

# Instrumente si Tehnici de Baza in Informatica

Semestrul I 2025-2026

Vlad Olaru

# Outline

- pornirea sistemului (procesul de boot)
- procesul de login utilizator
- interfata cu utilizatorul
- fisiere si directoare

# Bootarea sistemului

- la pornirea calculatorului, executia incepe intr-un loc fix din memorie
- SO trebuie sa fie facut disponibil HW ca sa-l poata porni
  - o mica bucata de cod – **bootstrap loader**, **BIOS**, stocat in **ROM** sau **EEPROM** localizeaza kernelul, il incarca in memorie si il porneste
  - uneori e un proces in doi pasi, utilizand un bloc de boot aflat la o adresa fixa in codul ROM, care apoi incarca bootstrap loader-ul de pe disc
  - sistemele moderne inlocuiesc BIOS cu **Unified Extensible Firmware Interface (UEFI)**
- un bootstrap loader uzual este **GRUB**, permite selectia kernelului de pe discuri multiple, cu versiuni si optiuni diferite
- programul kernel se incarca si apoi sistemul ruleaza
- boot loader-erele permit frecvent diferite stari de boot, cum ar fi de pilda single user mode

# Procesul de boot Unix

- primul sector al discului de boot (MBR, respectiv succesorul sau GPT)
  - tabela de partitii de disc
  - cod de bootstrap (boot loader)

```
Disk /dev/sda: 500GB
Sector size (logical/physical): 512B/4096B
Partition Table: msdos
Disk Flags:
```

Number	Start	End	Size	Type	File system	Flags
1	1049kB	1574MB	1573MB	primary	ntfs	boot
2	1574MB	211GB	209GB	primary	ntfs	
4	211GB	481GB	270GB	extended		
6	211GB	465GB	254GB	logical	ext4	
5	465GB	481GB	16.4GB	logical	linux-swap(v1)	
3	481GB	500GB	18.9GB	primary	ntfs	

# Procesul de boot Unix (cont.)

- loader-ul identifica partitia de boot si incarca codul kernel (nucleul sistemului de operare)
- obs: la acest nivel nu exista notiunea (abstractia) de fisier, ci doar sectoare de disc => 2 solutii posibile
  - 1. loader-ul cunoaste o harta a sectoarelor de disc care contin codul kernel
    - solutie hardcoded, implica actualizari alei hartilor atunci cand imaginea kernelului pe disc se schimba, la defragmentarea discului, etc
  - 2. loader-ul are acces la drivere care inteleg structura sistemului de fisiere de pe disc si pot identifica astfel kernelul ca pe un fisier oarecare (folosind calea fisierului)
- ex boot loaders Linux: Lilo, Grub

# Procesul de boot (cont.)

- fisierul cu imaginea kernelului (eg, */boot/vmlinuz* pt Linux) se incarca in memorie si kernelul preia controlul masinii HW
- subsecvent, kernelul executa:
  - secventa de intializare a componentelor HW
  - instantiaza principalele componente: controlul proceselor, gestiunea memoriei, gestiunea fisierelor, accounting, gestiunea timpului sistem, mecanismele de protectie HW si de securitate, etc
  - ramane rezident in memorie in asteptarea unor evenimente externe (“program interrupt-driven”)
  - la sfarsitul secventei de initializare executa primul proces (ID = 1): */sbin/init*
  - *init* seteaza modul de operare (*runlevel*)
    - defineste starea masinii de calcul dupa boot
    - traditional definit de un numar intre 0 si 6
    - istoric (sistemele Unix), *init* cauta runlevel-ul in */etc/inittab*
    - apoi, apeleaza scripturi de initializare a serviciilor sistem cf. runlevelului selectat  
*/etc/rc0.d/, /etc/rc1.d, ..., /etc/rc6.d, /etc/rcS.d*

# Runlevels

- asignate modului de operare al masinii
  - 0 , power-off
  - 1, single-user mode
  - 2, multi-user fara retea
  - 3, multi-user cu retea dar fara interfata grafica
  - 4, in general nedefinit, rezervat pt utilizari speciale
  - 5, multi-user cu retea si interfata grafica
  - 6, reboot
- Linux: *init* s.n. *systemd*, iar runlevel-urile sunt definite ca *targets*, manipulate cu comenzi specifice (*systemctl*)
- comenzi care manipuleaza runlevels:

\$ runlevel	# afiseaza runlevelul current, similar cu “who -r”
\$ telinit <runlevel>	# comuta kernelul in runlevelul specificat
\$ telinit 6	# reboot

# Sisteme cu sau fara GUI

- *init* este responsabil si pt. pornirea proceselor de login pt utilizator:
  - in functie de runlevel: */sbin/getty* respectiv desktop manager-ul de interfete grafice de tip X Window (*xdm*, *gdm*, etc)
  - runlevel 3: *init* porneste *getty* pe un numar prestabilit de terminale
  - runlevel 5: *init* porneste *getty* + desktop manager
  - istoric (sistemele Unix), *init* cauta in */etc/inittab* asocierea terminal – program de login (*getty* sau *xdm/gdm*)
- comutarea sistemului intre runleveluri cu sau fara interfata grafica

\$ telinit 3            # dezactiveaza GUI

\$ telinit 5            # activeaza GUI inapoi

Obs: combinatii de taste (gen Ctrl-Alt-F1 in Linux) permit comutarea intre terminale si GUI (uzual, Ctrl-Alt-F7) in runlevel 5, dar nu se dezactiveaza GUI !



# Logarea utilizatorului in sisteme fara GUI

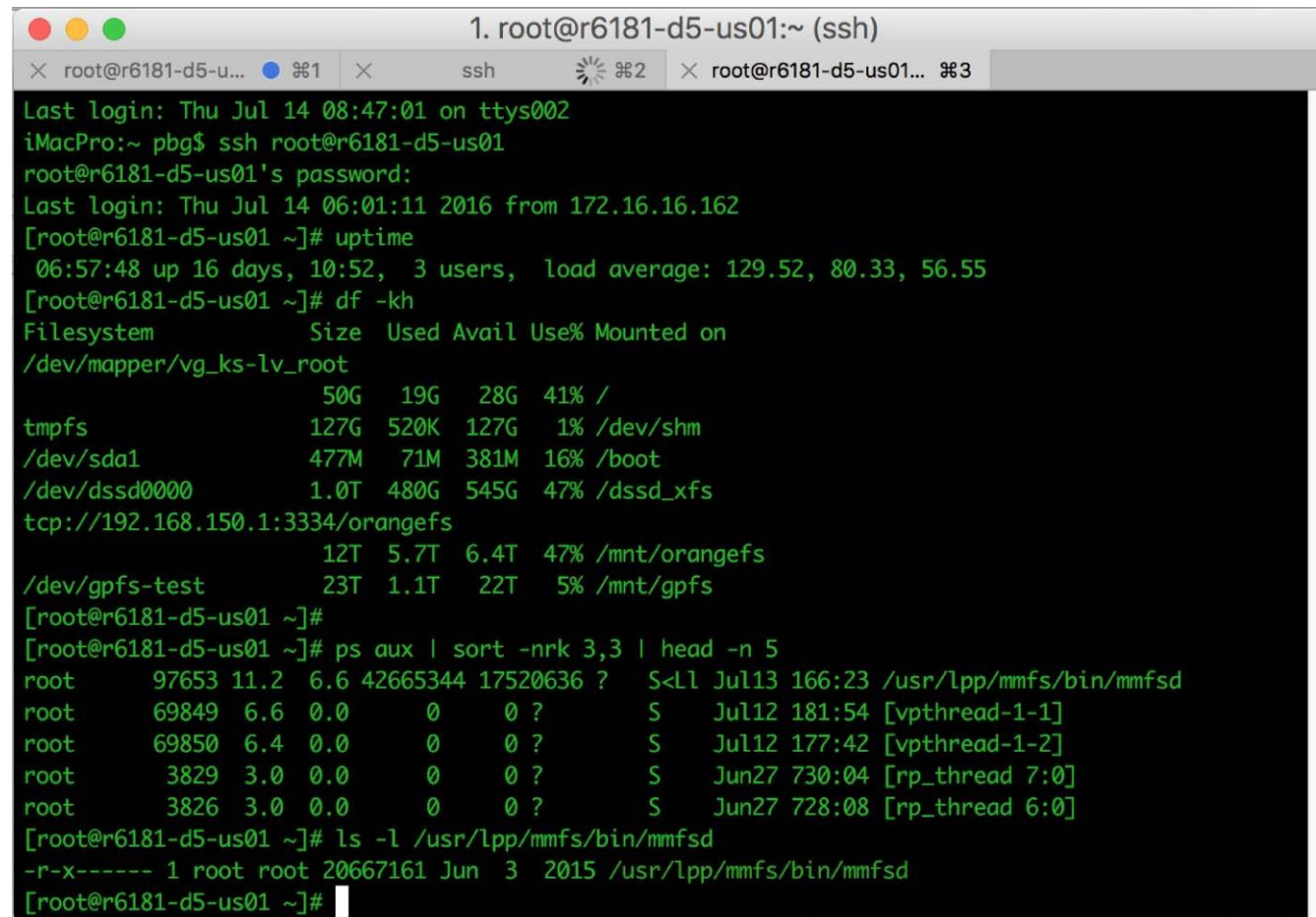
- *getty* afiseaza prompt-ul de login
- utilizatorul introduce numele de utilizator
- *getty* apeleaza */bin/login* care stabileste o noua sesiune de lucru
- *login* afiseaza promptul de parola
  - cauta in */etc/passwd* o intrare corespunzatoare numelui de utilizator
  - verifica parola (de regula stocata criptat in alt fisier, eg. */etc/shadow*)
  - pt parola corecta, executa interpretorul de comenzi (*shell*-ul) asociat intrarii identificate
  - asociaza cu *shell*-ul variabile de mediu (environment)
    - unele variabile importante (USER, SHELL, HOME ) initializate cu valorile din campurile citite din intrarea corespunzatoare din */etc/passwd*
- *shell*-ul afiseaza un prompt specific (eg, \$) si asteapta comenzile utilizatorului
- *init* monitorizeaza sesiunea de lucru a utilizatorului
  - la terminarea activitatii (*shell* exit), reporneste o instanta a programului *getty* pe terminalul respectiv

# Interpretorul de comenzi

- Command Line Interpreter (CLI), permite introducerea directa a comenzilor
  - program de sistem care preia comenzile utilizator si le executa
  - utilizabil deopotriva in mod interactiv cat si batch (folosind *shell script*-uri)
  - executa atat comenzi interne (executate in cadrul interpretorului) cat si externe (programe incarcate de pe disc)
  - functionalitati principale
    - asigurarea unui mediu de lucru utilizatorului (v. comanda *env*)
    - comenzi de manipulare a fisierelor si directoarelor
    - comenzi de control al executiei programelor
    - controlul si monitorizarea activitatilor de I/O
    - administrarea sistemului (rezervata unui utilizator special cunoscut in mod uzual sub numele de *root*, cu `UID = 0`, v. prima intrare din */etc/passwd*)
    - *samd.*
- ex. interpretoare de comenzi: Bourne Shell (`/bin/sh`), Bourne Again Shell (`/bin/bash`), C Shell (`/bin/csh`), Korn Shell (`/bin/ksh`), etc.

`/etc/shells` contine interpretoarele de comenzi disponibile pe sistem

# Bourne Shell (CLI)

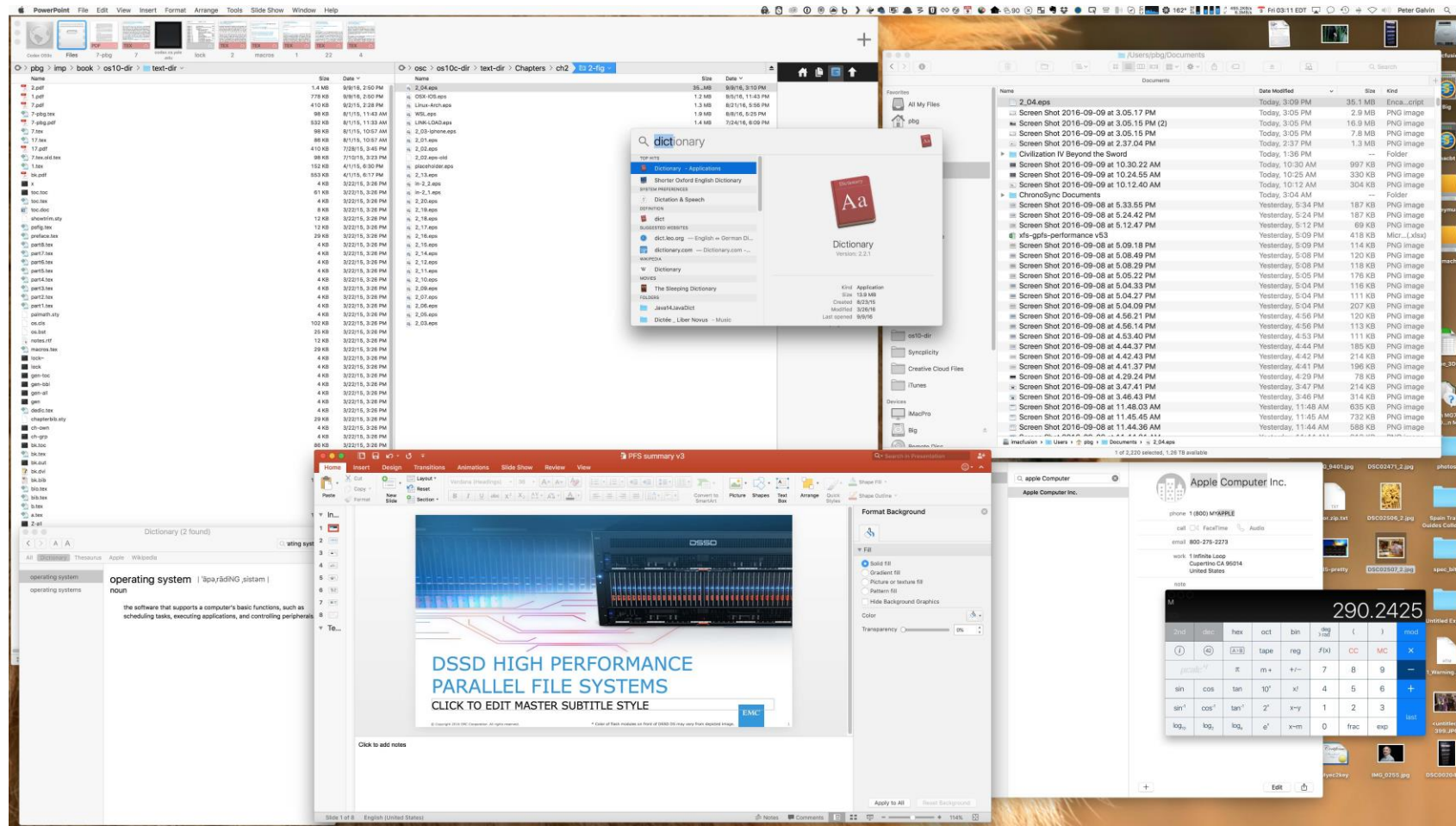


```
1. root@r6181-d5-us01:~ (ssh)
root@r6181-d5-us01:~ (ssh)
Last login: Thu Jul 14 08:47:01 on ttys002
iMacPro:~ pbg$ ssh root@r6181-d5-us01
root@r6181-d5-us01's password:
Last login: Thu Jul 14 06:01:11 2016 from 172.16.16.162
[root@r6181-d5-us01 ~]# uptime
 06:57:48 up 16 days, 10:52,  3 users,  load average: 129.52, 80.33, 56.55
[root@r6181-d5-us01 ~]# df -kh
Filesystem      Size  Used Avail Use% Mounted on
/dev/mapper/vg_ks-lv_root
50G   19G   28G  41% /
tmpfs           127G  520K  127G   1% /dev/shm
/dev/sda1       477M   71M  381M  16% /boot
/dev/dssd0000    1.0T  480G  545G  47% /dssd_xfs
tcp://192.168.150.1:3334/orangefs
12T   5.7T   6.4T  47% /mnt/orangefs
/dev/gpfs-test  23T   1.1T   22T   5% /mnt/gpfs
[root@r6181-d5-us01 ~]#
[root@r6181-d5-us01 ~]# ps aux | sort -nrk 3,3 | head -n 5
root      97653 11.2  6.6 42665344 17520636 ?    S<Ll  Jul13 166:23 /usr/lpp/mmfs/bin/mmfsd
root      69849  6.6  0.0      0      0 ?        S    Jul12 181:54 [vpthread-1-1]
root      69850  6.4  0.0      0      0 ?        S    Jul12 177:42 [vpthread-1-2]
root       3829  3.0  0.0      0      0 ?        S    Jun27 730:04 [rp_thread 7:0]
root       3826  3.0  0.0      0      0 ?        S    Jun27 728:08 [rp_thread 6:0]
[root@r6181-d5-us01 ~]# ls -l /usr/lpp/mmfs/bin/mmfsd
-r-x----- 1 root root 20667161 Jun  3  2015 /usr/lpp/mmfs/bin/mmfsd
[root@r6181-d5-us01 ~]#
```

# Interfata grafica- GUI

- interfata user-friendly
  - compusa uzual din mouse, tastatura, si monitor
  - **icoanele** reprezinta fisierele, programele, actiuni, etc
  - actionarea butoanelor mouse peste obiecte din interfata determina diverse actiuni (furnizare de informatii, optiuni, executia de functii, deschiderea de directoare, etc)
  - inventata la Xerox PARC
- multe sisteme de azi includ atat CLI cat si GUI
  - Microsoft Windows are GUI si CLI “command” shell
  - Apple Mac OS X are GUI cu kernel UNIX dedesubt si shell-uri disponibile
  - Unix si Linux au CLI (shell-uri) cu GUI optional (CDE, KDE, GNOME)

# Mac OS X GUI



# Identificarea utilizatorului

- la login, utilizatorul primește un ID propriu (valoare întreaga nenegativă prin care utilizatorul este identificat în SO), user ID-ul (UID)
  - obținut din intrarea corespunzătoare utilizatorului din */etc/passwd*
  - unic
  - asignat de către *root*, singurul care are permisiunea de a scrie în */etc/passwd*
  - nu poate fi schimbat de către utilizator
  - folosit de kernel pentru a verifica dacă procesele utilizatorului au dreptul să execute anumite operații
  - UID = 0 rezervat pt *root* sau *superuser* (administratorul sistemului)
- procesele *root* au privilegii de superuser și de regulă circumventează verificările pe care kernelul le face pentru o serie de operații
- unele dintre funcțiile kernelului pot fi executate doar de procese *root*
- *root*-ul are control total asupra sistemului de calcul
  - Obs: din acest motiv, este puternic descurajată inițiativa utilizatorilor sistemului care știu parola de *root* să ruleze programe obișnuite în calitate de *root* (UID = 0)

# Identificarea utilizatorului (cont.)

- la login utilizatorul primește și un GID (Group ID), setat tot de *root* în intrarea corespunzătoare din */etc/passwd*
  - permite partajarea de resurse între membrii aceluiași grup, chiar dacă au UID-uri diferite
  - în schimb, utilizatorii cu GID diferit nu pot accesa aceste resurse partajate ale grupului
  - ex: intrările de director pentru fiecare fișier din sistem conțin perechea (UID,GID) a proprietarului fișierului respectiv
  - comanda shell *ls -l* permite afișarea ID-urilor proprietarului fișierului
- */etc/group*
  - asignează nume lizibile GID-urilor utilizator
  - modificabil doar de către *root*
  - listează și *supplementary GIDs*, i.e. același utilizator poate avea mai multe GID-uri (poate aparține mai multor grupuri)
- */usr/bin/id* afișează UID/GID
 

\$ id	# UID/GID pt utilizatorul shell-ului
\$ id root	# UID/GID pt alt utilizator (root)

# Fisiere si directoare

- **fisier**: abstractie de nivel SO pentru stocarea permanenta a datelor
  - ascunde detaliile stocarii efective a datelor pe disc
  - model usor de inteles al structurii datelor (eg, stream de octeti in Unix)
  - grupate in directoare
  - referite prin nume (poate contine orice caracter mai putin '/')
  - attribute: tip, dimensiune, proprietar, permisiuni, timpul ultimei modificari, etc
  - eg Unix, comanda uzuala pentru afisarea atributelor

```
$ ls -l <nume-fisier>
```

- **director/folder**: colectie de fisiere
  - poate contine alte directoare (subdirectoare)
  - modalitate de a organiza informatia, uzual de-o maniera ierarhica



# Fisiere si directoare (cont.)

- sistemele tip Unix folosesc o structura ierarhica de directoare
  - incepe dintr-un director special numit *root* (radacina), desemnat prin caracterul “/”
- directoare speciale create automat atunci cand se creeaza un nou director
  - . directorul curent (directorul nou creat)
  - .. directorul parinte (directorul in care a fost inserata o noua intrare corespunzatoare noului director creat)
  - in cazul directorului radacina (*root*) ./ si ../ reprezinta acelasi director, si anume “/”
- cale (path): secventa de nume de fisiere separate de caracterul /
  - cai absolute: incep intotdeauna cu /
  - cai relative: nu incep cu /, fiind interpretate relativ la directorul de lucru curent (*current working directory*)
- la login, directorul de lucru curent este setat la valoarea obtinuta din */etc/passwd* pentru utilizatorul logat (s.n. *home directory*)
- comanda de tiparire a intrarilor intr-un director: */bin/ls*
  - Obs: un program executabil este si el reprezentat printr-o cale in sistemul de fisiere

# Sistemul de fisiere

- componenta speciala a SO care gestioneaza fisierele si directoarele
- structureaza datele pe disc intr-un anumit *format*
- ofera utilizatorului o interfata uniforma de acces la date
  - eg Unix, ierarhie arborescenta de directoare, cu o radacina comuna
- SO moderne sunt capabile sa integreze sisteme de fisiere cu format diferit in aceeasi ierarhie de directoare
  - VFS – Virtual Filesystem Switch (ext3, ext4, ntfs, vfat, etc)
- devin disponibile utilizatorului ca urmare a operatiei de *mount*

```
$ mount -t ext4 /dev/sda1 /
```

- directorul in care se instaleaza discul formatat s.n. *mountpoint*
- */etc/fstab*: tabela system-wide cu mountpoint-uri inspectata la bootarea SO
  - la bootare, mountpoint-urile din tabela se instaleaza ca si cand s-ar fi apelat

```
$ mount -a
```

# Mountpoints

```
$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            7.7G     0  7.7G   0% /dev
tmpfs           1.6G   9.6M   1.6G   1% /run
/dev/sda6       233G  218G   3.5G  99% /
tmpfs           7.7G  221M   7.5G   3% /dev/shm
tmpfs           5.0M   4.0K   5.0M   1% /run/lock
tmpfs           7.7G     0  7.7G   0% /sys/fs/cgroup
tmpfs           1.6G   72K   1.6G   1% /run/user/1000
$ mount -t ntfs /dev/sda2 /mnt
$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            7.7G     0  7.7G   0% /dev
tmpfs           1.6G   9.6M   1.6G   1% /run
/dev/sda6       233G  218G   3.5G  99% /
tmpfs           7.7G  223M   7.5G   3% /dev/shm
tmpfs           5.0M   4.0K   5.0M   1% /run/lock
tmpfs           7.7G     0  7.7G   0% /sys/fs/cgroup
tmpfs           1.6G   72K   1.6G   1% /run/user/1000
/dev/sda2       195G  147G   49G  76% /mnt
$
```

# Descriptori de fisiere

- intregi nenegativi folositi pentru identificarea fisierele deschise in sistem
  - alocati de kernel la deschiderea/crearea unui fisier prin program
  - folositi subsecvent de catre program pentru citirea/scrierea fisierului
- descriptori speciali
  - la pornirea oricarui program, shell-ul deschide pentru acesta trei descriptori de fisiere speciali:
    - 0 standard input
    - 1 standard output
    - 2 standard error
  - uzual, asociati cu terminalul de login (sau terminalul de lucru, intr-un mediu grafic cu multiple X terminale)
  - `/usr/bin/tty` afiseaza terminalul asociat unui shell (in general, nu doar terminalul de login)

# Redirectarea operatiilor de I/O

- redirectarea operatiilor de I/O se poate face programatic sau direct din shell
- shell-ul intelege constructii sintactice de tipul urmator ca fiind redirectari ale operatiilor de I/O

[n] < filename	redirecteaza citirile de pe descriptorul <i>n</i> catre fisierul desemnat; daca <i>n</i> lipseste, se foloseste <i>stdin</i>
<< marker	redirecteaza <i>stdin</i> catre tastatura folosind un marker de end of file (altfel e Ctrl-d); folosit pentru introducerea documentelor ad-hoc
[n] > filename	redirecteaza scrierile pe descriptorul <i>n</i> catre fisierul desemnat; daca <i>n</i> lipseste, se foloseste <i>stdout</i>
[n] >> filename	adauga scrierile pe descriptorul <i>n</i> la sfarsitul fisierului desemnat ("append"); daca <i>n</i> lipseste, se foloseste <i>stdout</i>

# Redirectarea operatiilor de I/O

- ex:

*\$ echo "redirectarea stdout in fisierul out" > out*

*\$ echo "adaugam la sfarsitul fisierului out inca o linie" >> out*

*\$ cat < out*

*\$ cat << EOF >> out*

*> mai adaugam o line la sfarsitul fisierului out*

*> EOF*

*\$*