Instrumente si Tehnici de Baza in Informatica

Semestrul I 2025-2026

Vlad Olaru

Curs 3 - outline

- interpretorul de comenzi
- fisiere si directoare (revizitat)

Interpretorul de comenzi (recapitulare)

- mod de lucru interactiv (comanda-raspuns) sau batch (automatizarea lucrului cu scripturi)
- in mod interactiv, afiseaza un prompt
  - indica faptul ca se asteapta o comanda (interna/externa) de la utilizator
  - uzual, $ sau % pt utilizatori obisnuiti, # pt root
  - definit de continutul variabilei de mediu PS1

$ echo $PS1

  - prompt de continuare in variabila de mediu PS2, uzual >

$ cat << EOF >> out

> mai adaugam o line la sfarsitul fisierului out

> EOF

$

Mediul de lucru (environment)

- lista de perechi *name* = *value*
- *name* este numele unei variabile interne a shell-ului
  - Obs: nu orice variabila interna a shell-ului este variabila de mediu
- valorile variabilelor de mediu influenteaza comportamentul shell-ului, respectiv al comenzilor externe lansate de acesta
  - eg: shell (PS1/2, PATH), gcc (CPATH, LIBRARY_PATH), ld (LD_LIBRARY_PATH), samd
  - setate de SO sau utilizator
- numele scris cu litere mari: PS1, SHELL, HOME, PATH, etc
- valoarea dereferentiata cu ajutorul simbolului $, eg

$SHELL = /bin/bash

Variabile de mediu

- setate cu comanda interna *export*
  - marcheaza variabila ca fiind variabila de mediu

$ export PS1="my-new-prompt> "

- afisate cu comanda */usr/bin/env*
- intr-un program C, accesibile in al treilea parametru al functiei *main* a programului lansat in executie de catre shell

int main(int argc, char argv[], char *envp[])*

The Bourne-Again SHell

- */bin/bash*, urmasul primului shell istoric, */bin/sh* (Bourne Shell)
- fisiere de configurare
  - fisiere de start-up inspectate doar la login
    - system-wide: */etc/profile*
    - locale, in home directory: *~/.profile, ~/.bash_profile, ~/.bash_login*
    - continutul lor e executat automat la login
  - fisiere de start-up inspectate la crearea fiecarui terminal (rc file, *run commands file*)
    - *~/.bashrc*
    - continutul lor poate fi executat voluntar cu comanda *source* (sau ".")

      source .bashrc

      . .bashrc
  - fisier de logout: *~/.bash_logout*
    - continutul executat la iesirea din shell (cu *exit* sau Ctrl-d)
    - Obs: Ctrl-d in Unix este caracterul EOF, tiparirea Ctrl-d la prompt termina shell-ul
- istoria comenzilor inregistrata in *~/.bash_history*

Structura comenzilor bash

- pipeline-uri

  $ cmd_1 | cmd_2 | … |cmd_n # executie paralela a comenzilor

  $ cmd_1 |& cmd_2 |& … |& cmd_n # "|&" e totuna cu "2>&1 |"

- liste de comenzi

  $ cmd_1; cmd_2; …; cmd_n # executie secventiala a comenzilor

  $ cmd_1 && cmd_2 && … && cmd_n # executie conditionata a comenzilor

  $ cmd_1 || cmd_2 …. || cmd_n

- variabila bash "?" contine codul de terminare (*exit status*) al ultimei comenzi executate (valoarea zero inseamna succes)

  $ *echo $?*

Obs: *?* nu e variabila de mediu !

Job control

- doua categorii de programe
  - executate in foreground, au acces R/W la terminal
  - executate in background
- comanda incheiata un “&” ruleaza in background
  - shell-ul returneaza imediat utilizatorului promptul

$ cmd &

- executia unei comenzi in foreground se poate suspenda cu ^Z (Ctrl-z)
  - de fapt, e semnalul SIGTSTP (*kill –SIGTSTP <pid>*)
  - executia comenzii poate fi reluata ulterior, fie in foreground, fie in background
- comanda *jobs* listeaza procesele (joburile) rulate la momentul curent de shell
  - joburile identificate prin numar
  - numarul job-ului poate fi folosit impreuna cu urmatoarele comenzi

$ kill %n	# termina procesul/job-ul cu nr *n*
$ fg %n	# muta in foreground procesul cu nr *n*
$ bg %n	# muta jobul *n* in background
$ %n &	# muta jobul *n* in background

Controlul istoriei comenzilor

- *~/.bash_history*
- exemple

!n	re-executa comanda cu nr *n*
!-n	re-executa comanda curenta – *n*
!string	re-executa cea mai recenta comanda care incepe cu *string*
!?string?	re-executa cea mai recenta comanda care contine *string*
^str1^str2	repeta comanda anterioara inlocuind *str1* cu *str2*

- interactiv: *Ctrl-r* urmat de un substring al comenzii cautate din istoric

Comenzi

- *interne*: executate direct de catre bash

  cd <dir>

  alias l='ls –l'

  fg/bg/kill <job#>

  exit <status> # termina shell-ul cu cod de retur *status*

  exec <cmd> # inlocuieste imaginea bash cu imaginea noului proces

  # de ex: $ exec firefox

- *externe*: programe de pe disc lansate de catre shell

  pwd

  echo string

  ex escape chars:

  echo –e \a # bell

  echo –e "aaa\tbbb" # horizontal tab

  echo –e "aaa\v\bbbb" # vertical tab + backspace

  echo –e "aaa\t\rbbb" # carriage return

  echo –e "aaa\t\nbbb" # newline

Tipuri de fisiere

- *fisiere obisnuite* (*regular files*): contin date (text sau binare)
- *directoare*: contin numele altor fisiere si informatii despre ele
  - pot fi citite de catre procesele care au permisiunile potrivite
  - DOAR kernelul poate scrie in ele !
- *fisiere speciale, tip device*
  - *caracter*: pt device-uri caracter (ex: tty, seriala)
  - *bloc*: pt device-uri orientate pe bloc (ex: discuri)
  - operatiile de R/W nu se fac prin intermediul FS ci al driverelor

  Obs: orice device (echipament) din sistem e fie fisier bloc, fie caracter
- *FIFO*: *named pipe*, mecanism IPC (Inter-Process Communication)
  - | s.n. *anonymous pipes*, leaga procese *inrudite*
  - conecteaza procese fara legatura
  - fisiere de pe disc, cu nume si politica de acces FIFO
  - bidirectionale, spre deosebire de |

Tipuri de fisiere (cont.)

- *socket*: abstractie pentru IPC peste retea
  - canal de comunicatie local (socket Unix, un fel de FIFO)
  - canal de comunicatie intre masini conectate in retea (socket TCP/IP)
- *link simbolic*: fisier care refera un alt fisier
  - practic fisierul destinatie (*link-ul simbolic*) contine numele fisierului sursa (fisierul referit)

  $ ln –s <fisier-sursa> <link-simbolic>
- formatul lung al comenzii *ls* marcheaza in primul caracter tipul fisierului:

  -, d, c, b, p, s, l
- comanda generala, distinge si tipuri de fisiere regulate (text, executabile, imagini, etc):

  $ file <nume-fisier>

Set UID, set GID

- fiecare proces (program in executie) are asociat
  - UID, GID real: identitatea reala a utilizatorului provenita din */etc/passwd*
  - UID, GID efectiv
  - set-UID, set-GID salvate (copii ale UID/GID efectiv)
- in mod normal, UID/GID real = UID/GID efectiv
- cand se executa un program exista posibilitatea de a seta un flag in atributele fisierului executabil a.i.:

> "pe durata executiei acestui program UID/GID efectiv al procesului devine UID/GID-ul proprietarului fisierului program"

Ex: comanda de schimbare a parolei utilizator

> *$ passwd*
>
> */usr/bin/passwd* este un program *set-UID* la *root* pt a avea drepturi de scriere in */etc/shadow*

Permisiuni de acces la fisiere

- atribute ale fisierului
- grupate in trei categorii
  - permisiuni utilizator: S_IRUSR, S_IWUSR, S_IXUSR
  - permisiuni grup: S_IRGRP, S_IWGRP, S_IXGRP
  - permisiuni pt. alti utilizatori: S_IROTH S_IWOTH, S_IXOTH
  - permisiuni speciale: set-uid, set-gid, sticky bit
- modificabile din shell cu ajutorul comenzii *chmod*

  ex: *chmod u+rw <fisier>, chmod g-x <fisier>, chmod o-rwx <fisier>, etc*

  sau in octal

  chmod 755 <fisier>, chmod 644 <fisier> , etc
- accesul la un fisier: conditionat de combinatia dintre UID efectiv (respectiv GID efectiv) al comenzii executate si bitii de permisiune

umask

- orice fisier nou creat are setata o masca implicita a permisiunilor
  - setata cu comanda *umask* (comanda interna shell)

$ umask 022

- bitii setati in *umask* sunt off in permisiunile noului fisier creat
- de regula, masca setata in fisierele de configurare shell (eg, */etc/profile*)

Stergerea fisierelor

- un fisier poate avea m.m. *link-uri* la aceeasi structura interna din kernel (*i-node*)
  - nume diferite ale aceluiasi fisier
  - s.n. *link-uri hard*
  - create cu comanda *ln*

$ ln <fisier-sursa> <fisier-destinatie>

  - stergerea unui link nu inseamna stergerea fisierului de pe disc !
  - stergerea ultimului link sterge si fisierul
- pt. a sterge o intrare de fisier dintr-un director se foloseste *rm*

$ rm <nume-fisier>

- stergerea necesita doua permisiuni:
  - permisiunea de a scrie in director
  - permisiunea de a cauta in director (bitul *x* de executie setat in directorul din care stergem fisierul)

Link-uri simbolice

- limitari *link-uri hard*
  - link-ul si fisierul linkat trebuie sa se afle pe acelasi sistem de fisiere (disc formatat)
  - doar *root-ul* poate crea linkuri hard catre directoare
- *link simbolic*: fisier care contine numele fisierului referit (un string)
- utilizate pentru a circumventa limitarile link-urilor hard
- create cu comanda *ln* si flag-ul *–s*

$ ln –s /etc/profile ~/.system-wide-profile

- in general, comenzile shell dereferentiaza linkul simbolic
  - exceptii: *lstat*, *remove*, *rename*, *unlink*, samd
  - ex: pt. linkul simbolic creat mai sus

```
$ rm ~/.system-wide-profile      # sterge link-ul simbolic, nu sursa, adica /etc/profile
$ ls –l ~/.system-wide-profile   # afiseaza atributele linkului symbolic, nu ale sursei
$ cat ~/.system-wide-profile     # afiseaza continutul /etc/profile
```

Lucrul cu directoare

- create cu *mkdir*, sterse cu *rmdir*

  $ mkdir <director>

  $ mkdir –p </director>/<subdirector> # creeaza si directoarele
  # inexistente on the fly

  $ rmdir <director>

- Obs: *rmdir* nu poate sterge un director decat daca e GOL !
- schimbarea directorului curent

  $ cd <director> # schimba directorul in <director>

  $ cd # ⇔ cd $HOME

  $ cd - # schimba directorul curent in directorul anterior

- aflarea directorului de lucru curent (current working directory)

  $ pwd