GHID DE UTILIZARE LINUX (IV)

Lucrul la linia de comandă în UNIX, partea a III-a:

Interpretoare de comenzi UNIX, partea I – prezentare generală

Cristian Vidrașcu
cristian.vidrascu@info.uaic.ro

Martie, 2025

Sumar

Introducere

Într-un sistem `UNIX`, interpretorul de comenzi este un program ce îndeplinește aceleași sarcini ca și în `MS-DOS` sau `Windows`, oferind două funcționalități de bază:

- preia comenzile introduse de utilizator, le interpretează și le execută, realizând astfel interfața dintre utilizator și sistemul de operare;
- oferă facilități de programare într-un limbaj de comandă specific, cu ajutorul căruia se pot scrie *script*-uri, *i.e.* fișiere text ce conțin secvențe de comenzi `UNIX`.

Observație: în sistemele de operare din familia `UNIX` avem la dispoziție mai multe interpretoare de comenzi (denumite uneori și *shell*-uri, de la termenul folosit în limba engleză), precum ar fi: `sh` (*Bourne SHell*), `bash` (*Bourne Again SHell*), `csh` (*C SHell*), `ksh` (*Korn SHell*), `ash`, `zsh`, ș.a.

Principala caracteristică prin care se diferențiază interpretoarele între ele este sintaxa limbajului de comandă specific fiecăruia dintre ele. În plus, *shell*-urile disponibile în sistemele `UNIX` sunt mai puternice decât interpretoarele de comenzi din `MS-DOS` și `Windows` (`command.com`, respectiv `cmd.exe`), oferind limbaje de comandă asemănătoare cu limbajele de programare de nivel înalt d.p.d.v. al sintaxei: au structuri de control alternative și repetitive (de genul `if`, `case`, `for`, `while`, etc.), ceea ce permite scrierea de programe complexe ca simple fișiere cu secvențe de comenzi (*i.e.*, *script*-uri).

Vom prezenta în continuare facilitățile comune tuturor *shell*-urilor `UNIX`, prin care se realizează prima sarcină pomenită mai sus, cu referiri explicite la sintaxa utilizată de interpretorul `bash`.

Agenda

4 / 29

Comenzi simple – definiție

Comenzile simple sunt "componentele" individuale ce pot fi "asamblate" în comenzi compuse, folosind anumiți *operatori sintactici de compunere*, ce vor fi descriși în secțiunea următoare.
Însă, pentru început, să ne reamintim faptul că în sistemele de operare din familia `UNIX` există două categorii de *comenzi simple*:

- *Comenzi interne*: sunt implementate în interpretoarele de comenzi.
  Exemple: `cd`, `help`, ș.a.
- *Comenzi externe*: sunt implementate de sine stătător (*i.e.*, se găsesc fiecare în câte un fișier, având același nume cu comanda respectivă), în:
  - fișiere executabile (*i.e.*, programe executabile obținute prin compilare din programe sursă scrise în C sau în alte limbaje de programare).
    Exemple: `passwd`, `ls`, ș.a.
  - fișiere text cu secvențe de comenzi, numite *script*-uri.
    Exemple: `.profile`, `.bashrc`, ș.a.

5 / 29

Modul de execuție a comenzilor simple

Forma generală de lansare în execuție a unei comenzi simple, internă sau externă:

`UNIX>` *`numele_comenzii`* `[`*`opțiuni`*`]` `[`*`argumente`*`]` `[`*`redirectări I/O`*`]`

Observații:

- Textul "`UNIX>` " este *prompterul* afișat de interpretorul de comenzi, la care acesta așteaptă să tastați comanda dorită, urmată de apăsarea tastei `ENTER`.
  Notă: textul afișat ca și prompter este configurabil – vom discuta ulterior cum anume.
- Drept caracter separator între cuvintele din linia de comandă se utilizează `SPACE` sau `TAB`.
- Opțiunile și/sau argumentele specificate după numele comenzii pot eventual lipsi (*i.e.*, sunt opționale, lucru indicat printr-o pereche de paranteze '`[...]`' ; acestea nu trebuie tastate).
- Prin convenție, opțiunile sunt precedate de caracterul '-' (sau '`--`' , în cazul opțiunilor lungi).
- Semnificația argumentelor depinde de comandă (*e.g.*, cel mai adesea sunt nume de fișiere).
- Comenzile externe pot fi specificate și prin numele complet (*i.e.*, *calea absolută sau relativă*) al fișierului respectiv.
- O comandă mai lungă (*i.e.*, cu mulți parametri) poate fi introdusă pe mai multe linii, caz în care fiecare linie trebuie terminată cu caracterul '\' urmat de `ENTER`, cu excepția ultimei linii (terminarea acesteia se face apăsând doar tasta `ENTER`).
- Despre ce înseamnă redirectări I/O vom discuta ulterior, tot în această prezentare.

Modul de execuție a comenzilor simple (cont.)

O altă posibilitate de a lansa în execuție o comandă simplă, valabilă numai pentru comenzi externe ce sunt *script*-uri, este prin apelul unui anumit *shell*:

`UNIX> bash` *`script`* `[`*`opțiuni`*`]` `[`*`argumente`*`]` `[`*`redirectări I/O`*`]`

În acest caz, secvența de comenzi din fișierul cu numele *script* va fi executată, într-o sesiune de lucru **neinteractivă**, de către o instanță a *shell*-ului invocat pe prima poziție din linia de comandă (în acest exemplu, *shell*-ul `bash`).

Observație: dacă nu se specifică niciun script după numele *shell*-ului invocat pe prima poziție din linia de comandă, se va crea o nouă sesiune de lucru **interactivă**, controlată de o instanță de execuție a *shell*-ului specificat.

Iar o a treia posibilitate, tot numai pentru comenzi externe ce sunt *script*-uri, este următoarea, cu două forme sintactice echivalente:

`UNIX> .` *`script`* `[`*`opțiuni`*`]` `[`*`argumente`*`]` `[`*`redirectări I/O`*`]`

`UNIX> source` *`script`* `[`*`opțiuni`*`]` `[`*`argumente`*`]` `[`*`redirectări I/O`*`]`

Execuția comenzilor simple în *background*

În cazul formelor de lansare în execuție descrise anterior, spunem că acea comandă simplă este executată în *foreground* (*i.e.*, în "planul din față"), deoarece interpretorul așteaptă terminarea execuției acelei comenzi și abia apoi reafișează prompterul, oferind utilizatorului posibilitatea să introducă o nouă comandă pentru execuție.

O altă manieră de execuție a comenzilor ar fi în *background* (*i.e.*, în "planul din spate"), adică interpretorul să nu mai aștepte terminarea execuției acelei comenzi, ci să reafișeze imediat prompterul, oferindu-i astfel utilizatorului posibilitatea să introducă imediat o nouă comandă pentru execuție. Comenzile executate în *background* mai sunt denumite și *comenzi asincrone*.

Sintactic, specificarea modului de execuție în *background* a unei comenzi se face adăugând caracterul '`&`' la sfârșitul liniei de comandă:

`UNIX>` *`numele_comenzii`* `[`*`opțiuni`*`]` `[`*`argumente`*`]` `[`*`redirectări I/O`*`]` `&`

Notă: în cazul comenzilor externe ce sunt *script*-uri, se poate adăuga '`&`' la finalul oricăreia dintre cele trei forme de lansare în execuție specificate anterior.

8 / 29

Redirectări I/O

Există trei dispozitive logice de I/O standard:

- *intrarea standard* (`stdin`), de la care se citesc datele de intrare în timpul execuției unei comenzi (prin funcțiile de I/O ce accesează *file-descriptorul* `0`);
- *ieșirea normală standard* (`stdout`), la care sunt scrise datele de ieșire în timpul execuției unei comenzi (prin funcțiile de I/O ce accesează *file-descriptorul* `1`);
- *ieșirea de eroare standard* (`stderr`), la care sunt scrise mesajele de eroare în timpul execuției unei comenzi (prin funcțiile I/O ce accesează *file-descriptorul* `2`).

În mod implicit, dispozitivul logic `stdin` este atașat dispozitivului fizic tastatură (*i.e.*, terminalul de intrare), iar dispozitivele logice `stdout` și `stderr` sunt atașate dispozitivului fizic ecran (*i.e.*, terminalul de ieșire).

Însă, interpretorul de comenzi poate "forța" o comandă ca, pe parcursul execuției sale, să primească datele de intrare dintr-un fișier specificat, în locul citirii lor de la tastatură, precum și să trimită datele de ieșire și/sau mesajele de eroare într-un fișier specificat, în locul afișării lor pe ecran.

9 / 29

Redirectări I/O (cont.)

Această "forțare" poartă numele de *redirectarea* fluxului (sau fluxurilor) I/O respective, și este valabilă doar pentru acea execuție a comenzii și, mai ales, fără să fim nevoiți să facem modificări în codul sursă al comenzii și s-o recompilăm!

Specificarea redirectărilor I/O se face folosind următoarea sintaxă:

- *redirectarea intrării standard* (`stdin`):
  `UNIX> numele_comenzii [parametri] < fișier_intrare`
- *redirectarea ieșirii normale standard* (`stdout`), în modul *rewrite* vs. *append*:
  `UNIX> numele_comenzii [parametri] > fișier_iesire`
  `UNIX> numele_comenzii [parametri] >> fișier_iesire`
- *redirectarea ieșirii de eroare standard* (`stderr`), în modul *rewrite* vs. *append*:
  `UNIX> numele_comenzii [parametri] 2> fișier_iesire_err`
  `UNIX> numele_comenzii [parametri] 2>> fișier_iesire_err`

Iată un exemplu, având ca efect concatenarea conținuturilor primelor două fișiere în cel de-al treilea:
`UNIX> cat fis1 fis2 > fis3`

Se pot specifica mai multe redirectări într-o comandă, evaluarea lor efectuându-se de la stânga spre dreapta. În plus, există și sintaxa `n>&m`, unde `n` și `m` sunt *file-descriptori*. Iată un exemplu:
`UNIX> ls -l .bashrc un_fișier_inexistent >listing.txt 2>&1`

Valoarea de exit (codul de terminare)

Valoarea de exit (în engleză, *exit status*) a unei comenzi simple executate de către o instanță de *shell* în *foreground* este valoarea returnată de apelul de sistem `waitpid` (sau alt apel echivalent), prin care acea instanță a interpretorului de comenzi așteaptă terminarea execuției acelei comenzi. (Pentru o comandă asincronă, valoarea este `0`.)

Valoarea de exit a unei comenzi simple (precum și a comenzilor compuse și, respectiv, a comenzilor interne ale acelui *shell*) este întotdeauna o valoare întreagă din intervalul `0` - `255`, însă interpretorul de comenzi poate folosi, în circumstanțe speciale, valori peste `125` pentru a indica moduri specifice de eșec al execuției comenzii respective (a se vedea slide-ul următor).

Valoarea de exit a *ultimei comenzi executate* este disponibilă în variabila specială `$?` a instanței de *shell* ce a executat-o.

Pentru scopurile interpretorului, o comandă care se termină cu valoarea de exit zero a reușit: o valoare de exit zero indică *succesul* execuției comenzii. O valoare de exit diferită de zero indică *un eșec* al execuției comenzii.

Valoarea de exit (cont.)

Circumstanțe speciale de eșec :

- Când o comandă se termină datorită unui semnal fatal `N`, interpretorul `bash` folosește valoarea `128+N` ca valoare de exit.
- Dacă o comandă nu este găsită, procesul copil creat pentru a o executa returnează valoarea `127`.
- Dacă o comandă este găsită, dar nu este executabilă, starea returnată este `126`.
- Dacă o comandă eșuează din cauza unei erori în timpul expansiunii sau redirectării I/O, valoarea de exit este mai mare decât zero.

Comenzile interne ale unui *shell* returnează valoarea de exit `0` (*true*) dacă au succes și, respectiv, o valoare diferită de zero (*false*) dacă apare o eroare în timpul executării lor. Toate comenzile interne returnează valoarea de exit 2 pentru a indica utilizarea incorectă (*i.e.*, opțiuni în general nevalide sau argumente lipsă).
Când execuția unei instanțe a interpretorului `bash` se termină, ea returnează valoarea de exit a ultimei comenzi executate, cu excepția cazului în care apare o eroare de sintaxă la interpretarea acelei comenzi, caz în care se termină cu o valoare diferită de zero.

Comenzi compuse

Agenda

Comenzi compuse – definiție

Două sau mai multe comenzi simple se pot "grupa" într-o *comandă compusă*, prin scrierea lor pe un singur rând (la prompterul liniei de comandă sau într-un *script*), separate prin următorii operatori sintactici de "compunere" a comenzilor simple:

- operatorul ';' – *execuția secvențială* a mai multor comenzi
- operatorul '|' – *execuția paralelă, înlănțuită* a mai multor comenzi simple
- operatorul '&' – *execuția paralelă, neînlănțuită* a mai multor comenzi
- operatorii '&&' și '||' – *execuția condițională* de comenzi

Semantica (*i.e.*, semnificația) fiecăruia dintre acești operatori de compunere este descrisă sumar în cele ce urmează. *Avertisment*: descrierea se axează pe sintaxa folosită de interpretorul de comenzi `bash`; în cazul altor *shell*-uri `UNIX`, sintaxa comenzilor compuse s-ar putea să difere (*i.e.*, să se utilizeze alți operatori de "compunere", etc.).
Notă: pentru o descriere detaliată, vă recomand consultarea documentației oficiale a *shell*-ului `bash`, disponibilă aici.

14 / 29

Lanțuri de comenzi simple

Înlănțuirea mai multor comenzi simple, într-un așa-numit *lanț de comenzi* (sau *pipeline*), se realizează folosind simbolul '|' (*i.e.*, caracterul *pipe*), astfel:
`UNIX>` `[time]` *comanda_1* `|` *comanda_2* `| ... |` *comanda_N*
Simbolul '|' marchează "conectarea" ieșirii normale standard a unei comenzi la intrarea standard a comenzii următoare din lanțul de comenzi; comunicația între cele două comenzi se face printr-un canal de comunicație anonim (astfel se elimină necesitatea comunicării prin intermediul unor fișiere temporare, precum se întâmplă în `MS-DOS` sau `Windows`).
Cuvântul cheie `time` are drept efect afișarea timpilor de execuție a lanțului.
Notă: este permis și $N = 1$, adică un lanț format dintr-o singură comandă simplă.

Modul de execuție a unui lanț de comenzi:
Toate comenzile simple din lanțul respectiv sunt executate simultan, în "același" timp (*i.e.*, în paralel, și nu secvențial una după alta!), fiecare comandă fiind executată de către o nouă instanță de execuție a *shell*-ului. Practic, se creează procese multiple, câte un proces pentru fiecare comandă din lanț.

15 / 29

Lanțuri de comenzi simple (cont.)

Iată câteva exemple de lanțuri de comenzi:
UNIX> `ls -Al | wc -l`
Efect: afișează numărul total de fișiere de orice tip (inclusiv subdirectoare) aflate în directorul curent.
UNIX> `who | cut -f1 -d" " | sort -u`
Efect: afișează lista ordonată a numelor utilizatorilor conectați la sistem.
UNIX> `cat /etc/passwd | grep -w so`
Efect: ?

Demo: a se vedea toate exemplele de lanțuri de comenzi din suportul de laborator, disponibil aici.
Primul exemplu descrie și efectul folosirii cuvântului cheie `time`.

Sintaxa extinsă: în locul oricărui simbol '`|`', dintr-un lanț cu $N \geq 2$, se poate utiliza perechea '`|&`', care este o prescurtare pentru '`2>&1 |`'. Prin aceasta se realizează "conectarea" ambelor ieșiri standard (și cea normală, și cea de eroare) a unei comenzi la intrarea standard a comenzii următoare din lanțul de comenzi.

Valoarea de exit: *shell*-ul așteaptă ca toate comenzile dintr-un lanț să se termine înainte de a returna o valoare ; valoarea returnată de lanțul de comenzi este valoarea de exit a ultimei comenzi din lanț.
Observație: dacă opțiunea `pipefail` este activată, valoarea de exit a lanțului este valoarea de exit a ultimei comenzi (*i.e.*, cea mai din dreapta) care se termină cu un eșec, sau zero dacă toate comenzile se termină cu succes.

Execuția secvențială a mai multor comenzi

Mai multe comenzi (simple sau lanțuri cu $N \geq 2$) pot fi separate prin caracterul '`;`' și vor fi executate secvențial, de către aceeași instanță de execuție a *shell*-ului respectiv (*i.e.*, sunt executate una după alta, în ordinea în care apar ; *shell*-ul așteaptă fiecare comandă să se termine înainte de a o executa pe următoarea).
Sintaxa folosită (unde fiecare *comanda_i* este un lanț de comenzi simple) :
UNIX> *comanda_1 ; comanda_2 ; ... ; comanda_N*

Echivalent, cele N comenzi se pot scrie fiecare pe câte un rând, la prompter sau într-un *script*, astfel:
comanda_1
comanda_2
...
comanda_N

Valoarea returnată de o secvență de comenzi este valoarea de exit a ultimei comenzi executate.

Iată și două exemple:
UNIX> `ls -A ; cd Desktop ; ls -l`
UNIX> `mkdir d1 ; echo "Salut!" > d1/f1.txt ; cd d1 ; stat f1.txt`
Efectele acestor comenzi : ?

Execuția paralelă, neînlănțuită a mai multor comenzi

Mai multe comenzi (simple sau lanțuri cu $N \geq 2$) pot fi separate și prin caracterul '&' și vor fi executate (aproape) simultan, fiind lansate pe rând în execuție în *background* (*i.e.*, fără a se aștepta, pe rând, terminarea fiecăreia și fără a avea fluxurile I/O standard "înlănțuite" prin *pipe*, precum la lanțurile de comenzi). Practic, fiecare comandă este executată de către o nouă instanță de execuție a *shell*-ului și rulează în mod independent de celelalte (*i.e.*, fără "înlănțuiri" între ele).
Sintaxa folosită (unde fiecare *comanda_i* este un lanț de comenzi simple) :
UNIX> *comanda_1* & *comanda_2* & . . . & *comanda_N* [&]

Observație: ultima comandă va fi rulată fie în *background* (dacă este terminată cu caracterul '&'), fie în *foreground* (în caz contrar). *Întrebare*: care va fi valoarea de exit returnată, în fiecare caz ?

Echivalent, cele N comenzi se pot scrie fiecare pe câte un rând, la prompter (dar astfel "defazajul" de start al lor va crește, depinzând de viteza cu care le introduceți de la tastatură) sau într-un *script*, astfel:
comanda_1 &
comanda_2 &
. . .
comanda_N [&]

Iată și un exemplu:
UNIX> cat /etc/passwd & cat /etc/group &
Efect: ?

Execuția condițională a mai multor comenzi

Execuția unei comenzi poate fi condiționată de rezultatul execuției unei alte comenzi.
Sintaxa folosită (unde *comanda_1* și *comanda_2* sunt lanțuri de comenzi simple) :

- operatorul '&&' – *"conjuncția" a două comenzi* (lista AND) :
  UNIX> *comanda_1* && *comanda_2*
  Modul de execuție: mai întâi se execută prima comandă, iar apoi se va executa a doua comandă numai dacă execuția primei comenzi se *termină cu succes* (*i.e.*, întoarce codul de terminare 0).
- operatorul '||' – *"disjuncția" a două comenzi* (lista OR) :
  UNIX> *comanda_1* || *comanda_2*
  Modul de execuție: mai întâi se execută prima comandă, iar apoi se va executa a doua comandă numai dacă execuția primei comenzi *eșuează* (*i.e.*, întoarce un cod de terminare nenul).

Observații:
i) Se poate remarca analogia cu evaluarea scurt-circuitată a expresiilor logice booleene.
ii) Valoarea de exit a listelor AND și OR este valoarea de exit a ultimei comenzi executate din listă.
iii) Sunt permise și secvențe formate din mai mult de două comenzi separate prin operatorii '&&' și '||', ce au aceeași precedență, iar pentru execuția lor se va aplica asociativitatea la stânga a operatorilor.

Sintaxa extinsă: liste de comenzi și comenzi compuse

O *listă de comenzi* este o secvență de una sau mai multe *pipelines* (*i.e.*, lanțuri de comenzi simple), separate între ele prin operatorii ';', '&', '&&', sau '||', și opțional terminată cu unul dintre caracterele ';', '&', sau <newline>.

Modul de execuție a unei *liste de comenzi*: se evaluează secvența de lanțuri de la stânga la dreapta, ținând cont de ordinea dată de *precedența operatorilor*: '&&' și '||' au aceeași precedență, mai mare decât cea a operatorilor ';' și '&', care de asemenea au aceeași precedență. Apoi se execută fiecare lanț din listă, în ordinea astfel stabilită, aplicând pentru fiecare operator semnificația sa.

Observație: valoarea de exit a unei liste de comenzi se obține pe baza regulilor specificate pentru valoarea de exit în cazul fiecărui operator. Iată și câteva exemple:

UNIX> `stat /etc/passwd | grep Uid && ls -l /non-existent-folder || echo "Ok"`

```
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
ls: cannot access '/non-existent-folder': No such file or directory
Ok
```

UNIX> `uname -srv && date ; cat non-existent-file || echo "Ok"`

```
Linux 6.8.0-53-generic #55-Ubuntu SMP PREEMPT_DYNAMIC Fri Jan 17 15:37:52 UTC 2025
Tue Feb 18 12:25:30 EET 2025
cat: non-existent-file: No such file or directory
Ok
```

UNIX> `pwd || date ; stat non-existent-file && echo "Ok"`

```
/home/vidrascu
stat: cannot statx 'non-existent-file': No such file or directory
```

UNIX> `pwd || { date ; stat non-existent-file ; } && echo "Ok"`

Efectul acestei comenzi: ?

Sintaxa extinsă: liste de comenzi și comenzi compuse (cont.)

Două dintre tipurile de *comenzi compuse* sunt următoarele : `(list)` și `{ list; }` .
i) *Modul de execuție* a unei *comenzi compuse* `(list)` : se creează un *subshell* (*i.e.*, o nouă instanță a interpretorului respectiv) care va executa lista de comenzi `list`.
ii) *Modul de execuție* a unui *grup de comenzi* `{ list; }` : lista de comenzi `list` se va executa de către instanța curentă de *shell*.

Notă: valoarea de exit a fiecăreia dintre cele două comenzi compuse este valoarea de exit a listei `list`.
Iată 2 exemple :

```
UNIX> ps -f ; var=outside ; (ps -f ; var=inside) ; echo "list_exit=$?, var=$var"
```

```
UID          PID    PPID  C STIME TTY          TIME CMD
vidrascu    2158    2109  0 15:32 pts/0    00:00:00 bash
vidrascu    2194    2158  0 15:36 pts/0    00:00:00 ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
vidrascu    2158    2109  0 15:32 pts/0    00:00:00 bash
vidrascu    2195    2158  0 15:36 pts/0    00:00:00 bash
vidrascu    2196    2195  0 15:36 pts/0    00:00:00 ps -f
list_exit=0, var=outside
```

```
UNIX> ps -f ; var=outside ; { ps -f ; var=inside; } ; echo "list_exit=$?, var=$var"
```

```
UID          PID    PPID  C STIME TTY          TIME CMD
vidrascu    2158    2109  0 15:32 pts/0    00:00:00 bash
vidrascu    2202    2158  0 15:37 pts/0    00:00:00 ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
vidrascu    2158    2109  0 15:32 pts/0    00:00:00 bash
vidrascu    2203    2158  0 15:37 pts/0    00:00:00 ps -f
list_exit=0, var=inside
```

Alte tipuri de *comenzi compuse* (a se vedea cursul următor) : structurile de control alternative `if`, `case` și repetitive `for`, `select`, `while` și `until` , precum și expresiile aritmetice `((arithmetic-expression))` și condiționale `[[ conditional-expression ]]` .
Observație: pentru o descriere mai detaliată a lanțurilor de comenzi, a listelor de comenzi și a tipurilor de comenzi compuse vă recomand consultarea documentației, disponibilă aici.

Agenda

Specificarea fișierelor individuale

Specificarea numelui unui fișier oarecare, fie ca argument pentru comenzi, fie drept nume al unei comenzi externe, se poate face în trei moduri diferite:

- prin *cale absolută* (*i.e.*, numele complet, pornind de la directorul rădăcină)
  Exemplu: `/home/vidrascu/so/file0003.txt`
- prin *cale relativă* la directorul curent de lucru (*i.e.*, pornind din directorul curent)
  Exemplu (presupunând că directorul curent de lucru este `/home/vidrascu`) : `so/file0003.txt`
- prin *calea relativă la directorul home al unui anumit utilizator* (*i.e.*, pornind din directorul *home* al acestuia)
  Exemplu: `~vidrascu/so/file0003.txt`
  Observații: i) dacă lipsește numele utilizatorului (*e.g.*, `~/so/file0003.txt`), atunci se consideră directorul *home* al utilizatorului curent;
  ii) acest al treilea mod de specificare se poate folosi doar la linia de comandă sau în *script*-uri, dar NU și ca argumente ale funcțiilor apelate în programe C.

Șabloane pentru specificarea multiplă de fișiere

Se poate specifica o listă de fișiere, ca argumente pentru comenzi, utilizând un singur "șablon" de specificare multiplă, cu ajutorul următoarelor *caractere cu interpretare specială*:

- caracterul '*' : se înlocuiește cu orice șir de caractere, inclusiv șirul vid
  Exemplu: `~vidrascu/so/file*.txt`
- caracterul '?' : se înlocuiește cu orice caracter (exact un caracter)
  Exemplu: `~vidrascu/so/file000?.txt`
- specificatorul "mulțime precizată de caractere" [...] : se înlocuiește cu exact un caracter, dar nu cu orice caracter posibil, ci doar cu cele specificate între parantezele '[' și ']', sub formă de enumerare (separate prin ',' sau nimic) și/sau interval (dat prin capetele intervalului, separate prin '-')
  Exemple: `~vidrascu/so/file000[1,3,579].txt`,
  `~vidrascu/so/file00[0-9][3-9].txt`,
  `~vidrascu/so/file000[1-3,57-9].txt`.

24 / 29

Șabloane pentru specificarea multiplă de fișiere (cont.)

(*continuare*)

- specificatorul "mulțime exclusă de caractere" [^...] : se înlocuiește cu exact un caracter, dar nu cu orice caracter posibil, ci doar cu cele din complementara mulțimii specificate între parantezele '[' și ']' similar ca mai sus, exceptând faptul că primul caracter de după '[' trebuie să fie '^' pentru a indica complementariere
  Exemplu: `~vidrascu/so/file000[^1-3].txt`
- caracterul '\' : se folosește pentru a inhiba interpretarea operator a caracterelor speciale anterioare, și anume \c (unde c este unul dintre caracterele '*', '?', '[', ']', '^', '\') va interpreta acel caracter c ca text obișnuit (*i.e.*, prin el însuși) și nu ca operator (*i.e.*, prin "șablonul" asociat lui în felul descris mai sus)
  Exemple: `ce_mai_faci\?.txt` , `lectie\[lesson].txt`.

Important: fiecare astfel de șablon este înlocuit, în poziția din linia de comandă în care apare, cu lista tuturor numelor de fișiere existente ce satisfac acel șablon, dar numai dacă lista respectivă este nevidă. În caz contrar, șablonul rămâne nemodificat în urma interpretării sale.

Notă: se mai pot specifica și *șabloane negate*. Un exemplu ar fi: `ls !(*.sh)`

25 / 29

Agenda

26 / 29

Inițializarea sesiunilor interactive ale *shell*-ului

Interpretoarele de comenzi din `UNIX` pot folosi anumite fișiere ce stochează diverse comenzi de configurare a sesiunilor interactive de execuție a *shell*-ului respectiv, numele acestor fișiere fiind specifice acestuia.
Astfel, în cazul interpretorului de comenzi `bash`, se folosesc următoarele fișiere ([3]) :

- *fișiere de inițializare* a sesiunilor `bash` de *login* :
  - un *script* global, executat pentru toți utilizatorii : `/etc/profile`
  - *script*-uri locale, specifice utilizatorului respectiv : `~/.bash_profile`, `~/.bash_login` sau `~/.profile`
- *fișier de finalizare* a acestor sesiuni – doar un *script* local : `~/.bash_logout`
- *fișiere de inițializare* a sesiunilor `bash` interactive, cu excepția celor de *login* :
  - un *script* local, specific utilizatorului respectiv : `~/.bashrc`
  - un *script* global, executat pentru toți utilizatorii : `/etc/bashrc` (însă acesta trebuie apelat explicit din *script*-ul local)

27 / 29

Istoricul comenzilor executate

- *fișier de istoric* al comenzilor introduse și executate la linia de comandă `bash` – un fișier local, cu numele `~/.bash_history`

Conținutul acestui fișier poate fi vizualizat (și) cu comanda `history`.
Numerele afișate de această comandă pot fi folosite pentru a executa în mod repetat comenzile din istoric, fără a fi nevoie să mai fie re-tastate. Și anume, pentru a invoca din nou o comandă, se tastează la prompterul liniei de comandă caracterul '!' urmat imediat de numărul asociat comenzii respective.

O altă facilitate utilă a interpretorului `bash`, ce permite invocarea comenzilor tastate anterior, precum și editarea lor, constă în navigarea prin istoric prin apăsarea, la prompter, a tastelor `UP` și `DOWN` (*i.e.*, tastele săgeată-sus și săgeată-jos). După selectarea din istoric a liniei de comandă dorite, ne putem deplasa în cadrul ei cu tastele săgeată-stânga și săgeată-dreapta pentru a modifica parțial linia respectivă (*e.g.*, modificăm, ștergem sau adăugăm vreun parametru al comenzii) și apoi apăsăm tasta `ENTER` pentru a se executa comanda astfel modificată.

Referințe bibliografice

Bibliografie obligatorie

[1] Cap. 2, §2.3 din cartea "*Sisteme de operare – manual pentru ID*", autor C. Vidrașcu, editura UAIC, 2006. *Notă*: este accesibilă, în format PDF, din pagina disciplinei "Sisteme de operare":

- `https://edu.info.uaic.ro/sisteme-de-operare/SO/books/ManualID-SO.pdf`

[2] Suportul de laborator online asociat acestei prezentări:

- `https://edu.info.uaic.ro/sisteme-de-operare/SO/support-lessons/bash/suport_lab3.html`

Bibliografie suplimentară:

[3] Documentația interpretorului de comenzi `bash` : `man 1 bash` și "GNU Bash manual"

[4] *Linux Documentation Project Guides* → "Bash Guide for Beginners"

[5] Cartea "Bash Pocket Reference" (1st edition), by A.Robbins, O'Reilly Media Inc., 2010.