COPIA DE SEGURIDAD

versión 1.4
15-1-2023

Índice

1 Hacia atrás en el tiempo con el programa `tar`

1.1 Copias completas, incrementales y diferenciales

En un medio de almacenamiento tenemos un directorio llamado `origen` con los datos a guardar. Generamos un archivo comprimido con todos los datos y lo guardamos en otro medio de almacenamiento llamándolo `año-mes-dia-hora-minutos-segundos-nivel-0.tar.gz`. Esta es una copia completa o copia de nivel 0. En la siguiente copia de seguridad podemos hacer dos cosas:
a) Volver a realizar una copia de nivel 0, con el inconveniente del tiempo que lleva y del espacio que usa además de que muchos archivos no hayan cambiado y los tengamos en las dos copias innecesariamente.
b) Realizar una copia solo con los ficheros que han cambiado desde la última copia de seguridad llamándolo `año-mes-dia-hora-minutos-segundos-nivel-1.tar.gz`. Esta sería una copia incremental o de nivel 1. Requiere poco tiempo y poco espacio. Si tenemos que recuperar datos tenemos que descomprimir la copia 0 y la copia 1.

En la siguiente copia de seguridad tenemos tres opciones:
a) Realizar una copia de nivel 0.
b) Realizar otra copia de nivel 1. Si tenemos que recuperar datos hay que descomprimir la copia de nivel 0 y las dos copias de nivel 1.
c) Realizar una copia de los ficheros que han cambiado desde la última copia de nivel 0 llamándola `año-mes-dia-hora-minutos-segundos-nivel-2.tar.gz`. Esta sería una copia diferencial o de nivel 2. Si queremos recuperar datos tenemos que descomprimir la copia de nivel 0 y la de nivel 2.

Las copias incrementales requieren poco tiempo para crearlas y, si las tenemos todas juntas, se descomprimen rápido con el comando `find`. Antiguamente se tenían que guardar en varias cintas magnéticas y por eso requería mucho tiempo restaurarlas. Las copias diferenciales requieren más tiempo para crearlas y antiguamente se restauraban más fácilmente que las incrementales. Si modificamos un fichero varias veces entre copias, la copia incremental guardará varias versiones del fichero pero la copia diferencial solo guardará la última versión. Por eso actualmente parece más práctico hacer copias incrementales. Los archivos que hayamos borrado entre copias de nivel 0 permanecen en las copias de seguridad, así que si restauramos las copias volveremos a tener dichos ficheros. Eso puede ser bueno si hemos borrado algo que luego queremos tener o malo si por confidencialidad no queremos volver a tenerlo o simplemente para no volver a saber qué tenemos que borrar. Si los datos a guardar ya están comprimidos, como las fotos, vídeos, documentos odt, docx, etc., generar un fichero comprimido requiere mucho tiempo y no se gana espacio. En ese caso podemos generar un archivo `tar` sin comprimir.

1.2 Ejemplos de esquemas de copia de seguridad

1. Incremental. Hacemos una copia de nivel 0 el domingo y el resto de días copias incrementales o de nivel 1. El siguiente domingo volvemos a comenzar realizando una nueva copia de nivel 0. Para restaurar los datos por ejemplo al jueves tenemos que restaurar la copia de nivel 0 y las copias incrementales del lunes, martes, miércoles y jueves.
2. Diferencial. Hacemos una copia de nivel 0 el domingo y el resto de días copias diferenciales o de nivel 2. El siguiente domingo volvemos a comenzar realizando una nueva copia de nivel 0. Para restaurar los datos, por ejemplo al jueves, tenemos que restaurar la copia de nivel 0 y la copia diferencial del jueves.
3. Incremental y diferencial. Hacemos una copia de nivel 0 el domingo y el resto de días de la semana copias incrementales. El siguiente domingo creamos una diferencial y volvemos a crear incrementales el resto de la semana. Continuamos con este esquema hasta el mes siguiente en el que volvemos a hacer una copia completa o de nivel 0.
Esta estrategia nos permite restaurar el sistema a un determinado día desempaquetando la completa del primer domingo del mes y luego dependiendo de la semana, desempaquetaremos la diferencial del domingo de esa semana y posteriormente las incrementales.
Con estos esquemas disponemos de copias de lo que hacemos a diario. Pero hemos de tener en cuenta que tan importante es realizar las copias como comprobar periódicamente que funcionan correctamente.

1.3 Copias incrementales con tar

Usaremos la opción `-g` para que `tar` guarde toda la información sobre la copia de seguridad en un fichero. En la siguiente copia `tar` lee dicho fichero para saber qué ha cambiado y copiarlo.
Usamos varios comandos concatenados con `&&` (si un comando se ejecutó con éxito entonces ejecuta el siguiente) para crear la copia completa del directorio que queremos respaldar. Como el comando es muy largo usamos la barra `\` para separarlo en varias líneas y que se lea mejor (la barra le dice a `bash` que no interprete el retorno de carro). Los ejecutamos en el directorio `tar`. El comando crea una carpeta para cada copia para

que al restaurar las copias no se sobrescriban y nos quedemos solo con la última versión de cada fichero. Para ello creamos una variable de `bash` para almacenar la fecha y la llamamos `fecha`. Obtenemos su valor escribiendo `${fecha}`. Mediante `$(comando)` sustituimos `$(comando)` por la salida de `comando`, en este caso la salida del comando `date +%Y-%m-%d-%H-%M-%S`:

```
fecha="$(date +%Y-%m-%d-%H-%M-%S)-nivel-0" && \
mkdir "incr/${fecha}" && \
tar -cpvzf "incr/${fecha}/${fecha}.tar.gz" -g incr/snapshot origen
```

Es más seguro usar una ruta relativa para el directorio `origen`. Si usamos una ruta absoluta `tar` le quita la `/` inicial para evitar sobreescribir la carpeta original al descomprimir (así que guarda `home/usuario/origen` y no `/home/usuario/origen`), no obstante hay que tener cuidado porque puede que usemos una versión de `tar` que no tenga esa precaución. Incluso usando rutas relativas o empaquetando sin la `/` inicial podemos sobrescribir el contenido original, en el ejemplo anterior sería descomprimiendo desde el directorio en el que está `origen`. Por eso es mejor descomprimir en un directorio diferente para no equivocarse.
La copia incremental la creamos con:

```
fecha="$(date +%Y-%m-%d-%H-%M-%S)-nivel-1" && \
mkdir "incr/${fecha}" && \
tar -cpvzf "incr/${fecha}/${fecha}.tar.gz" -g incr/snapshot origen
```

Cuando tengamos que restaurar ejecutamos el comando:

```
find incr -iname "*.tar.gz" -execdir tar -zxf "{}" \;
```

Y si queremos obtener el directorio `origen-copia` y que sea una réplica del directorio `origen` (pero incluyendo los archivos borrados, como ya hemos dicho):

```
find incr -iname origen | sort | xargs -I "{}" \
cp -av "{}" origen-copia
```

Si en las copias de seguridad tenemos varias versiones de ficheros el comando anterior copia primero los ficheros más antiguos y luego los más nuevos (sobreescribiendo) garantizando por tanto que al final tendremos la última versión.
Lo más cómodo es tener estos comandos en un guión de `bash` y automatizar la copia periódica con el programa `cron`. En dicho guión iríamos borrando las copias antiguas para controlar el espacio de disco.

Ejercicio: haz una copia de nivel 0 de `origen`. Borra el fichero `1.sh`, modifica el fichero `2.sh` y añádele los ficheros `5.sh` y `6.sh` del directorio `mas_ficheros`. Haz una copia incremental. Añádele los ficheros `7.sh` y `8.sh` y haz otra copia incremental. Restaura la copia con `find` y comprueba que todo funciona correctamente.

1.4 Copias diferenciales con `tar`

Usaremos su opción `-N`. Lo que nos permite esta opción es ordenar a `tar` que solo copie aquellos datos que han sido cambiados o agregados desde una determinada fecha hasta la fecha de ejecución del comando. Por ejemplo hacemos la copia de nivel 0 el 2015-5-1 ejecutando el comando desde el directorio `tar`:

```
fecha="$(date +%Y-%m-%d-%H-%M-%S)-nivel-0" && \
mkdir "dif/${fecha}" && \
tar -cpvzf "dif/${fecha}/${fecha}.tar.gz" origen
```

y creamos la siguiente copia (que es incremental) el 2015-5-2 con los cambios ocurridos desde 2015-5-1:

```
fecha="$(date +%Y-%m-%d-%H-%M-%S)-nivel-2" && \
mkdir "dif/${fecha}" && \
```

```
tar -cpvzf "dif/${fecha}/${fecha}.tar.gz" origen \
-N "2015-5-1 21:24:00"
```

Para la siguiente copia (que ahora sí es diferencial) antes debemos borrar la anterior:
`rm -rv dif/*-nivel-2`
y ejecutamos la copia:

```
fecha="$(date +%Y-%m-%d-%H-%M-%S)-nivel-2" && \
mkdir "dif/${fecha}" && \
tar -cpvzf "dif/${fecha}/${fecha}.tar.gz" origen \
-N "2015-5-1 21:24:00"
```

Cuando tengamos que restaurar ejecutamos el comando:
`find dif -name "*.tar.gz" -execdir tar -zxf "{}" \;`
Y si queremos obtener el directorio `origen-copia` y que sea una réplica del directorio `origen` (pero incluyendo los archivos borrados, como ya hemos dicho):
`find dif -iname origen | sort | xargs -I "{}" cp -av "{}" origen-copia`

Si en las copias de seguridad tenemos varias versiones de ficheros el comando anterior copia primero los ficheros más antiguos y luego los más nuevos (sobreescribiendo) garantizando por tanto que al final tendremos la última versión.
Lo más cómodo es tener estos comandos en un guión de `bash` y automatizar la copia periódica con el programa `cron`.

Ejercicio: haz una copia de nivel 0 de `origen`. Borra el fichero `1.sh`, modifica el fichero `2.sh` y añádele los ficheros `5.sh` y `6.sh` del directorio `mas_ficheros`. Haz una copia incremental. Añádele los ficheros `7.sh` y `8.sh` y haz una copia diferencial. Para hacerlo todo seguido tendrás que ajustar bien la fecha de `-N`, minutos y segundos incluidos. Restaura la copia con `find` y comprueba que todo funciona correctamente.

1.5 Otras opciones del comando `tar`

`--same-owner`
permite mantener los propietarios de los ficheros.

`--exclude`
permite excluir ficheros y directorios, por ejemplo:

```
tar -cpvzf "prueba.tar.gz" origen --exclude="papelera"
tar -cpvzf "prueba.tar.gz" origen --exclude="local/papelera"
```

2 Hacia delante en el tiempo con el programa `rsync`

2.1 Copias completas, incrementales y diferenciales

En un medio de almacenamiento tenemos un directorio llamado `origen` con los datos a guardar. Generamos una copia de dicho directorio en otro medio de almacenamiento llamándolo `espejo`. Para ello usamos el programa `rsync`. Los datos los copiamos sin comprimir, así que podemos buscar directamente en `espejo`. Esta es una copia completa o copia de nivel 0. Esta en concreto comienza un ciclo de copias. En la siguiente copia de seguridad podemos hacer tres cosas:
a) Volver a realizar una copia de nivel 0 con `rsync`. Si en `origen` hay datos nuevos, modificados o borrados `rsync` los copiará, modificará o borrará también en `espejo`. Por tanto el esfuerzo en tiempo es el de una copia incremental con la ventaja de tener una copia completa. Con este esquema cada vez que hacemos una copia comenzamos un ciclo nuevo. Lo más práctico es tener un guión de `bash` y automatizar la copia periódica

con `cron`. El inconveniente de este método es que no guardamos los ficheros modificados o borrados de la primera copia. Si borramos algo, ¡o todo!, al hacer la copia lo borramos también en el espejo. Puede que borremos algo a conciencia pero que luego nos arrepintamos, que sea un borrado accidental, un virus que borre nuestros datos o incluso otra persona que lo borre adrede o accidentalmente.
b) Volver a realizar una copia de nivel 0 con `rsync`. Pero esta vez los datos modificados o borrados de `origen`, antes de copiarlos a `espejo`, los copiamos de `espejo` a otra carpeta con la fecha. Así tenemos una copia de nivel 0 y una incremental. En la siguiente copia volvemos a crear una carpeta con la fecha y copiamos a ella los datos modificados desde la última copia. Con este esquema podemos tener siempre una réplica exacta de `origen` y cuantas copias incrementales, respecto de la copia anterior, deseemos y solo con el esfuerzo de una copia incremental. Lo más práctico es tener un guión de `bash` y automatizar la copia periódica con `cron`. El guión también se encargará de borrar todas las copias antiguas después de un número determinado de copias para controlar el espacio de disco y así comenzar un ciclo nuevo, aunque lo más eficiente es el que el guión calcule el espacio necesario y el disponible y que borre solo el mínimo de copias antiguas.
c) Volver a realizar una copia de nivel 0 con `rsync`. Pero esta vez los datos modificados o borrados de `origen`, antes de copiarlos a `espejo`, los copiamos de `espejo` a otra carpeta llamada `dif`. Así tenemos una copia de nivel 0 y una incremental. En la siguiente copia añadimos los ficheros modificados o borrados otra vez a `dif`. Con este esquema tenemos siempre una réplica exacta de `origen` y una copia diferencial respecto del comienzo de ciclo. Lo más práctico es tener un guión de `bash` y automatizar la copia periódica con `cron`. El guión también se encargará de borrar el directorio `dif` después de un número determinado de copias para controlar el espacio de disco y así comenzar un ciclo nuevo.

Como vemos, la copia incremental y la diferencial requieren el mismo esfuerzo pero si modificamos un fichero varias veces entre copias, la copia incremental guardará varias versiones del fichero mientras que la diferencial solo guardará la última versión (ocupando menos espacio que en la copia incremental). Por eso la elección depende del espacio disponible.

Las copias incrementales y diferenciales hechas con `tar` y con `rsync` guardan los mismo ficheros pero los distribuyen en diferentes directorios. Con `tar` vamos hacia delante en el tiempo partiendo de la réplica congelada de `origen` y con `rsync` vamos hacia atrás en el tiempo a partir de la réplica actualizada de `origen`. Con `tar` no podemos obtener una réplica exacta de origen por culpa de los ficheros borrados, que no sabemos cuáles son. Además `tar` copia el fichero entero aunque solo haya cambiado muy poco. Sin embargo a `rsync` podemos pasarle la opción `--no-whole-file` para que solo copie la parte del fichero que haya cambiado, aunque comparar ficheros para ver qué ha cambiado también requiere su tiempo, por lo que solo suele ser útil si hacemos la copia a través de redes lentas. `tar` y `rsync` permiten comprimir y ocupar menos espacio pero a costa de tardar más tiempo, así que si no hay problemas de espacio es mejor no comprimir.

2.2 Espejo simple con `rsync`

Ejecutamos el comando en el directorio `rsync`. La opción `-a` hace que se guarden los

permisos, el propietario, la fecha de modificación, etc. y la opción `-v` hace que muestre información y la segunda `-v` hace que muestre más información todavía:

```
rsync -avv origen/ espejo
```

la barra final en `origen` significa copia todo el contenido (incluidos ocultos) de `origen`. Si no la ponemos nos crea el directorio `origen` en `espejo`. Para aclararlo del todo, estos dos comandos hacen lo mismo:

```
rsync -avv origen/  espejo/origen
rsync -avv origen   espejo
```

Para las siguientes copias usamos la opción `--delete` para borrar en `espejo` lo borrado en `origen`:

```
rsync -avv --delete origen/ espejo
```

Para saber cuándo realizamos las copias podemos ejecutar estos comandos en vez de los anteriores (creamos un archivo en `origen` con `touch` con la fecha de la copia):

```
fecha="rsync-nivel-0-$(date +%Y-%m-%d-%H-%M-%S)" && \
touch "origen/${fecha}" && \
rsync -avv --delete origen/ espejo
```

También está disponible una interfaz gráfica muy completa de `rsync` llamada `grsync`. La interfaz gráfica es útil para un uso interactivo y nos permite guardar las opciones para la siguiente copia, pero para usar en guiones `bash` tendremos que usar `rsync`.

Ejercicio: haz una copia de nivel 0 de `origen`. Borra el fichero `1.sh`, modifica el fichero `2.sh` y añádele los ficheros `5.sh` y `6.sh` del directorio `mas_ficheros`. Actualiza el directorio `espejo`. Añádele los ficheros `7.sh` y `8.sh` y actualiza de nuevo. Comprueba los cambios.

2.3 Espejo más copia incremental con `rsync`

Comenzamos el ciclo:

```
fecha="rsync-nivel-0-$(date +%Y-%m-%d-%H-%M-%S)" && \
touch "origen/${fecha}" && \
rsync -avv origen/ espejo
```

Para las siguientes copias usamos la opción `-b` para que cree copia de seguridad de los archivos modificados o borrados y le decimos dónde guardarlos con `--backup-dir`:

```
fecha="rsync-nivel-1-$(date +%Y-%m-%d-%H-%M-%S)" && \
mkdir "incr/${fecha}" && \
rsync -avvb --delete --backup-dir="../incr/${fecha}" origen/ espejo
```

El directorio que indicamos en `--backup-dir` es relativo al directorio `espejo`. Si no hubiéramos puesto `../` lo habríamos creado en `espejo` y en la siguiente copia se hubiera borrado. Otra solución es usar una ruta absoluta.

Ejercicio: haz una copia de nivel 0 de `origen`. Borra el fichero `1.sh`, modifica el fichero `2.sh` y añádele los ficheros `5.sh` y `6.sh` del directorio `mas_ficheros`. Actualiza el directorio `espejo` guardando los ficheros modificado o borrados en un directorio que tenga por nombre la fecha y hora a la que se hace la copia. Borra los ficheros `2.sh` y `5.sh`, añádele los ficheros `7.sh` y `8.sh` y actualiza de nuevo `espejo` y crea otro directorio para los ficheros borrados y guardados. Comprueba los cambios.

2.4 Espejo más copia diferencial con `rsync`

Comenzamos el ciclo:

```
fecha="rsync-nivel-0-$(date +%Y-%m-%d-%H-%M-%S)" && \
touch "origen/${fecha}" rsync -avv origen/ espejo
```

Para las siguientes copias:

```
fecha="rsync-nivel-2-$(date +%Y-%m-%d-%H-%M-%S)" && \
touch "origen/${fecha}" \
rsync -avvb --delete --backup-dir=../dif origen/ espejo
```

Ejercicio: haz una copia de nivel 0 de origen. Borra el fichero `1.sh`, modifica el fichero `2.sh` y añádele los ficheros `5.sh` y `6.sh` del directorio `mas_ficheros`. Actualiza el directorio `espejo` guardando los ficheros modificado o borrados en el directorio `dif`. Borra los ficheros `2.sh` y `5.sh`, añádele los ficheros `7.sh` y `8.sh` y actualiza `espejo` de nuevo y guarda los ficheros modificado o borrados otra vez en `dif`. Comprueba los cambios.

2.5 Otras opciones

Excluir directorios:

```
rsync -avv  --exclude="papelera" origen/ espejo
rsync -avv  --exclude="/local/papelera" origen/ espejo
```

Mostrar el progreso (tendrás que tener un fichero grande en `origen`):

```
rsync -avvP  origen/ espejo
```

2.6 Copiar datos con `rsync`

Cuando tenemos que copiar muchos datos no es conveniente usar el comando `cp` porque si se interrumpe la copia hay que volver a empezar de nuevo. Tampoco es conveniente con el navegador de archivos porque aunque este te pregunte si no quieres sobrescribir puede que algún fichero se haya quedado a medias al interrumpirse la copia y no lo copiarías correctamente. Por eso es mejor usar `rsync`. Te permite interrumpir la copia y además te copia todos los metadatos de los ficheros (permisos, propietario, fecha de modificación, etc.). Con `tar` y con `cp` con la opción `-a` también puedes copiar los metadatos pero no puedes interrumpir la copia.
Simplemente usaríamos:

```
rsync -av origen/ destino
```