

# Pràctica 5: Còpies de Seguretat

## Objectius de la pràctica

Aprendre a dissenyar i implementar sistemes de còpies de seguretat tot utilitzant eines bàsiques de UNIX.

## Abans de començar

Per aquesta sessió haurieu de ser capaços de contestar les següents preguntes abans de començar:

- Com es pot empaquetar i desempaquetar un(s) fitxer(s) utilitzant la comanda `tar`?
- Què és un enllaç dur (hard link)? I quina diferència hi ha entre fer una còpia (`cp file_a file_b`) i fer un *hard link* (`ln file_a file_b`)?

## Introducció

Una de les tasques més importants de l'administrador de sistemes és la realització de còpies de seguretat que permeten restaurar el sistema complet en una quantitat acceptable de temps quan es produeix una fallada del sistema amb pèrdua de dades. Aquestes pèrdues poden ser degudes a múltiples factors com poden ser fallades de hardware, de software, accions humanes (accidentals o premeditades) o desastres naturals (1).

Abans de començar a realitzar còpies de seguretat l'administrador del sistema ha de decidir una política tenint en compte aspectes com(2):

- Seleccionar el tipus correcte de medi físic per fer les còpies de seguretat tenint en compte la grandària, el cost, la velocitat, la disponibilitat, l'usabilitat i la confiabilitat.
- Decidir quins fitxers necessiten una còpia de seguretat i on són aquest fitxers. Són més importants els fitxers de configuració del sistema i els fitxers dels usuaris (normalment ubicats a `/etc` i `/home` respectivament) que els fitxers temporals o els binaris del sistema (`/tmp` i `/bin`)
- Decidir la freqüència i el tipus de planificació de les còpies de seguretat. Això depèn de la variabilitat de les dades. Una base de dades pot necessitar múltiples còpies de seguretat diàries, mentre que un servidor web pot requerir només una còpia diària i altres sistemes de fitxers poden requerir només una còpia setmanal.
- Analitzar altres aspectes com: on s'han d'emmagatzemar les còpies, per quan temps s'han de mantenir i amb quina rapidesa es necessita poder recuperar cada tipus de fitxer.

Utilitzant tota la informació anterior es pot decidir finalment una estratègia de

còpies de seguretat. Això inclou decidir la freqüència de les còpies i el tipus. Una estratègia comú es fer còpies completes i còpies incrementals. D'aquesta manera es pot disminuir el temps i la grandària de les transferències de dades de les còpies però també s'incrementa la complexitat de la restauració de les dades.

Una estratègia típica consisteix en realitzar còpies completes (també conegudes com de nivell 0) setmanalment i còpies incrementals (conegudes com de nivell 1 o més gran) diàriament. Si el grau de variabilitat dels fitxers és molt gran es pot modificar l'anterior model setmanal per un model mensual on cada més es realitza una còpia de nivell 0, cada setmana una còpia de nivell 1 (incremental setmanal sobre el nivell 0) i cada dia es fa una còpia de nivell 2 (incremental diari sobre el nivell 1).

Per últim s'han de decidir les eines més adequades per implementar l'estratègia de còpies de seguretat que s'ha dissenyat. En aquesta pràctica usarem el mateix disc dur com medi físic per fer les còpies de seguretat tot i que no és el més convenient habitualment a causa del alt risc de que una pèrdua de dades del disc afecti també a les còpies de seguretat. Per fer les còpies utilitzarem dos tipus diferents d'aplicacions de còpia de seguretat: **tar**, que realitza les còpies a través del sistema de fitxers i **rsync** que permet sincronitzar discs amb moltes opcions de configuració i per tant permet implementar diferents estratègies de còpies de seguretat.

## Partició per emmagatzemar les còpies de seguretat.

Per tal de guardar els fitxers de les còpies de seguretat que generarem al llarg de la pràctica, creeu un nou directori `/backup`. Creeu una nova partició a l'espai lliure que teniu al disc i doneu-li format `extended3`. Munteu aquesta partició sobre el directori `/backup` de forma que tan sols `root` tingui permisos d'accés. La resta d'usuaris no han de tenir ni tan sols permís de lectura al directori ja que el contingut de les còpies de seguretat podria ser informació confidencial.

Quines comandes heu utilitzat per crear la partició, donar format al sistema de fitxers, muntar la partició i canviar els permisos del directori `backup`?

```
mkfs.ext4 /dev/sdb4      mount /dev/sdb4 /backup
chmod go-rx /backup.
```

Per afegir més proteccions a aquest directori es pot muntar en mode de escriptura només quan s'escriguin els *backups* i la resta del temps muntar-lo en mode de només lectura. Normalment, s'hauria de desmuntar i tornar a muntar canviant les opcions per defecte. Però és possible canviar les opcions d'una partició sense desmuntar-la si feu servir l'opció **remount**.

Muntar només per lectura:

```
mount -o remount,ro /dev/usb4 /backup
```

Muntar per lectura i escriptura:

```
mount -o remount,rw /dev/usb4 /backup
```

Quina modificació és necessària fer perquè aquesta nova partició es munti automàticament durant el boot amb mode de només lectura?

`fstab`

`/dev/sdb4 /backup ext4 ro 0 2`

## Realització de còpies amb TAR

### Realització de còpies completes

Realitzeu una còpia completa del directori `/root` (comproveu que existeixen fitxers en aquest directori) amb la comanda `tar`. Useu noms significatius per als fitxers de *backup*: feu que el nom del fitxer tingui informació sobre el contingut del fitxer, la data i hora en que es va fer la còpia, i si la còpia és completa o incremental, i el nivell de la còpia (0, 1, ...).

Com es pot fer perquè el nom del fitxer de *backup* inclogui automàticament la data del *backup*? per exemple que sigui `backup-etc-nivell0-200712041030` (any mes dia hora minut segon). Utilitza la comanda `date`.

`date +%Y%m%d%H%M%S`

Quina comanda heu fet servir per fer la còpia completa del directori `/etc`?

`tar -cf /backup / backup-etc-nivell0-`date +%Y%m%d%H%M%S`.tar /etc`

Per què no és aconsellable comprimir el fitxer de *backup*?

Augmenta el temps de recuperació

Si volguéssim comprimir el fitxer de *backup* quina opció afegiríem a la comanda `tar`?

`-z`

A vegades quan es realitza la còpia completa es requereix excloure certs fitxers. Per això es pot construir un fitxer amb una llista de fitxers que no haurien de ser al *backup*.

Feu de nou la copia completa però aquesta ocasió exclouent el fitxers que siguin

al fitxer *excludes*. (poseu el nom d'alguns fitxers al fitxer *excludes*)

Quina opció de **tar** permet excloure un llistat de fitxers del *backup*?

-X *excludes*.      posar abans de la còpia  
a fer còpia (filtre)

A més a més de protegir el directori de les còpies de seguretat és importat utilitzar algun mecanisme que permeti verificar que els fitxers de *backup* no hagin estat modificats després d'haver-los creat. Per això es comú utilitzar algun mecanisme de signatura digital, com són els *hashs* MD5, que permeten verificar la integritat d'un fitxer. Una vegada hagueu realitzat la còpia, utilitzeu la comanda **md5sum** (utilitzeu el **man** per saber com fer servir aquesta comanda) amb la còpia del directori i guardeu-vos el resultat en un fitxer *<nomdelacòpia>.asc*.

Com heu utilitzat la comanda **md5sum** per produir la signatura md5?

```
md5sum "arxiv" > "nomarxiv".asc  
comprovar: md5sum -c "nomarxiv".asc "arxiv"
```

## ***Realització de còpies incrementals***

Per tal de dur a terme còpies incrementals, serà necessari modificar alguns fitxers dins el directori */root*

- Genereu nous fitxers i subdirectoris
- Modifiqueu el contingut d'alguns fitxers
- Useu la comanda **touch** per canviar la data de modificació d'alguns fitxers.

Per fer còpies incrementals amb **tar** tenim l'opció **--newer** que només inclou els fitxers que hagin estat modificats des d'una data determinada. Aquesta data es pot especificar de dues formes: la primera, posant-la directament, per exemple: "**--newer=2007-11-28 12:10**". La segona manera consisteix en agafar la data d'un fitxer, això vol dir que la data serà la de l'última modificació del fitxer, per exemple: "**--newer=./file**".

Ara realitzeu una còpia incremental del directori */root* respecte a la còpia completa que heu fet abans usant la comanda **tar**. Feu també un **md5sum** i guardeu-lo en un arxiu.asc amb el nom diferent al nom que heu utilitzat abans.

Quina comanda heu utilitzat per fer la còpia incremental?

```
tar --newer="data" -cf /backup/backupinc.tar /root
```

Quin problema potencial hi ha al utilitzar el fitxer .tar de la còpia completa per obtenir la data del backup per fer la còpia incremental? Com es pot resoldre aquest problema?

que només es read only, muntar i desmuntar.  
(remount)

Realitzeu una segona ronda de modificacions al directori /root per tal de provocar una segona còpia incremental:

- Genereu nous fitxers
- Modifiqueu el contingut d'alguns fitxers
- Useu la comanda **touch** per canviar la data de modificació d'alguns fitxers.
- Esborreu algun dels fitxers que heu generat per la primera còpia incremental anterior.

Realitzeu una segona còpia incremental del directori /root (respecte la primera còpia incremental) amb la comanda **tar**. També feu un **md5sum** de la segona còpia incremental i poseu-li un nom apropiat.

Quina comanda heu fet servir per fer la segona còpia incremental?

```
tar --newer="data" -cf /backup/backupinc.tar /root
```

Com es pot verificar que el contingut del fitxer de *backup* sigui el mateix que el directori que s'ha copiat?

```
tar -tvf
```

Com es pot verificar, fent ús de la comanda **md5sum**, la integritat d'una còpia de seguretat, és a dir, que el fitxer no ha estat modificat des que es va realitzar la còpia?

Comprovar: `md5sum -c "nomarxiv".asc "arxiv"`

## **Restauració d'una còpia de seguretat**

Renombreu el directori /root per /root.old per simular l'efecte que es produiria si esborréssim el directori. Ara restaureu la còpia de seguretat del directori /root, la qual cosa implica restaurar els tres fitxers que hem creat: la còpia completa i les dos incrementals.

En quin ordre cal restaurar els fitxers per tal que el resultat final sigui el desitjat? Quines comandes heu utilitzat?

total i incrementals per ordre

tar -zP

Què ha passat amb els fitxers que havíeu esborrat abans de fer la segona còpia incremental? Com es pot detectar que aquest fitxers han estat esborrats? Quan seran esborrats de les còpies de seguretat?

han estat esborrats i han tornat a aparèixer

no es pot detectar.

## Restauració d'un fragment

Renombreu un dels subdirectoris dins del directori /root per simular l'efecte que es produiria si esborréssim el subdirectori. Restaureu només aquest directori a partir de la còpia de seguretat que hem fet amb **tar**. Això implica restaurar únicament aquest subdirectori a partir dels tres fitxers que hem creat: la còpia completa i les dos incrementals.

Quines comandes heu fet servir per recuperar només una part de les còpies de seguretat fetes amb **tar**?

tar -xvf foo.tar home/foo/bar

\* sense barra

només extrec el bar

## Realització de còpies usant RSYNC (RS)

Fins ara hem vist com fer *backups* i guardar-los en la mateixa màquina, però el més comú és tenir diferents màquines en què fem *backups* i una altra màquina en xarxa que emmagatzema aquests *backups*. Per fer això tenim la comanda **rsync** que permet copiar un directori (o un conjunt de fitxers) a un altre directori a través de una connexió de xarxa. Per això **rsync** usa un algorisme de *checksum* eficient per transmetre només les diferències entre els dos directoris i al mateix temps comprimeix els fitxers per fer més ràpida la transmissió de dades. Aquesta eina permet copiar fitxers des de o cap a un directori situat en una màquina remota, o entre directoris de la mateixa màquina. El que no permet és copiar directoris entre dos màquines remotes. A més a més **rsync** permet copiar enllaços, dispositius, i preservar permisos, grups i propietaris. També suporta llistats de exclusió i connexió remota amb

*shell* segur (ssh) entre altres possibilitats. Per a més informació veure **man rsync**.

## **Realització de backups a través d'una xarxa**

Com ja hem dit abans, **rsync** permet fer còpies en una màquina remota. Això es podria fer utilitzant **rsh** o posant **rsync** en mode servidor, però això pot ser perillós perquè en una xarxa local alguna altra màquina podria estar “escoltant” la connexió i podria agafar informació confidencial. Per aquesta raó, **rsync** permet fer connexions segures amb **ssh**. Per realitzar les nostres proves utilitzarem la nostra pròpia màquina com servidor **ssh**. En primer lloc inicialitzeu el *daemon* de **ssh**.

## **Realització de còpies completes**

Creeu un directori per fer les còpies en la partició de *backups* i després feu la següent comanda:

```
rsync -avz /root -e ssh root@localhost:/backup/backup-rsync/
```

Nota: Perquè l'anterior comanda funcioni bé és necessari activar el compte del root i posar-li una contrasenya vàlida.

Quin és el significat de les opcions -avz de l'**rsync**?

- a archive mode
- v verbose
- z Comprimir.

Creeu un arxiu en el directori root amb:

```
echo "nou arxiu" > /root/arxiu_nou.txt
```

i torneu a fer el mateix **rsync** d'abans.

Ara esborreu l'arxiu que heu creat abans i torneu a sincronitzar.

Què ha passat amb el fitxer esborrat?

Que encara hi es.

Amb quin paràmetre podríeu sincronitzar exactament els dos directoris?

```
rsync -avz --delete /root -e ssh root@localhost:  
/backup/backup-rsync.
```

Com faríeu per copiar tots el arxius del directori /home excepte els que tenen extensió .txt?

```
--exclude "*"*.txt'
```

Quina diferencia hi ha entre fer `rsync /source /destí` i `rsync /source/ /destí/`?

`/source` → Busca la carpeta mare i copia tot a dins  
`/source/` → Busca carpetes `/source/` a tot l'arbre de directoris descrit

## Realització de còpies incrementals inverses

Com hem vist a l'apartat anterior, cada vegada que realitzem una còpia i sincronitzem, el directori en què tenim el *mirror* queda exactament igual que el directori d'origen. Això és un problema perquè no tenim control dels canvis realitzats. Per solucionar aquest problema podem utilitzar l'opció `--backup` i `--backup-dir`. Els *backups* generats amb les opcions `--backup` i `--backup-dir` es diuen inversos perquè la còpia completa es la més recent i no la més antiga com amb **tar**. Amb aquesta opció la còpia completa correspon a la última data en que s'ha fet el *backup* i les incrementals a les dels dies anteriors.

A continuació teniu un script senzill per fer *backups* incrementals amb **rsync**. Completeu-lo amb les dades que facin falta.

```
#!/bin/bash

SOURCE_DIR=/root

DEST_DIR= /backUp / back-up-rsync

# excludes file: list of files to exclude

EXCLUDES=

# the name of the backup machine

BSERVER= root@localhost : /backUp / back-up-rsync

# the name of the incremental backups directory

# put a date command for: year month day hour minute second
```



```
BACKUP_DATE= date +%Y%m%d%H%M%S

# options for rsync

OPTS="--ignore-errors --delete-excluded --exclude-from=$EXCLUDES \
--delete --backup --backup-dir=$DEST_DIR/$BACKUP_DATE -av"

# now the actual transfer

rsync $OPTS $SOURCE_DIR root@$BSERVER:$DEST_DIR/complet
```

Ara creeu un fitxer arxiu.txt i sincronitzeu-lo amb l'script anterior. Després modifiqueu aquest arxiu i torneu a sincronitzar. Finalment esborreu el fitxer arxiu.txt i feu una sincronització més.

Què observeu en modificar un fitxer i fer una sincronització? I en esborrar-lo?

*funciona correctament..*

### **Realització de còpies incrementals inverses tipus snapshot (3)**

Una possibilitat que dóna *rsync* és fer *backups* incrementals on, utilitzant una propietat dels enllaços durs, és possible fer que les còpies incrementals semblin còpies completes. Per això analitzarem primer algunes propietats dels enllaços durs.

#### **Repàs d'enllaços durs**

El nom d'un fitxer no representa el fitxer mateix, per al sistema és només un enllaç dur al *inode*. Això permet que un fitxer (*inode*) pugui tenir més d'un enllaç dur. Per exemple si teniu un fitxer *file\_a* es pot crear un enllaç cridat *file\_b*:

In *file\_a* *file\_b*

Amb la comanda **stat** es pot saber quants enllaços durs té un fitxer:

**stat filename**

Com es pot comprovar que `file_a` i `file_b` pertanyen al mateix *inode*?

*Comprovant el comp inode.*

Què passa amb el fitxer `file_b` si es fa un canvi a `file_a`?

*canvia la data de modif*

I si es fa un canvi de permisos a `file_a`?

*es posen els mateixos permisos,*

I si copiem un altre fitxer sobreescrivint el fitxer `file_a` (`cp file_c file_a`)? I si es sobre escriu amb l'opció `--remove-destination`?

*el inode*  *no canvia*

I què passa amb `file_b` si `file_a` és esborrat?

*perde un link*

La comanda **cp** té una opció per fer còpies en què realment no es fa una còpia sinó un enllaç dur (**cp -l**). Una altra opció interessant es **-a** que fa una còpia recursiva i preserva els permisos de accés, els temps i els propietaris dels fitxers.

### **Backups tipus “snapshot” amb rsync i cp -al**

Es poden combinar **rsync** i **cp -al** per crear *backups* que semblin múltiples còpies completes d'un sistema de fitxers sense que sigui necessari gastar tot l'espai en disc requerit per totes les còpies, en resum es podria fer:

```
rm -rf backup.3
mv backup.2 backup.3
mv backup.1 backup.2
cp -al backup.0 backup.1
rsync -a --delete source_directory/ backup.0/
```

Si les comandes anteriors s'executen cada dia, `backup0`, `backup1`, `backup2` i `backup3` apareixeran com si fossin còpies completes del directori `source_directory` com estava avui, ahir, abans d'ahir i tres dies abans respectivament. Però en realitat l'espai extra serà igual a la grandària del directori `source_directory` més la grandària total dels canvis dels últims tres dies. Exactament el mateix que un *backup* complet més els *backups*

incrementals con heu fet abans amb **tar** i el mateix **rsync**. L'únic problema és que els permisos i propietaris de les còpies anteriors serien els mateixos que els de la còpia actual.

Hi ha una opció d'**rsync** que fa directament la còpia amb enllaços durs ( **--link-dest**) i d'aquesta manera no seria necessari utilitzar la comanda **cp**, a més a més que preserva els permisos i propietaris de les còpies anteriors. Amb aquesta opció l'esquema platejat anteriorment quedaria així:

```
rm -rf backup.3
mv backup.2 backup.3
mv backup.1 backup.2
mv backup.0 backup.1
rsync -a --delete --link-dest=../backup.1 \
    source_directory/ backup.0/
```

### ***Script per fer backups tipus snapshot***

Per fer *backups* del directori **/root** utilitzarem l'script **backup-rsync-snapshot.sh**, que està disponible en el servidor ftp al directori **sources**. En primer lloc modifiqueu les variables de la secció “file locations” per posar els valors adequats del vostre sistema. Després completeu la secció amb la comanda **rsync** amb els valors apropiats per fer la còpia tipus *snapshot*.

Com ha quedat la comanda **rsync** a l'script?

Ara feu modificacions als fitxers del directori origen (per exemple crear un nou fitxer, modificar el contingut i la data d'accés a un fitxer o esborrar un fitxer) i torneu a executar l'script. Feu això varies vegades fins que tingueu una còpia actual i tres còpies anteriors.

Què observeu al modificar un fitxer i fer una sincronització?

Quina és la grandària del directori **/backup.0** i dels altres directoris **backup1**, **backup2** i **backup3**?

Finalment, voldríem fer una restauració. Canvieu el nom del directori **/root** per

simular una pèrdua de dades. Feu una restauració d'aquest directori amb la còpia de seguretat més recent.

Amb quina comanda es pot fer això?

## Referències

- (1) Lars Wirzenius, Joanna Oja, Stephen Stafford, Alex Weeks, **The Linux System Administrator's Guide** Version 0.9, <http://tldp.org/LDP/sag>
- (2) Eelen Frisch, **Essential System Administration**. O'Reilly. 2002.
- (3) Mike Rubel. **Easy Automated Snapshot-Style Backups with Linux and Rsync**. [http://www.mikerubel.org/computers/rsync\\_snapshots/](http://www.mikerubel.org/computers/rsync_snapshots/)
- (4) **Rsnapshot**. *a remote filesystem snapshot utility based on rsync for making backups of local and remote systems*. <http://www.rsnapshot.org/>

## Apèndix. Codi de l'script per còpies tipus *snapshot*

```
#!/bin/bash

# -----

# mikes handy rotating-filesystem-snapshot utility
# http://www.mikerubel.org/computers/rsync_snapshots
# Modified by Mauricio Alvarez: http://people.ac.upc.edu/alvarez
# -----

# ----- system commands used by this script-----

ID=/usr/bin/id

ECHO=/bin/echo
```

```
MOUNT=/bin/mount

RM=/bin/rm

MV=/bin/mv

CP=/bin/cp

TOUCH=/usr/bin/touch

RSYNC=/usr/bin/rsync


# ----- file locations -----

MOUNT_DEVICE=

SNAPSHOT_MOUNTPOINT=

SNAPSHOT_DIR=

EXCLUDES=

SOURCE_DIR=


# ----- the script itself-----


# make sure we're running as root

if (( `ID -u` != 0 )); then { $ECHO "Sorry, must be root. Exiting..."; exit; } fi


# attempt to remount the RW mount point as RW; else abort

$MOUNT -o remount,rw $MOUNT_DEVICE $SNAPSHOT_MOUNTPOINT ;

if (( $? )); then

{

    $ECHO "snapshot: could not remount $SNAPSHOT_MOUNTPOINT readwrite";

    exit;

}

fi;
```

```
# rotating snapshots of /$SNAPSHOT_DIR
```

```
# step 1: delete the oldest snapshot, if it exists:
```

```
if [ -d $SNAPSHOT_MOUNTPOINT/$SNAPSHOT_DIR/daily.3 ] ; then
```

```
    $RM -rf $SNAPSHOT_MOUNTPOINT/$SNAPSHOT_DIR/daily.3
```

```
fi
```

```
# step 2: shift the middle snapshotss back by one, if they exist
```

```
if [ -d $SNAPSHOT_MOUNTPOINT/$SNAPSHOT_DIR/daily.2 ] ; then
```

```
    $MV      $SNAPSHOT_MOUNTPOINT/$SNAPSHOT_DIR/daily.2
$SNAPSHOT_MOUNTPOINT/$SNAPSHOT_DIR/daily.3
```

```
fi
```

```
if [ -d $SNAPSHOT_MOUNTPOINT/$SNAPSHOT_DIR/daily.1 ] ; then
```

```
    $MV      $SNAPSHOT_MOUNTPOINT/$SNAPSHOT_DIR/daily.1
$SNAPSHOT_MOUNTPOINT/$SNAPSHOT_DIR/daily.2
```

```
fi
```

```
if [ -d $SNAPSHOT_MOUNTPOINT/$SNAPSHOT_DIR/daily.0 ] ; then
```

```
    $MV      $SNAPSHOT_MOUNTPOINT/$SNAPSHOT_DIR/daily.0
$SNAPSHOT_MOUNTPOINT/$SNAPSHOT_DIR/daily.1
```

```
fi;
```

```
# step 3: rsync from the system into the latest snapshot
```

```
$RSYNC
```

```
# complete here what is missing for the rsync command:
```

```
# - basic options:

# - excludes:

# - --link-dest=

# - source and destination directories


# step 5: update the mtime of daily.0 to reflect the snapshot time
$TOUCH $SNAPSHOT_MOUNTPOINT/$SNAPSHOT_DIR/daily.0 ;

# now remount the RW snapshot mountpoint as readonly
$MOUNT -o remount,ro $MOUNT_DEVICE $SNAPSHOT_MOUNTPOINT ;

if (( $? )); then
{
    $ECHO "snapshot: could not remount $SNAPSHOT_MOUNTPOINT readonly";
    exit;
} fi;
```

/etc/ssh/sshd-config

PermitRootLogin

→ Restart daemon



~~prohibit password~~  
↓  
yes 😊