unui pipe; in functie de aceasta, sintaxa de utilizare difera:

```
# efectuarea de modificari in cadrul unui fisier
sed -ri 's/regex/inlocuitor/g' fisier

# modificari operate asupra output-ului comenzii precedente
comanda1 | comanda2 | … | sed -r 's/regex/inlocuitor/g'
```

Comanda identifica sirurile ce corespund regex-ului specificat si le inlocuieste cu cel de-al doilea sir.

Optiunea **-r** face ca sed ca interpreteze regex-urile ca fiind de tip extended, iar **-i** face ca modificarile sa fie produse direct in fisier (in mod normal sed ar afisa rezultatul procesarii pe ecran).

Exemplele din tabelul urmator folosesc fisierul *f1* din sectiunea anterioara:

Comanda	Output/continut fisier
sed -ri 's/D/d/g' f1	democratia,

	dictatura majoritatii
sed -ri 's/^d/aparent d/g' f1	Democratia, aparent dictatura majoritatii
cat f1|sed 's/Demo/Biro/g'|sed 's/majo/medioc/g'	Birocratia, dictatura mediocritatii