

# Instrumente și Tehnici de Baza în Informatică

Semestrul I 2025-2026

Vlad Olaru

# Curs 8 - outline

- demoni
- gestiunea timpului
  - cron
  - anacron
  - at

# Demoni

- procese (servere) care ruleaza in background si nu sunt asociate cu un terminal de control
- in general porniti ca servicii de sistem la sfarsitul operatiei de boot
- serverele pornite de utilizator dintr-un shell (terminal) se dezasociaza de terminal pt a evita
  - interactiuni nedorite cu controlul proceselor executat de shell
  - interactiunea cu gestiunea sesiunii de terminal
  - tiparirea mesajelor pe terminal (avand in vedere ca ruleaza in background)
- exemple
  - serviciile sistem din */etc/rc.d* pornite cu privilegii de superuser (*inetd/xinetd*, *Web*, *sendmail/postfix*, *syslogd* etc)
  - job-uri *cron/at* invocate de demonul de *cron/at*
  - servere/programe pornite din terminal

# Transformarea programelor in demoni

- (1) detasarea de terminal
- (2) schimbarea directorului de lucru curent
  - demonul schimba directorul de lucru curent in / (sau alt director la alegere)
  - fisierele core generate de demon se salveaza in directorul de lucru curent
  - daca directorul de lucru al demonului era pe un sistem de fisiere si demonul continua sa lucreze acolo, sistemul de fisiere respectiv nu va putea fi dezinstalat (*umount*)
- (3) inchiderea tuturor descriptorilor de fisiere deschisi
  - se inchid toti descriptorii de fisiere mosteniti de la procesul care a creat demonul (in mod normal, shell-ul)
- (4) redirectarea *stdin*, *stdout* si *stderr* la */dev/null*
  - garantaza ca descriptorii de fisiere corespunzatori sunt deschisi, dar citirea intoarce EOF iar scrierea se pierde
  - necesar pt ca functiile de biblioteca care presupun folosirea acestor descriptori sa nu esueze
- (5) utilizarea *syslogd* pentru logarea erorilor

# Syslogd

- fara terminal de control, demonii necesita metode specifice pt a afisa mesaje (ex mesaje de eroare, urgență, logging, etc)
- metoda standard folosește funcția *syslog* care trimite mesaje catre demonul *syslogd* (*rsyslogd* pe sisteme Linux)
  - fisier de configurare *syslog.conf*, specifică ce trebuie facut cu fiecare mesaj primit de demon:
    - se adauga la sfârșitul unui fisier
    - se loghează pe consola (*/dev/console*)
    - se forwardează catre demonul *syslogd* al altor mașini
  - serviciul funcționează pe portul 514 UDP (v. */etc/services*)
  - la primirea semnalului SIGHUP se recitește fisierul de configurare (în sistemele moderne se folosește comanda *service reload*)
    - ex: `$ kill -HUP <pid syslogd>`
  - controlat de *init/systemd* prin interfețele discutate anterior
- mesajele se pot trimite direct pe portul 514 UDP, dar ușor se folosește funcția *syslog* sau comanda *logger*

# Functia syslog

- in lipsa terminalului de control demonul nu poate folosi functii de genul *fprintf la stderr*
- functia *syslog*, respectiv comanda *logger* permit scrierea mesajelor in *log-uri*
  - uzual, fisiere din */var/log* sau *consola*
- mesajele logate sunt caracterizate de o combinatie intre *nivel si facilitate* (*level/facility*)
  - combinatia se mai numeste si *prioritate*
- nivelul mesajelor
  - valoare intre intre 0 si 7
  - determina importanta mesajelor: urgență, alertă, critice, eroare, atenționare, etc
  - permite ca toate mesajele de un anumit nivel sa fie tratate unitar
  - valoare implicită: LOG\_NOTICE (nivel 5, notificare normală dar semnificativă)

# Functia syslog (cont.)

- *facilitate*

- specifica tipul de program care trimite mesajul
- ex: mesaje kernel, de autentificare, temporale (*cron/at*), mail, imprimanta, etc
- valoarea implicită: LOG\_USER (mesaje generice de nivel utilizator)
- valoarea facilității permite ca toate mesajele venite de la o anumita facilitate să fie tratate la fel în *syslog.conf*
- ex:

kern.*	/dev/console
local7.debug	/var/log/cisco.log

# Utilizare syslog

- vizualizarea mesajelor logate in timp real

```
$ tail -f /var/log/syslog
```

- *head/tail*

- afiseaza partea de inceput, respectiv sfarsit a unui fisier
  - uzual afisarea se face in termen de linii, cu optiunea *-n*

```
$ head -n 5 /etc/passwd
```

- logarea mesajelor din linie de comanda

```
$ logger -p user.info Some user message of certain importance
```

# Crond

- demon care executa comenzi planificate pentru rulare la un moment dat (uzual periodic)
- pornit ca serviciu de sistem la bootare
  - varianta interactiva, nedemonizata  
`$ cron -f`
- incarca in memorie tabele cu planificările taskurilor utilizator, *crontabs*
  - disponibile in */var/spool/cron/crontabs/\$USER*
  - create/editate cu comanda *crontab*
  - */etc/crontab*, tabela de sistem (cu variantele */etc/cron.hourly*, */etc/cron.daily*, */etc/cron.weekly*, */etc/cron.monthly*)

# Functiune crond

- la fiecare 60s examineaza continutul crontab-urilor pt a stabili daca respectivele comenzi trebuie rulate in timpul minutului curent
- rezultatele executiei comenzilor sunt trimise pe mail proprietarului crontab sau utilizatorului specificat in MAILTO
- in plus, in fiecare minut verifica daca directorul de spool sau tabela de sistem s-au modificar
  - daca da, reincarca continutul tuturor crontab-urilor
  - *crontab* modifica implicit si timpul de modificare al directorului de spool

# Crontab

- comanda pentru gestiunea tabelelor utilizator
- in general, tablelele nu sunt editate direct
- instalarea unei table crontab

*\$ crontab <file> # fisierul contine planificarea in format specific*

- */etc/cron.allow si /etc/cron.deny* se pot folosi pt controlul permisiunii de a folosi un crontab
  - format: un nume de utilizator singur pe o linie
  - daca nu exista, depinde de setarile sistemului, fie doar root-ul poate folosi crond, fie toti utilizatorii
  - precedenta: allow, deny

# Operatii crontab

- editare
  - lanseaza \$VISUAL sau \$EDITOR sau */usr/bin/editor*

*\$ crontab -e*

- stergere

*\$ crontab -r*

- afisare

*\$ crontab -l*

# Formatul crontab

- spatiul, newline ignorate
- # marcheaza comentarii
- linii active de doua feluri
  - setari de variabile de mediu
  - comenzi
- dereferentierea variabilelor de mediu nu functioneaza

Ex: PATH=\$HOME/bin:\$PATH

- variabile de mediu recunoscute:
  - SHELL=/bin/sh implicit
  - LOGNAME, HOME preluate din */etc/passwd* (LOGNAME nu se poate schimba)
  - PATH=/usr/bin/:/bin , valoare implicita
  - MAILTO, valori: utilizator, lista de utilizatori, nedefinita implica \$LOGNAME

# Formatul crontab (cont.)

- <min> <ora> <zi> <luna> <zi a sapt.>      <cmd> “\n”
- campurile referitoare la timp pot fi:
  - valori uzuale, ex: min: 0-59, ore: 0-23, zi: 1-31, luna:1-12, zi a sapt.: 0 – 7, mon-sun (0 si 7 duminica, mon, tue, wed, samd.)
  - intervale, ex: 8-11 pt ore inseamna orele 8,9,10,11
  - liste, ex: “1,2,5,9”, “0-4,8-12”
  - pasi (steps), ex: “0-23/2” pt ore, din 2 in 2 ore
  - “\*”: wildcard, semnificatia fiind intreg intervalul posibil (ex: 0-23 pt ore, 1-31 pt zile)
- comenzi
  - executate de /bin/sh sau \$SHELL
  - “%” : prima aparitie e newline, dupa el *stdin*
  - “%” poate fi escaped cu “\”

# Exemplu crontab

```
# utilizeaza /bin/bash pt a rula comenzi, in loc de /bin/sh
SHELL=/bin/bash
# utilizatorul catre care se trimit pe mail rezultatele
MAILTO=root
#
# ruleaza la 5 minute dupa miezul noptii in fiecare zi
5 0 * * *      $HOME/bin/daily.job >> $HOME/tmp/out 2>&1
# ruleaza la 2:15pm in prima zi a lunii
15 14 1 * *    $HOME/bin/monthly
# ruleaza la 10 pm in zilele de lucru deranjandu-l pe Joe
0 22 * * 1-5   mail -s "It's 10pm" joe%Joe,%%Where are your kids?%
23 0-23/2 * * * echo "run 23 minutes after midn, 2am, 4am ..., everyday"
5 4 * * sun    echo "run at 5 after 4 every sunday"
# ruleaza fiecare a doua sambata din luna
0 4 8-14 * *   test $(date +\%u) -eq 6 && echo "2nd Saturday"
```

# Anacron

- *cron* necesita functionarea in permanenta a calculatorului
- *anacron*
  - ruleaza comenzi periodic, cu frecventa precizata in zile
  - nu pp. rularea continua a sistemului
  - lista de job-uri in `/etc/anacrontab`
  - format linie: `<perioada in zile> <delay in min> <job id> <comanda>`
  - verifica daca s-a executat comanda in ultima perioada (nr de zile de mai sus)
    - daca nu, se executa comanda dupa delay-ul specificat
  - la terminarea comenzii, se salveaza data (fara ora) intr-un fisier in `/var/spool/anacron` pt job-ul respectiv pt a se sti cand sa se execute din nou
  - job id-ul trebuie sa se potriveasca cu unul dintre argumentele comenzii
  - rezultatul job-urilor e trimis fie catre *root* fie catre \$MAILTO, daca exista

# At/atd

- comanda folosita pentru controlul executiei intarziate a comenzilor
- comenzi sunt plasate in cozi de asteptare (desemnate prin litere: a-z, A-Z)
- *at* – executa o comanda la un timp specificat
- *atq* – listeaza job-urile utilizatorului programate pt executie ulterioara aflate intr-o anumita coada (*a* e coada implicita pt *at*)
- *atrm* – sterge un job programat pt executie ulterioara
- specificarea timpului
  - HH:MM, MM/DD/YY, DD.MM.YY, samd
  - now + nr unitati

Ex:

“4pm + 5 days”, “11am Jun 23”, “3pm tomorrow”, etc

# Exemplu at

```
$ at now + 1 minute
```

```
warning: commands will be executed using /bin/sh
```

```
at> ps auxw
```

```
at> <EOT>
```

```
job 4 at Mon Nov 25 20:07:00 2024
```

```
$ atq
```

```
4 Mon Nov 25 20:07:00 2024 a guest
```

```
$ atrm 4
```

```
$ atq
```

```
$
```

# Gestiunea timpului, ceasul HW

- doua ceasuri: HW si sistem
- *hwclock*
  - comanda pt accesul la ceasul HW al sistemului (RTC, CMOS)
  - afiseaza/seteaza timpul HW
    - `$ hwclock --set --date`
  - seteaza ceasul HW la valoarea ceasului sistem si invers
    - `$ hwclock --hctosys`
    - `$ hwclock --systohc`
  - compenseaza drift-ul ceasului HW (folosind istoria din */etc/adjtime*)
    - `$ hwclock --adjust`
  - compara cele doua ceasuri
    - `$ hwclock -c` # compara periodic (10s)
  - prezice valori viitoare ale ceasului HW pe baza vitezei de drift
    - `$ hwclock --predict`
  - samd

# Ceasul sistem

- componenta a kernelului bazata pe un timer (intrerupere de timp)
- are sens doar cat merge calculatorul
- reprezinta nr de secunde de la 1 Ian 1970 00:00:00
- ceasul care conteaza in sistem, initializat la boot din ceasul HW
- *date*
  - afiseaza/seteaza ceasul sistem
  - foloseste parametri de formatare introdusi cu “+”

<code>\$ date +%d</code>	# afiseaza ziua lunii (e.g., 02)
<code>\$ date +%m</code>	# luna
<code>\$ date +%y</code>	# anul
<code>\$ date +%H</code>	# ora
<code>\$ date +%M</code>	# minutul

samd.

# Timpul de executie al unei comenzi

- masurarea timpului de executie al unei comenzi

```
$ time find / -name crontab
```

```
$ time sleep 5
```

- *time* e comanda interna shell, dar exista si varianta externa
  - afiseaza trei timpi:
    - timpul real
    - timpul petrecut in spatiul utilizator
    - timpul petrecut in kernel
  - in plus, sumarizeaza folosirea resurselor sistemului
    - se foloseste un format specific
    - in *bash* se foloseste comanda externa */usr/bin/time*