Backups

Eugenia Damonte, Ariel Fideleff y Martín Goñi

Índice

1. Backups **1**

1.1. Que es un backup 1

1.2. Tipos de backups 1

2. rsync **3**

2.1. Que es rsync 3

2.2. Instalación de rsync 3

2.3. Uso básico de rsync 4

3. Comandos usados **16**

1. Backups

1.1. Que es un backup

En el mundo del IT un backup es una copia de parte o toda la información de una computadora, almacenada en una unidad de almacenamiento distinta a la de la computadora. Esta puede luego ser usada para recuperar la informacón original en caso de que ocurra una pérdida de datos. Es importante recalcar que una pérdida de datos puede ocurrir no solo debido a daño al sistema, ya sea de hardware o software, sino que también puede ser provocada por un error humano(por ejemplo borrar un archivo importante). Para cumplir su función un backup debe contener por lo menos una copia de toda la información que se considere vale la pena guardar. Esto nos introduce a un dilema muy importante, ¿que información vale la pena guardar?

En principio uno podría pensar que simplemente deberíamos hacer un backup de todo el sistema, para no tener que tomar esta decisión. Sin embargo a medida que crece el tamaño y complejidad del sistema se vuelve cada vez mas costoso, en todo sentido, realizar backups completos. Algo que también se debe tomar en cuenta al hacer esta decisión es el valor del sistema en sí, es decir cuanto vale el sistema operativo, sus configuraciones y ajustes. Si bien a simple vista esto puede parecer algo no muy importante en sistemas grandes y complejos, que requieren una gran cantidad de conocimiento y experiencia para configurar la configuración puede ser igual de valiosa que la información que almacena el sistema.

1.2. Tipos de backups

Antes de poder elegir que tipo de backup hacer hay que elegir que método utilizar para el mismo, los dos que se usan hoy en día son:

- **Backup por archivos:** El backup por archivos es la forma original en que se hacían los backups. En este todos los archivos y carpetas a los que se les debe realizar un backup son copiados utilizando las utilidades proveídas por el sistema operativo. Este método si bien es

simple también es lento y consume una gran cantidad de recursos.[1]

- **Backup por imágenes:** Otra opción que esta ganando popularidad es el backup por imágenes, este método sobrepasa gran parte de las utilitades del sistema operativo, copiando bloques del disco duro de manera directa. Esto le permite ser mucho más eficiente a la hora de copiar archivos que han sido modificados, esto es porque no es necesario copiar todo el archivo, solo los bloques que han sido modificados.

Cabe destacar que estos métodos no son mutuamente exclusivos, se pueden usar en conjunto para obtener mayor eficiencia y robustez. Por ejemplo se puede tener un sistema que haga un backup por imágen diariamente y uno por archivos semanalmente.

Una vez que se decidió que metodo utilizar para hacer los backups ahora hay que decidir que método usar para los mismos. Los backups se dividen en tres tipos:

- **Backup completo:** Es el mas simple y el método original que se usaba para hacer los backups. Copia toda la información en el sistema especificado. Lo bueno de este método es que el backup es autocontenido, esto significa que no se requiere de ningún otro tipo de información o archivo para que este funcione. Por el otro lado, se necesitan grandes cantidades de espacio y pueden ser casi idénticos a backups completos anteriores.

- **Backup diferencial:** Este tipo de backup solo copia las diferencias entre el sistema actual y el del último backup completo. La principal ventaja de este método es que es mucho mas rápido y ocupa mucho menos espacio que un backup completo. La desventaja es que para poder recupera la informaciñ con un sistema de backup diferencial se necesita el últim backup completo junto con el backup diferencial.

- **Backup incremental:**El backup incremental solo copia diferencias entre el el sistema actual y el último backup completo, diferencial o incremental. La ventaja es que es aún mas rápido y ocupa menos espacio que un backup diferencial. El gran inconveniente con esta forma

[1]Esto se debe a que para copiar un archivo utilizando el sistema operativo se debe: Encontrar los bloques en el disco duro donde se encuentra la carpeta que contiene al archivo, leer la carpeta, buscar el archivo especificado, determinar en que bloques se encuentra y finalmente copiarlo.

de backup es que para recuperar la información se necesitan todos los backups incrementales anteriores junto con el último backup completo. Debido a esto recuperar información con este tipo de sistema puede ser un proceso largo.

2. rsync

2.1. Que es rsync

rsync es una utilidad que permite transferir y sincronizar archivos, entre otras cosas, entre una computadora y un disco duro. También es capaz de realizar esta tareas usando dispositivos de red. Su uso es muy común en sistemas basados en Unix, dada su simplicidad y facilidad de uso.

El programa inicial fue escrito por Andrew Tridgell y Paul Mackerras, en C. Su primera versión se anunció en Junio de 1996, luego en 1999 Tridgell habló sobre el diseño y la implementación de **rsync** en su tesis. Es similar a la utilidad `rdist -c` creada por Ralph Campbell en 1983. Actualmente Wayne Davison se encarga de mantener el proyecto

2.2. Instalación de rsync

Antes de poder usar **rsync** tuvimos que instalarlo. Si bien hoy en día suele venir incluido con la gran mayoría de las distros, debido a que Debian 7 es bastante viejo no la incluye. Para instalarlo usamos el comando `sudo apt-get install rsync`.

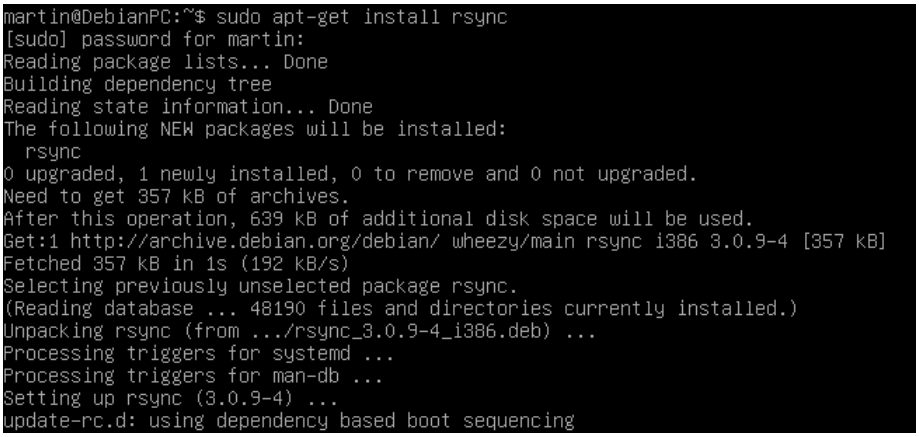

Figura 2.1: Instalamos **rsync** con `sudo apt-get install rsync`.

2.3. Uso básico de rsync

2.3.1. Preparando el disco

Lo primero que necesitamos para hacer un backup de cualquier tipo es una unidad de almacenamiento para guardar el backup. En nuestro caso simplemente creamos otro disco duro y lo "conectamos" a la VM. Para hacer esto con la máquina apagada abrimos la configuración y fuimos a `Storage`. Allí apretamos el botón para añadir un disco duro, esto nos llevo a un menú dond elegimos crear un nuevo disco. Especificamos el tipo y tamaño del disco y presionamos `Create`. Finalmente montamos el disco, para esto volvimos a apretar el botón para añadir una disco duro y seleccionamos el creado.

Es importante destacar que no se debe usar un pen drive para realizar backups. Esto es porque los transistores que almacenan información en los mismos no están hechos para soportar el número de escrituras que se necesitan para una unidad de backup. Esto a largo plazo causa que algunos de los transistores en ellos se queden "trabados"[2] en una posición haciendo imposible contiunar usándolo. Con el tiempo esto causa deteriorio en la capacidad y velocidad de funcionamiento de la unidad y puede incluso causar perdida de datos.

[2]Realmene no se quedan "trabados" sino que el transistor, normalmente un FGT, pierde su capacidad de cargarse y descargarse, y por tanto de cambiar de estado.

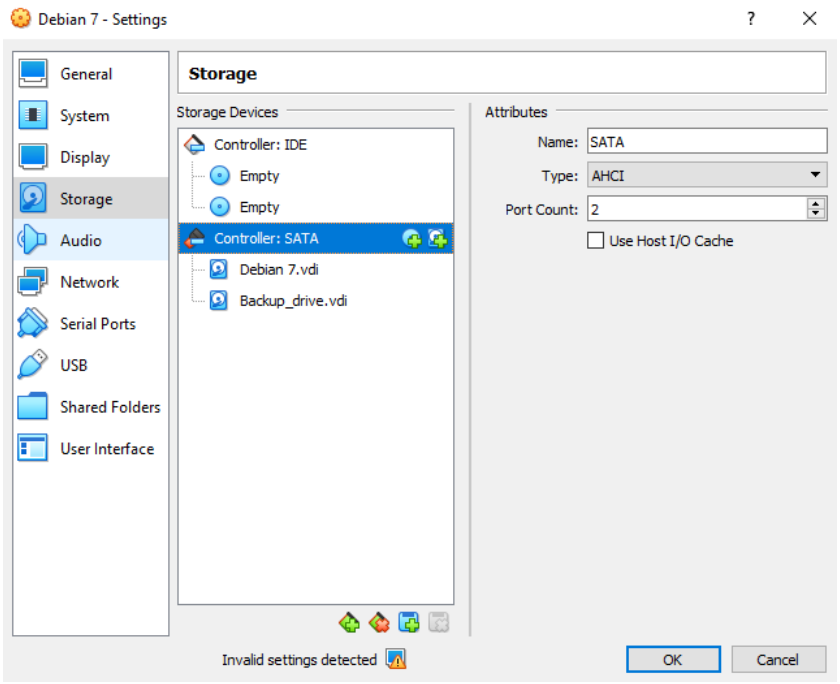

Figura 2.2: Creamos un nuevo disco y lo montamos en la VM

Ahora que nuestra VM podía ver el disco había que montarlo y configurarlo. Primero utlizamos el comando `sudo fdsik -l` para verificar que el disco fuese detectado por el sistema. Al usar el comando este nos mostró que había un disco llamado `/dev/sdb` de 10GB, ese era el disco que habíamos montado.

```
martin@DebianPC:~$ sudo fdisk -l
[sudo] password for martin:

Disk /dev/sda: 16.1 GB, 16106127360 bytes
255 heads, 63 sectors/track, 1958 cylinders, total 31457280 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x0007252e

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *        2048      499711      248832   83  Linux
/dev/sda2          501758    31455231    15476737    5  Extended
/dev/sda5          501760    31455231    15476736   8e  Linux LVM

Disk /dev/sdb: 10.7 GB, 10737418240 bytes
255 heads, 63 sectors/track, 1305 cylinders, total 20971520 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000
```

Figura 2.3: Usamos el comando `sudo fdsik -l` para verificar que el disco fuese detectado por el sistema.

Sabiendo el nombre del disco procedimos a particionarlo y formatearlo. Para esto usamos el comando `sudo cfdisk /dev/sdb`, este abrió `cfdisk` una utilidad que permite crear particiones de discos. Primero nos aseguramos de que este fuese el disco que habíamos montado y no solo uno con esa capacidad mirando el tipo de sistema de archivos, este decia `Free Space`, es decir que no estaba formateado, era nuestro disco. Entonces seleccionamos la opción `New` para crear una nueva partición, dejamos el tamaño especificado por `cfdisk`, que es el máximo que permite la unidad. Esto nos devolvió al menú principal, para confirmar los cambios usamos la opción `Write` para escribir los cambios a la tabla de discos del sistema.

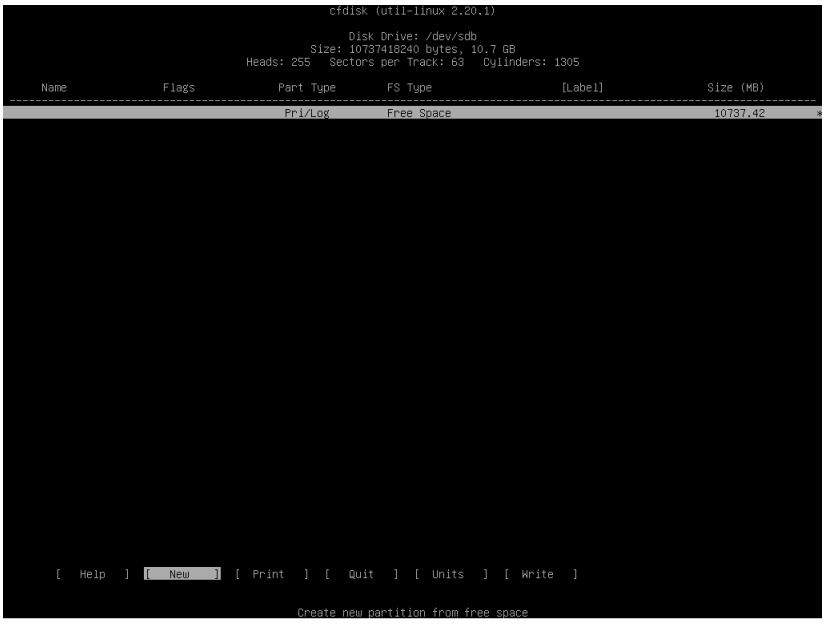

Figura 2.4: Usamos `cfdisk` para crear una partición en el disco nuevo.

Finalmente confirmamos que la operación se había realizado de manera exitosa usando la opción `Print`. Esta nos mostró que efectivamente había una partición en el disco.

```
Partition Table for /dev/sdb

        ---Starting----      ----Ending-----    Start     Number of
 # Flags Head Sect  Cyl   ID  Head Sect  Cyl     Sector    Sectors
-- ----- ---- ---- ----- ---- ---- ---- ----- ---------- ----------
 1  0x00    1    1     0 0x83  106   17  1305         63    20971457
```

Figura 2.5: Una vez creada la partición la verificamos con la opción `Print`.

Ahora que ya teníamos una particion que podíamos usar llego el momento de formatearla para eso usamos el comando `sudo mkfs.ext4 /dev/deb`. Lo que hizo fue formatear el disco con el formato `ext4`,[3] para poder así montarlo.

```
martin@DebianPC:~$ sudo mkfs.ext4 /dev/sdb
[sudo] password for martin:
mke2fs 1.42.5 (29-Jul-2012)
/dev/sdb is entire device, not just one partition!
Proceed anyway? (y,n) y
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
655360 inodes, 2621440 blocks
131072 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=2684354560
80 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done
```

Figura 2.6: Formateamos el disco con el comando `sudo mkfs.ext4 /dev/deb`.

Como último paso montamos el disco para poder usarlo. Para esto primero creamos una carpeta en la cual montar el disco, normalmente estas se encuentran en el directorio `/mnt`. Entonces creamos la carpeta `/mnt/sdb` con el comando `sudo mkdir /mnt/sdb`.

```
martin@DebianPC:~$ sudo mkdir /mnt/sdb
[sudo] password for martin:
```

Figura 2.7: Creamos el directorio donde montar el disco con `sudo mkdir /mnt/sdb`.

Finalmente montamos el disco para poder usarlo con `sudo mount /dev/sdb /mnt/sdb`. El problema con solo hacer esto es que tendríamos que volver a montar el disco cada vez que iniciacemos el sistema. Para solucionar esto cambiamos el archivo `/etc/fstab`, que almacena los discos que deben ser montados al iniciar el sistema. Abrímos el archivo para

[3] `ext4`(fourth extended filesystem) es un sistema de archivos transacional que reemplazo a ext3.

editarlo con el comando `sudo vi /etc/fstab`, luego añadimos lo siguiente al final del mismo:

```
/dev/sdb /mnt/sdb ext4 defaults 0 0
```

Figura 2.8: Editamos `sudo vi /etc/fstab` para que el disco se monte automaticamente al iniciar la máquina.

El primer elemento es el camino del disco, el segundo a donde debe ser montado y el tercero el sistema de archvios. Los demás los dejamos con sus valores por defecto. Habíendo hecho esto el disco debería montarse automaticamente cada vez que iniciemos la máquina. Como una verificación usamos el comando `mount | grep ‘‘sdb’’`, lo que hace es listar todos los discos montados y buscar uno llamado `‘‘sdb’’`.

Figura 2.9: Verificamos que el disco este estuviese montado correctamente con `mount | grep ‘‘sdb’’`.

Al ejecutar el comando vimos que efectivamente había un disco montado llamado `‘‘sdb’’`, confirmando que el disco estaba montado correctamente. Ahora que el disco estaba montado y funcionando decidimos probar almacenar algo en el, solo para probar. Nos movimos a el mismo con `cd /mnt/sdb` y luego usamos el comando `touch Prueba` para intentar crear un archivo. Al ejecutar el comando el sistema nos dijo que no teníamos los permisos para crear un archivo. Cuando listamos los permisos del disco con `ls -l ..`, se volvio claro porque. El dueño del disco era el usuario `root`, no nuestro usuario.

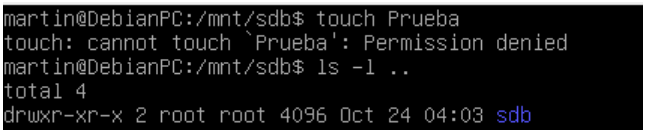

Figura 2.10: Intentamos crear un archivo con `touch Prueba`.

Para solucionar esto nos movimos a la carpeta `/mnt`(donde `sdb` se encuentra) y usamos el comando `sudo chown -R martin:martin sdb` para cambiar el dueño de la carpeta de `root` a `martin`. Habiendo hecho esto repetimos la prueba con `touch` y esta fue exitosa.

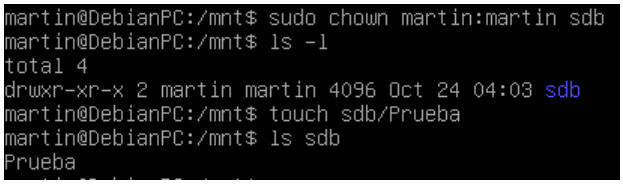

Figura 2.11: Cambiamos el dueño de `sdb` para solucionar el problema y lo verificamos.

Finalmente con el disco funcionando correctamente decidimos comenzar con la próxima parte del trabajo, hacer backups con **rsync** usando el nuevo disco.

2.3.2. Usando rsync

Habiendo llegado la hora de usar **rsync** decidimos hacer un backup completo de una carpeta en nuestro directorio propio como prueba de uso. Seleccionamos la carpeta `C`, que tenía todo el código correspondiente a el trabajo practico sobre software en formato fuente. La elegimos porque era una carpeta que a su vez tenía varias sub carpetas con archivos en ellas, lo que nos permitía probar las opciones recursivas de **rsync**. Antes de hacer el backup creamos la carpeta a donde los guardaríamos, `/mnt/sdb/Backups`, lo hicimos con el comando `mkdir /mnt/sdb/Backups`.

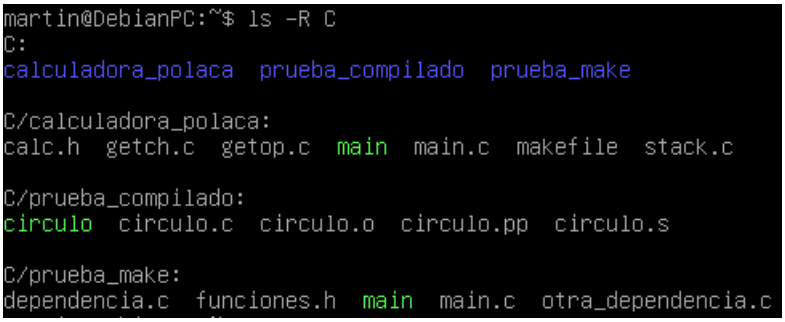

Figura 2.12: Los contenidos del directorio `C`.

El uso básico de **rsync** es bastante simple, primero se pone el directorio desde el que se quiere copiar y luego hacia cual se quiere copiar. Para empezar esto fue lo que hicimos, usamos el comando `rsync ~/C /mnt/sdb/Backup`, al hacerlo el mensaje *skipping directory C* apareció. Cuando revisamos los contenidos de `Backup` vimos que estaba vacío. Esta situación se explica si miramos los contenidos del directorio `C`, dentro de el solo hay carpetas, no archivos. Entonces al no encontrar archivos que copiar dentro de la carpeta especificada **rsync** simplemente la omitió y no copió nada. Dado este problema añadimos la opción `-r` a el comando, para copiar tambíen los contenidos de todas las subcarpetas. Al ejecutar el comando y luego verificar los contenidos de `Backup` vimos que ahora había una carpeta llamada `C` en ella, indicando que los archivos efectivamente se habían copiado.

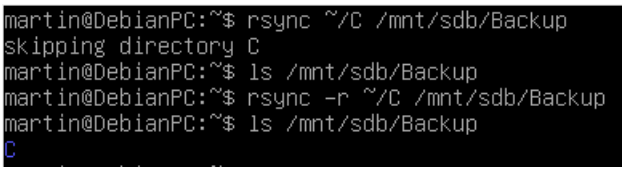

Figura 2.13: Nuestro primer intento de usar **rsync**.

Para asegurarnos de que la copia hubiese efectivamente ocurrido listamos los contenidos de `Backup` con `ls -Rl /mnt/sdb/Backup`, esto nos reveló algo interesante, todos los archivos tenían la fecha de modificación de cuando habíamos realizado el backup. Esto tenía sentido ya que esos archivos se crearon y modificaron al momento en que nosotros realizamos el backup. Sin embargo nos preguntamos si había alguna manera de conservar la fecha original. Entonces decidimos leer las opciones de **rsync**, para ver si se podía,

junto con tal vez encontrar otras opciones útiles.

```
martin@DebianPC:~$ ls -Rl /mnt/sdb/Backup
/mnt/sdb/Backup:
total 4
drwxr-xr-x 5 martin martin 4096 Oct 25 00:28 C

/mnt/sdb/Backup/C:
total 12
drwxr-xr-x 2 martin martin 4096 Oct 25 00:28 calculadora_polaca
drwxr-xr-x 2 martin martin 4096 Oct 25 00:28 prueba_compilado
drwxr-xr-x 2 martin martin 4096 Oct 25 00:28 prueba_make

/mnt/sdb/Backup/C/calculadora_polaca:
total 32
-rw-r--r-- 1 martin martin  239 Oct 25 00:28 calc.h
-rw-r--r-- 1 martin martin  989 Oct 25 00:28 getch.c
-rw-r--r-- 1 martin martin  727 Oct 25 00:28 getop.c
-rwxr-xr-x 1 martin martin 8063 Oct 25 00:28 main
-rw-r--r-- 1 martin martin 1486 Oct 25 00:28 main.c
-rw-r--r-- 1 martin martin  302 Oct 25 00:28 makefile
-rw-r--r-- 1 martin martin  601 Oct 25 00:28 stack.c

/mnt/sdb/Backup/C/prueba_compilado:
total 56
-rwxr-xr-x 1 martin martin  3649 Oct 25 00:28 circulo
-rw-r--r-- 1 martin martin   236 Oct 25 00:28 circulo.c
-rw-r--r-- 1 martin martin  1188 Oct 25 00:28 circulo.o
-rw-r--r-- 1 martin martin 40115 Oct 25 00:28 circulo.pp
-rw-r--r-- 1 martin martin   835 Oct 25 00:28 circulo.s

/mnt/sdb/Backup/C/prueba_make:
total 24
-rw-r--r-- 1 martin martin  181 Oct 25 00:28 dependencia.c
-rw-r--r-- 1 martin martin   58 Oct 25 00:28 funciones.h
-rwxr-xr-x 1 martin martin 5556 Oct 25 00:28 main
-rw-r--r-- 1 martin martin  288 Oct 25 00:28 main.c
-rw-r--r-- 1 martin martin  147 Oct 25 00:28 otra_dependencia.c
```

Figura 2.14: Los contenidos de `Backup`, mostrados con ls -Rl /mnt/sdb/Backup.

Luego de leer la página de `man` de **rsync** encontramos no solo opciones útiles sino también información sobre como mejor usar el comando y adaptarlo mejor a nuestras necesidades.

Lo primero que encontramos fue una sección referida a el uso de la barra(`/`). Si se pone al final del directorio desde el que se quiere copiar cambia la forma en la que **rsync** almacena los archivos copiados. Lo que hace es evitar crear un directorio adicional, copiando directamente los contenidos del directorio especificado sin el directorio en sí. La diferencia se puede observar

mejor en la figura 2.15.

```
martin@DebianPC:~$ rsync -r ~/C /mnt/sdb/Backup/
martin@DebianPC:~$ ls -R /mnt/sdb/Backup | head -3
/mnt/sdb/Backup:
C

martin@DebianPC:~$ rm -R /mnt/sdb/Backup/*
martin@DebianPC:~$ rsync -r ~/C/ /mnt/sdb/Backup/
martin@DebianPC:~$ ls -R /mnt/sdb/Backup | head -4
/mnt/sdb/Backup:
calculadora_polaca
prueba_compilado
prueba_make
```

Figura 2.15: La diferencia entre usar y no usar la barra en **rsync**.

En cuanto a las opciones encontramos varias que nos parecieron bastante útiles. La primera que probamos fue `-v`(verbose), como indica su nombre lo que hace es hacer la salida de **rsync** mucho más verbosa haciendo que muestre que es lo que esta copiando asi como cuantos bytes copia y cuanto tiempo le lleva. Otra opción que también nos pareció útil pero algo peligrosa es `--delete`. Lo que hace es eliminar cualquier archivo en el directorio de destino que no este en el directorio a copiar. La razón de su peligrosidad es su principio de funcionamiento, si hay algun archivo en la carpeta de destino que desamos conservar pero este no esta en la carpeta a copiar sera borrado.

Luego de un poco mas de búsqueda encontramos la opción `-a`(archive), esta es equivalente a un gran número de opciones. Es la manera de decir que uno quiere preservar casi todos los atributos sobre los archivos copiados, la unica exepción es que no preserva los enlaces duros.[4] Las opciones a las que `-a` se expande son:

- **-r:** Hace que el directorio a copiar se copie de manera recursiva, permitiendo copiar subdirectorios y sus contenidos.
- **-l:** Mantiene los enlaces símbolicos encontrados.
- **-p:** Mantiene los permisos originales de los archivos copiados. Sin esta opción los archivos existentes(incluyendo los que se actualizan) mantienen sus permisos. Para los nuevos los bits de permisos son iguales

[4]El motivo por el que no son preservados es porque encontrar archivos que poseen múltiples enlaces es costoso en cuanto a recursos. Para preservarlos se debe usar la opción `-H`.

```
martin@DebianPC:~$ rsync -rv ~/C/ /mnt/sdb/Backup/
sending incremental file list
.swp
calculadora_polaca/
calculadora_polaca/calc.h
calculadora_polaca/getch.c
calculadora_polaca/getop.c
calculadora_polaca/main
calculadora_polaca/main.c
calculadora_polaca/makefile
calculadora_polaca/stack.c
prueba_compilado/
prueba_compilado/circulo
prueba_compilado/circulo.c
prueba_compilado/circulo.o
prueba_compilado/circulo.pp
prueba_compilado/circulo.s
prueba_make/
prueba_make/dependencia.c
prueba_make/funciones.h
prueba_make/main
prueba_make/main.c
prueba_make/otra_dependencia.c

sent 78185 bytes  received 366 bytes  157102.00 bytes/sec
total size is 76948  speedup is 0.98
```

(a) **rsync** con la opción `-v`.

```
martin@DebianPC:~$ echo "Hola" > /mnt/sdb/Backup/Prueba.txt
martin@DebianPC:~$ ls /mnt/sdb/Backup/
Prueba.txt
martin@DebianPC:~$ rsync -r --delete ~/C/ /mnt/sdb/Backup/
martin@DebianPC:~$ ls /mnt/sdb/Backup/
calculadora_polaca  prueba_compilado  prueba_make
```

(b) **rsync** con la opción `--delete`.

Figura 2.16: Probamos usar las opciones `-v` y `--delete` con **rsync**.

a los originales tras ser enmascarados con los bits de permiso de los predeterminados del directorio de destino.

- **-t:** Hace que los tiempos de modificación se transfieran de los archivos originales a los copiados. Si esta opción no se incluye, durante el próximo backup todos los archivos van a ser actualizados, es decir copiados nuevamente.
- **-g:** Mantiene el grupo de los archivos copiados. Si esta opción no se incluye los archivos copiados toman el grupo de el usuario que haya ejecutado el comando.
- **-o:** Mantiene el dueño de los archivos copiados. Si esta opción no se incluye el dueño de los archivos copiados pasa a ser el usuario que haya ejecutado el comando.

- **-D:** Esta opción se expande a dos opciones más, son:
  - **--devices:** Hace que se copien los archivos de character y bloqueo de dispositivo a el destino para recrear los dispositivos. Esta opción no tiene efecto si **rsync**no se ejecuta como `root`.
  - **--specials:** Hace que se copien todos los archivos especiales.

Habiendo analizado las opciones de **rsync**decidimos que la configuración ideal, al menos en nuestro caso, sería `rsync -avH  /C/ /mnt/sdb/Backup`. Teniendo nuestra configuración probamos usar el comando y funciono perfectamente, haciendo un backup de todos los elementos en el directorio `C`.

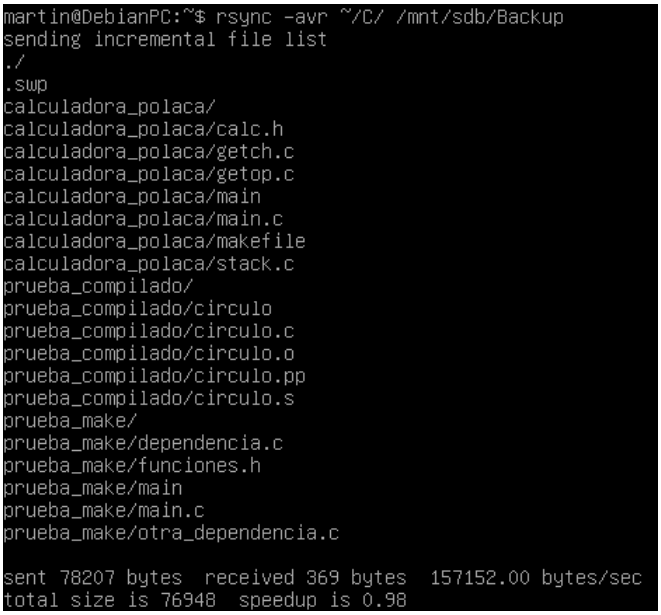

Figura 2.17: Usamos **rsync** con todas las opciones que decidimos.

Para verificar que el backup hubiese funcionado correctamente listamos los contenidos de el `Backup`con el comando `ls -lr /mnt/sdb/Backup/`. Efectivamente había funcionado de manera correcta, todos los archivos estaban y tenían todos los atributos correctos(dueño, permisos, última modificación, etc).

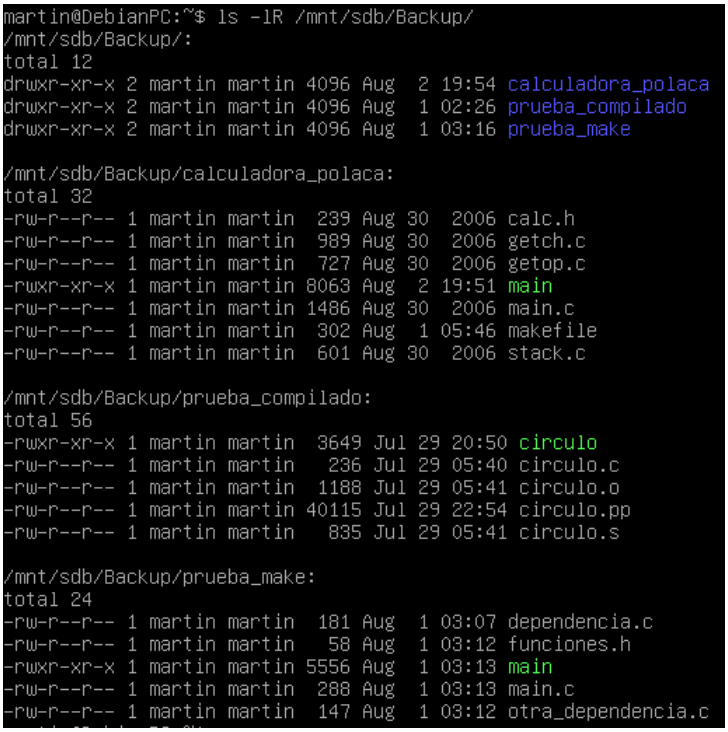

Figura 2.18: Verificamos que los resultados del backup fuesen correctos.

Como nota final creemos que vale la pena aclarar que debido nuestra configuración **rsync** hace backups incrementales,[5] dado que los archivos copian su fecha de modificación. Por lo que si intentamos hacer un backup de la misma carpeta sin cambiar nada nignún archivo se copiará.

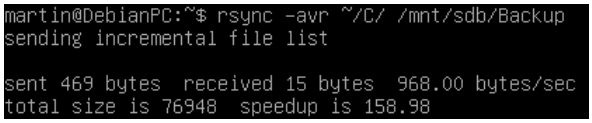

Figura 2.19: Revisamos si **rsync** hacía backups diferenciales con nuestra configuración.

[5]Referirse a sección 1.2 - Tipos de backups.

3. Comandos usados

A continuación se encuentran todos los comandos utilizados en este trabajo, correspondientes a las imágenes presentadas.

```
sudo apt install rsync
```

[2.1]

```
sudo fdisk -l
```

[2.3]

```
sudo mkfs.ext4 /dev/sdb
```

[2.6]

```
sudo mkdir /mnt/sdb
```

[2.7]

```
mount | grep ‘‘sdb’’
```

[2.9]

```
touch Prueba
ls -l
```

[2.10]

```
sudo chown martin:martin sdb
ls -l
touch Prueba
ls sdb
```

[2.11]

```
ls -R C
```

[2.12]

```
rsync ~/C /mnt/sdb/Backup
ls /mnt/sdb/Backup
rsync -r ~/C /mnt/sdb/Backup
ls /mnt/sdb/Backup
```

[2.13]

```
ls -Rl /mnt/sdb/Backup
```

[2.14]

```
rsync -r ~/C /mnt/sdb/Backup/
ls -R /mnt/sdb/Backup | head -3
rm -R /mnt/sdb/Backup/*
rsync -r ~/C/ /mnt/sdb/Backup/
ls -R /mnt/sdb/Backup | head -4
```

[2.15]

```
rsync -rv ~/C/ /mnt/sdb/Backup/
echo ‘‘Hola’’ > /mnt/sdb/Backup/Prueba.txt
ls /mnt/sdn/Backup
rsync -r --delete ~/C/ /mnt/sdb/Backup/
ls /mnt/sdn/Backup
```

[2.16]

```
rsync -avr ~/C /mnt/sdb/Backup
```

[2.17]

```
ls -lR /mnt/sdb/Backup
```

[2.18]

```
rsync -avr ~/C /mnt/sdb/Backup
```

[2.19]