



EL KONSULTORIO DE KLAUS



Klaus Knopper es el creador de Knoppix y co-fundador de la LinuxTag Expo. En la actualidad trabaja como profesor, programador y consultor. Si tiene algún problema de configuración, o simplemente quiere conocer mejor cómo funciona Linux, no dude en escribir sus preguntas a: klaus@linux-magazine.com

Añadir RAID

? ¿Cómo puedo montar un disco RAID 1 mediante conexión en caliente con mdadm en Debian y con el kernel 2.6.16?

He de decir que quiero añadir un segundo disco duro con el sistema ya instalado y con la siguiente configuración:

```
dev/hda1=boot; ⤴
dev/hda2=swap; ⤴
dev/hda3=/;
dev/hda5=usr; ⤴
```

```
dev/hda6=/var; ⤴
dev/hda8 =home.
```

No puedo hacer las particiones porque el sistema ya existe y tiene que usar mdadm[1] para administrar el software del array RAID.



Encontrará un completo estudio de este asunto en el HOWTO de mdadm-Raid con Debian [2]. Pero primero intentemos algunos pasos previos. Para su configuración, supongo que el primer disco duro es `/dev/hda` (el disco maestro en el primer controlador) y el segundo es `/dev/hdb` (disco esclavo en el primer controlador). En teoría obtenemos una mejor tasa de transferencia cuando configuramos el segundo disco como maestro en la segunda controladora, pero usualmente ya hay un CD-ROM, lo que podría ralentizar la transferencia de datos cuando estamos usándolo sobre el mismo cable con un disco duro. Dejemos entonces el disco maestro y el esclavo en el primer controlador: `hda` para el disco original, y `hdb` para el que sea disco imagen. Los jumpers del nuevo disco deberían estar configurados como “disco esclavo” de manera concordante, y los conectamos al mismo cable del primer disco.

Paso 0: Hacemos una copia de seguridad de toda la información relevante, que es todo lo que no seremos capaces de reinstalar desde un disco Debian (por ejemplo, nuestros trabajos y configuraciones: `/home` y al menos `/etc`).

Para la autodetección de RAID recomiendo ejecutar un kernel con RAID compilado en él o un disco RAM inicial con módulos RAID. Alternativamente podemos usar un pequeño sistema de arranque en una partición no RAID que contenga los

módulos necesarios. (En otras palabras, nuestra partición de root incluyendo `/lib/modules` estaría en una partición simple sin RAID). Probaremos la configuración completa de RAID, donde todas las particiones (excepto la de swap) tienen su mirror en `hdb`.

Para hacer autodetectable el array cambiamos el tipo de partición de todas las particiones Linux (excepto la swap) al tipo `fd`, que es la *Linux raid autodetect*. Yo uso `fdisk` para esto, pero `cfdisk` también funcionaría sin problemas. Como no estamos cambiando las ubicaciones de las particiones y los tamaños, esto no tiene efectos secundarios, y el sistema seguiría arrancando normalmente. El comando:

```
fdisk -l /dev/hda
```

debería reportar ahora algo como la salida del Listado 1 para nuestra tabla de particiones existente (`hda1 = boot; hda2 = swap; hda3 = /; hda5 = usr; hda6 = /var; hda8 = home`). Este Listado 1 omite el comienzo y fin de las particiones, dado que estos parámetros dependen de nuestra configuración.

No necesitamos volver a formatear estas particiones y mantener los archivos del sistema previamente instalados. Ahora copiamos la tabla de particiones (no la información) desde `hda` a `hdb`:

```
sfdisk -d /dev/hda | ⤴
sfdisk /dev/hdb
```

El procedimiento para añadir los discos vacíos para formar un array RAID 1 es un poco extraño. En lugar de “sincronizar” cada partición de `hda` a `hdb`, creamos archivos de sistema en un array RAID 1 “degra-

dado” donde falta el primer disco (*hda*). El software RAID tiene que almacenar alguna metainformación fuera de la estructura de datos del sistema de archivos, y si éste ocupa todas las particiones RAID, esta metainformación podría ser sobrescrita o el sistema de archivos podría dañarse. El procedimiento recomendado es crear una estructura RAID 1 en *hdb* que contenga la información de *hda*, arrancamos desde este array RAID incompleto, y añadimos *hda* como “spare disk”.

Si el kernel falla al arrancar desde un array RAID 1, aún tenemos la configuración antigua en *hda* hasta la sincronización.

Creamos el array:

```
mdadm --create /dev/md0 -l1 ↗
-n2 /dev/hdb1 missing
mdadm --create /dev/md1 -l1 ↗
-n2 /dev/hdb3 missing
mdadm --create /dev/md2 -l1 ↗
-n2 /dev/hdb5 missing
mdadm --create /dev/md3 -l1 ↗
-n2 /dev/hdb6 missing
mdadm --create /dev/md4 -l1 ↗
-n2 /dev/hdb8 missing
```

Dado que no me comentó qué había en *hda7*, lo he omitido. Añadimos *hda7* en caso necesario. Formateamos todas las particiones RAID con el sistema de archivos que prefiramos:

```
for i in `seq 0 4`; do
mkreiserfs /dev/md$i
done
```

Esto sólo escribirá en las particiones *hdb*, ya que nuestro RAID todavía no sabe nada de *hda*. A continuación, montamos y copiamos desde *hda* hasta el array. Se muestra un ejemplo para la partición *hda1* (de arranque). Copiamos las otras particiones monta-

Listado 1: Particiones	
01 Disk /dev/hda: xxx heads, 63 sectors, xxxx cylinders	07 /dev/hda3 fd Linux raid autodetect
02 Units = cylinders of xxxxx * 512 bytes	08 /dev/hda4 5 Extended
03	09 /dev/hda5 fd Linux raid autodetect
04 Device Boot Start End Blocks Id System	10 /dev/hda6 fd Linux raid autodetect
05 /dev/hda1 * fd Linux raid autodetect	11 /dev/hda7 83 Linux raid autodetect
06 /dev/hda2 fd Linux raid autodetect	12 /dev/hda8 fd Linux raid autodetect
82 Linux swap	

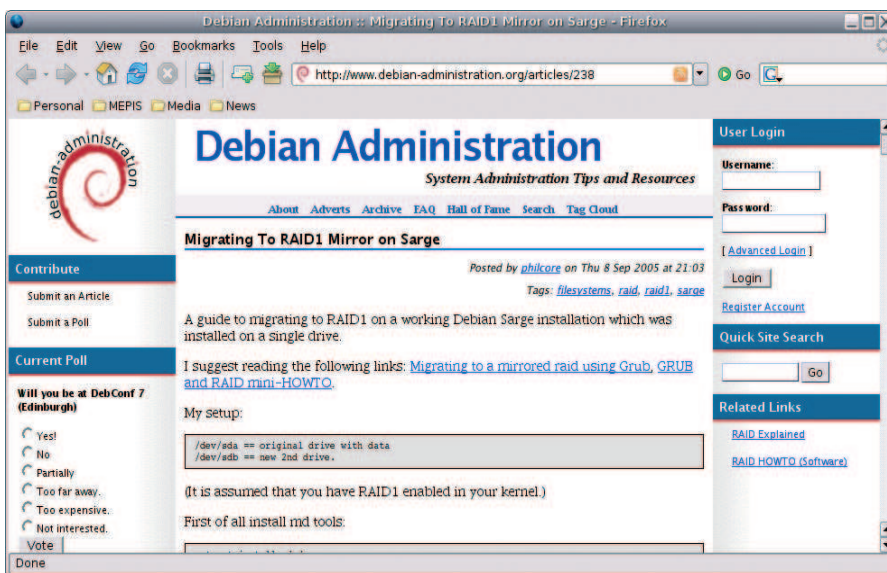


Figura 1: debian-administrator.org es una excelente fuente de información acerca de Linux RAID y otros temas de configuración

das de manera concordante. La opción *-x* de *rsync* se asegura de que sólo se copian los archivos de los sistemas de archivos montados, sin subdirectorios montados:

```
mkdir -p /mnt
mount /dev/md0 /mnt
rsync -Hax /boot/ /mnt
umount /mnt
```

Si no tenemos *rsync*, un *cp -dpRx* hace lo mismo.

Tras copiar todas las particiones a un dispositivo RAID de esta manera, tenemos que modificar */etc/fstab* en el RAID. La partición root RAID de nuestro ejemplo es */dev/md1*, por lo que montamos esto bajo */mnt* y cambiamos los archivos de dispositivo de */boot*, */*, */home*, */usr* y */var* a las correspondientes particiones */dev/md**. Para hacer uso de la partición swap duplicada usamos *mkswap* */dev/hdb2* y añadimos de igual manera esta partición como espacio swap adicional.

/mnt/etc/fstab debería contener ahora líneas similares al ejemplo del Listado 2, y podemos hacer *umount /mnt/* de nuevo. Para arrancar desde RAID, tenemos que indicarle al bootloader la nueva partición de root:

```
mount /dev/md0 /mnt
```

Editamos */mnt/boot/grub/menu.lst*, hacemos una copia de seguridad de las entradas antiguas, y añadimos las nuevas entradas de RAID 1, lo cual sería algo como:

```
title Linux (new)
root (hd0,0)
kernel /vmlinuz-2.6.21 ↗
root=/dev/md1 md=1,↗
/dev/hda3,/dev/hdb3 ro
boot
title Linux (RAID Recovery)
```

Listado 2: Ejemplo fstab			
01 /dev/md0	/boot	reiserfs	
defaults	1	2	
02 /dev/md1	/	reiserfs	
defaults	1	1	
03 /dev/md2	/usr	reiserfs	
defaults	1	2	
04 /dev/md3	/var	reiserfs	
defaults	1	2	
05 /dev/md4	/home	reiserfs	
defaults	1	2	
06 /dev/hda2	none	swap	
defaults	0	0	
07 /dev/hdb2	none	swap	
defaults	0	0	

```
root (hd1,0)
kernel /vmlinuz-2.6.21
root=/dev/md1 md=1,
/dev/hdb3 ro
boot
```

Recordemos que `/dev/md1` es nuestra nueva partición de `root`. Modificamos `/vmlinuz-2.6.21` para que coincida con nuestro kernel en la partición `boot`, y si nuestro sistema usa un `initrd`, añadimos:

```
initrd /initrd.img-2.6.21
```

justo después de la línea `kernel`, como en las entradas existentes. Para poder usar esta configuración al arrancar desde el primer disco, tecleamos:

```
cp /mnt/boot/grub/menu.lst
/boot/grub/menu.lst
```

y podemos hacer de nuevo `umount /mnt`.

Ya deberíamos estar listos para reiniciar. Generalmente, no tendríamos que reinstalar GRUB en `hda`, pero en caso de que sea necesario, tecleamos la siguiente línea:

```
grub-install /dev/hda
```

Tras reiniciar, elegimos la entrada GRUB *Linux (RAID recovery)*, que es la opción segura. Si esto no funciona, debemos vigilar los mensajes de arranque (o `panic`) atentamente en busca de lo que haya ido mal. En caso de fallo simplemente debemos arrancar nuestra vieja entrada de GRUB, ya que no hemos cambiado nada (excepto la configuración de GRUB) en el disco `hda`.

Si el sistema arranca en un modo RAID

degradado, entonces un `df` debería mostrarnos todas las particiones que antes eran `/dev/hda*` como `/dev/md*`.

Si todo funciona sin problemas, podemos iniciar la sincronización RAID de `hda` desde el `hdb` en ejecución:

```
mdadm --add /dev/md0 /dev/hda1
mdadm --add /dev/md1 /dev/hda3
mdadm --add /dev/md2 /dev/hda5
mdadm --add /dev/md3 /dev/hda6
mdadm --add /dev/md4 /dev/hda8
```

Toda la información de las particiones de `hda` se sobrescriben. En el shell podemos monitorizar el progreso con:

```
watch cat /proc/mdstat
```

Dejamos que se ejecute esto hasta que se complete la sincronización.

Pregunta sobre Arranque Dual



Me gustaría probar distintas distros, y usualmente hago arranque dual con dos sistemas Linux. Algunas veces la primera o segunda distro instalada no muestra el otro sistema en el bootloader. ¿Cómo podría conseguir un bootloader que muestre el primer sistema operativo?



No es un problema de qué bootloader usar. Podemos incluso usar el bootloader de Windows para arrancar diferentes instalaciones Linux y editar `boot.ini` de manera adecuada. Sin embargo, el bootloader tiene que saber qué sistemas operativos están instalados y dónde, por lo que no importa cual usemos (LILO, GRUB o cualquier otro): tendremos que editar el

archivo de configuración para poder indicarle qué sistemas debe mostrar en el menú de arranque. Suponiendo que el bootloader que tenemos instalado en el MBR es GRUB, y su configuración de arranque se ubica en `/boot/grub/menu.lst`, puede estudiar el Listado 3 que muestra distintos ejemplos para añadir diferentes sistemas operativos al menú de arranque.

UMASK



Soy bastante novato en Linux y he descubierto `umask`. Cuando configuro `umask 0007` y uso `chmod ug +s` en la ubicación deseada, debería ser posible hacer `log out` y registrarme de nuevo. El `umask` debería sobrevivir a la desconexión, pero no lo hace. Estoy usando Ubuntu 6.06 y 6.1, Centos 4.3 y openSUSE 10.2, y todos tienen el mismo problema. Me gustaría obtener una solución general con la que pueda terminar usando NFS, ftp o cualquier otra cosa.



El comando interno de shell `umask`, en resumen, fija una máscara para los bits de permiso que deben usarse con el conjunto completo de permisos de los archivos creados dentro del shell. En otras palabras, `umask 022` elimina los permisos de escritura de los archivos recientemente creados, mientras que `chmod 644` cambia los permisos a lectura-escritura para el propietario y a sólo lectura para el resto. `umask` se aplica sólo en el shell actual y sus descendientes, y debe fijarse con cada login. Es posible indicarle al sistema que configure esta opción de manera automática mediante `.profile` (para los shells con login), `.bashrc` (para todos los shells), o con `/etc/profile` o `/etc/bash.bashrc` (de manera global para todos los usuarios). Por supuesto, `umask` también funciona al comienzo de un script de shell.

Los permisos de `umask` en el proceso cliente se aplican cuando se accede a archivos remotos mediante NFS. Para Samba, los parámetros relevantes de `smb.conf` serían `create mask` para archivos y `directory mask` para directorios creados del lado servidor. Para ftp, la configuración de `umask` depende del servidor que elijamos. ■

Listado 3: Añadir Entradas al Menú de Arranque de GRUB

```
01 # /boot/grub/menu.lst for GRUB          12
    MBR                                     13 # Fedora on Partition 3
02 # By default boot the first            14 title Fedora
    menu entry.                             15 root (hd0,2)
03 default 0                               16 kernel /boot/kernel-2.6.18.1
04                                         17 root=/dev/hda3
05 # Allow 30 seconds before              17 initrd /boot/initrd-2.6.18.1
    booting the default.                     18
06 timeout 30                               19 # Partition 2 is Linux Swap,
07 ignore.                                    ignore.
08 # Debian GNU/Linux on                  20
    Partition 4                               21 # Windows on Partition 1
09 title Debian Etch                       22 title Windows
10 root (hd0,3)                             23 rootnoverify (hd0,0)
11 kernel /boot/vmlinuz-2.6.20.1          24 chainloader +1
    root=/dev/hda4
```

RECURSOS

[1] `mdadm`: <http://neil.brown.name/blog/mdadm>

[2] `mdadm` con Debian: <http://www.debian-administration.org/articles/238>