
RA3. Automatització de tasques del sistema

Índex

1. Per què automatitzar? Avantatges i riscos	1
2. Planificació de tasques amb cron	1
2.1 Format d'una línia de crontab	1
2.2 Crontabs per usuari	2
2.3 Crontab del sistema i /etc/cron.d/	2
2.4 Errors habituals en scripts cridats des de cron	3
3. Tasques quan la màquina no sempre és encesa: anacron	3
4. Tasques puntuals: at i batch	3
5. L'alternativa moderna: temporitzadors de systemd	4
6. Restriccions de seguretat	5
6.1 Control d'accés a cron i at	5
6.2 Permisos dels scripts i fitxers de tasques	6
6.3 Principi de mínim privilegi	6
7. Casos pràctics de planificació	6
7.1 Tasca repetitiva: rotació i neteja de logs propis	6
7.2 Tasca repetitiva: còpia de seguretat incremental	7
7.3 Tasca puntual: manteniment programat fora d'hores	7
7.4 Comprovació d'eficiència	7
8. Automatització de la gestió de comptes	8
8.1 Creació massiva d'usuaris des d'un fitxer	8
8.2 Baixa i bloqueig automàtic de comptes inactius	8
8.3 Caducitat de contrasenyes i comptes	8
9. Eines gràfiques: instal·lació i ús	9
9.1 Webmin -- mòdul "Scheduled Cron Jobs"	9
9.2 KCron (entorns KDE Plasma)	9
9.3 crontab-ui (eina web lleugera, multiplataforma)	9
9.4 Comparativa ràpida	10
10. Documentació dels processos programats	10
10.1 Capçalera de comentari a cada tasca	10
10.2 Inventari centralitzat	11
10.3 Comandes útils per auditar què hi ha planificat en un sistema	11

Cicle: Administració de sistemes informàtics en xarxa (ASIX)

Mòdul: 0374. Administració de sistemes operatius

1. Per què automatitzar? Avantatges i riscos

Un servidor sol executar, dia rere dia, un conjunt reduït de tasques repetitives: rotació de logs, còpies de seguretat, neteja de fitxers temporals, comprovacions d'estat, actualitzacions de bases de dades d'antivirus, sincronització d'hora, etc. Fer-les a mà té tres problemes:

- **Es descuiden.** Ningú recorda fer una tasca cada dia a les 3 h de la matinada.
- **No són reproduïbles.** Un procés manual depèn de qui l'executa i de com estigui de cansat aquell dia.
- **No escalen.** Amb 1 servidor és factible; amb 50, no.

L'automatització resol això, però introdueix riscos propis que cal gestionar:

Avantatge	Risc si no es controla
Regularitat garantida	Una tasca mal planificada pot repetir-se sense parar
Estalvi de temps de l'administrador	Un script amb permisos massa amplis és un vector d'atac
Reducció de l'error humà	Un error en un script automatitzat es repeteix... automàticament
Traçabilitat (si es registra la sortida)	Si no es documenta, ningú sap per què existeix una tasca al cap d'un any

La regla d'or: **tota tasca automàtica ha de tenir sortida registrada, permisos mínims necessaris i estar documentada.**

2. Planificació de tasques amb cron

cron és el dimoni clàssic de planificació a Linux. Llegeix taules de tasques (*crontabs*) i n'executa les entrades quan es compleix la data/hora indicada.

2.1 Format d'una línia de crontab

```
# minut hora dia_mes mes dia_setmana ordre
   *      *      *      *      *      comanda
```

Camp	Rang
minut	0-59
hora	0-23
dia del mes	1-31
mes	1-12 (o jan-dec)
dia de la setmana	0-7 (0 i 7 = diumenge, o sun-sat)

Es poden combinar amb * (qualsevol valor), llistes (1, 15), rangs (1-5) i passos (* / 10).

Exemples:

```
# Cada dia a les 3:30
30 3 * * * /usr/local/bin/backup.sh

# Cada 15 minuts
*/15 * * * * /usr/local/bin/comprova_estat.sh

# Cada dilluns a les 8h
0 8 * * 1 /usr/local/bin/informe_setmanal.sh

# El dia 1 de cada mes a mitjanit
0 0 1 * * /usr/local/bin/tancament_mensual.sh
```

2.2 Crontabs per usuari

Cada usuari té la seva pròpia crontab, gestionada amb la comanda crontab:

```
crontab -e          # edita la crontab de l'usuari actual
crontab -l          # llista la crontab de l'usuari actual
crontab -r          # elimina la crontab de l'usuari actual
crontab -u ramon -e # edita la crontab d'un altre usuari (cal ser
↪ root)
```

Aquests crontabs individuals es desen internament a `/var/spool/cron/crontabs/<usuari>` i **no s'han d'editar mai directament amb un editor de text**: cal fer-ho sempre amb `crontab -e`, perquè així es valida la sintaxi abans de desar.

2.3 Crontab del sistema i `/etc/cron.d/`

A banda de les crontabs d'usuari, hi ha dos mecanismes pensats per a tasques d'administració del sistema (paquets, scripts propis):

- **/etc/crontab**: com una crontab normal, però amb una columna extra abans de l'ordre que indica **amb quin usuari** s'executa la tasca.

```
# min hora dia mes diaSetm usuari ordre
30 3 * * * root /usr/local/bin/backup.sh
```

- **/etc/cron.d/**: directori on cada paquet o cada administrador pot deixar un fitxer independent amb el mateix format que `/etc/crontab` (amb usuari). És la manera recomanada d'afegir tasques del sistema, perquè no cal editar un fitxer compartit i els canvis queden aïllats per paquet/servei.
- **/etc/cron.hourly/**, **/etc/cron.daily/**, **/etc/cron.weekly/**, **/etc/cron.monthly/**: directoris on n'hi ha prou de deixar un script executable (sense necessitat d'escriure sintaxi cron); un-part s'executa amb la periodicitat indicada pel nom del directori. Còmode per a scripts senzills que no necessiten una hora exacta.

2.4 Errors habituals en scripts cridats des de cron

El motiu número 1 pel qual “el meu script funciona a mà, però no per cron” és l'entorn:

- cron no carrega ~/.bashrc ni ~/.profile, per tant, \$PATH és mínim (sol ser /usr/bin:/bin). **Solució:** utilitzar sempre camins absoluts (/usr/bin/rsync en lloc de rsync) o definir PATH a la mateixa crontab.
- No hi ha terminal ni variable \$DISPLAY. Cap ordre que necessiti interfície gràfica funcionarà.
- La sortida (stdout/stderr) es descarta llevat que es redirigeixi o hi hagi un MTA configurat que l'envii per correu a l'usuari propietari de la tasca.

```
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
30 3 * * * /usr/local/bin/backup.sh >> /var/log/backup.log 2>&1
```

3. Tasques quan la màquina no sempre és encesa: anacron

cron assumeix que la màquina està encesa a l'hora exacta prevista: si l'equip està apagat, la tasca simplement no s'executa fins al pròxim cicle. Per a portàtils o màquines que no estan sempre engegades, anacron és la solució: comprova, en arrencar, si una tasca diària/setmanal/mensual s'ha “perdut” i, en cas afirmatiu, l'executa (amb un retard configurable, per no saturar l'arrencada).

```
cat /etc/anacrontab
```

#	periode	retard	identificador	ordre
	1	5	cron.daily	run-parts --report /etc/cron.daily
	7	10	cron.weekly	run-parts --report /etc/cron.weekly
	@monthly	15	cron.monthly	run-parts --report /etc/cron.monthly

A la majoria de distribucions, /etc/cron.daily, .weekly i .monthly ja estan orquestrats per anacron quan aquest paquet està instal·lat, complementant cron.

4. Tasques puntuals: at i batch

cron és per a tasques **repetitives**. Quan cal executar una ordre **una sola vegada**, en un moment futur concret, l'eina és at.

```
# Instal·lació (Debian/Ubuntu)
sudo apt install at

# Programar una tasca puntual
echo "/usr/local/bin/reinicia_servei.sh" | at 23:00

# De forma interactiva
at now + 30 minutes
```

```
at> /usr/local/bin/notifica.sh
at> <Ctrl+D>
```

```
# Amb data concreta
at 09:00 25.12.2026
```

Gestió de la cua:

```
atq                # llista les tasques pendents (per a l'usuari
↪ actual)
at -c <job_id>      # mostra el contingut d'una tasca
atrm <job_id>       # elimina una tasca pendent
```

batch és similar a at, però en lloc d'una hora fixa, executa l'ordre **quan la càrrega del sistema baixa per sota d'un llindar** (per defecte 1.5), útil per a tasques pesades que no urgeixen:

```
echo "/usr/local/bin/reindexa_bd.sh" | batch
```

5. L'alternativa moderna: temporitzadors de systemd

En distribucions amb systemd, es pot substituir cron per un parell de fitxers unitat: un `.service` (què s'executa) i un `.timer` (quan). Els avantatges respecte a cron són: registre integrat a `journalctl`, dependències entre serveis, i possibilitat de recuperar tasques perdudes (`Persistent=true`), similar a `anacron` però nadiu.

/etc/systemd/system/backup.service

```
[Unit]
Description=Còpia de seguretat diària

[Service]
Type=oneshot
ExecStart=/usr/local/bin/backup.sh
```

/etc/systemd/system/backup.timer

```
[Unit]
Description=Executa backup.service cada dia a les 3:30

[Timer]
OnCalendar=*-*-* 03:30:00
Persistent=true
RandomizedDelaySec=300

[Install]
WantedBy=timers.target
```

Activació i comprovació:

```
sudo systemctl daemon-reload
sudo systemctl enable --now backup.timer

systemctl list-timers                # mostra tots els temporitzadors
↳ i la propera execució
journalctl -u backup.service        # registre de les execucions
systemctl status backup.timer
```

Persistent=true fa que, si la màquina estava apagada quan tocava executar-se, ho faci en la pròxima arrencada (equivalent a anacron). RandomizedDelaySec evita que, en un parc de màquines, totes executin la mateixa tasca exactament al mateix segon.

6. Restriccions de seguretat

Automatitzar sense controlar qui pot fer-ho és obrir una porta d'escalada de privilegis: un script programat amb l'usuari root que sigui editable per un usuari normal és, de facto, root per a aquest usuari.

6.1 Control d'accés a cron i at

Els dimonis cron i at comproven, en aquest ordre, dos fitxers:

```
/etc/cron.allow  /etc/cron.deny
/etc/at.allow    /etc/at.deny
```

- Si existeix *.allow: **només** els usuaris llistats hi poden accedir (llista blanca). *.deny s'ignora.
- Si no existeix *.allow, però sí *.deny: tothom **excepte** els llistats hi pot accedir (llista negra).
- Si no existeix cap dels dos: normalment només root pot fer servir cron/at (comportament per defecte a moltes distribucions modernes).

Bona pràctica: crear sempre un cron.allow explícit amb els usuaris que realment necessiten planificar tasques, en lloc de confiar en el comportament per defecte:

```
echo "ramon" | sudo tee /etc/cron.allow
echo "backupuser" | sudo tee -a /etc/cron.allow
sudo rm -f /etc/cron.deny # que no quedin fitxers contradictoris
```

6.2 Permisos dels scripts i fitxers de tasques

- Els scripts cridats per root han de ser propietat de root i **no** escriptibles per grup ni altres (chmod 700 o 750 segons el cas). Si un script root-owned és escriptible per un usuari normal, aquest usuari pot injectar-hi codi que s'executarà com a root.
- Els fitxers a /etc/cron.d/ han de tenir permisos 644 i ser propietat de root:root; cron ignora silenciosament fitxers amb permisos massa oberts (group/other amb permís d'escriptura) com a mesura de protecció.
- Per a temporitzadors systemd, es pot restringir encara més l'execució amb directives de sandboxing al .service: User=, ProtectSystem=strict, NoNewPrivileges=true, ReadOnlyPaths=, etc.

```
[Service]
User=backupuser
NoNewPrivileges=true
ProtectSystem=strict
ReadWritePaths=/var/backups
ExecStart=/usr/local/bin/backup.sh
```

6.3 Principi de mínim privilegi

Regla pràctica: si una tasca només necessita llegir fitxers d'un usuari i pujar-los a un servidor de còpies, **no s'executa com a root**. Es crea un usuari de servei dedicat (backupuser), se li donen només els permisos necessaris (per exemple, via ACL o pertinença a un grup concret) i la tasca es planifica amb crontab -u backupuser -e o amb User= al fitxer .service.

7. Casos pràctics de planificació

7.1 Tasca repetitiva: rotació i neteja de logs propis

```
#!/bin/bash
# /usr/local/bin/neteja_logs.sh
# Elimina logs de l'aplicació amb més de 30 dies

LOGDIR="/var/log/app"
find "$LOGDIR" -name "*.log" -mtime +30 -delete
echo "$(date '+%F %T') - Neteja completada" >>
  ↪ /var/log/neteja_logs.log
```

```
0 2 * * * /usr/local/bin/neteja_logs.sh
```

7.2 Tasca repetitiva: còpia de seguretat incremental

```
#!/bin/bash
# /usr/local/bin/backup.sh
DEST="/mnt/backup/$(date +%F)"
mkdir -p "$DEST"
rsync -a --link-dest=/mnt/backup/latest /srv/dades/ "$DEST"/
ln -sf "$DEST" /mnt/backup/latest
```

7.3 Tasca puntual: manteniment programat fora d'hores

```
echo "/usr/local/bin/actualitza_sistema.sh" | at 23:30 tomorrow
```

7.4 Comprovació d'eficiència

Un criteri important (i sovint oblidat) és **no sobrecarregar el sistema amb tasques mal repartides**. Bones pràctiques:

- Repartir les tasques pesades (backups, indexacions) en franges horàries de baixa activitat i **no totes a la mateixa hora en punt** (RandomizedDelaySec a systemd, o triar minuts diferents de 0 a cron).
- Fer servir nice/ionice per no competir per CPU/disc amb serveis en producció:

```
0 3 * * * nice -n 19 ionice -c3 /usr/local/bin/backup.sh
```

- Evitar encavalcament: si una tasca pot trigar més que l'interval de planificació, cal un mecanisme de bloqueig (flock):

```
*/5 * * * * flock -n /tmp/sync.lock /usr/local/bin/sync.sh
```

8. Automatització de la gestió de comptes

8.1 Creació massiva d'usuaris des d'un fitxer

```
#!/bin/bash
# /usr/local/bin/alta_usuaris.sh
# Format del CSV: usuari;nom_complet;grup

while IFS=';' read -r usuari nom grup; do
    useradd -m -c "$nom" -G "$grup" -s /bin/bash "$usuari"
    echo "${usuari}:${(openssl rand -base64 12)}" | chpasswd
    chage -d 0 "$usuari" # obliga a canviar la contrasenya en el
    ↪ primer accés
done < usuaris.csv
```

8.2 Baixa i bloqueig automàtic de comptes inactius

```
#!/bin/bash
# /usr/local/bin/bloqueja_inactius.sh
# Bloqueja comptes sense accés en els últims 90 dies

for usuari in $(lastlog -b 90 | awk 'NR>1 {print $1}'); do
    usermod -L "$usuari"
    echo "$(date '+%F') - Compte bloquejat per inactivitat: $usuari"
    ↪ >> /var/log/comptes.log
done
```

```
0 4 * * 1 /usr/local/bin/bloqueja_inactius.sh
```

8.3 Caducitat de contrasenyes i comptes

chage permet fixar polítiques de caducitat que es comproven automàticament en cada inici de sessió, sense necessitat de cap tasca planificada addicional:

```
chage -M 90 -W 7 usuari # contrasenya caduca als 90 dies, avisa
    ↪ 7 dies abans
chage -E 2026-12-31 usuari # el compte caduca en una data concreta
chage -l usuari # consulta la configuració actual
```

Combinar això amb un script planificat que revisi `chage -l` per a tots els usuaris i enviï un avís per correu als que caduquen aviat és una pràctica habitual d'automatització.

9. Eines gràfiques: instal·lació i ús

Encara que l'administració via terminal és la més potent i la que millor es documenta i versiona, molts entorns (especialment els mixtos, amb administradors menys experimentats en shell) es beneficien d'eines gràfiques.

9.1 Webmin -- mòdul “Scheduled Cron Jobs”

Instal·lació (Debian/Ubuntu):

```
curl -o setup-repos.sh https://raw.githubusercontent.com/webmin/webmin/master/setup-repos.sh
sudo sh setup-repos.sh
sudo apt install webmin
```

Un cop instal·lat, s'accedeix via navegador a https://<ip_servidor>:10000. Al menú **System** → **Scheduled Cron Jobs** es poden crear, editar i eliminar tasques cron amb un formulari (usuari, minut, hora, dia, mes, ordre), sense tocar cap fitxer a mà. Webmin també té un mòdul **System** → **Scheduled Commands** equivalent a at.

9.2 KCron (entorns KDE Plasma)

```
sudo apt install kcron
```

kcron obre un gestor gràfic de la crontab de l'usuari actual (o de qualsevol usuari si s'executa com a root), amb pestanyes per a tasques i variables d'entorn.

9.3 crontab-ui (eina web lleugera, multiplataforma)

Útil quan es vol una interfície web independent del gestor d'escriptori:

```
sudo apt install npm
sudo npm install -g crontab-ui
crontab-ui
```

S'obre a <http://localhost:8000> i permet crear/editar/activar/desactivar tasques amb registre d'execucions integrat.

9.4 Comparativa ràpida

Eina	Entorn	Punt fort
Webmin	Web, qualsevol distro	Gestiona tot el sistema, no només cron
KCron	Escriptori KDE	Integrat, molt senzill per a un únic usuari
crontab-ui	Web, lleuger	Registre d'execucions integrat, fàcil d'exposar només a la xarxa interna
GNOME (sense eina nadiua)	GNOME	Cal recórrer a Webmin o crontab-ui

IMPORTANT

Qualsevol d'aquestes eines web (Webmin, crontab-ui) exposa un servei de xarxa que permet planificar ordres arbitràries. Cal protegir-lo amb HTTPS, autenticació forta i, idealment, restringir l'accés per IP/tallafoç a la xarxa d'administració.

10. Documentació dels processos programats

Una tasca automàtica que ningú entén al cap de sis mesos és un risc, no un estalvi. Convé seguir una convenció mínima:

10.1 Capçalera de comentari a cada tasca

```
# -----  
# Nom:          Neteja de logs de l'aplicatiu X  
# Responsable:  ramon  
# Objectiu:     Elimina logs >30 dies per evitar omplir el disc  
# Freqüència:   Diària, 02:00  
# Script:       /usr/local/bin/neteja_logs.sh  
# Log:          /var/log/neteja_logs.log  
# Data alta:    2026-09-01  
# -----  
0 2 * * * /usr/local/bin/neteja_logs.sh
```

10.2 Inventari centralitzat

Més enllà del comentari en línia, és recomanable mantenir un inventari (full de càlcul, wiki interna o fitxer README.md al mateix directori dels scripts) amb una fila per tasca:

Tasca	Usuari	Planificador	Freqüència	Script	Log	Responsable
Backup diari	backupuser	cron (/etc/cron.d/backup)	03:30 diari	/usr/local/bin/backup.sh	/var/log/backup.log	ramon
Neteja logs	root	cron	02:00 diari	/usr/local/bin/neteja_logs.sh	/var/log/neteja_logs.log	ramon
Bloqueig d'inactius	root	cron	04:00 dilluns	/usr/local/bin/bloqueja_inactius.sh	/var/log/comptes.log	ramon

10.3 Comandes útils per auditar què hi ha planificat en un sistema

```
for u in $(cut -f1 -d: /etc/passwd); do
    echo "== $u =="
    crontab -u "$u" -l 2>/dev/null
done

ls -la /etc/cron.d/ /etc/cron.daily/ /etc/cron.weekly/
↪ /etc/cron.monthly/
systemctl list-timers --all
```

Executar aquesta petita auditoria abans d'una migració o d'un canvi d'administrador és un pas que estalvia moltes sorpreses.

Versions d'aquest document

- [HTML](#)
- [PDF](#)
- [ODT](#)
- [MD](#)

[Domini Públic \(CC0\)](#)