Entrega SO

RAID

En este punto se va a crear un RAID Level 5 por software. Para simular los 3 discos necesarios para un RAID de este nivel se deberán generar 3 particiones del mismo tamaño. Una vez realizado este paso se generará el RAID. Para esto se utilizará la herramienta `mdadm` que deberá instalarla en la VM provista por la cátedra usando el siguiente comando: `apt-get install mdadm`.

1. ¿Qué es un RAID? Explique las diferencias entre los distintos niveles de RAID.

   RAID(Redundant Array of Independent Disks) es una técnica que permite usar múltiples discos en forma conjunta con el fin de construir un sistema de discos más rápido, más grande y más confiable. Originalmente, definía los niveles 1, 2, 3, 4 y 5; sin embargo, en 1989, se introdujo un nuevo nivel: el RAID 6 que trajo, entre otras cosas, ventajas sobre un solo disco, incremento de la perfomance, mayor capacidad y aumento de la confiabilidad.

   - RAID 0: no es un nivel de RAID ya que no existe redundancia. Se requieren al menos dos discos para conformarlo. Es un array de discos con *striping* a nivel de bloque. La capacidad del RAID está definida por la suma de la capacidad de los discos. Si falla un disco, los datos de todos los otros se vuelven inaccesibles.
   - RAID 1: asegura redundancia mediante mirroring. Mínimo de 2 discos: trabaja con pares de discos. No hay *striping*. Almacena datos duplicados en discos separados o independientes. Es ineficiente por la escritura en espejo y desperdicia el 50% de la capacidad total.
   - RAID 2: distribuye los datos entrelazados a nivel de bit. El código de error se intercala a través de varios discos también a nivel de bit. Todo giro del cabezal de disco se sincroniza y los datos se distribuyen en bandas de modo que cada bit secuencial está en una unidad diferente. La paridad se calcula a través de y los bits correspondientes y se almacena en al menos un disco de paridad. Este nivel es sólo significante a nivel histórico y teórico, ya que actualmente no se utiliza.
   - RAID 3: un RAID 3 divide los datos a nivel de bytes en lugar de a nivel de bloques. Los discos son sincronizados por los drivers para funcionar al unísono. Éste nivel actualmente no se usa. Permite tasas de transferencias extremadamente altas. Un RAID 3 necesitaría un mínimo de tres discos, utilizando uno para datos de paridad. En estos se copian los datos en distribución RAID 0 en los 2 primeros discos, sin embargo, en el tercer disco, se crea el byte de paridad. Esto quiere decir que si, por ejemplo, perdemos un byte de uno de los discos, siempre podremos recuperarlo mediante el byte de paridad que se ha generado anteriormente.
   - RAID 4: un RAID 4, también conocido como IDA (Independient Disk Array), usa división a nivel de bloques con un disco de paridad dedicado. Necesita un mínimo de 3 discos físicos. El RAID 4 es parecido al RAID 3 excepto porque divide a nivel de bloques en lugar de a nivel de bytes. Esto permite que cada miembro del conjunto funcione independientemente cuando se solicita un único bloque. Si el driver de disco lo permite, un conjunto RAID 4 puede servir varias peticiones de lectura simultáneamente. En principio también sería posible servir varias peticiones de escritura simultáneamente, pero al estar toda la información de paridad en un solo disco, éste se convertiría en el cuello de botella del conjunto.
   - RAID 5: es una de las implementaciones más utilizadas. Requiere al menos 3 discos. El *striping* se da a nivel de bloque y la paridad es distribuida. La información de esta está distribuida entre

todos los discos del array. El rendimiento es alto y no hya cuellos de botella; sin embargo, no hay solución al fallo simultáneo de discos.
    - RAID 6: es únicamente recomendado cuando se tienen varios discos, ya que requiere al menos 4. El *striping* se da a nivel de bloque y la paridad es doble y distribuida. Tiene una alta tolerancia a fallos (hasta dos discos); sin embargo, dado que la paridad es doble las operaciones de escritura son más lentas.

2. Utilizar el comando `parted -l` (debe ejecutarlo con `sudo`) para ver la tabla de particiones. Conteste:

    1. ¿Cómo llama Linux al dispositivo físico? ¿Cuál es su tamaño total?

        Lo llama *sda*. Su tamaño total es 172GB.

    2. ¿Cuántas particiones existen? ¿Qué tipo de file-system contiene cada una?

        Existen 3 particiones: *sda1* que es una primaria de tipo ext4, *sda2* que es otra primaria de tipo linux-swap(v1) y *sda3* que es otra primaria de tipo ext4(esta última es la que tuve que crear antes para el último punto de la primera parte).

    3. ¿Qué significa el flag *boot*?

        Es lo que le indica al master boot record qué partición ha de bootear.

3. Usando el comando `parted` crear una nueva partición de tipo extendida, si es que no existe previamente (debe seleccionar el dispositivo donde se van a generar las particiones):

    1. `sudo parted`

    2. *(parted) print*

        Este comando te muestra tus particiones actuales, es útil porque en el próximo comando te va a pedir valores de acá.

    3. *(parted) mkpart*

        Este es el comando para crear una partición. Inicia el proceso interactivo.

    4. Tipo de partición: extendida.

    5. Inicio: #MB (igual al tamaño final de la partición existente más alta.

        Se utiliza el tamaño final de la partición existente más alta para que sea lo más contiguo posible.

    6. Fin: #MB (igual al tamaño máximo del dispositivo físico).

        Si bien el tamaño máximo es 172GB, estoy seguro que no se va a usar ni un cuarto de eso y no voy a asignar disco en algo que no voy a usar, por lo que decidí bajar a 50GB y que la partición tenga 10GB (que también me parece mucho, pero más razonable). La cantidad que de espacio que va a tener la partición se define como *end - start*, esto es, si mi start fue 40GB y mi end 50GB, la partición va a tener 10GB.

    Dentro de esta partición extendida crear 3 nuevas particiones de 300MB cada una. Para esto utilizar nuevamente el comando `mkpart`, pero debe seleccionar logical como tipo de partición. Como tipo de

sistema de archivos elija ext4.

Para este caso usé el comando `mkpart` con parámetros y lo construí de la siguiente manera: `mkpart part-type fs-type start end`. Las particiones quedaron aproximadamente de 300MB(287, 299 y 299 MB respectivamente).

Observación: como las particiones lógicas se crean dentro de una partición extendida, los valores de inicio/fin de cada partición lógica deben estar dentro de los valores seleccionados en la partición extendida.

4. ¿Por qué es necesario crear una partición extendida? Si se usase GPT, ¿Sería necesario este tipo de particiones?

   La tangente pasa por el tipo de tabla de particiones que tiene el disco definido de la máquina virtual: es un MBR, lo cual tiene un límite de cuatro particiones. Esto significa que si yo quisiera crear tres particiones más no podría, ya que antes ya habían otras dos (la partición del sistema y la swap) creadas. Como mucho podría crear otras dos, pero esto no es lo que se busca. Para eso es que uso la partición extendida, que me permite crear unas cuantas particiones lógicas dentro de esta (aparentemente el máximo es 65536, pero en linux se limita a 60, lo importante es que me permite crear todas las particiones extras que necesito). Se crea una partición extendida y luego otras tres lógicas dentro de los límites de la primera y asunto resuelto.

   Si se usara GPT, la historia sería completamente diferente. Aparentemente, GTP permite definir tantas particiones como yo quiera, aunque el límite lo define el sistema operativo (en este caso, el límite es de 128 particiones, pero lo importante, nuevamente, es que ya con ese número podemos definir las 3 particiones que necesitábamos, sin necesidad de recurrir a la partición extendida--de hecho, a priori, solo se pueden crear particiones primarias en GPT, aunque también es posible convertir de una primaria a una extendida en GPT--)

5. Mirar nuevamente la tabla de particiones para ver las nuevas particiones y salga del entorno `parted`.

   Para salir usé `parted quit`. Era necesario refrezcar el */etc/fstab*, por lo que reinicié la máquina virtual (hay otras formas, pero esa era la más cómoda).

6. Utilizar el comando `mdadm` para crear un RAID 5 utilizando las 3 particiones lógicas que se generaron en el punto anterior (`fdisk -l` para ver el nombre de las particiones que generaron).

   ```
   mdadm --create --verbose /dev/md0 --level=5 --raid-devices=3 /dev/sda5 /dev/sda6 /dev/sda7
   ```

   Observación: *md0* es el nombre que le dará al nuevo RAID.

7. ¿Qué significan los valores sda5, sda6 y sda7?

   SD es la forma en que se nomenclan los discos duros en Linux. La letra A significa que estamos hablando del primer disco duro detectado. Los diferentes número hacen referencia a las diferentes particiones del disco, en este caso, las particiones 5, 6 y 7.

8. Ejecutar la siguiente consulta y contestar:

   ```
   mdadm --detail /dev/md0
   ```

1. ¿Cuál es el tamaño total del RAID?

   El tamaño total que me figura es 570.43 MB. El campo que usé es *Array Size*. La línea decía lo siguiente: **Array Size : 557056 (544.00 MiB 570.43 MB)**

2. ¿Cuál es el tamaño total utilizable para almacenar datos?

   El tamaño total que me figura es 544.00 MB. El campo que usé es *Array Size*. La línea decía lo siguiente: **Array Size : 557056 (544.00 MiB 570.43 MB)**

9. Analizar el contenido del siguiente comando:

   ```
   cat /proc/mdstat
   ```

   El /proc/mdstat muestra una instantánea del estado del RAID del kernel. La línea que dice *personalities* indica qué niveles de RAID soporta el kernel. Para soportar más niveles basta cambiar los módulos referentes al RAID o recompilar el kernel. La línea de abajo indica el estado del dispositivo RAID(debería ser *active*. Si es *inactive*, estamos en problemas) y los dispositivos que lo conforman. El orden de estos no significa nada. La siguiente línea muestra el tamaño del array, el tamaño usable y el algoritmo que usa. Al final de esa línea aparece algo similar a esto: [n/m]. Lo que significa es que, idealmente, el array tendría que tener **n** dispositivos, pero los cierto es que **m** dispositivos están en uso. Si m >= n, entonces está todo OK. Después de [n/m], aparece algo similar a esto: [UUU]. Eso representa el estado de los dispositivos. La cantidad de U varían y también podrían aparecer _. Las U representan dispositivos *up* y los _, dispositivos *down*.

   Observación: puede suceder que al reiniciar la VM el RAID se vea como de solo lectura y con el número 127. Para solucionar esto deben ejecutar los comandos `mdadm –stop /dev/md127` para parar el RAID y `mdadm –assemble –scan` para volverlo a generar como *md0* y de lectura/escritura. Esto se debe hacer cada vez que se inicia la VM. Si quiere que quede en forma persistente a través de los reboots debe guardar la configuración en el archivo **mdadm.conf**, `mdadm – assemble –scan >> /etc/mdadm/mdadm.conf` y luego `update-initramfs -u` (esto último puede tardar un poco de tiempo).

10. Ahora se va a probar la funcionalidad del RAID 5. Para esto completar los siguientes pasos:

    1. Crear un file system de tipo ext4 en el RAID 5 recién generado

       ```
       mkfs.ext4 /dev/md0
       ```

    2. Montar la partición con el file system generado en el directorio */mnt/rd5*

       Para montar el filesystem, antes hay que crear el directorio con `mkdir /mnt/rd5`. Una vez creado el directorio, se procede a montar el filesystem con el comando `mount` de la siguiente manera: `mount el/dispositivo/a/montar el/directorio/donde/lo/voy/a/montar`. En síntesis, el comando `mount` que usé fue `mount /dev/md0 /mnt/rd5`.

    3. Crear un directorio con dos archivos.

       Para esto, primero creo el directorio haciendo `mkdir /mnt/rd5/dir1`. Después, creo los archivos con `touch /mnt/rd5/dir1/file1 /mnt/rd5/dir1/file2`.

    4. Quitar una de las particiones del RAID. Para esto ponemos uno de los componentes en falla:

`mdadm --fail /dev/md0 /dev/sda7`

5. Observar el estado del RAID y contestar:

    1. ¿Cuál es el estado del RAID? ¿Cuántos dispositivos activos existen?

        El raid sigue en *active*. Hay dos dispositivos levantados y uno dado de baja por estar *faulty*.

    2. El tamaño total y disponible del RAID ¿Se modificó?

        Sigue siendo exactamente lo mismo.

    3. ¿Qué sucedería si se ejecuta el comando anterior sobre una de las particiones restantes?

        Se pierden los datos definitivamente. La pérdida de un disco degrada el rendimiento. Ya cuando se pierden dos, los datos se pierden definitivamente.

6. Quitar del RAID el componente puesto en falla en el paso anterior `mdadm --remove /dev/md0 /dev/sda7`

7. Observar nuevamente el estado del RAID y contestar:

    1. ¿Se puede acceder al directorio /mnt/rd5? ¿Están los archivos creados anteriormente?

        Sigue pudiéndose acceder y los archivos siguen estando ahí. ESto se da porque, si bien el sistema sabe que un disco ha fallado, sigue funcionando sólo con el fin de que el sistema operativo pueda notificar al administrador que una unidad necesita ser reemplazada: las aplicaciones en ejecución siguen funcionando ajenas al fallo. Las lecturas y escrituras continúan normalmente en el conjunto de discos, aunque con alguna degradación de rendimiento.

    2. ¿Qué hubiese sucedido si teníamos otra partición como “hot-spare”?

        Por lo general, las particiones *hot spare*(discos de reserva) se usan inmediatamente(y casi siempre automáticamente) tras algún fallo de un disco del RAID. Esto reduce el tiempo del período de reparación al acortar el tiempo de reconstrucción del RAID.

8. Por último, remover la partición permanentemente del RAID (Observación: esto es muy importante para que el próximo booteo `mdadm` no intente usar a esta partición como parte del RAID, lo que provocaría la pérdida de todos los datos):

```
mdadm --zero-superblock /dev/sda7
```

A partir de este momento la partición /dev/sda7 se puede utilizar como una partición común.

11. Para evitar la pérdida de datos es fundamental volver al RAID a un estado estable (sacarlo del estado degradado). Para esto se agregará nuevamente la partición /dev/sda7 que se quitó en el paso anterior.

    1. Ejecutar el comando `mdadm –add /dev/md0 /dev/sda7`

    2. Ejecutar el comando `mdadm –detail /dev/md0`

    3. ¿Qué hace el RAID con la nueva partición recientemente agregada? ¿Qué significa el estado “Rebuild Status”?

Lo que se hace es un resync, dado que deben copiarse los datos del RAID al nuevo disco agregado. El estado *rebuild* se da cuando el raid es inconsistente. Lo que se hace es precisamente reconstruir los discos para que sean idénticos.

4. ¿Es posible ingresar al recurso /mnt/rd5? ¿Se encuentran disponibles los datos creados en el punto anterior?

   Si, es posible ya que, a pesar de que uno de los discos se esté re-sincronizando, los otros dos están activos, por lo que el RAID usa eso. Si, ya que el RAID solo estuvo en estado degradado y no llegó al punto de perder definitivamente todos los datos.

12. Como los datos que mantiene el RAID son muy importantes es necesario tener un disco (partición en nuestro ejemplo) de respaldo. Para esto se va a agregar una partición como "hot-spare".

13. Usando el comando `parted` generar una nueva partición /dev/sda8 (con igual tamaño a las anteriores).

    Para esto primero hay que abri el `parted`. Luego, usé el comando `mkpart` con parámetros y lo construí de la siguiente manera: `mkpart part-type start end`. La partición quedó de 299MB.

14. Agregar la nueva partición al RAID: `mdadm --add /dev/md0 /dev/sda8`

    1. ¿Cómo se agregó la nueva partición? ¿Por qué?

       La partición se agregó como *spare*, es decir como partición extra, ya que el superbloque del RAID fue declarado solo con 3 particiones, por lo que siempre van a estar activas tres como mucho.

15. Volver a poner en falla a la partición /dev/sda7. Ver el estado del RAID y contestar:

    `mdadm --detail /dev/mda0`

    1. ¿Qué hace el RAID con la partición que estaba como spare?

       Automáticamente le cambia el estado a activa y empieza a sincronizar.

16. Por último, se eliminará el RAID creado en los pasos anteriores.

    1. Desmontar el RAID (comando umount).

       Para desmontar el RAID usé `umount /mnt/rd5`.

    2. Para cada una de las particiones del RAID ejecutar los pasos realizados cuando se quitó una partición del RAID (`mdadm` con las opciones `--fail` y `-remove`). Por cada partición que se quita ir mirando el estado del RAID para ver como se comporta.

       Cuando se marca como *faulty* la partición *sda8*, no pasa nada. Cuando se hace lo mismo con *sda7*, el estado pasa a *degraded*. Finalmente, cuando se aplica lo mismo a *sda6*, ahí se pasa a *failed*, idem con *sda5*.

    3. Remover los superbloques de cada una de las particiones:

       `mdadm --zero-superblock /dev/sda5 /dev/sda6 /dev/sda7`

    4. Remover el RAID:

```
mdadm --remove /dev/md0
```

Observación: si existe, quitar la linea ARRAY... del archivo */etc/mdadm/mdadm.conf*

5. Reiniciar y comprobar que el RAID ya no existe.

LVM (Logical Volumen Management

A continuación se creará un LVM utilizando las particiones **/dev/sda5**, **/dev/sda6** y **/dev/sda7** (respetar el tamaño y nombre de los volúmenes y directorios). En principio solo se utilizarán las particiones 5 y 6, luego se agregará la partición 7 para incrementar el tamaño de los volúmenes.

1. ¿Qué es LVM? ¿Qué ventajas presenta sobre el particionado tradicional de Linux?

   LVM es una implementación de un gestor de volúmenes lógicos para el kernel Linux. Provee un método más flexible que los convencionales esquemas de particionamiento para asignar espacio en los dispositivos de almacenamiento masivo. Sus principales componentes son:

   - Physical Volume (PV): dispositivos físicos o particiones que serán utilizados por LVM.
   - Volume Group (VG): grupo de physical volumns. Representa el data storage.
   - Logical Volume (PV): cada una de las partes en las que se dividen los volume groups.
   - Physical Extent (PE): unidades direccionables en las que LVM divide cada physical volumns.
   - Logical Extent (LE): unidad de asignación básica en los LVs.

   Algunas ventajas de usar LVM son:

   - Particionamiento simplificado: con el uso de LVM, el disco completo puede ser asignado a un único grupo lógico y definir distintos volúmenes lógicos para almacenar /home u otros directorios. En el caso de que nos quedemos sin espacio, por ejemplo, en /home, y tenemos espacio en /opt, podríamos redimensionar /home y /opt y usar el espacio que le hemos quitado a /opt y añadírselo a /home. Hay que tener en cuenta, que para realizar esto, nuestro sistema de ficheros debe soportar el redimensionado por arriba y por abajo, como ReiserFS.
   - Administración simplificada: administrar un sistema con muchos discos es un trabajo que consume tiempo, y se hace particularmente complejo si el sistema contiene discos de distintos tamaños. Equilibrar los requerimientos de almacenamiento de distintos usuarios (a menudo conflictivos) puede ser una tarea muy laboriosa y compleja. Los distintos grupos de usuarios pueden tener sus volúmenes lógicos y éstos pueden crecer lo que sea necesario, y el administrador puede realizar las operaciones oportunas sobre dichos volúmenes.

2. ¿Cómo funcionan los *snapshots* en LVM?

   Las instantáneas (*snapshots*) permiten al administrador crear un nuevo dispositivo que será una copia exacta del LV, congelada en algún punto del tiempo. Normalmente esto se realiza de forma automática, para no alterar el funcionamiento normal del sistema. Cuando la instantánea ha terminado, el administrador puede quitar el dispositivo sin mayor complicación. Además, no es necesario que los datos en el LV se encuentren en un estado consistente, ya que muchos sistemas de ficheros en el kernel 2.6 lo realizan de forma automática.

3. Instalar la herramienta lvm2: `apt-get install lvm2`

4. Ejecutar el siguiente comando:

   `pvcreate /dev/sda5 /dev/sda6`

   ¿Qué es lo que realiza?

   El comando anterior crea dos volúmenes físicos.

5. Mediante el comando `pvdisplay` observar el estado del volumen físico recientemente creado.

6. Crear un grupo de volúmenes (volume group, VG) llamado *so_X*, donde X es el número de grupo asignado.

   `vgcreate so_X /dev/sda5 /dev/sd6`

   En mi caso, yo era el grupo 55, por lo que ejecuté `vgcreate so_55 /dev/sda5 /dev/sd6`

7. Utilizar el comando `vgdisplay` para ver el estado del VG ¿Cuál es tamaño total del VG? ¿Qué significa PE? ¿Qué define?

   El tamaño total es 556 MiB. Cada volumen físico se divide en chunks de data, conocidos como *physical extent* (PE), los cuales tienen el mismo tamaño que los logical extents para el grupo de volúmenes. El tamaño total del VG lo saqué del campo *VG size*. A su vez, los campos sucesivos(*PE size* y *total PE*) arrojan información acerca de los physical extents.

8. Crear dos volúmenes lógicos (logical volume, LV) de 8MB y 117MB respectivamente

   `lvcreate -l 2 -n lv_vol1 so_X lvcreate -L 117M -n lv_vol2 so_X`

9. ¿Cuál es la diferencia entre los dos comandos utilizados en el punto anterior?

   La diferencia es que el primero se utiliza para el número de logical extents a asignar para el nuevo volumen lógico. Al otro le das directamente el tamaño que tendrá el nuevo volumen lógico.

10. ¿Con qué tamaño se generó el LV lv_vol2? ¿Por qué?

    Se generó con 120MiB porque lvcreate redondea al extent más cercano. Siempre redondea para arriba, nunca para abajo.

11. Formatear los dos LV generados en el paso anterior con un file system de tipo ext4:

    `mkfs.ext4 /dev/so_X/lv_vol1`

    Dado que yo era el grupo 55, los comandos que ejecuté fueron `mkfs.ext4 /dev/so_55/lv_vol1` y `mkfs.ext4 /dev/so_55/lv_vol2`.

12. Crear dos directorios, vol1 y vol2, dentro de /mnt y montar ambos LVs en estos directorios (montar el LV lv_vol1 en el directorio vol1 y lv_vol2 en vol2).

    Vamos de a partes (dijo Jack):

    - Crear los dos directorios en `/mnt`:

      `mkdir /mnt/vol1 /mnt/vol2`

- Luego, montar cada filesystem en un directorio:

```
mount /dev/so_55/lv_vol1 /mnt/vol1
mount /dev/so_55/lv_vol2 /mnt/vol2
```

13. Ejecutar el comando proof (Puede tomar un rato su ejecución. Este comando estará disponible en la plataforma y deberán copiarlo a la VM)

14. Crear un nuevo archivo en /mnt/vol1. ¿Es posible? ¿Por qué? ¿Qué debería hacerse para solucionarlo?

    Intenté crear el archivo con `touch /mnt/vol1/dir1/file1` y no fue posible ya que se había superado la máxima cantidad de inodos. Para poder crear más archivos son necesarios más inodos, por lo que se necesita extender el volumen.

15. Para aplicar la solución propuesta en el punto anterior, para esto primero se debe incrementar el tamaño del LV correspondiente:

    1. Extender el LV lv_vol1 en 20M

       `lvextend -L +20M /dev/so_X/lv_vol1`

    2. Ejecute el comando `df -h` ¿Se refleja en la salida del comando el incremento del espacio? ¿Por qué?

       No se refleja porque hasta ahora se extendió el LV, pero falta que el filesystem se entere del cambio: para eso es que se va a usar el `resize2fs`.

    3. Incrementar el tamaño del file system:

       `resize2fs /dev/so_X/lv_vol1`

16. Después de la operación previa, ¿Siguen estando los datos disponibles?

    Si, siguen estando disponibles porque extendí el filesystem

17. Intentar crear un nuevo archivo en */mnt/vol1* ¿Es posible? ¿Por qué?

    Primero creé el archivo con `touch /mnt/vol1/dir1/file1`. En este caso, es posible ya que se ha resizeado el tamaño del filesystem y ahora si se pueden usar los inodos.

18. Se desea crear un nuevo LV de 500M ¿Hay suficiente espacio? ¿Cómo lo solucionaría?

    No hay suficiente espacio ya que tengo todo usado con los dos volúmenes lógicos. Necesitaría incorporar a sda7 y crear el nuevo LV.

19. Para aplicar la solución indicada en el punto anterior, realizar lo siguiente:

```
pvcreate /dev/sda7
vgextend so_X /dev/sda7
```

20. Comprobar con los comandos correspondientes que se haya extendido el tamaño del VG.

    Si se ejecuta `vgdisplay`, se puede apreciar en la parte de *Free PE / Size* que el tamaño ha crecido.

21. Generar el nuevo LV de 500M (llamarlo lv_vol3).

    Con el comando `lvcreate -L 500M -n lv_vol3 so_55` creé el nuevo volumen lógico de 500MB.

22. Montar este nuevo LV en el directorio /mnt/lv_vol3 y crear un directorio con dos archivos adentro (los nombre pueden ser cualquiera). En el siguiente paso se intentará reducir el tamaño del LV generado en el punto 18) de 500M a 400M.

    Antes de montar el LV, tuve que **definirle un filesystem** a *lv_vol3* con `mkfs.ext4`. Un vez creado el filesystem, para montarlo, antes hay que crear el directorio, por lo que primero hay que hacer un `mkdir /mnt/vol3`. Luego se procede a montar con `mount /dev/so_55/lv_vol3 /mnt/vol3`. Finalmente, se crea el directorio dentro de */mnt/vol3* con `mkdir` y, dentro de ese directorio, los archivos con `touch`:

    ```
    mkdir /mnt/vol3/dir1/
    touch /mnt/vol3/dir1/file1
    touch /mnt/vol3/dir1/file2
    ```

23. Desmontar el LV.

    Para desmontar a *lv_vol3*, hay que hacer un `umount /mnt/vol3`.

24. Reducir el LV *lv_vol3* (decir Sí/Yes al aviso que aparece al ejecutar el siguiente comando)

    `lvreduce -L 400M /dev/mapper/so_X-lv_vol3`

25. Montar el LV reducido en el paso anterior. ¿Es posible ver el directorio y los archivos generados en el paso anterior?

    Intenté re-montar el LV *lv_vol3* con `mount /dev/so_55/lv_vol3 /mnt/vol3` pero no se pudo ya que, efectivamente, como comenté arriba, cuando reducis podés perder el filesystem. En este caso, lo perdí (junto con los archivos y el directorio).

26. ¿Cuál es el error en el procedimiento anterior? ¿Cuáles serían los pasos correctos?

    El error era tipo de sistema de ficheros incorrecto (en otras palabras, el volumen no tiene filesystem o éste está corrupto).

27. Investigar y ejecutar los pasos necesarios, y de forma ordenada, para poder achicar el LV a 400M. A continuación se mostrará el funcionamiento de los snapshot en LVM.

    Pasos para achicar el LV a 400M (y no perder el filesystem en el intento):

    1. Desmontar el filesystem (`umount /mnt/vol3`).
    2. Reducir el tamaño del filesystem. Este es el paso que antes no habíamos hecho y la razón por la que se rompió todo. Primero hay que reducir el filesystem y luego el LVM. Para reducirlo, usar `resize2fs /dev/mapper/so_55-lv_vol3 400M`. Asegurarse que el tamaño a reducir sea múltiplo del extent para evitar que se rompa todo (400 es múltiplo de 4).

3. Reducir el tamaño del LVM usando `lvreduce` (`lvreduce -L 400M /dev/mapper/so_55-lv_vol3`). Va a aparecer una leyenda que dice que PUEDE que se destruyan todos los datos (como el filesystem, por ejemplo). Ingresar ***y*** porque PUEDE que pase, pero no va a pasar porque esta vez hicimos todo bien.
4. Listo! El LVM se debería haber reducido de tamaño sin perder los datos. Como extra, para asegurarnos que todo está bien, se le puede ejecutar, forzosamente, el comando `e2fsck` (`e2fsck -f /dev/mapper/so_55-lv_vol3`), que no indicará si hay errores en el filesystem (el cual, una vez ejecutado, mostrará que no hay errores) o también se puede re-montar el filesystem (`mount /dev/so_55/lv_vol3 /mnt/vol3`) y verificar que los archivos siguen ahí (`ls /mnt/vol3`).

28. Generar un LV de 100M, nombrarlo lv1, (o usar uno de los generados en pasos anteriores). Montarlo en el directorio /dir1.

    En este caso, lo que yo hice fue reducir el volumen anterior (el *lv_vol3*) a 100Mb usando el procedimiento anterior. Luego creé el dir1 en mnt y procedí a montar el LVM en /mnt/dir1. Los comandos fueron los siguientes:

    ```
    #Resizear el volumen a 100Mb
    umount /mnt/vol3
    e2fsck -f /dev/mapper/so_55-lv_vol3
    resize2fs /dev/mapper/so_55-lv_vol3 100M
    lvreduce -L 100M /dev/mapper/so_55-lv_vol3
    #Crear el directorio y montar el volumen ya resizeado
    mkdir /mnt/vol1
    mount /dev/so_55/lv_vol3 /mnt/dir1
    ```

29. Copiar desde /etc todo los archivos y directorios que comiencen con la letra a.

    El comando que usé (siempre estando parado sobre */etc*) fue `cp -r a* /mnt/mnt/dir1`.

30. Mediante el siguiente comando generar un snapshot del LV anterior.

    `lvcreate -L 30M -s /dev/so\_X/lv1 -n lvcopy` (-s indica que este LV será un snapshot)

31. Verificar la creación del snapshot con el comando `lvs`. Montarlo en el directorio /snap. ¿Qué contenido tiene en el snapshot?

    El contenido del snapshot es exactamente el mismo que el de */mnt/dir1* (lo que dejó el comando `cp -r a* /mnt/mnt/dir1`).

32. ¿Cuánto espacio hay consumido en el snapshot creado? ¿Por qué sucede esto?

    Solo hay 0.04% consumido. Esto es así porque al crear el snapshot, se copian los metadatos. Una vez que se empiezan a modificar los bloques en el LV original es que se empiezan a mover los datos al snapshot y ahí es cuando se incrementa la cantidad de datos usados (crece el espacio asignado).

33. Para probar el snapshot, elimine una carpeta del LV original (por ej, la carpeta `apt`). ¿Se eliminó en el LV original? ¿Y qué sucedió en el snapshot?

La carpeta se eliminó del LV original, pero no del snapshot.

34. Si se desea volver el LV a su estado original se debe hacer un “merge” entre el LV y su snapshot. Para esto primero deben desmontar el LV original y su snapshot correspondiente. Luego, realizar el “merge” de ambos LVs:

    `lvconvert --merge /dev/so_X/lvcopy`

    Primero tuve que desmontar los volúmenes con umount:

    ```
    umount /mnt/dir1
    umount /mnt/snap
    ```

    Después realicé el merge.

35. Comprobar si el LV original contiene nuevamente los datos eliminados anteriormente (Deberá montarlo nuevamente).

    Haciendo un `mount /dev/so_55/lv_vol3 /mnt/dir1` y luego un `ls -l /mnt/dir1`, se puede observar que se restauró la carpeta `apt`.

36. ¿Qué sucedió con el snapshot? Obs.: en caso que aparezca el error “Can’t merge over...” ejecutar los siguientes comandos para desactivar y activar el LV

    ```
    lvchange -an /dev/so_X/lv1
    lvchange -ay /dev/so_X/lv1
    ```

    En mi caso no apareció ningún error y observé que el snapshot ya no está. Esto me permite inferir que el snapshot se puede usar una vez y luego se descarta.