

**Índex**

[1. Introducció 3](#_gjdgxs)

[1.1. Objectiu 4](#_30j0zll)

[2. Abans de començar 4](#_1fob9te)

[3. Partició per emmagatzemar les còpies de seguretat. 4](#_3znysh7)

[Quina modificació és necessària fer perquè aquesta nova partició es munti automàticament durant el boot amb mode de només lectura? 4](#_2et92p0)

[3. Realització de còpies amb TAR 5](#_tyjcwt)

[3.1. Realització de còpies completes 5](#_3dy6vkm)

[3.2. Realització de còpies incrementals 6](#_1t3h5sf)

[3.3. Restauració d'una còpia de seguretat 11](#_4d34og8)

[3.4. Restauració d'un fragment 13](#_2s8eyo1)

[4. Realització de còpies usant RSYNC 14](#_17dp8vu)

[4.1. Realització de backups a través d'una xarxa 14](#_3rdcrjn)

[$ echo “nou arxiu” > /root/arxiu\_nou.txt 15](#_26in1rg)

[4.2. Realització de còpies incrementals inverses 18](#_lnxbz9)

[4.3. Realització de còpies incrementals inverses tipus snapshot 20](#_35nkun2)

[Repàs d'enllaços durs 20](#_1ksv4uv)

[4.3.1 Backups tipus “snapshot” amb rsync i cp -al 21](#_44sinio)

[4.4. Script per fer backups tipus snapshot 22](#_2jxsxqh)

[5. Referències 23](#_z337ya)

[6. Apèndix. Codi de l'script per còpies tipus snapshot 23](#_3j2qqm3)

# 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 .

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

* 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 el fitxers de configuració del sistema i els fitxers dels usuaris (normalment ubicats a /etc i /home respectivament) que el fitxers temporals o els binaris del sistema (/tmp i /bin)
* Decidir la freqüència i el tipus de planificació de les còpies de seguretat. Això depén de la variabilitat de les dades. Una base de dades pot necessitar múltiples copies de seguretat diàries, mentre que un servidor web pot requerir només una copia diària i altres sistemes de fitxers poden requerir només una copia 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 copies 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 copia 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. Com a primer pas en aquest objectiu 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 diferent estratègies de còpies de seguretat.

## Objectiu

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

# **Abans de començar**

contesteu les següents preguntes abans de començar:

1. Com es pot empaquetar i desempaquetar un(s) fitxer(s) utilitzant la comanda tar?

Per emaquetar hem d’utilitar:

tar -cvf nom\_arxiu.tar fitxer1 fitxer2 ..

Aquí, -c indica crear un arxiu, -v és per veure el procés (verbose), i -f especifica el nom de l'arxiu resultant. Pots afegir múltiples fitxers i directoris separant-los amb espais.

Per desempaquetar hem d’utilitzar:

tar -xvf nom\_arxiu.tar

On -x indica extreure el contingut de l'arxiu.

1. 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)?

Un enllaç dur és una referència directa al contingut físic d'un fitxer al disc. Quan crees un enllaç dur a un fitxer, estàs creant essencialment una altra entrada al mateix bloc de dades al disc. Això significa que el fitxer original i l'enllaç dur tenen el mateix identificador de node d'índex (inode) i qualsevol canvi en un es reflectirà en l'altre.

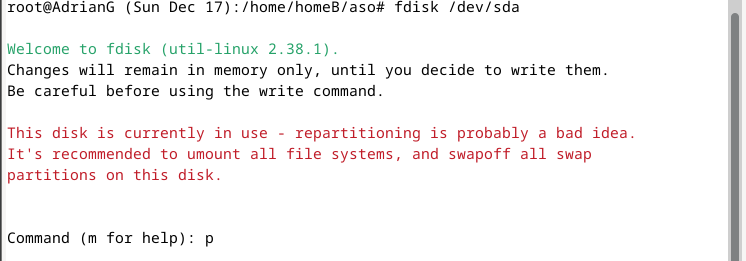
La diferència clau entre fer una còpia (cp file\_a file\_b) i crear un enllaç dur (ln file\_a file\_b) rau en com es tracta el fitxer en el sistema de fitxers. Amb cp, es crea una nova instància del fitxer, que posseeix un inode únic; això significa que els fitxers resultants són independents, i els canvis en un no afecten l'altre. En contrast, quan es crea un enllaç dur, no es genera una còpia física del fitxer, sinó una nova entrada al sistema de fitxers que apunta al mateix inode que l'original. Això fa que el fitxer original i l'enllaç dur siguin intercanviables, compartint els mateixos continguts, de manera que qualsevol canvi en un es reflecteix automàticament en l'altre.

# **Partició per emmagatzemar les còpies de seguretat.**

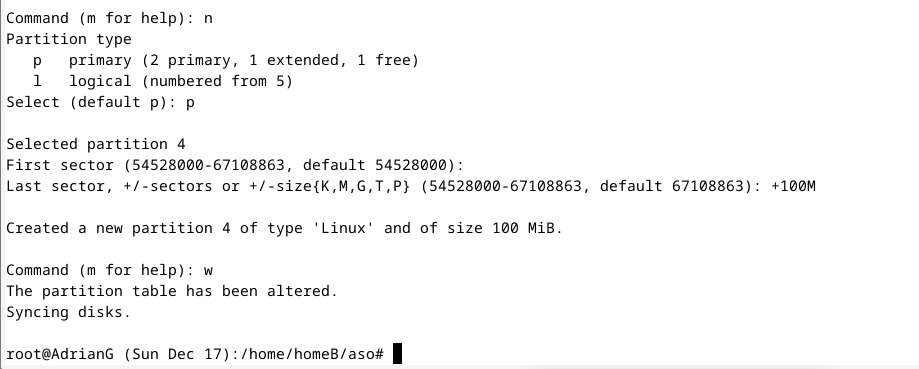
Creeu una nova partició a l'espai lliure que teniu al disc i doneu-li format extended3. Munteu aquesta partició sobre el directori /backup de forma que tan sols root tingui permisos d'accés. La resta d'usuaris no han de tenir ni tan sols permís de lectura al directori ja que el contingut de les còpies 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?

Primer de tot entrem al fdisk. Per fer-ho utilitzem “fdisk /dev/sda”.

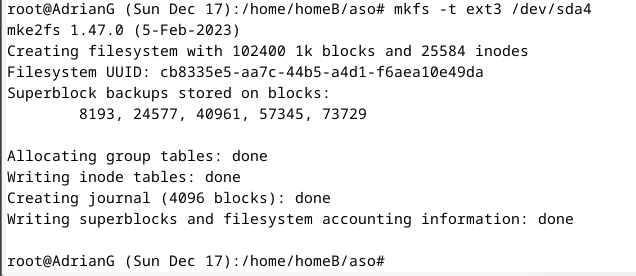


Creem una nova partició utilitzant “n”. Li asignem el valor de partició primària, i li donem un espai de 100 MB. Si després volem ampliar l’espai ja ho farem. Finalment guardem els canvis amb “w”.



Donem el sistema de fitxers ext3 amb la següent comanda.

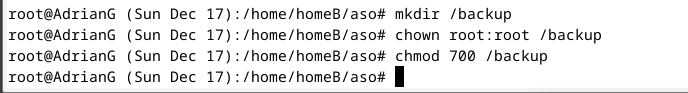
mkfs -t ext3 /dev/sda4



Ara hem de crear la carpeta del /backup amb mkdir. Una vegada creada utilitzem les següents comandes per donar-li només permisos al root per poder entrar a la carpeta.

chown root:root /backup

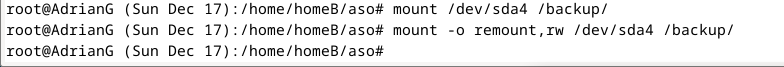
chmod 700 /backup



Montem el directori al sda4, i el remontem per a que sigui read-write amb les següents comandes.

mount /dev/sda4 /backup

mount -o remount,rw /dev/sda4 /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

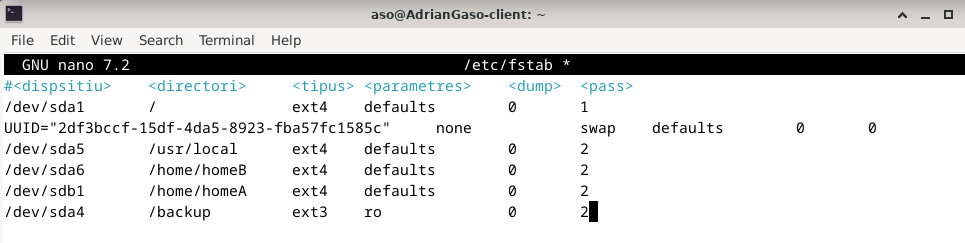
Muntar per lectura i escriptura:

$ 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?

Obrim el fstab fent “nano /etc/fstab”, i afegim la següent linia.

/dev/sda4 /backup ext3 ro 0 2

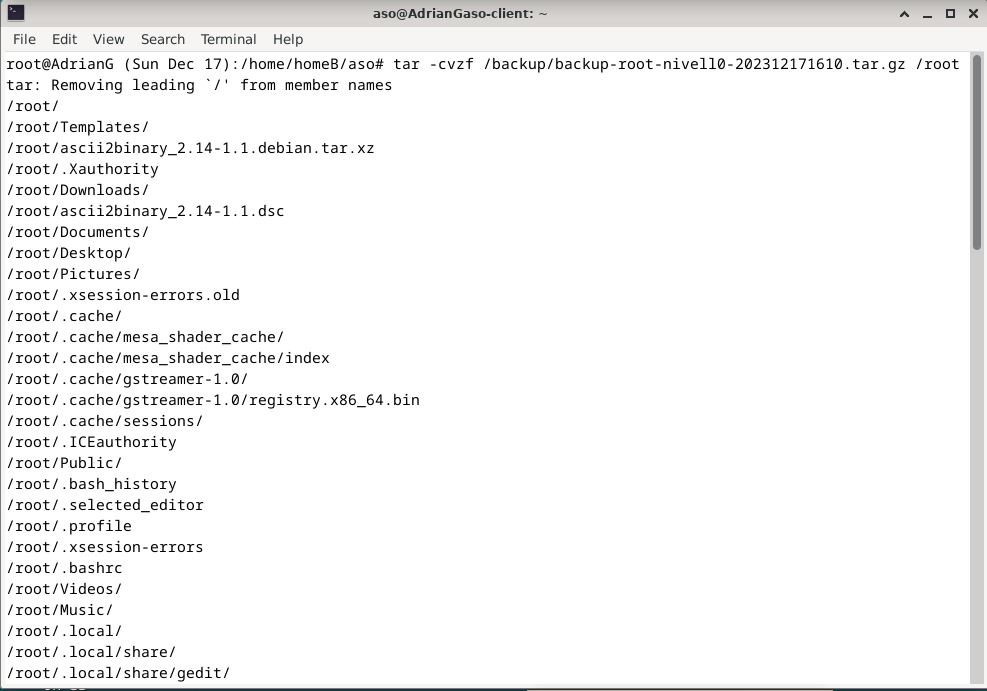


# Realització de còpies amb TAR

## 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, ...).

Utilitzem la comanda “tar -cvzf /backup/backup-root-nivell0-202312171610.tar.gz (directori on ho volem guardar) /root (carpeta que volem comprimir)”.



Com es pot fer perquè el nom del fitxer de *backup* inclogui automàticament la data del *backup*? per exemple que sigui backup-etc-nivell0-202112041030 (any mes dia hora minut segon). Utilitza la comanda **date.**

Utilitzem la comanda

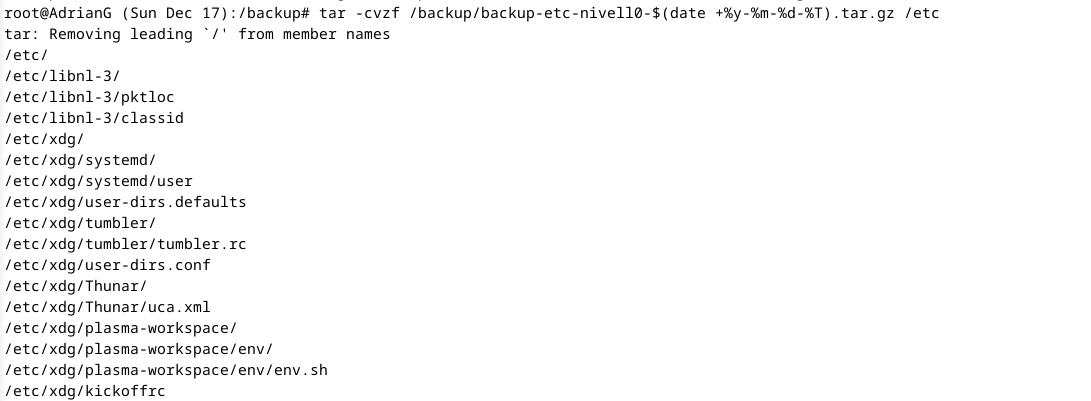
“tar -cvzf /backup/backup-root-nivell0-date-$(date +%y-%m-%d-%T).tar.gz /root”.

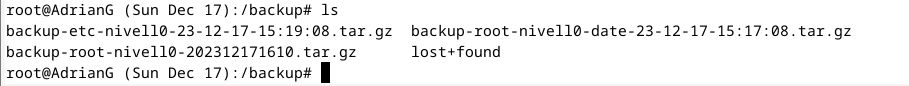




Quina comanda heu fet servir per fer la còpia completa del directori /etc?

tar -cvf /backup/backup-etc-nivell0-$(date +%y-%m-%d-%T).tar.gz /etc





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

Comprimir un fitxer de backup no és aconsellable principalment perquè pot augmentar el risc de corrupció de dades: si l'arxiu comprimit es danya, es podrien perdre totes les dades que conté. A més, la compressió pot complicar la restauració selectiva de dades, ja que caldria descomprimir tot l'arxiu per accedir a parts específiques. També, els processos de compressió i descompressió requereixen temps i recursos addicionals, el que pot ser inconvenient en situacions d'urgència on el temps de recuperació és crític.

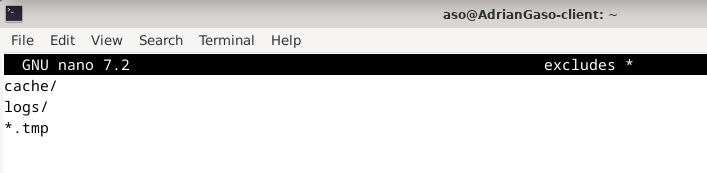
Si volguéssim comprimir el fitxer de *backup* quina opció afegiríem a la comanda tar?

Fariem “tar -cvzf backup.tar.gz /backup”. La z en permet comprimir en gzip el fitxer /backup

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.*

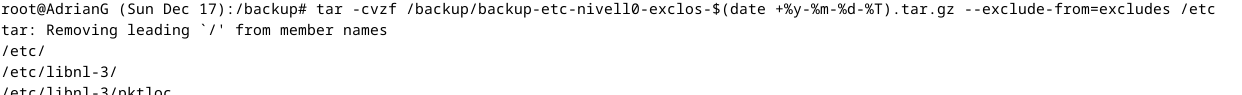
Feu de nou la copia completa però aquesta ocasió excloent el fitxers que siguin al fitxer *excludes*. (poseu el nom d'alguns fitxers al fitxer *excludes).* Quina opció de **tar** permet excloure un llistat de fitxers del *backup*?

Creem el fitxer excludes i fiquem alguns arxius de proba.



Aquest llistat exclouria tots els continguts dins dels directoris cache i logs, i qualsevol fitxer que acabi amb .tmp.  
Ara utilitzem la següent comanda.

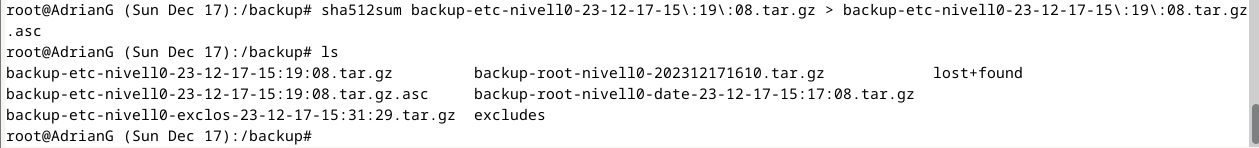
tar -cvzf /backup/backup-etc-nivell0-exclos-$(date +%y-%m-%d-%T).tar.gz. --exclude-from=excludes /etc



A més a més de protegir el directori de les còpies de seguretat és importat utilitzar algun mecanisme que permeti verificar que el fitxers de *backup* no hagin estat modificats després d'haver-los creat. Per això es comú utilitzar algun mecanisme de signatura digital, com son els *hashs* SHA, que permeten verificar la integritat d'un fitxer.

Una vegada hagueu realitzat la còpia, utilitzeu la comanda **sha512sum**(utilitzeu el **man** per saber com fer servir aquesta comanda) amb la còpia del directori i guardeu-vos el resultat en un fitxer <nomdelacopia>.asc.

Utilitzem la comanda sha512sum i fem:  
sha512sum backup-etc-nivell0-23-12-17-15\:19\:08.tar.gz > backup-etc-nivell0-23-12-17-15\:19\:08.tar.gz.asc



## Realització de còpies incrementals

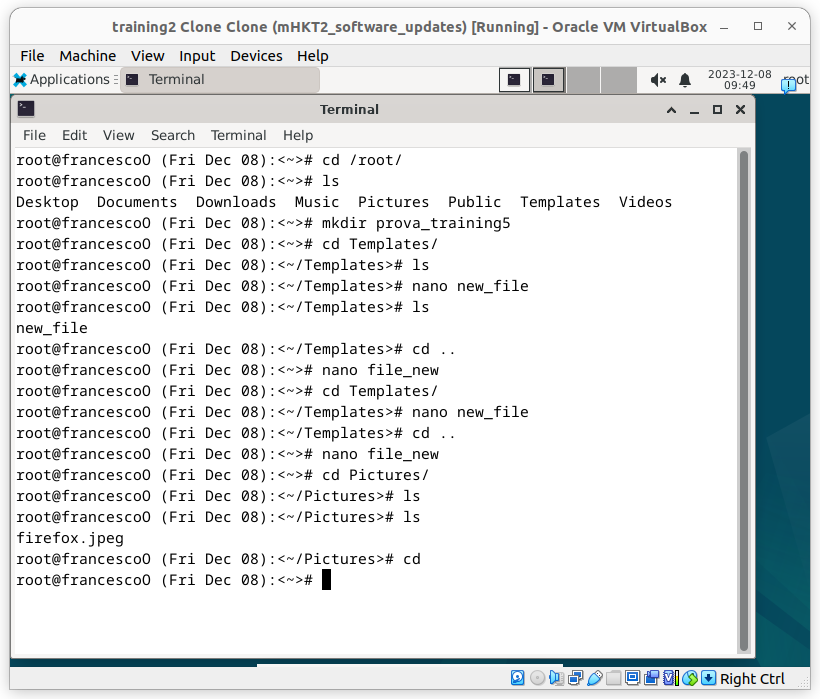
Per tal de dur a terme còpies incrementals, serà necessari modificar alguns fitxers dins el directori /root :

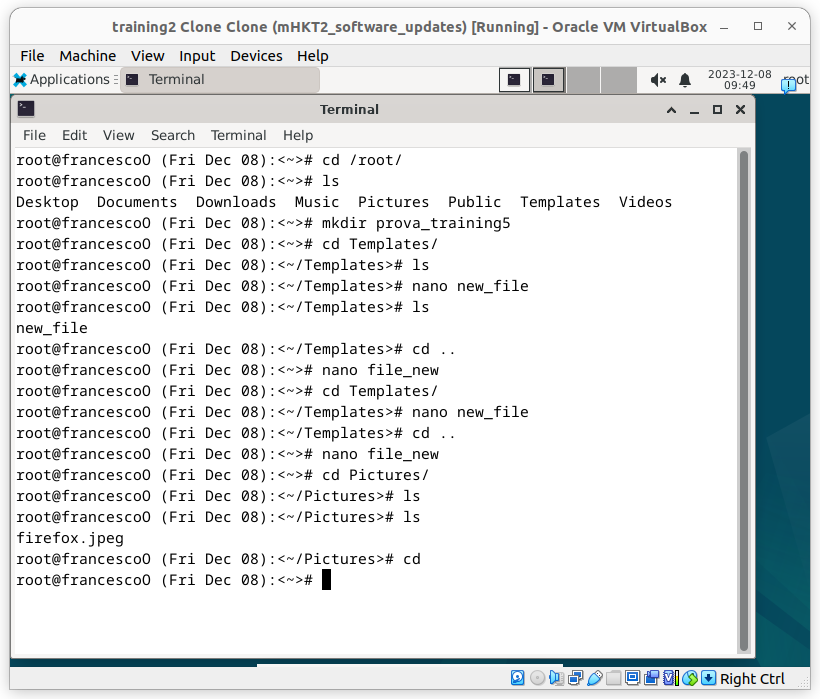


Per fer còpies incrementals amb tar tenim l'opció **--newer** que només inclou els fitxers que hagin estat modificats des d'una data determinada. Aquesta data es pot especificar de dues formes: la primera, posant-la directament, per exemple: “--**newer=”2021-11-28 12:10”**. La segona manera consisteix 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”.**

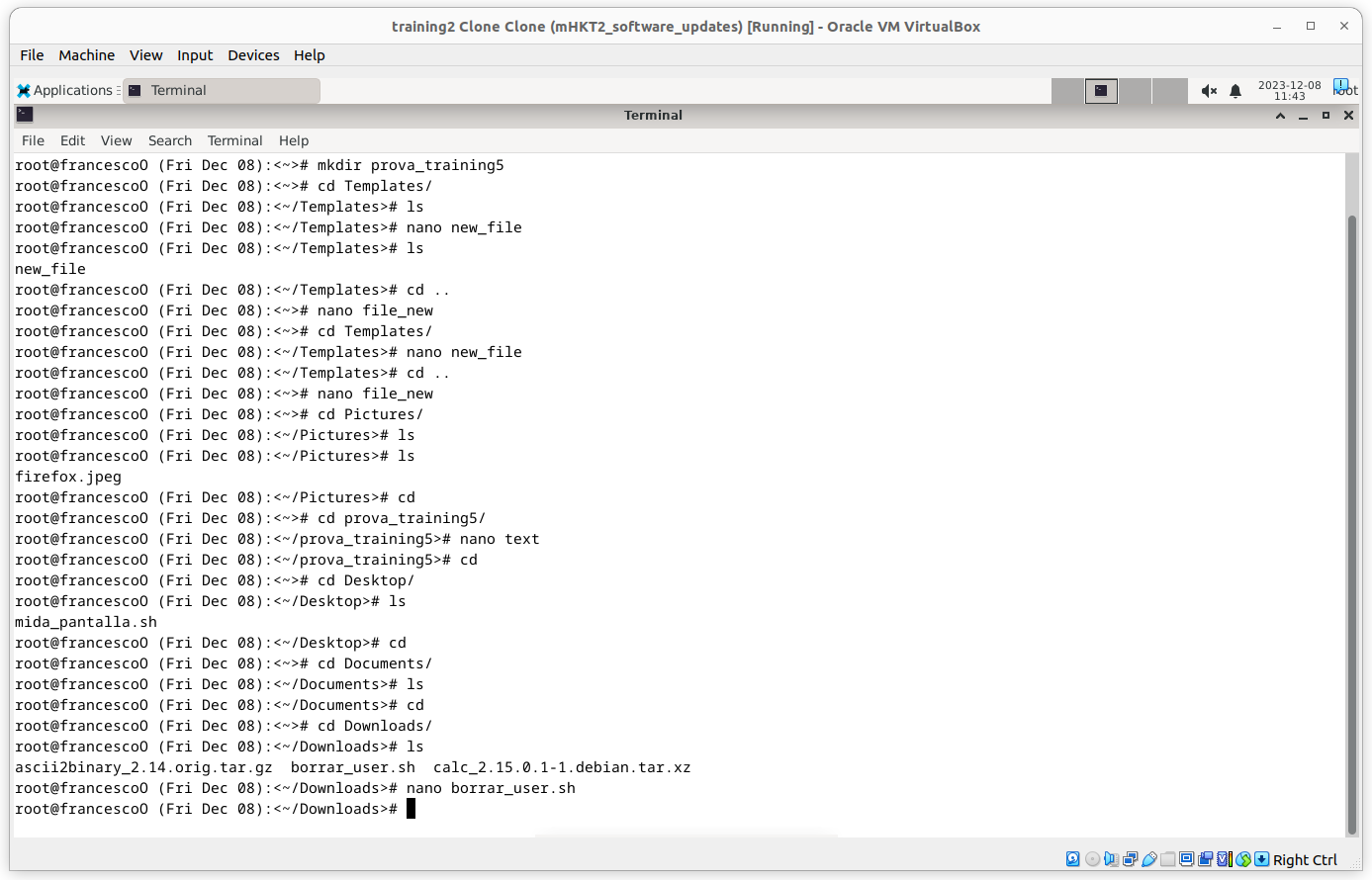
Ara realitzeu una còpia incremental del directori /root respecte a la còpia completa que heu fet abans usant la comanda tar. Feu també un **sha512sum** i guardeu-lo en un arxiu.asc amb el nom diferent al nom que heu utilitzat abans.

Hem creat aquests arxius i fitxers:

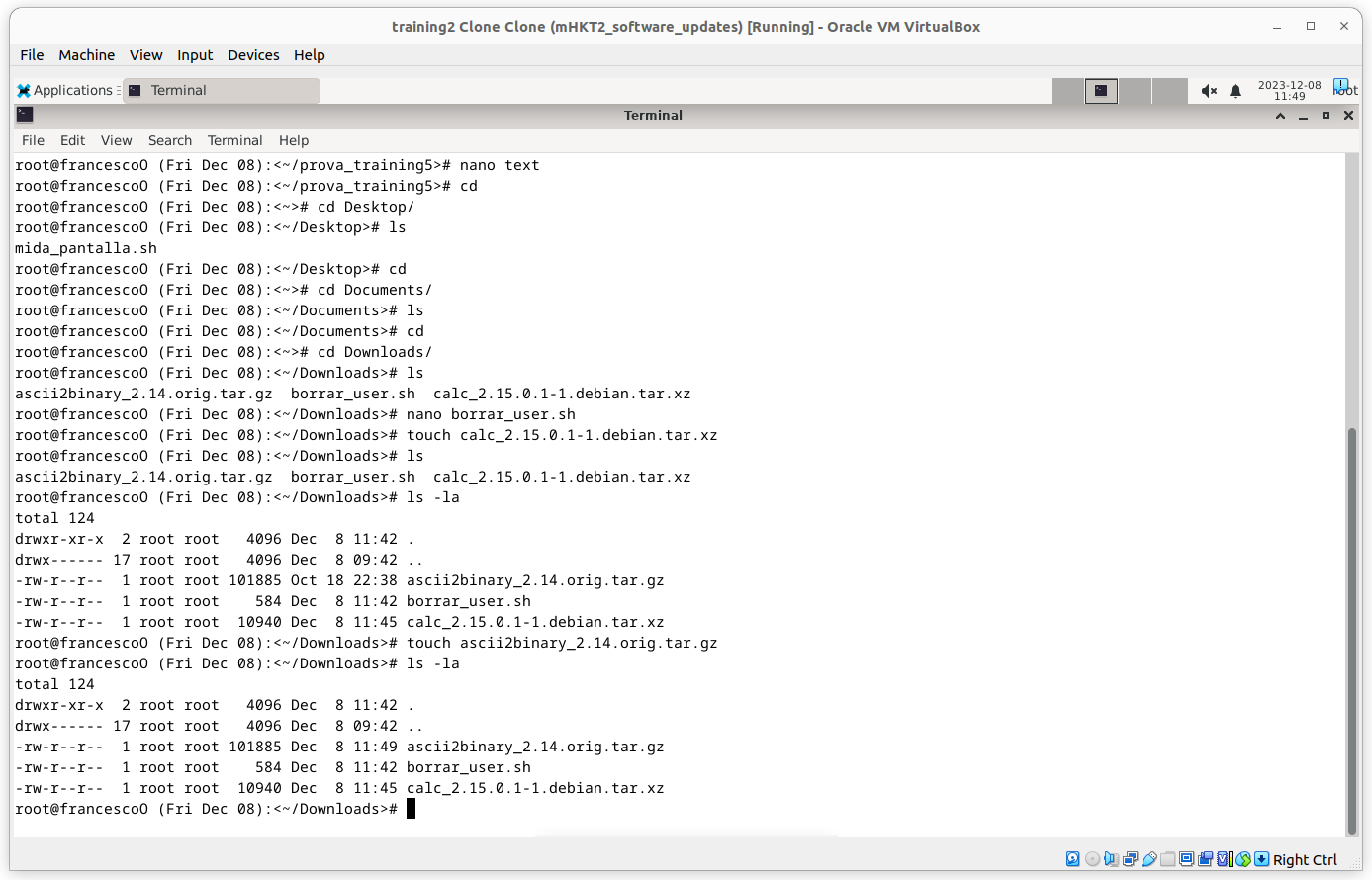




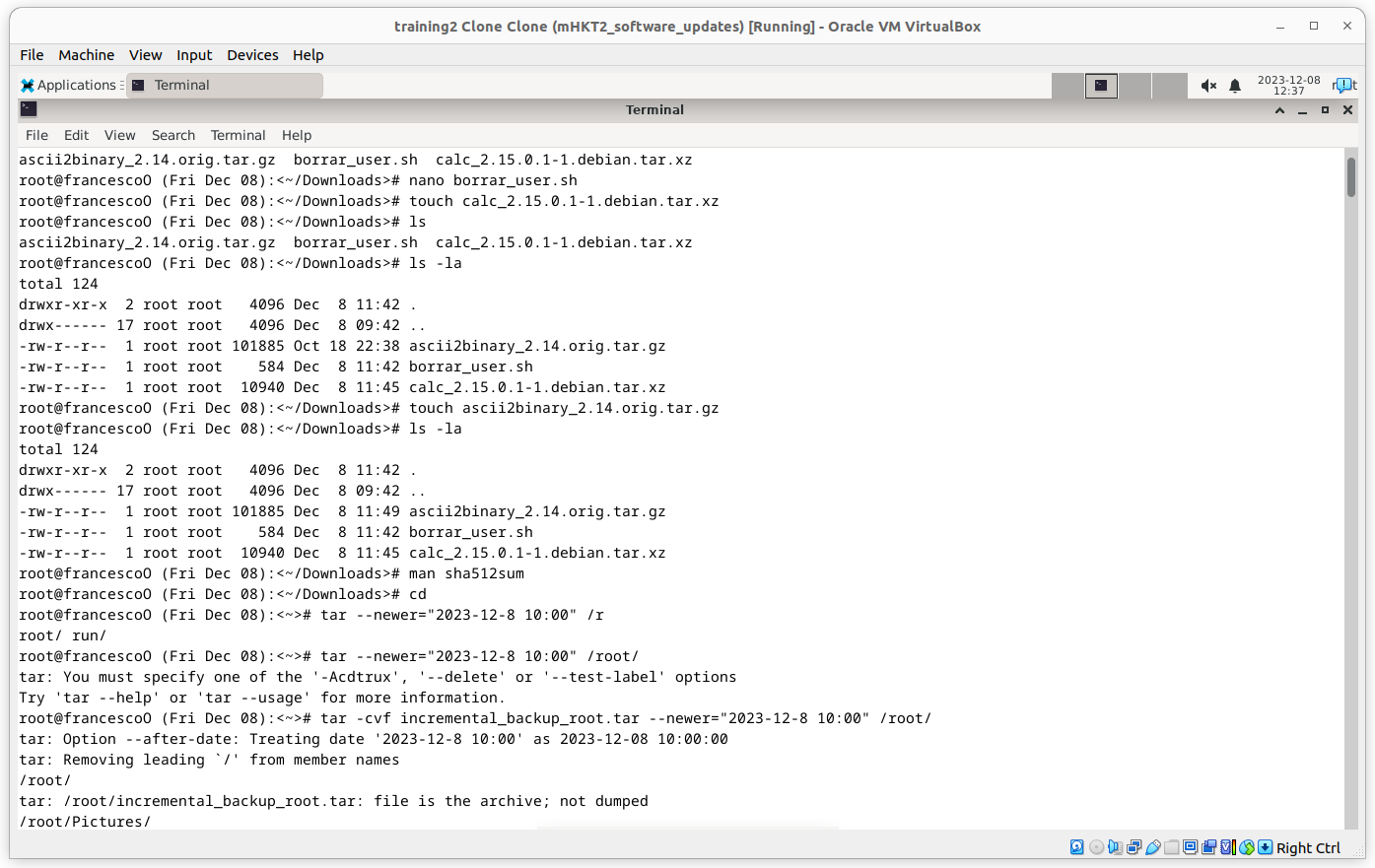
Hem modificat el contingut d’aquest arxiu:



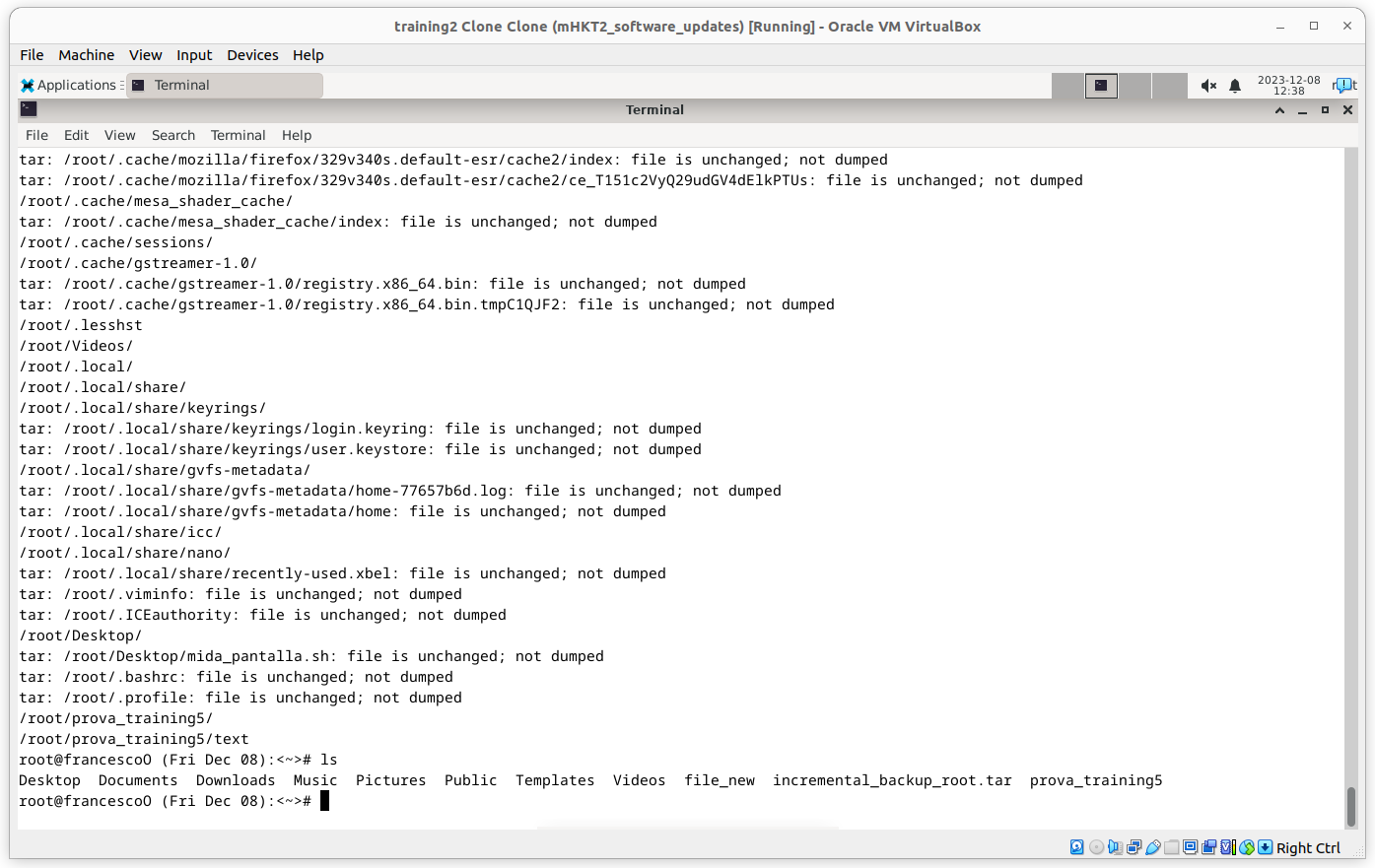
Hem canviat la data de modificació dels 2 arxius *.tar* amb la comanda touch:



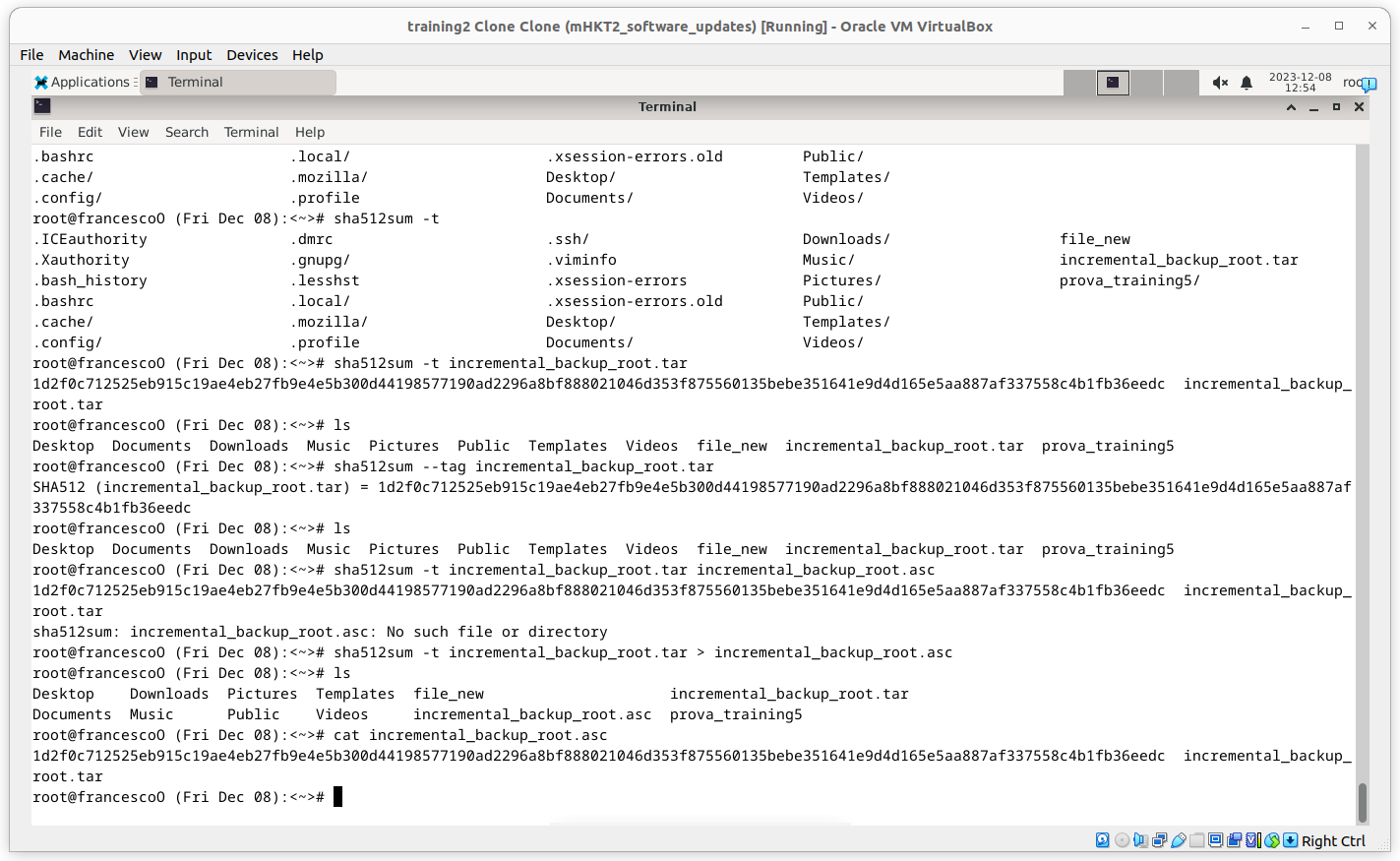
Imatge de la comanda utilitzada per fer la còpia incremental:



Imatge de la còpia incremental:



Si volem guardar el resultat del checksum del backup.tar podem direccionar-lo amb ‘>’ a l’arxiu que volguem:



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?

L'ús del fitxer .tar de la còpia completa per obtenir la data del backup per realitzar una còpia incremental pot presentar diversos problemes potencials:

1. Si el fitxer .tar es corromp o es modifica de qualsevol manera, això pot afectar la capacitat de determinar correctament la data de l'última còpia de seguretat.
2. Obtenir la data de la última còpia de seguretat a partir d'un fitxer .tar pot requerir que es descomprimeixi o es llegeixi el fitxer complet, la qual cosa pot ser molt ineficient, especialment si el fitxer és gran.
3. Si hi ha múltiples còpies de seguretat que s'executen en paral·lel o si el fitxer .tar s'actualitza o es modifica després de crear la còpia incremental, pot haver-hi inconsistències en les dades.

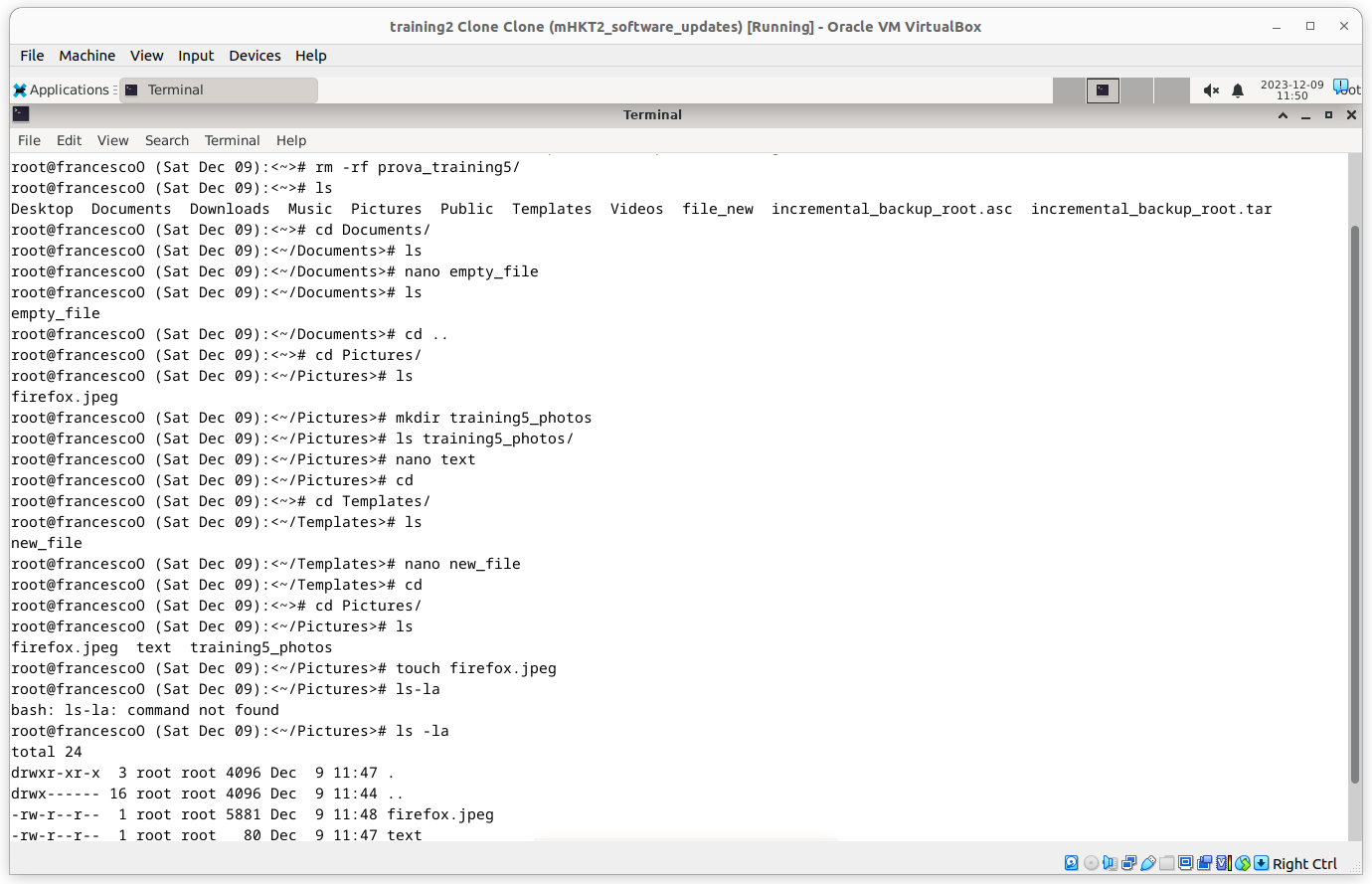
Aquests problemes es poden resoldre de diferents maneres:

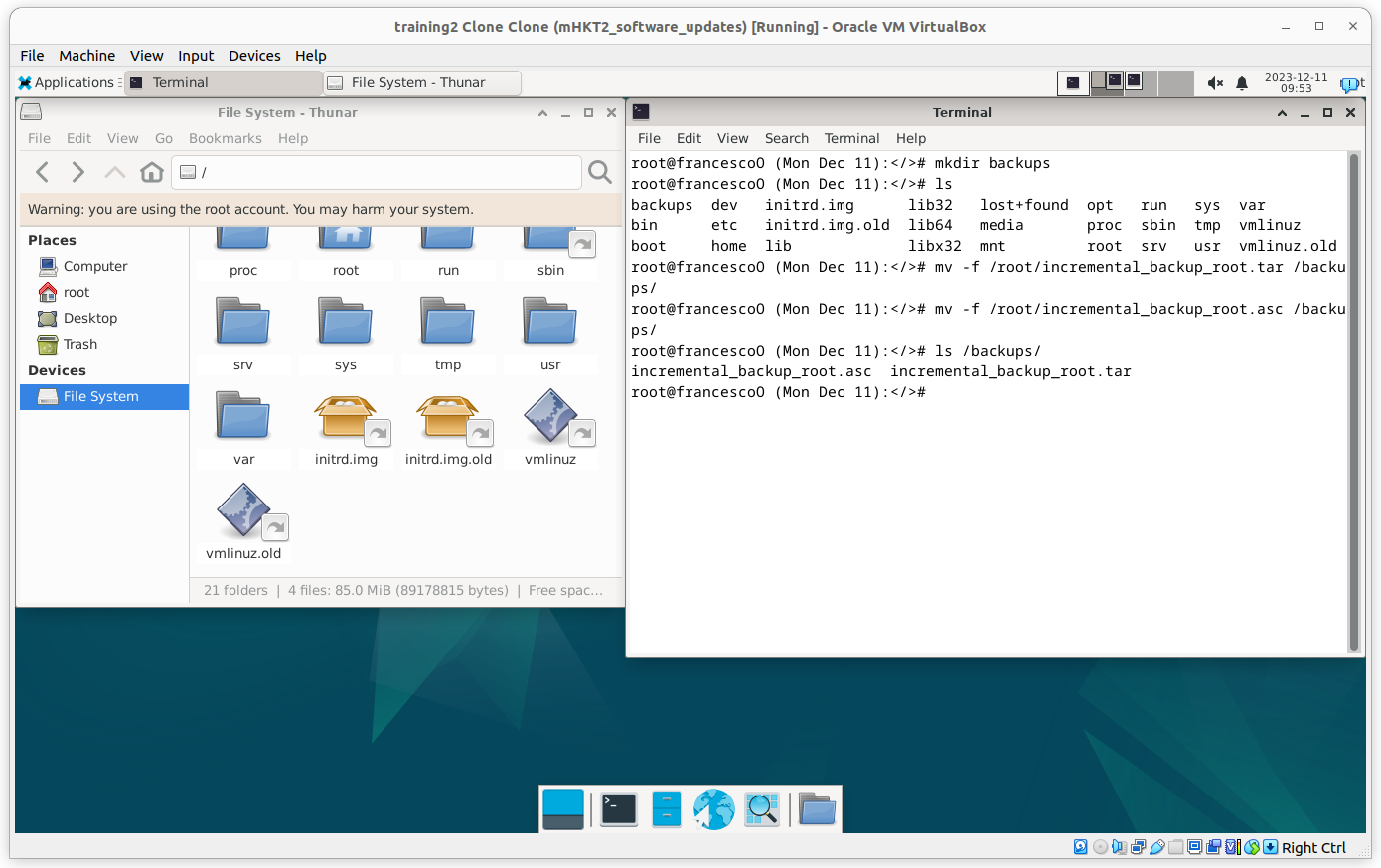
1. Utilitzar les metadades (autor, data de creació, data de modificació, mida del fitxer) del sistema de fitxers per obtenir la data de la última modificació dels fitxers.
2. Mantenir un registre separat de la data de l'última còpia de seguretat en un fitxer o base de dades.
3. Utilitzar eines de backup que suportin còpies de seguretat incrementals de manera nativa i que puguin manejar la lògica de determinació de quins fitxers han canviat des de l'última còpia de seguretat.
4. Fer servir checksums per validar la integritat dels fitxers abans de realitzar una còpia incremental, assegurant-se que els fitxers no s'han corromput o modificat de manera inesperada.
5. Considerar la possibilitat d'utilitzar còpies de seguretat basades en blocs en lloc de còpies basades en fitxers, la qual cosa pot ser més eficient i precisa per a determinar canvis.

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

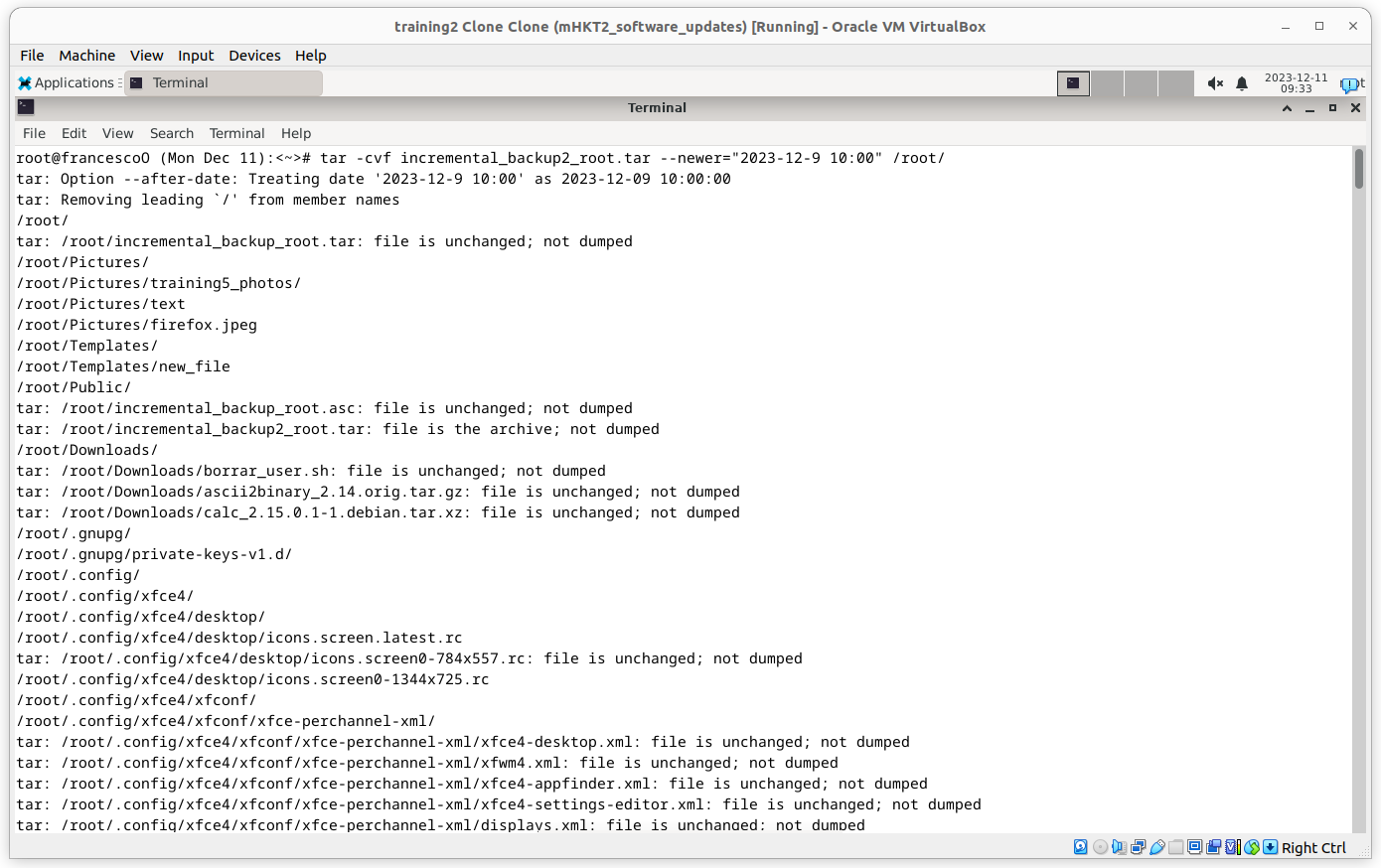
Realitzeu una segona còpia incremental del directori /root (respecte la primera còpia incremental) amb la comanda **tar**. També feu un **sha512sum** de la segona còpia incremental i poseu-li un nom apropiat.

Aquí hi ha una imatge de les modificacions que he fet per aquesta segona còpia incremental:

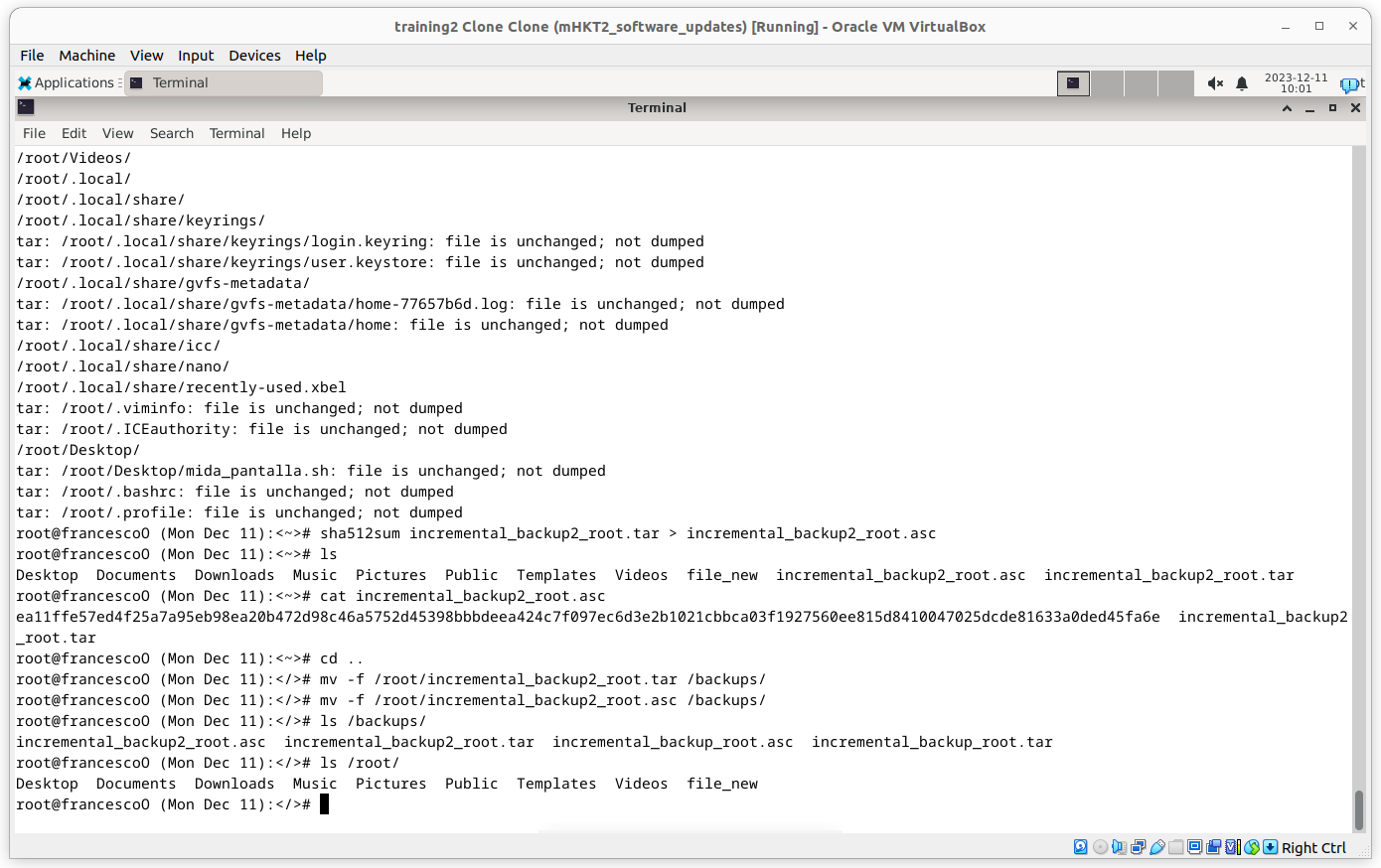


Per poder fer la còpia del root, sense que aquest tingui la còpia anterior, hem creat una carpeta backups fora del directori root, i hem mogut el .tar i .asc de la 1a còpia incremental:  


Segona còpia incremental:



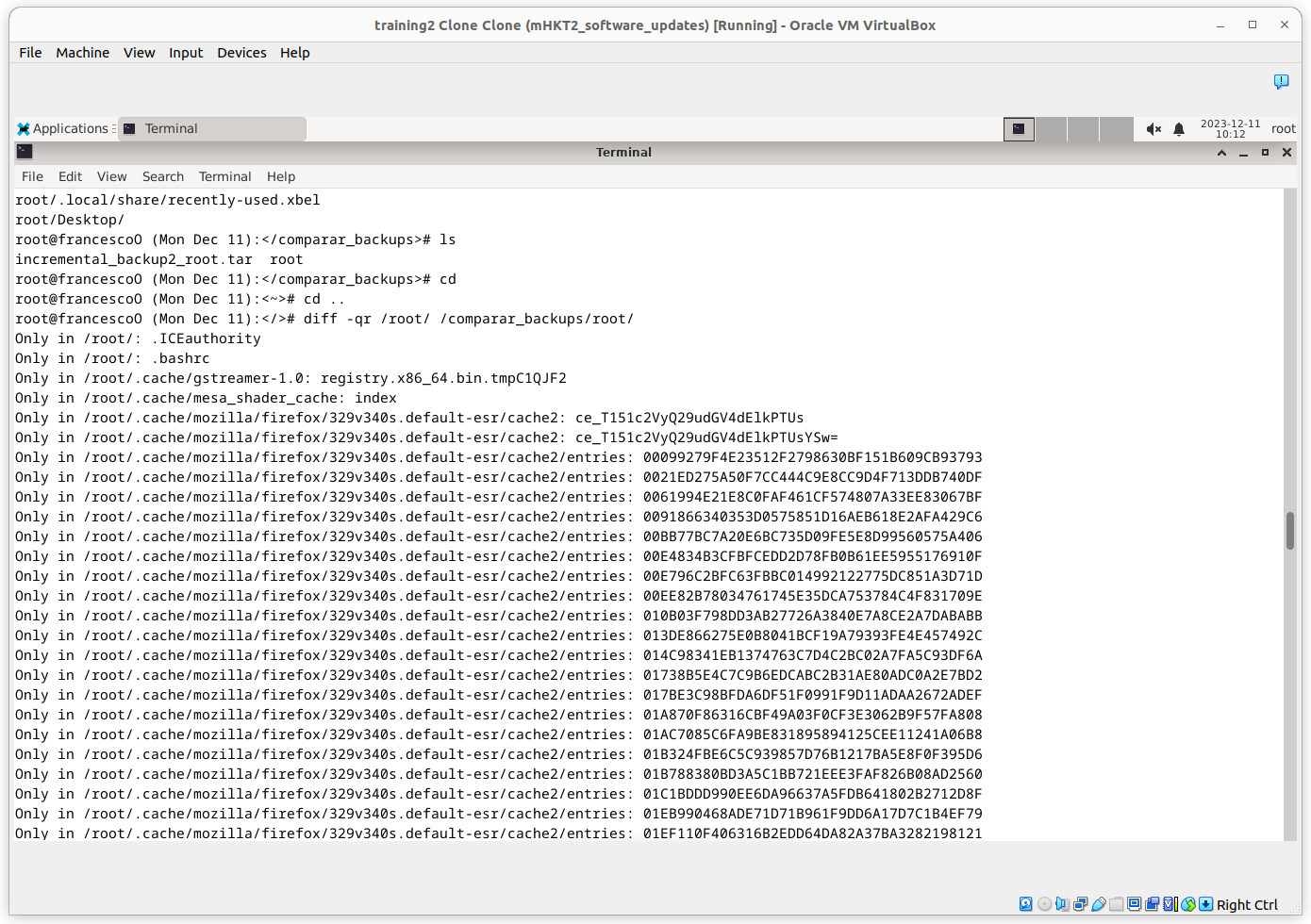
Imatge dels nous fitxers, .tar i .asc (checksum): els movem a la carpeta de backups per poder verificar que el contingut sigui el mateix.



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

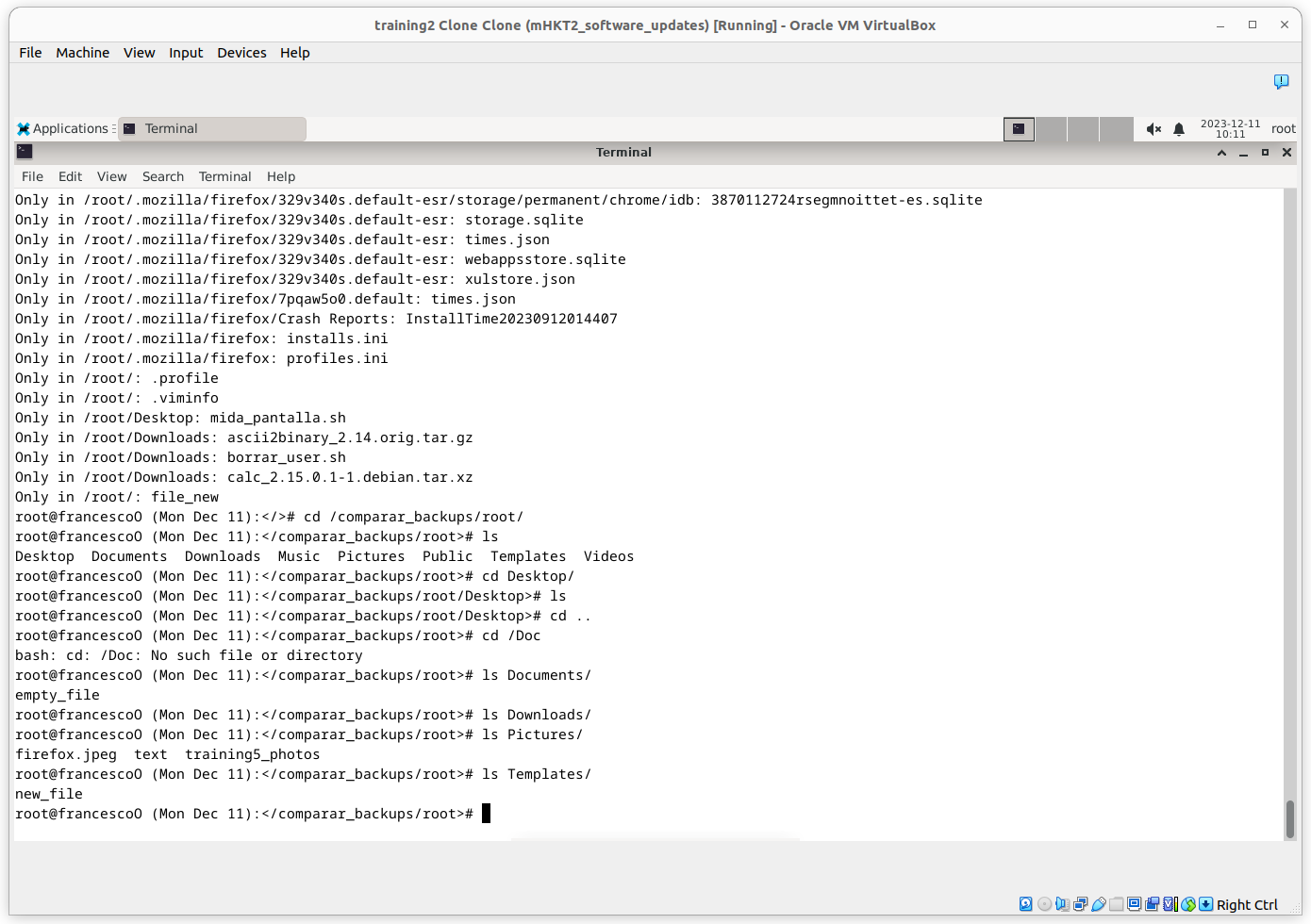
Per assegurar-nos que el contingut del fitxer de còpia és idèntic al directori original, hem de descomprimir la còpia incremental en un directori nou i comparar-lo amb el directori /root. Ho podem fer amb la comanda diff.

Per fer el diff utilitzem 2 opcions (-q i -r, el -q farà que només ens digui les diferències i el -r farà que miri dins els subdirectoris):



Quan fem el diff podem veure que hi ha moltíssimes diferències ja que a la còpia incremental només hem guardat els canvis nous que hem fet després de la primera còpia incremental.

Però podem comprovar que ha guardat els canvis fets després d’una data en concret (després de la primera còpia incremental):



Com es pot verificar, fent ús de la comanda **sha512sum**, la integritat d'una còpia de seguretat, és a dir, que el fitxer no ha estat modificat des que es va realitzar la còpia?

Per verificar la integritat hem de calcular el hash SHA-512 del fitxer de còpia, hem de comparar el nou hash amb el hash original. Si els dos coincideixen, la còpia no ha estat modificada. Si són diferents, el fitxer pot haver estat modificat o danyat.

Ho podem fer amb la comanda diff, creem un arxiu (en .asc) que contingui el resultat d’aquest checksum just després d’haver fet la còpia incremental, i quan vulguem comprovar si s’ha modificat el backup, tornem a fer el checksum i el comparem amb el primer chekcsum que vem fer. Si hi ha qualsevol diferència vol dir que ha estat modificat o danyat.

## Restauració d'una còpia de seguretat

Reanomeneu el directori /root per /root.old per simular l'efecte que es produiria si esborréssim el directori.

Per a renombrar el directori root utilitzarem la comanda mv: mv /root /root.old, aixó modificarà el nom de la nostre carpeta

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

L’ordre que s’ha de seguir es:

1. La copia completa
2. La copia incremental més antiga posible
3. La copia incremental més nova posible

D’aquesta manera no eliminarem ningún arxiu o el sobreescriurem de manera erronea

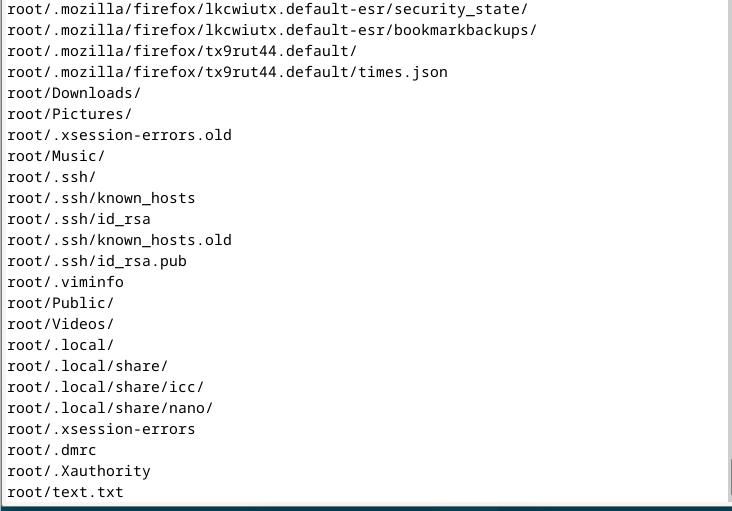
Ara restaureu la còpia de seguretat del directori /root, la qual cosa implica restaurar els tres fitxers que hem creat: la còpia completa i les dos incrementals.

Per a restaurar les nostres copies tindrem que descomprimir en l’ordre que hem dit anteriorment a la carpeta on es trobava el directori /root.

A continuació descomprimirem primer la copia completa, utilitzarem la comanda tar:

tar -xvf /backup/[nom\_del\_backup\_complet]



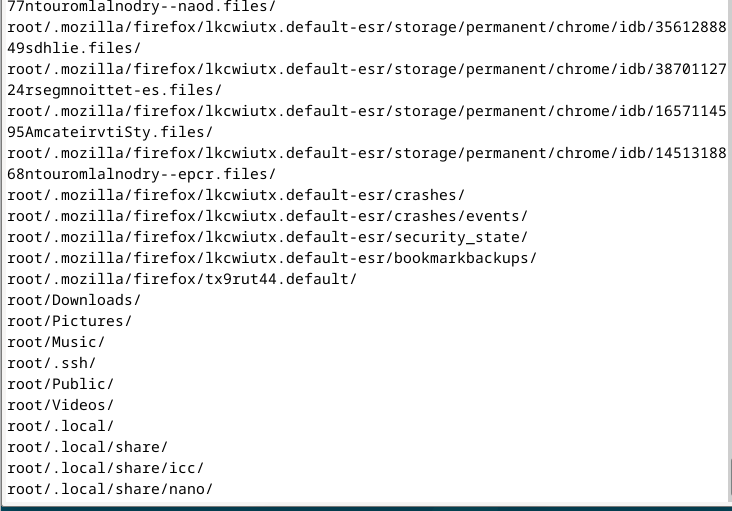


Una vegada descomprimit farem el tar de les altres dues incrementals en ordre de més vell a més nou:

tar -xvf /backup/[nom\_del\_backup\_incremental1]

tar -xvf /backup/[nom\_del\_backup\_incremental2]





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?

Els fitxers que s’han borrat en la segona copia incremental també s’han borrat una vegada hem descomprimit l’última copia de seguretat. Podem detectar que s’han borrat fent ús de la comanda [find] o comparant les dues còpies de seguretat amb un [diff]. Si la política de retenció no conserva les versions antigues, els fitxers eliminats es perdran en la propera rotació o neteja de les còpies de seguretat, ja que aquestes versions antigues s'eliminaran per alliberar espai per a les noves còpies.

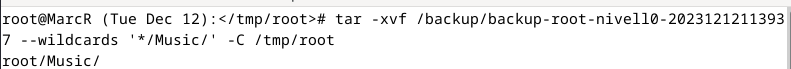
## Restauració d'un fragment

Reanomeneu 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.

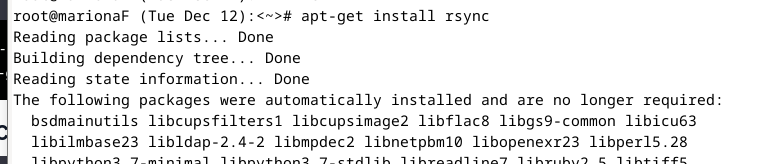
Per a obtenir només la carpeta que hem modificat, en el meu cas la de Music, utilitzarem la seguent comanda:



Una vegada haguem fet això amb la completa, si en les incrementals hem modificat la carpeta en algun moment tindrem que fer el mateix però amb els dos altres tars.

# Realització de còpies usant RSYNC

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**.

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. Recordeu instal.lar també el paquet ssh si us cal.

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

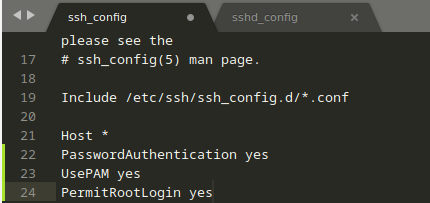
-a (o --archive): Aquesta opció és una combinació de diverses opcions. Activa el mode arxiu, el qual assegura que la transferència mantingui les propietats dels fitxers (com permisos, propietats, enllaços simbòlics, etc.) intactes. A més, també manté la informació de les carpetes (com per exemple, l'estructura de directoris).

-v (o --verbose): Aquesta opció fa que rsync sigui més xerraire. Proporciona informació detallada sobre la transferència de fitxers, mostrant quins fitxers estan sent transferits i un resum de les estadístiques de la transferència.

-z (o --compress): Aquesta opció habilita la compressió de dades durant la transferència. És útil quan s'estan transferint fitxers a través d'una xarxa amb una banda ampla limitada, ja que pot reduir la quantitat de dades que necessiten ser enviades a través de la xarxa.

Primer el que farem es configurar el ssh a la màquina a la que ens volguem conectar per poder conectar-nos com a root, de normal esta desactivat i només amb una comanda del rsync no podem copiar arxius de superusuaris.

Hem d’executar la comanda **nano /etc/ssh/ssh\_config** i afegim les següents instruccions (com a la foto).

**PasswordAuthentification yes, UsePAM yes, PermitRootLogin yes**

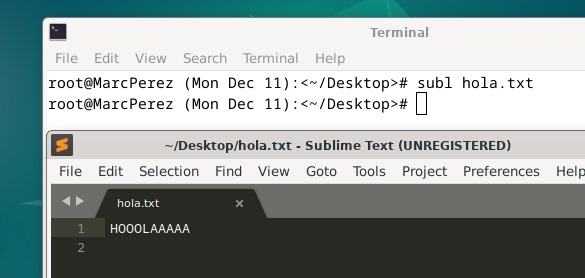
Farem el mateix amb l’altre arxiu anomenat sshd\_config. Mirem si estan les instruccions anteriors, i ens asegurem que tinguin el mateix valor que he descrit.

Per acabar executem **service ssh restart** i **service sshd restart**

Creeu un arxiu en el directori root amb:

### $ echo “nou arxiu” > /root/arxiu\_nou.txt

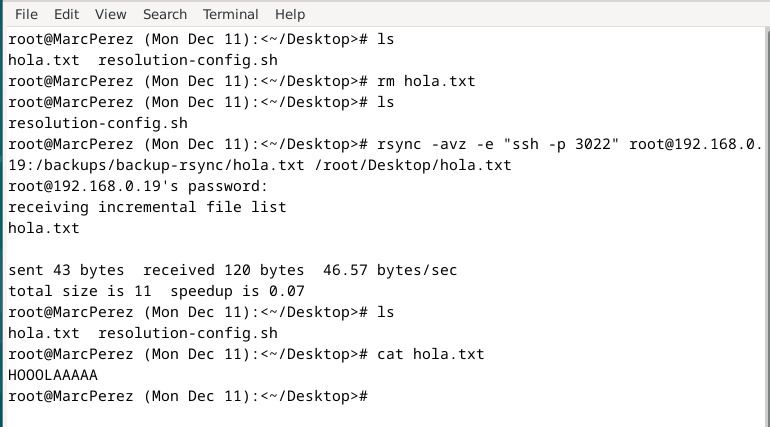
Crearem l’arxiu que anomenarem hola.txt



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



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



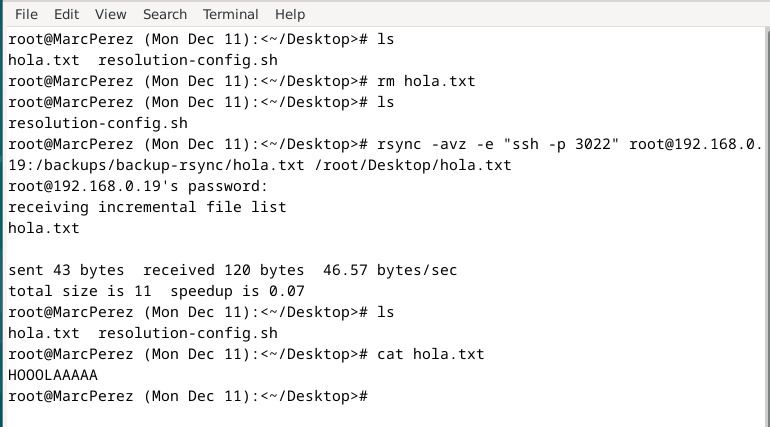
Què ha passat amb el fitxer esborrat?

S’ha eliminat de la nostre maquina, però seguim mantenint la copia a la maquina on hem fet el backup amb la comanda anterior ( en el nostre cas esta dins un directori anomenat backup-rsync dins la partició ext3, em montat el directori principal backups).



Amb quin paràmetre podríeu sincronitzar exactament els dos directoris?

Si utilitzem la mateixa comanda que em fet servir abans pero invertida. D’aquesta manera també recuperariem el arxiu.



En el cas que vulguem copiar tot el directori, i no només un arxiu, hauriem de fer servir la mateixa comanda però sense indicar el arxiu, deixant el directori nomes.

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

Utilitzariem la mateixa comanda del rsync, però afegint aquesta opció.

--exclude='\*.txt'

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

L'ús del caràcter / al final dels directoris en rsync té un significat especial que diu quins arxius són copiats i on són posats en el directori de destinació:

* **rsync /source /destí**:
  + Sense la barra inclinada al final de /source, rsync copiarà el directori source sencer (inclòs) dins de /destí.
  + Això significa que acabaràs amb /destí/source i tots els arxius i subdirectoris dins de source estaran dins d'aquest nou directori source.
* **rsync /source/ /destí/**:
  + Amb la barra inclinada al final de /source/, rsync copiarà tots els arxius i subdirectoris *dins* de source al directori destí, però no el directori source en si.
  + Això significa que el contingut de source serà copiat directament dins de destí, sense crear un subdirectori addicional. Així, si source conté file1 i file2, aquests arxius acabaran com /destí/file1 i /destí/file2.

## 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=  DEST\_DIR=  # excludes file: list of files to exclude  EXCLUDES=  # the name of the backup machine  BSERVER=  # the name of the incremental backups directory  # put a date command for: year month day hour minute second  BACKUP\_DATE=  # 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 |
| --- |

**script complet:**  
#!/bin/bash

# Directori d'origen per fer el backup

SOURCE\_DIR="/home/username/documents"

# Directori de destinació per guardar el backup

DEST\_DIR="/backup/documents"

# Fitxer que conté la llista d'exclusions

EXCLUDES="/home/username/backup\_excludes.txt"

# Nom o adreça IP del servidor de backup

BSERVER="192.168.1.15"

# Directori on es guardaran els backups incrementals

INCREMENTAL\_BACKUP\_DIR="incrementals"

# Comanda per generar la data actual

BACKUP\_DATE=$(date +"%Y-%m-%d\_%H-%M-%S")

# Opcions per a rsync

OPTS="--ignore-errors --delete-excluded --exclude-from=$EXCLUDES \

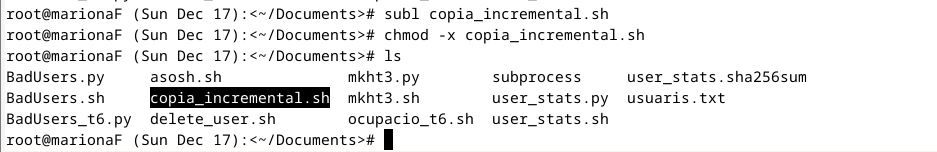
--delete --backup --backup-dir=$DEST\_DIR/$INCREMENTAL\_BACKUP\_DIR/$BACKUP\_DATE -av"

# Transferència actual

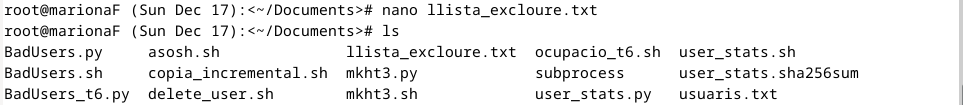
rsync $OPTS $SOURCE\_DIR root@$BSERVER:$DEST\_DIR/complet

Crear el fitxer per fer el script, en aquest cas es dirà: copia\_incremental.sh

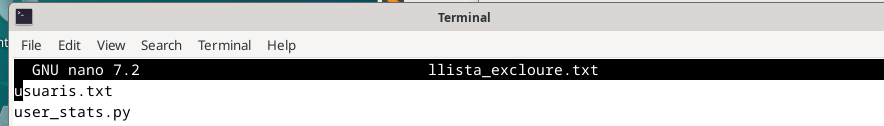
Donar-li les propetats perque pugui ser executat: chmod -x copia\_incremental.sh



Fer un fitxer amb la llista de fitxers que es volen excloure: llista\_exloure.txt



En aquest fitxer posar el nom dels fitxers que no es volen en la còpia incremental, per exemple no guardarem user\_stats.py i usuaris.txt

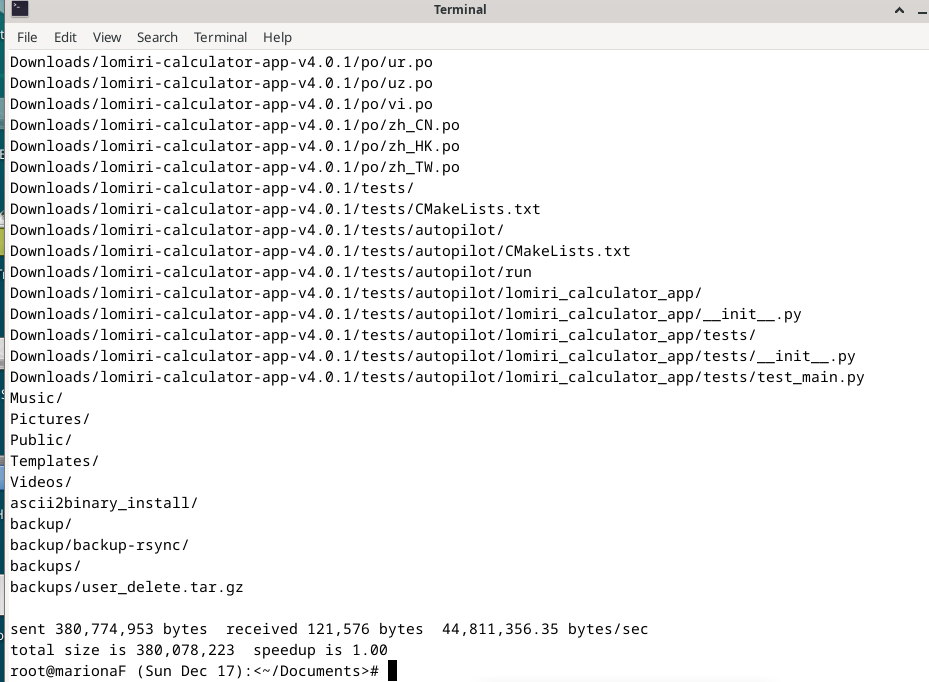


Dins del fitxer posar el codi, ara complet:

| #!/bin/bash  # Directori d'origen per fer el backup  SOURCE\_DIR="/root/"  # Directori de destinació per guardar el backup  DEST\_DIR="/backup/backup-rsync"  # Fitxer que conté la llista d'exclusions  EXCLUDES="/root/Documents/llista\_excloure.txt"  # Nom o adreça IP del servidor de backup  BSERVER="127.0.0.1"  # Nom del directori on es guardaran els backups incrementals  INCREMENTAL\_BACKUP\_DIR="incrementals"  # Comanda per generar la data actual amb format d'any, mes, dia, hora, minut i segon  BACKUP\_DATE=$(date +"%Y-%m-%d\_%H-%M-%S")  # Opcions per a rsync  OPTS="--ignore-errors --delete-excluded --exclude-from=$EXCLUDES \  --delete --backup --backup-dir=$DEST\_DIR/$INCREMENTAL\_BACKUP\_DIR/$BACKUP\_DATE -av"  # Assegura que el directori de backup incremental existeix  mkdir -p $DEST\_DIR/$INCREMENTAL\_BACKUP\_DIR/$BACKUP\_DATE  # Transferència actual amb rsync  rsync $OPTS $SOURCE\_DIR root@$BSERVER:$DEST\_DIR/complet |
| --- |

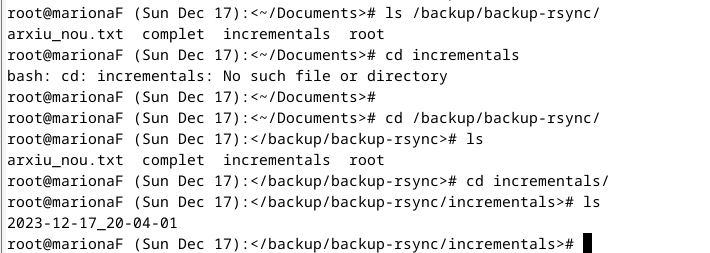
Executar el fitxer amb ./copia\_incremental.sh

Preguntarà per la contrasenya del root del localhost, serà la mateixa que el root

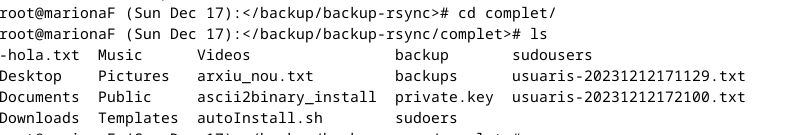


Si entrem a la carpeta de backups ens sortira la carpeta de les copies incremental i la completa amb tots els documents:

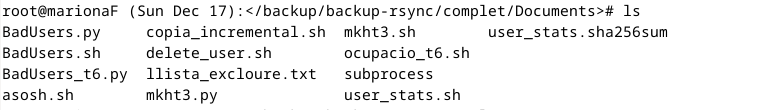
Incremental guarda la data quan s’ha fet la còpia:



Complet és on es guarda tots els documents :

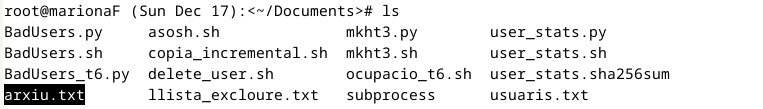


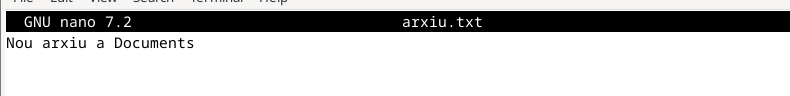
Podem observar que les fitxers que hem posat a la llista d’exclusió no sha fet la còpia:



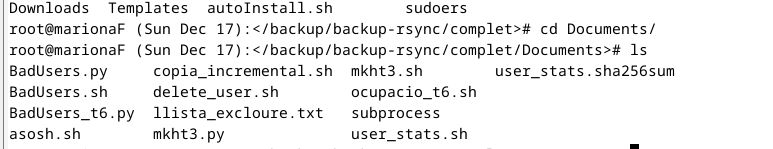
(No hi ha ni user\_stats.py ni usuaris.txt)

Ara creeu un fitxer ***arxiu.txt*** i feu-lo servir per comprovar el funcionament de l'script anterior

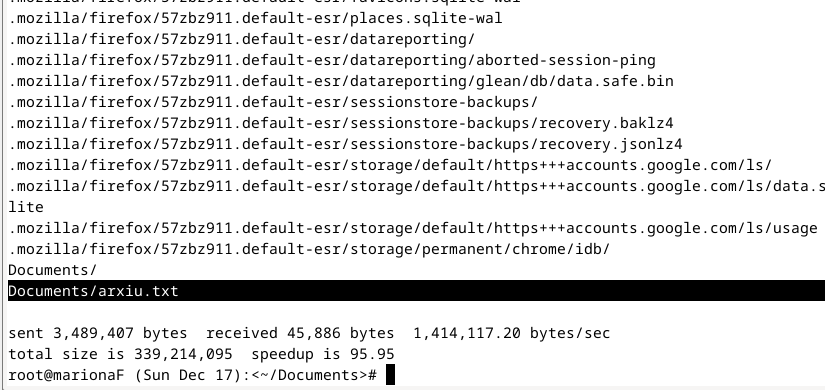




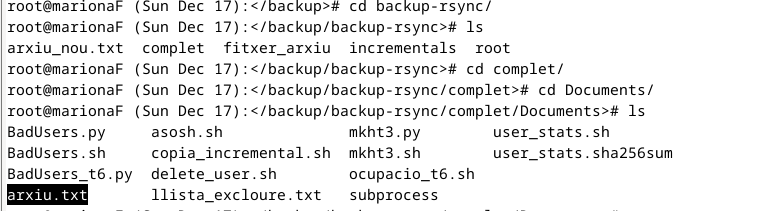
Abans de fer una altre copia la carpeta de Documents no hi ha el fitxer arxiu.txt:



Si es fa una còpia després de la seva creació, es pot veure que es un dels documents que es copia en el backup:



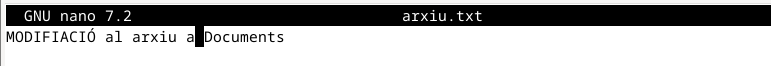
El arxiu s’ha pogut copiar correctament:



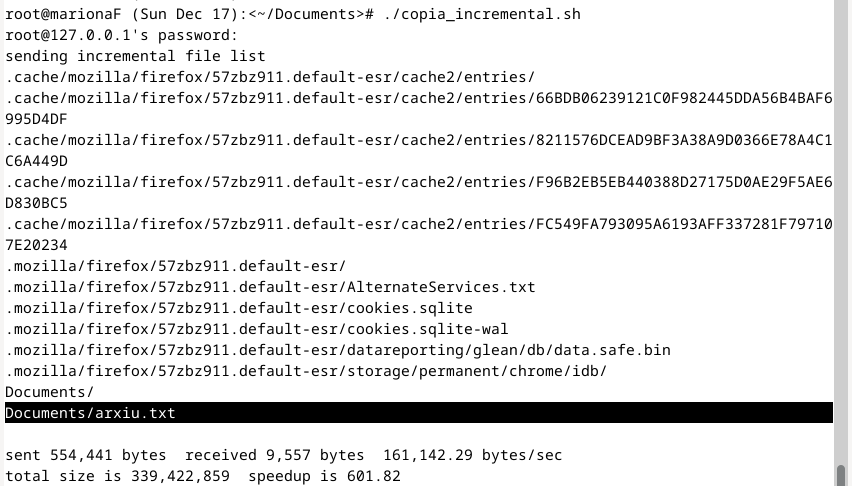


Després modifiqueu aquest **arxiu.txt** i torneu a sincronitzar.

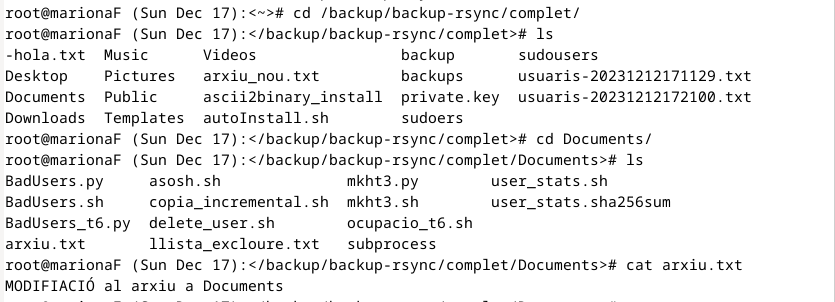
Modifiar el arxiu.txt des de la carpeta de Documents:



Fer una còpia:

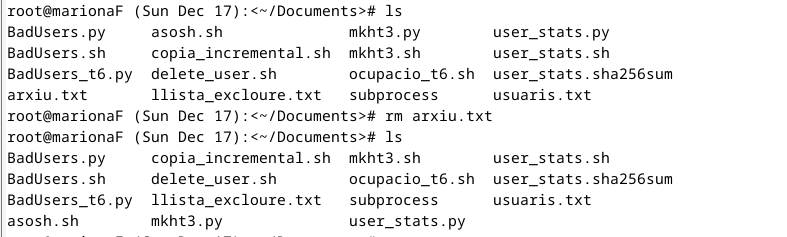


L’arxiu dins del backup sha modificat correctament:

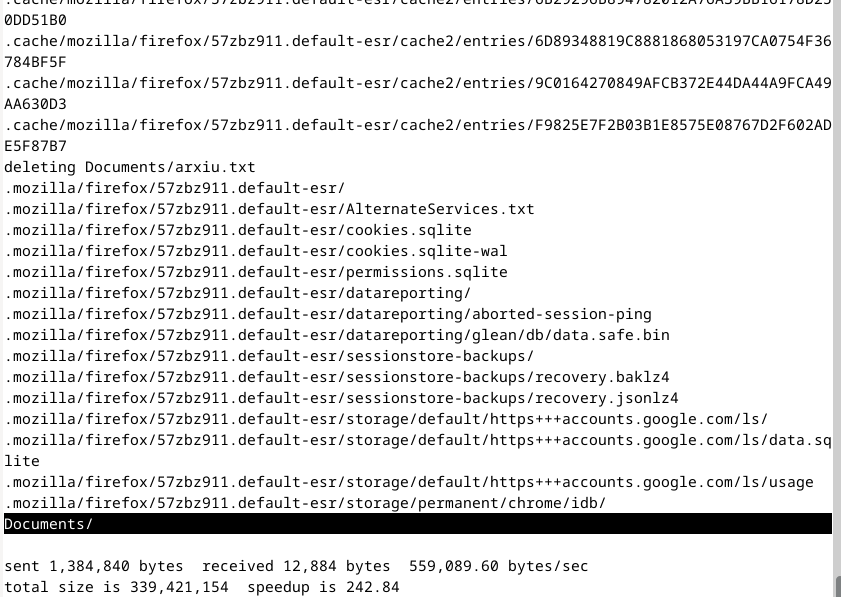


Finalment esborreu el fitxer **arxiu.txt** i feu una sincronització més.

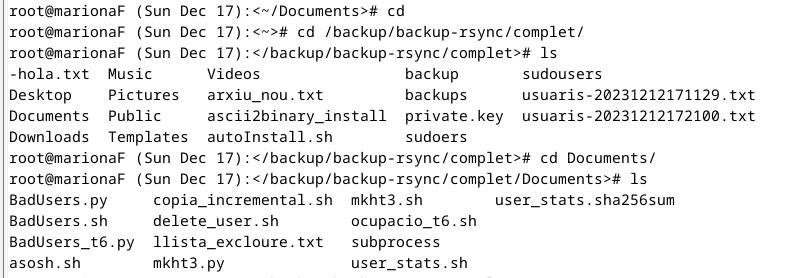
Borrar el fitxer:



Executar la copia incremental:



Com a la sortida de l’execució surt, s’haura borrat el fitxer en el backup complet:



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

## Quan es modifica un fitxer, ja sigui el nom o el contingut, quan es faci la còpia completa es guardarà la modificació feta, però si no es fa una copia, el fitxer del backup continuarà tenint la versió antiga, la ultima feta en la còpia.

## Si s’esborra algun fitxer, aquest també s’esborrarà del backup quan es faci una còpia de seguretat, mentres que no es faci aquell fitxer continuarà existint.

## Realització de còpies incrementals inverses tipus snapshot

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

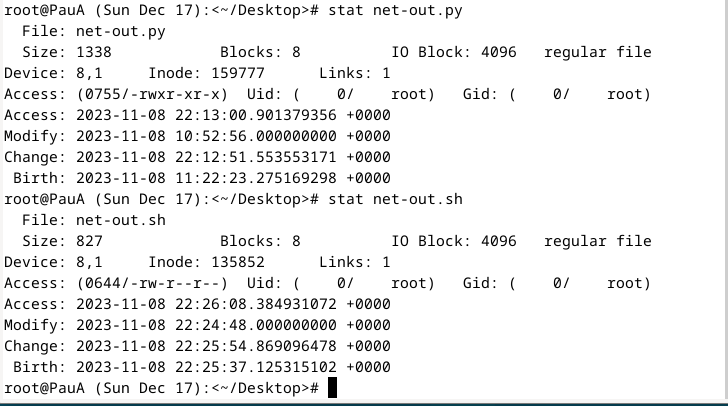
El nom d'un fitxer no representa el fitxer mateix, per al sistema és només un enllaç dur al *inode*. Això permet que un fitxer (inode) pugui tenir més d'un enllaç dur. Per exemple si teniu un fitxer file\_a es pot crear un enllaç cridat file\_b:

$ ln 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*?



Executar stat file\_a i anotar el número d'inode que apareix.

Executar stat file\_b i comparar el seu número d'inode amb el de file\_a.

Si els números d'inode són idèntics, llavors file\_a i file\_b són enllaços durs del mateix fitxer.

Què passa amb el fitxer file\_b si es fa un canvi a file\_a?

Quan es fa un canvi en file\_a, que és un dels noms associats a un inode a través d'un enllaç dur (file\_b és l'altre), el canvi s'aplicarà al contingut de l'arxiu que es troba emmagatzemat en l'espai de disc que és referenciat per l'inode. Ja que els enllaços durs file\_a i file\_b apunten al mateix inode, qualsevol canvi en el contingut del fitxer a través d'un d'aquests noms serà visible quan s'accedeix al fitxer a través de l'altre nom.

I si es fa un canvi de permisos a file\_a?

Si es canvia els permisos de file\_a utilitzant una comanda com chmod, aquests canvis afectaran també a file\_b, ja que els permisos del fitxer es guarden a l'inode. Com que tant file\_a com file\_b són enllaços durs al mateix inode, qualsevol modificació dels permisos s'aplicarà a l'inode i, per tant, serà reflectida a tots els noms que hi apunten.

I si copiem un altre fitxer sobreescrivint el fitxer file\_a (**cp file\_c file\_a**)? I si es sobreescriu amb l'opció –**remove-destination**?

Quan copies un fitxer sobre un altre utilitzant la comanda cp, el comportament dependrà de la manera específica en què s'executa la comanda:

Si executas cp file\_c file\_a sense cap opció adicional:

* El contingut de file\_c substituirà el contingut de file\_a.
* L'inode al qual apuntava file\_a mantindrà els mateixos enllaços durs, incloent file\_b, però ara amb el nou contingut copiat de file\_c.
* Els permisos i propietat del fitxer original file\_a es mantindran, ja que l'operació de copia no modifica l'inode; només es canvia el contingut del fitxer.

Si utilitzas cp --remove-destination file\_c file\_a:

* L'opció --remove-destination esborra el fitxer destí (file\_a) abans de realitzar la copia.
* Això significa que file\_a serà eliminat (juntament amb la seva entrada a l'inode), i després file\_c serà copiat en un nou fitxer anomenat file\_a.
* En aquest cas, file\_a tindrà un nou inode diferent de file\_b, ja que has eliminat l'enllaç original i creat un de nou amb el contingut de file\_c.
* El fitxer file\_b continuarà apuntant a l'antic inode (i per tant al contingut original) fins que aquest inode no tingui més referències (és a dir, quan tots els enllaços durs i referències de processos s'eliminin).

Per tant, en el primer cas, file\_a i file\_b seguiran sent enllaços durs al mateix inode i, per tant, continuaran tenint el mateix contingut després de la comanda cp. En el segon cas, file\_a serà ara un enllaç a un nou inode amb un contingut diferent, i file\_b no es veurà afectat i seguirà apuntant al contingut original.

I què passa amb file\_b si file\_a és esborrat?

Si file\_a és esborrat, això no afecta l'existència de file\_b ni el contingut del fitxer a què tots dos noms apuntaven originalment.

La comanda **cp** té una opció per fer còpies en què realment no es fa una còpia sinó un enllaç dur (**cp -l**). Una altre 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:

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:

## Script per fer backups tipus snapshot

Per fer *backups* del directori /root utilitzarem l'script **backup-rsync-snapshot.sh**, que està disponible a l'apèndix.

En primer lloc modifiqueu les variables de la secció “file locations” per posar els valors adequats del vostre sistema.

En la secció de “file locations” ficarem els següents valors:

MOUNT\_DEVICE = “/dev/sda4”

SNAPSHOT\_MOUNTPOINT = “/backup”

SNAPSHOT\_DIR = “root-backup”

EXCLUDES = “/root/exclude\_list.txt”

SOURCE\_DIR = “/root”

Després completeu la secció amb la comanda **rsync** amb els valors apropiats per fer la còpia tipus *snapshot*

$RSYNC -a --delete --exclude-from$EXCLUDES --link-dest=$SNAPSHOT\_MOUNTPOINT/$SNAPSHOT\_DIR/daily.1/$SOURCE\_DIR/ $SNAPSHOT\_MOUNTPOINT/$SNAPSHOT\_DIR/daily.0/

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?

**Creació de nous fitxers:** Quan afegeixes nous fitxers al directori origen i executes l'script, aquests nous fitxers es veuran reflectits en la còpia de seguretat més recent (daily.0). Cada vegada que fas una nova còpia de seguretat, es crea una nova instantània del sistema en aquest directori.

**Modificació de fitxers existents:** Quan modifiques el contingut d'un fitxer existent i executes l'script, aquest canvi es reflectirà només en la còpia de seguretat més recent (daily.0). Les còpies anteriors (daily.1, daily.2, etc.) continuen mantenint les versions anteriors d'aquests fitxers, ja que es fan servir enllaços durs per compartir els fitxers inalterats entre els snapshots. El mateix passa quan eliminem un fitxer al directori origen del snapshot, que només es reflectirà en la copia de seguretat més recent.

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

Ho podem saber executant la comanda:

du -h /backup\* //du=disk usage, -h= human-readable, /backup= directori pare.

, la qual ens mostra la grandària de cada directori backup. Podem esperar que la grandària del directori backup.0 serà la major, ja que les còpies més antigues probablement compartiran molts fitxers amb les còpies més recents a través d’enllaços durs, de manera que ocuparan menys espai de disc.

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.

En primer lloc, canviarem el nom del nostre directori root amb la comanda:  
 mv /root /root\_old

A continuació, restaurem la còpia de seguretat més recent amb la comanda:

cp -al /backup.0/daily.0 /root

Aquesta comanda farà una còpia enllaçada al directori /root. L'ús de -al assegura que es facin enllaços durs als fitxers per optimitzar l’ús de l’espai de disc.

# 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. AEleen 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/>
4. **Rsnapshot**. *a remote filesystem snapshot utility based on rsync for making backups of local and remote systems.* [*http://www.rsnapshot.org/*](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;