



Linux: Instalar el sistema de ficheros raíz sobre RAID

Martes 18 de enero de 2005

Enviado por [Pablo Iranzo Gómez](#)

Si queremos implementar un disco espejo en el sistema raíz, nos encontramos con que muchas distribuciones no están preparadas, siendo un incordio tener que prepararlas para esta finalidad... con este artículo veremos cómo conseguirlo.

RAID, son las siglas de Redundant Array of Inexpensive Disks, es decir, Matriz redundante de discos baratos.

Los raid tienen varios niveles, aunque los más extendidos son los linear, 0, 1, 0+1, 1+0 ó 5.

Cada sistema de raid tiene sus ventajas y sus inconvenientes y deberemos escoger el más adecuado a la finalidad que le queramos aportar:

| <i>Nivel RAID</i> | <i>Nombre</i> | <i>Ventajas</i> | <i>Desventajas</i> | <i>Descripción extendida</i> |
|-------------------|---------------------------------------|--|---|---|
| linear | linear | Permite unir varios discos en uno mayor, en caso de fallo permite recuperar datos al estar los datos a continuación como si cada disco fuera una parte de un disco mayor | No proporciona incrementos de velocidad | Este sistema permite que uniendo discos más asequibles (por ejemplo hoy en día varios discos de 160Gb son mucho más baratos que un sistema de almacenamiento de 1Tb). Estos discos se unen todos entre sí y permiten formar discos de mayor tamaño que de otra forma resultaría imposible |
| 0 | Striped (bandas) | Permite unir varios discos en uno mayor, aumenta la velocidad de lectura de los datos (se leen datos a la vez de varios discos) | En caso de fallo de un disco, no es posible recuperar los datos | Este sistema, al igual que el linear, permite unir muchos discos, pero va grabando alternativamente los datos en cada disco, de forma que la escritura se realiza a velocidad normal, pues debe escribir en cada disco por separado, pero a la hora de realizar la lectura, como se leen datos de varios discos a la vez, la velocidad es mucho mayor |
| 1 | Espejo/mirror | En caso de fallo físico de un disco, los datos no se han perdido, pues existe una copia idéntica en el otro | Pierde el 50% del espacio | Este sistema popular porque permite tener copias exactas de los datos en caso de fallo mecánico del disco, tiene el gran inconveniente de ser lento escribiendo y leyendo (funciona como si hubiera un único disco) y que si se utilizan dos discos de 160 Gb, sólo estarán disponibles para su uso 160Gb, ya que los otros 160 serán una copia idéntica del otro disco |
| 2 | Codificación de corrección de errores | Separa los datos a nivel de bits en lugar de a nivel de bloques | Raramente utilizado | |



| | | | | |
|-----|--------------------------------|--|---|---|
| 3 | Paridad | Aporta posibilidades extra de seguridad, pero en caso de fallo simultáneo de dos discos los datos se pierden | Accesos muy lentos | graba los datos separados en dos discos y un tercer disco extra donde almacena los datos de paridad, que pueden utilizarse para reconstruir cualquiera de los otros dos discos en el caso de que falle sólo uno |
| 4 | Disco paridad dedicado | bandas a nivel de bloque | Puede crear cuellos de botella | Parecido al nivel 3 |
| 5 | Paridad distribuida en bloques | Gran rendimiento y bastante tolerante a fallos | En caso de fallo simultáneo de dos discos se pierden los datos, requiere al menos tres discos | Los datos se almacenan distribuidos entre tres discos, grabando en uno de ellos los datos de paridad, de forma que puede reconstruirse en caso de fallo de un solo disco |
| 0+1 | Espejo de bandas | Permite un acceso rápido a los datos | Es menos fiable que el 1+0 por lo que se utiliza menos | Se crean dos conjuntos de bandas y encima de ellas se crea un espejo |
| 1+0 | Bandas de espejo | Es de los más extendidos por ser fiable y rápido | Requiere muchos discos | Se crean muchos conjuntos espejos y luego se crean conjuntos de bandas sobre ellos |

En el kernel de Linux tenemos soporte para algunos tipos de RAID, en la carpeta `/lib/modules/verkernel/drivers/md/` podremos ver los que soporta.

Para utilizar el soporte de dispositivos múltiples (Multiple Devices: MD), la forma más sencilla de empezar, es definir las particiones del sistema de antemano y tener clara la estructura de RAID a montar.

A partir de este punto tenemos tres posibilidades:

- ▣ Utilizar la instalación manual del sistema (hablaré de Debian y su debootstrap)
- ▣ Utilizar los nuevos instaladores de [Debian](#) (válido para [Ubuntu](#))
- ▣ Convertir un sistema en ejecución a sistema raid raíz

Instalación manual del sistema

En este caso, lo más cómodo consiste en arrancar con un Live CD como Knoppix y definir las particiones.

Dentro de fdisk deberemos establecer el tamaño de las particiones en cada uno de los discos y luego cambiarles el tipo a LINUX RAID AUTODETECT (tipo fd).

Una vez definidas todas, crearemos el fichero `/etc/raidtab`:

```
raiddev          /dev/md0
raid-level       1
nr-raid-disks   2
chunk-size      64k
persistent-superblock 1
nr-spare-disks  0
    device       /dev/hda2
    raid-disk    0
    device       /dev/hdc2
```



```

raid-disk      1

raiddev        /dev/md1
raid-level     0
nr-raid-disks  2
chunk-size     64k
persistent-superblock 1
nr-spare-disks 0
  device       /dev/hda3
  raid-disk    0
  device       /dev/hdc3
  raid-disk    1

```

En este caso, definimos dos unidades raid, en base a dos discos duros, la primera, md0, es un raid de nivel 1 (espejo), formado por dos particiones, con un tamaño de bloques de 64 kb, con superbloque persistente y sin discos de reserva, a continuación definimos el orden de los dispositivos y el disco que representan.

En el segundo caso, definimos un raid de bandas con dos discos y sin discos de reserva.

Para poder hacerlo, las particiones hda2, hdc2 deben tener el mismo tamaño y estar especificadas como tipo de partición "fd" en fdisk, a su vez, hda3 y hdc3 deben estar definidas como tipo "fd", pero el tamaño no importa ya que se une para crear una unidad mayor.

Una vez creado el archivo y salvado, podemos ejecutar las órdenes:

1. mkraid /dev/md0
2. mkraid /dev/md1
3. mkfs.ext3 /dev/md0
4. mkfs.ext3 /dev/md1

y a partir de ese momento, utilizar debootstrap tras montar las particiones:

1. mount /dev/md0 /target
2. mkdir /target/home
3. mount /dev/md1 /target/home
4. debootstrap woody /target (es necesario haber iniciado previamente el bind en el cd de la knoppix)

En este proceso de instalación se descargarán los paquetes y a continuación se instalarán utilizando un chroot.

En la página <http://linuxcordoba.org/node/view/19> tenéis una pequeña guía de cómo hacer la instalación entera.

Como últimos pasos, deberemos definir en el fstab que /dev/md0 es el dispositivo raíz, así como instalar los paquetes raidtools o raidtools2 y generar el initrd de nuevo para que soporte el raid (editando el mkinitrd.conf podemos especificar cuál será la partición raíz), también deberemos modificar el lilo.conf para adaptarlo a esta situación.

Es conveniente añadir a lilo la siguiente opción: raid-extra-boot=mbr, con ella conseguimos que lilo instale su código de arranque en el MBR de cada uno de los discos implicados en el RAID... (en algunas máquinas ha sido un verdadero quebradero de cabeza el conseguir que arrancara bien, hasta por un lado utilizar el kernel 2.6.10 (para soporte de SATA de NVIDIA) y por otro activar esta opción para que escribiera el MBR de ambos discos a la vez.

Para generar de nuevo el initrd, deberemos ejecutar `mkinitrd -o /boot/initrd.img-`uname -r` /lib/modules/`uname -r``



Tras el primer reinicio correcto arrancando desde el sistema raid, podremos quitar la línea `ROOT=/dev/md0` del `mkinitrd.conf` y dejarlo en su predeterminada (`PROBE`).

Instalación usando el nuevo instalador de Debian

Con el nuevo instalador de debian, es posible, durante la fase de particionado definir particiones para raid, e incluso definir los raids, pero, al menos con Ubuntu, se informa de que no será posible arrancar el sistema si se instala así el sistema operativo...

La solución consiste en hacer caso omiso y proseguir, una vez acabada la instalación y antes de reiniciar, accederemos al disco donde hemos instalado (estará montado en una carpeta porque se ha hecho la instalación).

En dicho paso, deberemos instalar el lilo, configurar el `lilo.conf` y adaptar el `mkinitrd.conf` para indicarle que usaremos raid.

Deberemos ejecutar `mkinitrd` como en el paso anterior:

```
mkinitrd -o /boot/initrd.img-$verkernel /lib/modules/$verkernel
```

Al finalizar la instalación y reiniciar, podremos cambiar al formato original el `mkinitrd.conf`, ya que Linux reconocerá que para poder montar el sistema raíz le hace falta el soporte para `raidtools`.

Migración en vivo de un sistema en ejecución

Esta parte requiere un par de reinicios del sistema en ejecución, pero permite llevar a cabo la mayor parte de la faena sin dejar de estar operativo (y por lo tanto, permite realizarlo remotamente sin excesivos riesgos).

Una vez añadido el segundo disco que utilizaremos para el RAID, procederemos a definir las particiones como consideremos adecuado y a definir el `raidtab`.

Suponiendo que el disco original es `hda` y el segundo disco instalado es `hdc`, crearemos un `raidtab` como el que sigue:

```
raiddev          /dev/md0
raid-level       1
nr-raid-disks   2
chunk-size      64k
persistent-superblock 1
nr-spare-disks  0
  device         /dev/hda2
  failed-disk    0
  device         /dev/hdc2
  raid-disk      1
```

De esta forma, indicamos al sistema que el sistema RAID estará formado por dos discos, sin discos de repuesto y que uno de los discos ha fallado (el que actualmente todavía tiene datos), de forma que `raidtools` creará el RAID sólo con el segundo disco.

Crearemos el raid con:

```
mkraid /dev/md0 y luego formatearemos la nueva unidad con mkfs.ext3 /dev/md0
```



Una vez realizado este paso crearemos una nueva carpeta donde montar la unidad de raid recién creada:

```
mkdir /target; mount /dev/md0 /target
```

Ahora viene la tarea "larga", que es proceder a la copia de los archivos, lo primero, deberíamos crear el directorio sys, proc, tmp etc que no copiaremos desde nuestro sistema actual, y empezaremos a copiar cada una de las carpetas, recomendaría hacerlo con el Midnight Commander, ya que podremos escoger de un árbol de directorios las que queramos y procederemos a su copia.

Una vez hechos todos los pasos, vendrá la faena "larga", y consiste en modificar el lilo.conf para que aparezca /dev/md0 en lugar de la actual, y el mkinitrd.conf.

Tras estos pasos, ejecutaremos lilo con el nuevo mkinitrd ya creado como en los pasos anteriores (mkinitrd -o /boot/initrd.img-\$verkernel /lib/modules/\$verkernel) e instalaremos el lilo tanto en el arranque del disco hda como el hdc y a ser posible, habremos reemplazado el initrd del primer disco por el nuevo con soporte de RAID.

Tras ejecutar lilo y asegurarnos de que no falla, podremos proceder a reiniciar el sistema, tras el primer reinicio, tendremos ya el sistema raíz montado sobre md0 (formado sólo por hdc), así que deberemos extenderlo a hda, para ello, volveremos a particionar hda y crearemos las particiones de forma idéntica a hdc, especificando correctamente el tamaño y el tipo.

Ahora, deberemos proceder a la sincronización del raid, para ello, por un lado, modificaremos el /etc/raidtab y cambiaremos la palabra failed-disk por raid-disk, y en la línea de comandos, añadiremos las particiones de hda al raid con:

```
raidhotadd /dev/md0 /dev/hda2
```

Haciendo cat /proc/mdstat, podremos ir viendo el proceso de la sincronización de los contenidos de ambas particiones...

Aunque pudiera parecer que ya está todo, queda un pequeño detalle... generalmente initrd crea los raids montando e insertando individualmente las particiones, con el problema de que ese sistema implica que cuando creamos el raid, no había partición hda2 incorporada al raid, así que al reiniciar, el raid se habrá vuelto a romper... para solucionarlo, deberemos generar de nuevo el initrd y volver a ejecutar lilo, para que se instale en los dos discos para que en caso de fallo de uno, pueda arrancar desde el otro.

Sería interesante instalar también la utilidad mdadm, que informa mediante correo de fallos en el raid...

Cabe indicar que al crear el raid como hemos indicado, el sistema es capaz de detectar automáticamente los raids (si por ejemplo arrancáramos con una knoppix y no quisieramos montar a mano el raid, haciendo mdrun -a, el sistema identificaría particiones y reensamblaría el raid para poder acceder a el con normalidad).

En estos casos conviene ser precavido, ya que el orden en el que lo hemos definido no quiere decir que los cree en el mismo, por ejemplo, si tenemos un raid espejo y luego uno de bandas, puede que los cree en distinto orden y que intercambie los nombres, aunque los datos seguirían estando a la perfección.

De hecho, mediante este sistema, es posible también usar raids 0+1 o 1+0, sin más que definir previamente los md1, md2 y luego unirlos sobre md0 o al revés (y también migrar sistemas sobre discos normales a discos Serial ATA, etc).

Espero que te sea útil