Bisoños Usuarios de GNU/Linux de Mallorca y Alrededores | Bergantells Usuaris de GNU/Linux de Mallorca i Afegitons

Instalar Debian en un RAID Promise PDC202xx en 20 minuts / Installing Debian on RAID Promise PDC202xx. (21143 lectures)

Per **Joan Miquel**, ***Joanmi*** (http://www.mallorcaweb.net/joanmiquel)

Creado el 16/12/2001 16:06 modificado el 02/03/2002 18:09

Els sistemes RAID (Redundant Array of Independent Disk) permeten combinar varis discs durs (o particións en el cas de Software-RAID) perque treballin com un tot sol oferint diversos avantatges segons el tipus (o combinació) d'array RAID que fem servir.
A més, gràcies a l'aparició de controladores IDE-RAID i al rendiment que es pot assolir ara amb el nou standard IDE-ATA100, podem conseguir prestacions similars als clàssics RAID scsi però a un cost molt inferior.

- ENGLISH VERSION[1] -

RAID systems (Redundant Array of Independent Disk) let you to combine many hard disks (or partitions in software-RAID case) to work as it were only one disk or partition also offering some other advantages depending on the RAID-type (or combination) that we use.
Althought, thanks to the new IDE-RAID controllers and to the performance availible because of the new IDE-ATA100 standard, we can obtain similar prestations than the obtained whith the classics scsi RAIDs but whith a so lower cost.

Evidentment el RAID per hardware sempre és la millor opció possible ja que no suposa cap càrrega pel sistema operatiu i, a més, pot paral·lelitzar realment les operacións de lectura/escritura als diferents discs durs connectats al RAID. ...Però també és la més cara ja que pot suposar un fort increment en el cost del nostre servidor.

Però com he comentat abans, aquest increment de cost pot ser molt més assequible si fem servir una controladora RAID IDE i discs durs IDE. Seguidament vos explicaré com ho he fet jo amb la controladora PROMISE PDC20267 del nou servidor de BULMA[2]. Però les instruccións aquí donades també haurien de ser extensibles a la resta de controladores de la sèrie PDC202xx soportades pel kernel (46, 62, 65 i 68 en el kernel 2.4.16 que és el que he fet servir jo).

En primer lloc, pels que encara no tengueu molt clar què és això del "RAID" una breu explicació de en que consisteix. Els que ja ho tengueu clar podeu passar dirèctament a la pàgina següent per començar la instalació de la vostra Debian.

RAID i tipus de RAIDs:

Com he explicat abans, una controladora RAID (em centraré en el hardware-raid que és el que ens ocupa) ens permet combinar diversos discs durs perque treballin com si fossin un tot sol. Per això, la controladora RAID tendrà una BIOS a la que podrem configurar diferents *arrays* de discs durs que seràn el que realment veurem des del nostre sistema operatiu.

...bé: això no és exactament així. Amb un WindowsXX _sí_ que només veurem el resultat de l'array, però des de Linux tenim accés tant a l'array com als discs durs per separat. Evidentement, el que passi si accediu independentment als discs individuals és només responsabilitat vostra.

Tipus d'arrays RAID:

Segur que ja fa estona que li estau donant voltes a com es combinen els discs durs d'un array RAID i que ja se vos han ocorregut vàries idees de com poden fer-ho i vos preguntau quina és la bona. Idò bé: Probablement la resposta a aquesta pregunta sigui que totes.

S'han estandaritzat diferents tipus d'arrays RAID que ofereixen diferents avantatges segons la manera en que combinen els discs durs que el composen. Seguidament es relacionen dos dels diferents tipus o "nivells" de RAID que existeixen. Però abans, comentar que un array pot estar composat de n discs durs o també de altres arrays d'un altre tipus. Per exemple, podriem definir dos arrays de tipus 0 de dos HD cada un i, després, fer un array 1 d'aquests dos arrays.

Els tipus de RAID més comuns són el 0 i l'1. Cada controladora RAID implementarà un conjunt determinat de tipus de RAID. A continuació s'explica amb una mica de detall què fan cada un dels dos tipus de RAID esmentats.

Es suposa que la controladora sempre accepta discs de tamanys diferents, però no sempre té perquè ser així. En tot cas això dependrà de la implementació concreta d'aquell model de controladora. En qualsevol cas mirau el manual per conèixer els requeriments exactes. Però allà on hi hagi algún tipus de paral·lelisme sempre és més recomanable fer servir discs durs iguals.

- **RAID 0 (disk stripping):** Uneix n discs de tamany igual o similar en un sol disc virtual.
  - La capacitat total de l'array és igual n * c (on c és la capacitat del disc més petit de l'array).
  - La informació es reparteix en paral·lel als dos discs.
  - Es dobla el rendiment en les lectures i escriptures ja que escrivim el doble d'informació en el mateix temps.
  - La informació d'un (o d'un subconjunt) dels discs és totalment inútil sense els altres.
- **RAID 1 (disk mirroring):** Realitza una còpia (mirror) exacta entre varis discs.
  - La capacitat total de l'array és igual a la del disc més petit.
  - La informació continguda als dos discs és en tot moment idèntica.
  - Les lectures poden fer-se (segóns la implementació de la controladora):
    - ◊ En un sol dels discs (en cas de fallada d'un passarien a fer-se de l'altre).
    - ◊ La meitat de cada disc, efectuant-se simultàniament i doblant així la velocitat de lectura.
  - Les escriptures es fan de manera simultània als dos i, per tant, no es perd rendiment (per contra del que passaria amb un mirror per software).

Si voleu més informació sobre els diferents tipus de RAID, jo he trobat aquest[3] enllaç al google.

La instalació:

En primer lloc un parell d'aclaracións:

- Jo he aconseguit fer anar el RAID gràcies al petit howto que podeu trobar a http://people.redhat.com/arjanv/pdcraid/ataraidhowto.html[4]. ...però vos recoman que continueu llegint aquest article perque hi ha MOLTS detalls que allà no es comenten prou a fons.
- Tant el software de la BIOS de la controladora, com el mòdul del kernel proporcionat pel fabricant, com els drivers per Windozes (que no ens interessen) fan servir emulació d'SCSI per accedir a la controladora. El soport que vé amb el kernel oficial **NO** fa servir tal emulació perque suposa un overhead innecessari en la implementació del driver. Això provocarà alguns petits inconvenients que veurém més endavant. També heu de tenir en compte que per arrancar des d'el RAID li heu de dir a la BIOS que arranqui desde scsi NO desde IDE.
- Per a la instalació jo he fet servir el CD de Debian 2.2 (Potato). Està a punt de sortir la Debian 3.0 (Woody), però en principi tots els canvis haurien de ser cap a millor. No obstant, segurament el procés definit aquí no es podrà seguir al peu de la lletra ja que és d'esperar que Debian 3.0 surti ja amb un kernel de la sèrie 2.4.x.
  - En concret, el mòdul que facilita Promise és per kernels de la sèrie 2.2 i, en principi, no funciona amb kernels de la sèrie 2.4, pel que, si el sistema d'instalació de Debian 3.0 no inclou els mòduls necessaris per fer anar el RAID, haurieu de compilar-los apart i carregar-os a mà (i els drivers del kernel són 6 mòduls diferents, no un totsol).
  - Si no vos va bé, sempre podeu instalar Potato i després actualitzar a Woody com he fet jo.
- En qualsevol cas, sempre necessitareu actualitzar a Woody perque la versió de Lilo de la Potato no pot manejar el dispositiu RAID (amb GRUB no he provat).

Bé, anam a començar:

Preparació:

En primer lloc necessitarem un disquet d'assistència amb tres fitxers que tot seguit vos explic quins son i com obtenir-los:

- El mòdul que ens permetrà accedir al nostre array durant la instalació : Es pot trobar a la web de Promise[5], però normalment ens vendrà al CD de la controladora. En el meu cas, està integrada a la placa mare (Supermicro P3TDDE) i ve al directori PROMISE del CD tant en versió monoprocessador com SMP (que per una instalació tant ens dona). S'ha de colocar al directori 'boot' del disket perque el sistema d'instalació de Debian l'agafi.
- Un kernel compilat des d'una altra màquina amb les següents opcións habilitades:
  - CONFIG_BLK_DEV_HPT366 HPT366 chipset support.
  - CONFIG_BLK_DEV-PDC202X PROMISE PDC202 {46|62|65|67|68} support
  - CONFIG_PDC202XX_FORCE Special FastTrack Feature
  - CONFIG_BLK_DEV_ATARAID Support for IDE Raid Controllers
  - CONFIG_BLK_DEV_ATARAID_PDC Support Promise software RAID (Fasttrak(tm))
  - CONFIG_BLK_DEV_ATARAID_HPT Highpoint 370 software RAID
- Un fitxer amb permís d'execució que anomenarem 'makedev' (preferiblement en minúscules per distingir-lo del que es pot trobar a http://people.redhat.com/arjanv/pdcraid/ataraidhowto.html[4] amb el següent contingut:

/floppy/makedev

```
#!/bin/sh
mkdir -p /target/dev/ataraid



cd /target/dev/ataraid
mknod d0 block 114 0
mknod d0p1 block 114 1
mknod d0p2 block 114 2
mknod d0p3 block 114 3
mknod d0p4 block 114 4
mknod d0p5 block 114 5
mknod d0p6 block 114 6
mknod d0p7 block 114 7
mknod d0p8 block 114 8
mknod d0p9 block 114 9
mknod d0p10 block 114 10
mknod d0p11 block 114 11
mknod d0p12 block 114 12
mknod d0p13 block 114 13
mknod d0p14 block 114 14
mknod d0p15 block 114 15
```

El sistema de fitxers tant pot ser FAT com ext2 perque el kernel del sistema d'instalació de Debian els reconeix tots dos.

Instalació:

FASE A: Instalació del sistema base

Per instalar el sistema base de la nostra Debian procedirem de la següent manera:

1. En primer lloc hem de configurar el nostre array a la BIOS de la controladora. Jo faig servir un raid1 (mirror) però això val per qualsevol tipus d'array que volguem definir. En el meu cas s'entra a la BIOS amb la combinació de tecles <CTRL>+'F'.

    - Si tenim un sol disc dur o volguessim fer servir discs independents, s'han de definir com a arrays d'un sol disc. Si no el mòdul del fabricant no detectarà cap disc a la controladora.
2. Després arrancarem normalment el nostre sistema d'instalació de Debian.
3. El primer que ens demana és la configuració de teclat. és un bon moment per indicar-li quin teclat tenim.
4. Després, si no tenim altres discs durs connectats a una controladora normal (soportada pel kernel), l'instalador ens suggerirà la opció "preload essential modules". Si el sistema ha detectat discs durs on es pugui fer la instalació de Debian, haurem de desplaçar el cursor fins a l'opció "Preload of modules" i seleccionar-la.
5. Picarem "intro" un parell de cops fins que ens carregui el mòdul (al CD de la P3TDDE anomenat 'fte.o') i ja tenim el nostre array visible com a primer disc scsi (/dev/sda).
6. Continuarem normalment amb la instalació del sistema (particionat, creació dels sistemes de fitxers, instalació del sistema base, etc...). També farem la instalació del kernel de la potato ("Install kernel and modules") encara que realment no el farem servir ja que així ens crearà un fitxer de configuració de lilo sobre el que podrem treballar més còmodament.
7. Finalment, la instalació ens durà a la opció "Make boot floppy". Podeu fer-lo si vos fa ilusió, però és una mica inútil, bàsicament perque _no_ arrancarà :-O. Anirem a la opció "Make Linux bootable directly from hard disk" (que tampoc funcionaria, però ens deixarà el lilo.conf a punt per modificar-lo al nostre gust).

FASE B: Fer arrancable el sistema

Ara només ens queda fer que el nostre sistema arranqui. Per això ens anirem a la consola que la instalació de Debian ens habilita a la tty2 pitjant <ALT>+F2, i <INTRO> per obtenir el prompt del sistema. Abans de començar però, una sèrie de reflexións per aclarir conceptes:

- En aquest moment tenim un sistema Linux (sistema d'instalació) correguent sobre un sistema de fitxers arrel ubicat en una ramdisk.
- El nostre disc dur (array) és visible gràcies a un mòdul que fa emulació SCSI i per tant el veim com /dev/sda.
- El pròxim cop que arranquem ja **NO** estarem fent emulació scsi i, per tant, ja no existirà /dev/sda.
- En comptes d'això podrem veure els discs durs INDIVIDUALMENT com a /dev/hdX: Típicament /dev/hde, /dev/hdf, /dev/hdg i /dev/hdh (masters i esclaus, dels dos busos de la controladora RAID, respectivament) a no ser que no tenguem compilat el soport per les controladores IDE normals.
- En principi **MAI** hem d'accedir dirèctament a aquests dispositius encara que el sistema ens ho permetrà. Especialment si formen un array conjuntament amb altres discs.
- En comptes d'això accedirem sempre als dispositius d'array que crearem al directori /dev/ataraid amb l'script makedev del nostre disquet.
- En aquest moment (encara estam en el sistema d'instalació) tenim montat tot el nostre arbre de directoris definitiu al directori /target, encara que el sistema de instalació està adaptat de manera que, per exemple, en editar el fitxer /etc/lilo.conf (/target/etc/lilo.conf, en realitat) podem (hem de) posar les rutes de fitxers i dispositius com si realment estiguessin ja a partir de /.

Ara ja estàm preparats per dur a terme els canvis que faràn que el nostre sistema pugui arrancar des del disc dur i no ens doni el "temible" *kernel panic*.

1. En primer lloc montarem el disket que ens hem fet a /floppy:
   #> mount /dev/fd0 /floppy
2. Seguidament ens situarem al directori /boot del nostre nou sistema:
   #> cd /target/boot
3. Copiarem el kernel que hem preparat al disket:
   #> cp /floppy/vmlinuz .
4. Crearem els dispositius per accedir al RAID amb el script que ens hem preparat:
   #> /floppy/makedev
5. Després editarem el fitxer /etc/lilo.conf (de moment /target/etc/lilo.conf). Per això farem servir l'editor ae que és l'únic del que disposam durant la instalació:
   #> ae /target/etc/lilo.conf
    - Canviarem la línia que diu:
      root=/dev/sdaX
      (on X és l'Id de la partició arrel) per:
      root=/dev/ataraid/d0pX
    - Cercarem la línia:

image=/vmlinuz
...i farem que apunti al kernel que hem copiat noltros:
image=/boot/vmlinuz

- **NO** tocarem la línia que diu "*boot=/dev/sda*" perque aquesta línia indica on s'ha d'escriure el nou MBR en instalar LILO i en aquest moment sí que estarem accedint a l'array amb el mòdul que fa emulació scsi.
- Salvarem els canvis amb <CTRL>+X <CTRL>+W i sortirem de l'editor amb <CTRL>+X <CTRL>+Q.

6. El pas següent és retocar el fitxer /etc/fstab (/target/etc/fstab) que ens ha creat l'instalador ja que els dispositius que ara son scsi en tornar a arrancar ja no ho seràn. Així substituïrem totes les ocurrències que trobem de "/dev/sdaX" per " /dev/ataraid/d0pX".
7. Salvarem els canvis i instalarem lilo:
   #> lilo
8. Rearrancarem el sistema:
   #> reboot

NOTA: Recordau que en arrancar el sistema amb el nou kernel ja no tendreu l'array a /dev/sda sino a /dev/ataraid/d0. Per tant, haureu de canviar la línia boot= de /etc/lilo.conf perque apunti a /dev/ataraid/d0.

APèNDIX: Ooops! Lilo ja no funciona!!!!

```
#> lilo
Fatal: Sorry, don't know how to handle device 0x7201
```

...en altres paraules: el nostre estimat lilo no reconeix el dispositiu /dev/ataraid/d0 i no sap com s'ho haurà de fer quan la BIOS el cridi per carregar el S.O.

La solució passa per actualitzar Lilo. Si hem de passar-nos a Woody el millor que podem fer és fer-ho ara. Un un cop actualitzat el sistema continuarem tenint un lilo que no pot arrancar desde el RAID, però bastarà fer un:
#> apt-get install lilo
...i veurem com ens elimina el paquet "debconf.tiny" per substituïr-lo per "debconf".

No he provat a fer una apt-get install debconf des de la Potato (si algú ho fa li agraïria comentàs el resultat); però sí un apt-get install lilo i no fa res.

- COMENTARIS[6] -

Disclaimer

This is a translation because of a petition of a friend who don't understands Catalan. Please, excuseme for the errors, but that is that I can do and feel free if you want to correct this and send me the result to publish here.
Thanks.

Obviously the hardware RAID is always the best possible solution because it don't mean any cost for our operating system and, also, can really paral·lelize read/write operations on our different disks. But it solution is more expensive than software one.

But as I had commented before, this cost growing can be more supportable if we use an IDE RAID controller whith IDE disks. Consecutively I will explain you how I did to take working whith the PROMISE PDC20267 of the new BULMA server[2]. But the instructions given here also sould be extensibles to the rest of PDC202XX series controllers supported by the Linux Kernel (46, 62, 65 and 68 on the 2.4.16 kernel which was the first kernel which I did use for this).

But at first time, and for who haven't so clear what is "RAID", here follows an brief explanation about in what consists it. If you know what is RAID yet and you don't need to read this, you can go directly to the next page to begin your Debian instalation.

RAID and RAID types:

As I explained above, a RAID controller (I will refer to hardware-raid for all the rest of this document) let us to combine many hard disks to operate as only one. For this reason, the RAID controller will have their own BIOS which we could cofigure different disk arrays which will be that we really could see from our operating system.

Well... it is not exactly this way. With an WindowsXX _it's true_ that we only will see the virtual disk which results from our array. But from Linux, we also have access to the separate disks. Obviously, that will occour if you try to access to one independent hard disk is only your responsability.

RAID array types:

I'm sure that you are now thinking about how are combined the different phisical disks of one RAID array and I suppose that you have yet some possible ideas and you are asking yourself what is the good one. Well: Probably the correct answer to this question is all of they.

There is different standarized array types which all offers differents advantages depending on which manner they combine the hard disks. Next, I will briefly explain two of the differents RAID "levels" (or types) which are the most commonly used. But before to begin, I want to comment that one array can be composed by n hard disks or also, by a number of other RAID arrays. For example, we can define two 0-type RAID arrays composed by two hard disks each one and then combine these two arrays in one 1-level array of the two firtst arrays.

The most common RAID types are '0' and '1'. Each RAID controller can implement a determined group of RAID types. The meaning of the two types mentioned above are explained below:

We suppose that hardware-RAID controller will support different capacity disks to combine in one array. But it can be not always true, depending on the contoller capacities and the array level which we want to use. In all cases, you should refer to the hardware documentation to know the exact capabilities of your controller. Also, when there is some paral·lelism on the disks, is more recomendable to use identical disks.

- **RAID 0 (disk stripping):** This RAID level combines one or (normally) more hard disks to made one unique virtual disk.
  - ♦ The disk capacity is n * c (where n is the number of implied disks and c is the smaller disk capacity).
  - ♦ The information is distributed equitablely thought all disks.
  - ♦ The performance is multiplied by the number of disks for reading and for writing because we read and write n times the information that we could read/write whith only one disk.
  - ♦ The information stored on one disk (or in a subconjunct) is totally inservible whithout the rest because the information of each logical block is reparted in one phisical block on each hard disk.
- **RAID 1 (disk mirroring):** Makes a exact copy (mirror) between various (two or more) physical disks.
  - ♦ The total capacity of the array is the capacity of the smaller disk.
  - ♦ The information contained in all disks is the same on each one.
  - ♦ The readings can be done (depending on the controller implementation):
    - ◊ In only one of the disks (in case of malfunction it will go to other disk on discarding the first one).
    - ◊ One part on each disk simultaneously multiplying the reading performance.
- The writings are made simultaneously on all disks and then, there is no performance reduction (in oposite than what success on software mirroring).

If you need more information about RAID types, i found this[3] link on Google.

The installation:

At first, some aclarations:

- I got working my RAID thanks to a small howto which you can find at http://people.redhat.com/arjanv/pdcraid/ataraidhowto.html[4]. ...but I recommend you to continue reading this document because there is SO MANY datails that there are not so explained.

- As the BIOS controller software, as the kerenel module provided by manufacturer as the Windozes drivers (which really don't interest us) use SCSI emulation to access to the controller. The native support which comes whith the lasts official kernels **DON'T** do this because it suppose an innecessary overhead in the driver implementation. This will cause many small inconvenients which we will see later. Althought you should to take in mind that to boot from RAID you should to say BIOS to boot from scsi, NOT from IDE.
- For the installation, I used the Debian 2.2 (Potato). The Debian 3.0 is just going to be released by Debian and I hope that all changes will be to better. But on the other hand, it is so possible that the process defined here will not be aplicable exactly "as is" because is so logical that the 3.0 release will come out whith a 2.4.x series kernel.
    - More concretely, the kernel module which provides the manufacturer (Promise Technology) is for 2.2.x Linux kernels and it don't work on 2.4.x kenrels. For this reason, if the future Debian 3.0 installation disk don't provide the necessary modules to support the RAID controller, you should compile it on other machine and load them by hand (and the kernel drivers are 6 different modules, not only one).
    - If you don't get right this way, you always can install a Potato base system, and then, upgrade to Woody as I did.
- In all cases, you will always need to upgrade to Woody because the Lilo version of Potatos distribution can not handle the IDE RAID device correctly to boot properly (I don't tryed with GRUB).

Well. We will begin:

Preparation:

The first thing that we will need is a usefull floppy whith the three files that I explain how to obtain below:

- The module which will let us to access to our array during the installation process: It can found on the PROMISE webpage[5], but normally it will come to us on the controller (or motherboard if integrated as in my case with Supermicro P3TDDE) CDROM. This module (normally called 'fte.o') must go on a directory named 'boot' on our assistance floppy.
- A kernel compiled on other machine with the next options enabled:
    - CONFIG_BLK_DEV_HPT366 HPT366 chipset support.
    - CONFIG_BLK_DEV-PDC202X PROMISE PDC202 {46|62|65|67|68} support
    - CONFIG_PDC202XX_FORCE Special FastTrack Feature
    - CONFIG_BLK_DEV_ATARAID Support for IDE Raid Controllers
    - CONFIG_BLK_DEV_ATARAID_PDC Support Promise software RAID (Fasttrak(tm))
    - CONFIG_BLK_DEV_ATARAID_HPT Highpoint 370 software RAID
- One file (whith execution privilege), named 'makedev' (preferently in lower case to distinct it from which you can find in http://people.redhat.com/arjanv/pdcraid/ataraidhowto.html[4] whith the next contents:

/floppy/makedev

```
#!/bin/sh
mkdir -p /target/dev/ataraid
cd /target/dev/ataraid
mknod d0 block 114 0
mknod d0p1 block 114 1
mknod d0p2 block 114 2
mknod d0p3 block 114 3
mknod d0p4 block 114 4
mknod d0p5 block 114 5
mknod d0p6 block 114 6
mknod d0p7 block 114 7
mknod d0p8 block 114 8
mknod d0p9 block 114 9
mknod d0p10 block 114 10
mknod d0p11 block 114 11
mknod d0p12 block 114 12
mknod d0p13 block 114 13
mknod d0p14 block 114 14
mknod d0p15 block 114 15
```

The floppy filesystem can be FAT or ext2 because I tested than the Debian installation system kernel supports they.

Installation:

FASE A: Base system installation

To install Debian base system, we will proceed this way:

1. At first, we must configure our array on the BIOS controller. I use a RAID1 (mirror) but this is usefull for any array type which we could want to define. In my case we can enter BIOS by pressing +'F' keys.
    - ♦ If we have only one hard disk or we want to use the disks independently one to the other, you can define only oune disk arrays. If you don't, the manufacturer module will not detect any disk attached to this controller.
2. Next, we will boot normally our Debian installation system.
3. The first thing that it ask us for is to configure keyboard distribution. This is a good moment to do this.
4. Then, if we haven't other hard disks attached to a normal contoller (natively supported by the kernel), the installer will suggest us to "preload essential modules". If the system has detected any hard disk on which the installation could be done, we will must to move the cursor to the "Preload of modules" option and select it.
5. Now we will press 'enter' repeatly until the only module in the floppy will be loaded, and then we get our disk array visible as scsi (emulated) device (/dev/sda if there is no other scsi disks on the system).
6. Continue whith the normal system instalation (partitioning, creating filesystems, base system installation, etc...). Also, we make the potatos kernel instalation althought we won't use it because this way we grant that a basic lilo.conf file will be created and we will only modify it to get right.
7. Finally, the installation will bring us to the "Make bot floppy". We really can make it if we like. But it will not boot :-O. For this reason we will go directly to the "Make Linux bootable directly from hard disk" (which didn't also work, but now we will be sure that we have a "correct" lilo.conf file as a base of our final configuration).

FASE B: Make system bootable

Now, it only left us to take our system bootable. Do do it, we go to the console which Debian installation system enabled to us on tty2 by pressing <ALT>+F2 and <INTRO> to obtain the system prompt. But before to staring, take knowledge about many reflections to clarify some concepts:

- In this moment, we hav a Linux System (installation system) running on a filesystem located on a ramdisk.
- Our hard disk (array) is visible because of a kernel module (provided by manufacturer) which let us access to hard disk by scsi emulation and then, we see it as scsi device (normally /dev/sda).
- Next time we boot, we will **NOT** make scsi emulation and then it will not exist /dev/sda. because we will be using Linux native (IDE) support to access the array.
- We will see the individual hard disks on /dev/hdx (tipically /dev/hde, /dev/hdf, /dev/hdg and /dev/hdh).
- We must **NEVER** access directly to these devices althougt the system will let us. Specially if they conform an array with another.
- In place of this, we will alwaysaccess to the array devices by the special files which we will create under the /dev/ataraid directory with our 'makedev' script on our floppy.
- At this moment (even we are on the installation system) we have all our system directory tree mounted at /target directory, althought the installation system had adapted to, for example, edit the /etc/lilo.conf (/target/etc/lilo.conf, really) with the filename paths pointing from / instead of /target..

Now, we are yet prepared to do the changes that will our system botable directly from our RAID array without giving us the "temible" *kernel panic* message followning next instructions.

1. Mount our assistence disk on /floppy:
   #> mount /dev/fd0 /floppy
2. Go to /boot directory of our new system on /target:
   #> cd /target/boot
3. Copy here the kernel which we preppared above.
   #> cp /floppy/vmlinuz .
4. Create the ataraid devices with the 'makedev' script: #> /floppy/makedev

5. Edit the /etc/lilo.conf (/target/etc/lilo.conf for now) using the 'ae' editor which comes with Debian installation system: #> ae /target/etc/lilo.conf
   - Change the line which says... *root=/dev/sdaX* (where X is the root partition ID) by: *root=/dev/ataraid/d0pX*
   - Change: *image=/vmlinuz* ...and make it pointing to our coppied kernel: *image=/boot/vmlinuz*
   - **DON'T** touch the 'boot=/dev/sda" line because this line says where to write your new MBR record wen installing LILO and we will install LILO after booting whith him wen we are accessing array by scsi emulation whith controller manufacturer module.
   - Save changes by pressing <CTRL>+X <CTRL>+W and exit with <CTRL>+X <CTRL>+Q.
6. The next step is to properly modify /etc/fstab (/target/etc/fstab) properly because devices which now are accessed via /dev/sdaX will become /dev/ataraid/d0pX.
7. Save the changes and install lilo:
   #> lilo
8. Reboot the system:
   #> reboot

NOTE: Remember that when boot with new kernel, you have your array attached at /dev/ataraid/d0, not /dev/sda; and now, you yet should modify "boot=" entry on /etc/lilo.conf to point to /dev/ataraid/d0 to lilo installs properly next time.

APPENDIX: Ooops! I can't install Lilo now!!!!

```
#> lilo
Fatal: Sorry, don't know how to handle device 0x7201
```

...in other words: our loved lilo don't recognize /dev/ataraid/d0 and don't know how to handle it to load Operating System from it on boot time.

The solution is to upgrade Lilo. If we plan to upgrade to Woody, this is the best moment to do it. When you have updated the system, you continue having lilo version which cannot be installed to boot from RAID, but it will be enought to do:

```
#> apt-get install lilo
```

...to see the system removing 'debconf.tiny' package and installing 'debconf' (which I suppose contains most updated and complete version of lilo).

I did'nt try to do "apt-get install debconf" from potato (if anyone do, please report me the results) but yes an "apt-get install lilo" and it had no sense.

- COMMENTS[6] -

Pàgina de comentaris.

Comments page.

Lista de enlaces de este artículo:

1. http://bulma.net/body.phtml?nIdNoticia=1067&nIdPage=3
2. http://bulma.net/body.phtml?nIdNoticia=1134
3. http://whatis.techtarget.com/definition/0,289893,sid9_gci214332,00.html
4. http://people.redhat.com/arjanv/pdcraid/ataraidhowto.html
5. http://www.promise.com
6. http://bulma.net/body.phtml?nIdNoticia=1067&nIdPage=5

E-mail del autor: joanmi _ARROBA_ bulma.net
Podrás encontrar este artículo e información adicional en: http://bulma.net/body.phtml?nIdNoticia=1067