Linux ATA RAID HOWTO

Murty Rompalli

murty@solar.m u r t y.n e t

April 26, 2002

Revision History

Revision 2.0	2002−05−10	Revised by: mr
Major enhancements		
Revision 1.3	2002−05−07	Revised by: jyg
format fixes		
Revision 1.2	2002−04−30	Revised by: mr
Minor fixes		
Revision 1.1	2002−04−28	Revised by: ldl
Some minor changes and sgml−improvements		
Revision 1.0	2002−04−26	Revised by: mr
Initial Release		

RAID is not limited to expensive SCSI disks anymore as more and more motherboard manufacturers are introducing motherboards with onboard RAID support for inexpensive IDE disks, known as ATA RAID. Promise Technolgy and HighPoint are two companies that dominate this ATA RAID market. This HOWTO document explains how to install Linux on an Intel Pentium compatible computer with an ATA RAID Controller (onboard chip or seperate card), single or multiple processors and atleast two hard disks. Currently, this document covers installing RedHat Linux 7.2 with Promise FastTrack ATA RAID Controller only.

Table of Contents

1. Introduction

The goal of this HOWTO document is to explain how to setup RAID 1 (mirroring) with the two hard disks and install bootable RedHat Linux Operating System on the mirror device. This document discusses both methods of achieving this goal: 1. Using Promise supplied driver 2. Using Linux native RAID. If you choose the first method (using Promise supplied driver), you must use the Kernel 2.4.7−10 that comes with RedHat 7.2 CD. If you choose the second method, you can upgrade the kernel to the latest kernel (2.4.18). This document does not discuss Striping or other disk configurations, although the author believes that this document might help setup those configurations. This document does not yet cover: 1. Installation using GRUB instead of LILO and 2. Installation with Latest development kernels (2.5.x).

First of all we need a bit of legalese. Recent development shows it is quite important.

1.1. Copyright Information

1.2. Disclaimer

No liability for the contents of this documents can be accepted. Use the concepts, examples and other content at your own risk. As this is a new edition of this document, there may be errors and inaccuracies, that may of course be damaging to your system. Proceed with caution, and although this is highly unlikely, the author does not take any responsibility for that.

All copyrights are held by their by their respective owners, unless specifically noted otherwise. Use of a term in this document should not be regarded as affecting the validity of any trademark or service mark.

Naming of particular products or brands should not be seen as endorsements.

You are strongly recommended to take a backup of your system before major installation and backups at regular intervals.

1.3. New Versions

The newest version of this HOWTO will always be made available on my website http://www.murty.net/ataraid/, or you may contact me directly at `<murty at solar . murty . net>` to check if there is a newer version.

Currently, this document is available in the following formats:

- HTML.
- plain text.
- Adobe PDF.
- gzipped postscript (US letter format).
- SGML source.
- gzipped DVI file.
- gzipped TeX (to be used with jadetex).

Note that paper sizes vary in the world, A4 and US letter differ significantly. You might also wish to consider using the *universal format* (8.27x11in; 210x279mm).

1.4. Credits

Your name here, if you contribute :)

Luc de Louw <luc at delouw.ch> corrected errors in my SGML source.

Joy Y Goodreau <joyg at us.ibm.com> corrected errors in my SGML source.

In this version I have the pleasure of acknowledging:

Alain Portal <alain.portal at free.fr>, Service Commun de Microscopie Electronique for proof–reading and correcting mistakes.

1.5. Feedback

Feedback is most certainly welcome for this document. Without your submissions and input, this document wouldn't exist. Please send your additions, comments and criticisms to the following email address : `<murty at solar. murty . net>`.

1.6. Translations

Please help International users who do not speak English. You are encouraged to translate this document to a foreign language. Please notify the author if you translated or would like to translate this document into a foreign language. The following translation efforts are already under way.

- French Translation is coming soon. Thanks go to Alain Portal.

2. Requirements

This section lists what items are required before you start installing on your computer.

- Two blank floppies, DOS formatted. On a working linux computer, you can type **mkfs.msdos /dev/fd0**
- Red Hat Linux 7.2 CDs 1 and 2 (3 and 4 contain SRPMS and are not required)
- Internet connection available
- Your computer with working CD, floppy,10/100 ethernet card support
- Patience

3. Prepare Promise Driver Floppy

1. Download appropriate driver from one of the two sites below and save the driver on one of the blank floppies. You can do this on any computer connected to Internet.

    - ♦ Driver for Single Processor Machine
    - ♦ Driver for Multiple Processor Machine
2. Insert the floppy with the driver tar ball into a working Linux computer and type:

```
mount /dev/fd0 /mnt/floppy
cd /mnt/floppy
cp rhsmp-ft12014.tgz /tmp
tar xvzf /tmp/rhsmp-ft12014.tgz
cd /
umount /mnt/floppy
```

3. Label the floppy »Promise FastTrack driver« and set it aside.

---

4. Preparing RedHat 7.2 CDs

NOTE

If you already have Red Hat Linux 7.2 CDs 1 and 2, skip this step. Otherwise, read this step to learn how you can get them for free.

1. Logon to a Windows computer that has a CD Writer drive installed and setup properly. Insert a blank CD−R into the CD Writer.
2. Point your browser to the Red Hat Web site: ftp://ftp.redhat.com/pub/redhat/linux/7.2/en/iso/i386/.
3. Browse and locate `enigma-disc1.iso` (the first disk of Red Hat 7.2), and save this file to your Windows desktop.
4. Right−click on the iso image created on your desktop and choose "Record to CD" It will then write the iso image onto your CD−R and create disk−1.
5. Repeat the procedure for the second iso file named `enigma-disc2.iso` on ftp.redhat.com/pub.
6. Test to make sure your CD−Rs are indeed readable. If you click on »My Computer« and click on the CD Writer Drive icon, you should be able to browse the contents of the CD−R.
7. Label the CD−Rs properly: RH 7.2 disk−1 and RH 7.2 disk−2

5. Installing Red Hat 7.2

Once you have created your CDs, you are ready to begin installing Red Hat 7.2 on your Linux system.

1. Restart the computer and press **Cntrl−F** when you see FastTrack BIOS prompt on screen. This will take you into the Promise FastTrack BIOS.
2. Inside this BIOS, choose "delete array," "define array," and "choose Mirror."
3. Press **Cntrl−Y** to save.
4. Choose »Create Only« and **ESC** to reboot. When it reboots, you should see that Promise FastTrack now has 1x2 RAID Mirror defined over your two hard disks connected to FastTrack controller.
5. Insert your Red Hat 7.2, disk−1 into your CD−ROM and reboot.
6. At the boot−prompt, type:

   ```
   linux noprobe
   ```
7. Now, installation will begin. Choose »Add device« and scroll through the list of available drivers to see if Promise FastTrack is listed. If it is not listed, press **F2** to load external driver from a floppy. Insert your Promise FastTrack driver floppy and hit "OK."
8. Continue with installation. Choose only ext3 type partitions for now. You can make ext2 partions at the very end, if you really need them.
9. When the system you to create Boot Floppy, insert a blank floppy disk. The Red Hat installation program will create a Boot Floppy disk.

   NOTE

   Please do not click on »Skip boot disk creation«. If you skip this step and do not create a Boot Floppy disk, you will be very sorry later.
10. When the install asks you to choose Boot loader configuration, please choose »LILO only«. Do not choose »GRUB« as your boot loader.

    When the installation is finished, you will see »Congratulations« screen.
11. Press **Cntrl−ALT−F2** to switch to `tty2`.
12. At the shell prompt, type:

    ```
    cd /mnt/sysimage/lib/modules/2.4.7−10/kernel/drivers/scsi
    ```
13. Type **ls** and make sure the file `ft.o` exists. If not, you are going to need to manually install the module `ft.o` as follows:

    a. Make sure your Driver floppy is mounted. Go into the floppy and type

    ```
    mv module.cgz /tmp
    cd /tmp
    gzip −dc module.cgz | cpio −idumv
    ```
    b. Now, you will see a bunch of directories created under `/tmp`.

    ```
    cd /tmp/`uname −r`
    cp ft.o /mnt/sysimage/lib/modules/`uname −r`/kernel/drivers/scsi
    ```
    c. At the shell prompt, type:

```
less /mnt/sysimage/etc/lilo.conf
```

Check to see that the `lilo.conf` looks good. Especially, it should have the `initrd=` line and the corresponding `initrd.img` file must exist in `/mnt/sysimage/boot`. If not, you are going to create it manually as follows:

```
/mnt/sysimage/usr/sbin/chroot /mnt/sysimage /sbin/mkinitrd \
--preload jbd \
--preload ext3 \
--preload scsi_mod \
--with ft \
/boot/initrd.img 2.4.7-10
```

14. Now, remove the floppy and the CD, and reboot. At this point, pray that your computer will boot without any problems.
15. If it does not boot, insert your Boot Floppy and reboot and login; And then repair as follows:

    a. Make sure `/boot/initrd.img` exists. Make sure `/lib/modules/2.4.7-10/kernel/drivers/scsi/ft.o` exists (If not you have to manually fix these issues as explained above)
    b. Type: **/sbin/lilo**
    c. Now, remove floppy and reboot. Your machine should boot into your new machine now using Promise Driver.
16. Type **df –k** and you should see your hard disks as `/dev/sdaX` instead of `/dev/hdaX`. This is because the Promise Driver is actually a special type of Software Emulation RAID, not exactly Hardware RAID. (Promise RAID works through a BIOS Hack).

If your machine is SMP, you will have to manually create `initrdsmp.img`, when you boot into Uniprocessor Kernel as shown below, and edit `/etc/lilo.conf` and then test to see you can boot into SMP system.

1. When you boot your machine into `2.4.7-10` uniprocessor kernel, type the following to make `initrdsmp.img` to be used for `2.4.7-10smp` kernel:

```
/sbin/mkinitrd \
--preload jbd \
--preload ext3 \
--preload scsi_mod \
--with ft \
/boot/initrdsmp.img 2.4.7-10smp
```

NOTE

If you are tired of remembering the command to create initrd files, dowload my geninitrd script and keep it handy.

2. Adjust your `/etc/lilo.conf` accordingly for `2.4.7-10smp` (the SMP kernel section), type **/sbin/lilo** and reboot into the SMP kernel. Here is how `/etc/lilo.conf` should look like.

At this point, you have a working Red Hat 7.2 machine with SMP support, if applicable. If you are happy with the fact that you are running a fake Hardware RAID from Promise FastTrack using SCSI Emulation, then read no further.

Because Promise Driver is a SCSI emulation, it puts a lot of load on CPU(s). Read on if you want to enable true Linux native RAID and get rid of this Promise SCSI emulation.

6. Installing Native Linux RAID

The first step you want to take is to configure your networking and connect your computer to Internet, which is still running under Promise Driver SCSI emulation. Next you will install native Linux RAID.

1. Go to www.kernel.org and download latest kernel `2.4.18.tar.gz`.
2. Configure your kernel:

```
cd /usr/src/
tar xvzf linux-2.4.18.tar.gz
cd linux                 # cd to kernel source directory just created by tar
cp config.txt .config    #(See NOTE below to find out where to get config.txt)
make menuconfig
```

NOTE

You can download a working `config.txt` file HERE. You can, of course, modify this to suit your needs either directly in a text editor or by typing **make menuconfig** as explained above. It is easy to make mistakes if you are editing `.config` directly in vi or emacs. Therefore, it is recommended to use Menu Interface by typing **make menuconfig**.

3. Enable all the following in the kernel statically (NOT as modules):

```
  ATA/IDE/MFM/RLL Support -->
        <*> ATA/IDE/MFM/RLL Support
        IDE/ATA/ATAPI Block Devices -->
                <*> Enhanced ATA/IDE/MFM/RLL disk/cdrom/tape/floppy support
                <*> Include IDE/ATA-2 Disk Support
                [*] Use multi-mode by default
                Include IDE/ATAPI CDROM support
                [*] Generic PCI IDE chipset support
                        [*] Sharing PCI IDE interrupt support
                        [*] Generic PCI Bus master DMA support
                        [*] Use PCI DMA by default when available
                        [*] Intel PIIXn chipset support
                                [*] PIIXn tuning support
                [*] Promise PDC202{46|62|65|67|68} support
                        [*] Special UDMA Feature
                        [*] Special FastTrack Feature
                [*] VIA 82CXXX chipset support
                <*> Support for IDE RAID controllers
                        <*> Support Promise Software RAID (fasttrack)
```

4. After you have enabled static features as shown above, make any more changes that you would like to make to suit your environment.
5. Save the kernel configuration. It is saved to `.config` in the current directory. Please back up this file. If you dont do so, you will regret it later. Copy `.config` file in a blank floppy or in `/root`
6. Build and install the kernel like you normally do, by typing:

```
make dep ; make clean ; make && make install
make modules && make modules_install
```

Look at /etc/lilo.conf to make sure new lines are added to boot your new kernel 2.4.18. Note that there should not be an initrd= line for this new kernel in /etc/lilo.conf. That is, our new kernel will boot itself without depending on a initrd.img unlike your current 2.4.7−10 kernel.

Now adjust /etc/lilo.conf as follows:

1. Locate the root=/dev/sdaX line for the new kernel in /etc/lilo.conf. Change this to root=/dev/ataraid/d0pX where X is a number 1 through 16. Save your changes.
2. Type:

```
/sbin/lilo
```
3. If you chose any ext2 partitions during installation, you should comment them out in /etc/fstab for now.(Best thing is: Forget ext2 in this whole process)
4. Remove all floppies and CDs, and reboot by typing:

```
sync;sync;reboot
```
5. At the LILO prompt, type your new kernel lable corresponding to 2.4.18 Your computer should then boot into your new kernel.
6. Login and type **df −k** to make sure you see /dev/ataraid/d0X entries instead of earlier /dev/sdaX entries.

NOTE

You may see some errors related to mounting swap device at the time of booting into new kernel. These are harmless. You should edit /etc/fstab to change any sdaX entries to ataraid/d0pX entries.
7. Now, connect your computer to Internet and download lilo−22.tar.gz (the latest version of lilo program) from the Internet.
8. Remove the existing lilo on your computer by typing: **rpm −e −−nodeps lilo**
9. Install new version of lilo as follows:

```
tar xvzf lilo-22.tar.gz
cd lilo-22      # cd to lilo source directory just created by tar
./QuickInst.sh
```
10. Say "Yes" to questions the system will ask you.
11. Ignore any errors except if /sbin/lilo is not created.
12. Adjust /etc/lilo.conf as follows:
    - Replace linear by lba32
    - Delete line "compact"
    - Change vga= line to vga=normal
    - Change boot=/dev/sda line to boot=/dev/ataraid/dN (where N is the partition number on which your root file system exists. Type **df −k /** to find out your root partition number).
    - Make sure that default=linux−2.4.18 (where linux−2.4.18 is the label given to boot your new kernel: 2.4.18)
13. Save changes to the file and type: **/sbin/lilo**
14. Reboot the system and cross your fingers.

Here is how the final `/etc/lilo.conf` should look like.

If everything comes up without any errors, time to celebrate!!

---

7. Installing on an existing Linux system

This section explains how to install Linux Native ATA RAID on non−OS disks you may have on a running Linux machine. Non−OS disks are those which do not contain any Linux OS partitions such as `/`, `/usr`, `/var`, `/boot`. In other words, we have a working Linux machine with two free disks and we want to setup ATA RAID mirror (raid 1) on those two disks. When we save important data on such a mirror device, our data is protected. Ofcourse, like any other RAID 1, we will experience improved read speeds while reading the data and slightly decreased write speeds while modifying or adding new data to the mirror device. Therefore, using RAID 1 for data disks is recommended if the data disk is more frequently read than written to. A web server hosting rarely changing web sites is a good example where web content is rarely modified but very frequently accessed by users.

Here are the steps to install non−OS ATA RAID if your RAID chip is from Promise Technology:

- Find out IO address numbers and IRQ number(s) for your Promise RAID chip/card.
- Edit `/etc/lilo.conf` and insert appropriate *Append Line*
- Enable ataraid support either by automatically loading ataraid module when your Linux machine starts up or by statically building ataraid support into your kernel

---

7.1. Append Line

For the purpose of understanding various tasks involving Promise FastTrack RAID such as upgrading or troubleshooting, we introduce a noun: *Append Line*

All the `ide` options you pass at the LILO `boot:` prompt at the time of booting, when put together as a string, make up the *Append Line*. All the `ide` options in double quotes after `append=` keyword in `/etc/lilo.conf` also make up the *Append Line*.

For example, if you type

linux−new ide2=0x0001,0x0009,9 ide3=0x2000,0x2009,10 ide4=none nousb expert root=/dev/hda3

at `boot:` prompt at the time of booting your Linux computer, then *Append Line* is the string

ide2=0x0001,0x0009,9 ide3=0x2000,0x2009,10 ide4=none.

Similarly, if your `/etc/lilo.conf` has the following section, your *Append Line* is

ide2=0x9400,0x9002 ide3=0x8800,0x8402.

```
image=/boot/vmlinuz-2.4.9-10
        label=linuxold
        read-only
        root=/dev/hde9
        append="nousb ide2=0x9400,0x9002 ide3=0x8800,0x8402"
        initrd="initrd.img"
```

When we experience problems booting a Linux machine with RAID, we may have to use an appropriate *Append Line*. Therefore, it is important to determine and write down the *Append Line*. This will help you fix your problems later or upgrade your kernel smoothly or add/remove additional hard disks.

7.2. Determining the Append Line

To determine the correct *Append Line*, we should first know how all our `ide` devices are connected. IDE devices can be hard disks, ATAPI CDROM(s) *etc*. Once we determine the *Append Line*, we can append it to the `boot:` options (at the time of booting) or we can alternatively assign it as a string value to the append paramater in `/etc/lilo.conf`. Unless you love to remember complicated `boot:` options and type them at boot time every time, you should choose the second method, *i.e.*, insert it into `/etc/lilo.conf` . You can do so by inserting `append="`*Your Append Line Here*`"`, saving file and then activating new `/etc/lilo.conf` by running the command **/sbin/lilo**.

For the purpose of understanding better, lets say that your `ide` devices are as follows:

- `ide0`: `hda` , `hdb` (hard disks)
- `ide1`: `hdc` , `hdd` (hard disks or other ide devices like CDROM)
- `ide2`: `hde` (first free disk)
- `ide3`: `hdg` (second free disk)

The two free disks above (`hde` and `hdg`) are the ones we would like to setup as RAID 1 to create `/dev/ataraid/d0` raid device. Note that we do not have `hdf` or `hdh` because that is how we used the IDE/RAID ports on Promise chip. It is not a good idea to connect two hard disks to the same Promise controller IDE port. In the above example, we used Primary Master and Secondary Master connections on the Promise Technology card.

If you do not know how various `ide` devices are connected in your computer, take a look at `/proc/devices` and `/proc/ide/*`. You can also carefully go through boot log file, `/var/log/bootlog` (or type **dmesg | more** right after your Linux system boots) to find your `ide` devices. Now type **less /proc/pci** and locate appropriate information about Promise Technology. In the output of **less /proc/pci**, you can see somewhere information about your Promise chip, something like:

```
Bus  0, device  17, function  0:
   Unknown mass storage controller: Promise Technology Unknown device (rev 2).
     Vendor id=105a. Device id=d30.
     Medium devsel.  IRQ 10.  Master Capable.  Latency=32.
     I/O at 0x9400 [0x9401].
     I/O at 0x9000 [0x9001].
     I/O at 0x8800 [0x8801].
     I/O at 0x8400 [0x8401].
     I/O at 0x8000 [0x8001].
     Non-prefetchable 32 bit memory at 0xd5800000 [0xd5800000].
```

From this output, we learn that our Promise Technology card uses IRQ 10 for both `ide` ports (`ide2` and `ide3`). Using same IRQ is perfectly alright as long as your kernel supports PCI IRQ Sharing. By default, our Linux kernel is configured to support PCI IRQ sharing. From the above output, we also learn that our Promise Technology card uses various IO addresses. For the purpose of identifying Promise Technology disks properly at boot time, we only want the IRQ number(s) and the first four IO Address numbers outside [ ]. Write down on a piece of paper this information. In this case, from the above output:

```
IRQ1 = 10
IRQ2 = 10
```

```
IO1 = 0x9400
IO2 = 0x9000
IO3 = 0x8800
IO4 = 0x8400
```

Now, we have to evaluate the following to obtain the correct *Append Line*. Then either specify this *Append Line* at boot time or specify it in lilo configuration file.

```
ideX=IO1,IO2+0x0002,IRQ1 ideY=IO3,IO4+0x0002,IRQ2
where ideX and ideY are the two IDE ports of Promise card our free disks are using.
```

In our example, the above *Append Line* will become:

`ide2=0x9400,0x9002,10 ide3=0x8800,0x8402,10`

If, for example, we want to boot kernel version 2.4.18, labelled `linux` according to `/etc/lilo.conf`, then we specify our *Append Line* in one of the following two methods:

1. At boot time
   ```
   boot: linux ide2=0x9400,0x9002,10 ide3=0x8800,0x8402,10
   ```
   If you choose this method, you should manually type the *Append Line* after the kernel label `linux` everytime you boot your Linux machine.
2. In `/etc/lilo.conf`
   ```
   image=/boot/vmlinuz-2.2.18
           label=linux
           read-only
           root=/dev/hda1
           append="ide2=0x9400,0x9002,10 ide3=0x8800,0x8402,10"
   ```

If you choose this method, you have to run lilo once to activate changes by typing **/sbin/lilo**. And you do not have to type anything extra at boot time.

7.3. Setting Up RAID 1

If you want to setup RAID 1 using Promise Technology proprietary driver (`ft.o`), you can download Promise Driver (`ft.o`) into `/lib/modules/kernel-version` and load the module by typing **modprobe −k ft**. You should then be able to access your new raid device as `/dev/sdc` or something like that. But if it does not work, then determine your *Append Line* and add it to `/etc/lilo.conf`. If you are setting up RAID on an existing Linux system and are using either Promise Technology `ft` driver or Linux native `ataraid` driver, then use of *Append Line* in `/etc/lilo.conf` is strongly recommended. Once you reboot with your new `/etc/lilo.conf` that contains *Append Line*, you can load either driver (`ft.o` from Promise Technology or `ataraid.o`, the Linux Native RAID module) to enable RAID except when your kernel has built−in ataraid support in which case you do not have to load ataraid module.

As Linux Native RAID is recommended, let us discuss that in detail. To setup Linux Native RAID on an existing Linux machine, insert *Append Line* into `/etc/lilo.conf` as explained above. Now, activate changes by typing **/sbin/lilo**. Then reboot your computer. After your computers reboots, load ataraid module manually if your kernel does not have ataraid built−in support or `ataraid.o` module failed to load for some reason. If you compiled your kernel with static ataraid support (ataraid not as module), then you can start formatting and using your mirror disk `/dev/ataraid/d0` right away.

But if you compile ataraid as a seperate module, then type **lsmod** and see ataraid is listed. If not, manually load it by typing **modprobe −k ataraid**. If you dont see any errors, then you can start using your mirror disk `/dev/ataraid/d0` right away. Format it, mount it and use it just like you normally do.

The fact that you can use `/dev/ataraid/d0` implies that you are successful in your mission. Please do not access `/dev/hde`, `/dev/hdg` or any of their partitions directly, although Linux will let you do that. Once you make a mirror device from two disks, you should always access the mirror device and not the disks directly.

8. Upgrading Kernel

Read this section carefully before you plan to upgrade kernel on your Linux machine with Promise FastTrack RAID. Unless you are not using Promise FastTrack at all in any way, you need this information to avoid problems.

Always backup your data before kernel upgrade. Also backup `/etc/fstab`, `/etc/lilo.conf`, `/boot/vmlinuz-currentversion` and `/boot/initrd.img` (if you use initrd). When you upgrade your kernel, do not delete old kernel or its dependent files in `/boot` and do not delete the lines corresponding to it in `/etc/lilo.conf`. If you upgrade kernel to say 2.4.19, just create another kernel section in `/etc/lilo.conf`. For example, add the following lines to `/etc/lilo.conf`:

```
# Begin Code for booting my brand new kernel: 2.4.19

image=/boot/vmlinuz-2.4.19
        label=linux-new
        read-only
        root=/dev/ataraid/d0p12

# End Code for booting my brand new kernel: 2.4.19
```

Also, do not change `default=linux` line in `/etc/lilo.conf` unless and until you successfully can boot into your new kernel by typing **`linux-new`** at the `boot:` prompt (or highlighting `linux-new` in the menu, if you are using LILO in curses menu mode).

Now lets discuss the actual process of upgrading kernel in the following four cases:

1. Promise Technology (`ft`) Driver with OS on RAID mirror
2. Promise Technology (`ft`) Driver with non−OS data on RAID mirror
3. Linux Native (`ataraid`) Driver with OS on RAID mirror
4. Linux Native (`ataraid`) Driver with non−OS data on RAID mirror

OS stands for "Operating System" or more specifically Red Hat Linux Operating System in our case. When you install Linux on mirrored partitions such as `/dev/ataraid/d0p1` (or `/dev/sda1` when using Promise Technology proprietary driver), you are said to have your OS on RAID mirror. If your Linux machine has only the main partitions such as `/` and `/boot` on RAID, it is also called (a case of) OS on RAID.

Partitions created by user to make use of available free partitions such as `/mydata1`, `/imp`, `/scratch` are not part of Linux OS because Linux does not install any files into them by default when you install Linux or upgrade any standard Linux software package. Any data in such user partitions becomes user data or alternatively, non−OS data. It is a good idea to use only ext3 and swap file systems for OS partitions. For non−OS partitions, you can use other file systems such as ext2 and dos (if you have another operating system like Windows on the same Linux computer or if you just feel like you love dos so much). However, why do you want to use ext2 if you dont have to and you have a better choice of using ext3 for OS or non−OS partitions?

8.1. Promise Technology (`ft`) Driver with OS on RAID mirror

Currently, Promise Technology supports Red Hat 7.2 and earlier versions only. Red Hat 7.2 uses kernel

2.4.7−10 by default. Also note that, you have to use initrd.img (Initial RAM disk image) in `/etc/lilo.conf`, when you install Linux with Promise supplied driver. At the time of installation, Promise Driver scripts are supposed to automatically generate initrd file and configure `/etc/lilo.conf` for you. Unfortunately, this does not work properly and you may have to manually create initrd.img and configure `/etc/lilo.conf` yourself.

You are stuck with the default kernel 2.4.7−10 and you cannot and should not upgrade kernel either by compiling or by automatic update programs such as **up2date** or **rpm** utility. If you really really want to upgrade kernel then do so but do not use raid. You can add *Append Line* to the `/etc/lilo.conf` as explained in Section 7.2 and do not load ataraid module (or do not compile your new kernel with ataraid feature built−in). By doing this, you are upgrading kernel to new version and sacrificing RAID feature because you are using Promise Technology card as a simple IDE extension card.

If Promise Technology releases new version of their `ft` driver for 2.4.19 in future, you can then upgrade your kernel first to 2.4.19 (using **up2date** or **rpm** but not by manually compiling) and place their new version of `ft.o` file in `/lib/modules/kernel−2.4.19`. You will also have to put `initrd−2.4.19` in `/boot` and append `initrd=` line to the new kernel section in `/etc/lilo.conf`. If your Linux machine uses Promise Technology driver, your OS is on RAID mirror and Promise Technology did not release any new versions for new kernel versions, please do not fool around with your current kernel. If you do fool around, remember that you can seriously damage your computer and not be able to boot or retrieve your data.

8.2. Promise Technology (`ft`) Driver with non−OS data on RAID mirror

Almost all of explanation in previous case (Promise Technology (`ft`) Driver with OS on RAID mirror) applies in this also, ofcourse you are now risking non−OS data instead of OS. That means, if your upgrade fails, you will be able to boot but not be able to see data in your RAID partitions. Additionally, you have the flexibility of loading and unloading Promise Proprietary driver (`ft.o`). However, this is not guaranteed to work smoothly because Promise Technology driver has a lot of issues.

Unless Promise Technology releases appropriate drivers for new kernel version, please do not attempt kernel upgrade in any method. You are stuck with 2.4.7−10 kernel. You may be required to add *Append Line* as discussed in Section 7.2 to boot your computer in some cases.

8.3. Linux Native (ataraid) Driver with OS on RAID mirror

You can upgrade kernel to any version above 2.4.18 by recompiling the kernel and not by any automatic upgrade method such as **up2date** or **rpm** utility. You need .config file from your previous kernel source directory (`/usr/src/linux−2.4.18/.config`). You did save your 2.4.18 .config file, did't you?

After you copy `.config` into `/usr/src/linux−2.4.19` directory, you may make changes by typing **make menuconfig** but generally you do not need to make changes because you are upgrading only to 2.4.19 and your machine hardware did not change. But if you do make changes by typing **make menuconfig**, remember to save and also backup your modified .config file. Save the .config file safely in /root or on a floppy disk. Here are the steps:

* Unzip new kernel in `/usr/src` and rename top directory to `linux−2.4.19`. So your new kernel source directory is `/usr/src/linux−2.4.19`.
* **cp /root/config.txt .config**
* **make menuconfig** (Make any required changes)
* **cp .config /root/config−2.4.19.txt** (Back up config file in a safe place)
* **make dep ; make clean ; make && make install** (Install new kernel)
* **make modules && make modules_install** (Install new kernel modules)
* Edit `/etc/lilo.conf` to add new lines to boot new kernel. Label the new kernel `linux−new`. Do not change `default=` line and do not delete lines corrosponding to current working kernel.
* **/sbin/lilo** (Activate changes in `/etc/lilo.conf`)
* **/sbin/lilo −R linux−new** (Tell LILO to treat `linux−new` as default boot kernel just for one time only)
* **sync;sync;reboot** (Reboot and hope that it works. If it does not boot, power off and power on again. Then login and investigate why your new kernel did not boot.)

8.4. Linux Native (ataraid) Driver with non−OS data on RAID mirror

To upgrade kernel in this case, follow same procedure as explained in previous section (Linux Native (ataraid) Driver with OS on RAID mirror). If you experience problems, you may be required to add an additional line, *Append Line*, to your `/etc/lilo.conf`. To find out what your *Append Line* is, see Section 7.2.

Similar to previous case, do not attempt to upgrade kernel by any automatic method (**up2date** or **rpm** utility). Manually compile and your install your new kernel as explained in the previous section.

9. Disabling RAID feature on Promise FastTrack

If you are reading this section, then you may be sick and tired of the fake RAID feature (formally known as quasi−hardware RAID) provided by Promise Technology FastTrack card or onboard chip on your motherboard.

9.1. Case 1: OS not using RAID

It is possible to disable the RAID feature and use the Promise FastTrack as a plain vanilla IDE card. It is very simple. Follow the below steps:

1. Determine your *Append Line*. See Section 7.2 for help with this
2. Modify `/etc/lilo.conf` to include `append="`*Append Line*`"` in your current kernel boot section. Again see Section 7.2 for details.
3. Type **/sbin/lilo** and reboot

Now you should see your hard disks seperately and you can use them. If you previously had partitions on the RAID mirror, you will see partitions accordingly on each of the two member disks. You will also see the data that you previously stored on the RAID mirror.

9.2. Case 2: OS is using RAID

If you want to disable RAID on a Linux computer that is using RAID mirror partitions for `/`, `/boot`, `/usr` *etc*., then first backup your data including `/usr/src/linux/.config`, `/etc/fstab`, `/boot/vmlinux`. Power off your computer. Then turn it on. At the `boot:` prompt, type **linux** followed by *Append Line*. If your system does not boot successfully, then insert your emergency rescue/boot floppy disk, boot off of that floppy disk and type at the `boot:` prompt **linux** followed by *Append Line*. If you still cannot boot, then you cannot disable RAID non−destructively, which means you have to boot off of the RedHat CDROM #1 and re−install Linux, again by typing **linux** followed by *Append Line* at the `boot:` prompt.

If your Linux machine boots successfully, then first make sure that all data is available and clean. Make sure that files in `/boot` are not corrupt. In some cases, the ASCII configuration files in `/boot` and LILO binary files may be corrupt. If that happens, you have to restore them from backup or reconfigure `/etc/lilo.conf`, run **/sbin/lilo** and recompile and reinstall the kernel (recompile with unmodified .config, *i.e.*, without Promise RAID support).

If your Linux machine boots up and all files are safe, then add *Append Line* to `/etc/lilo.conf`, run **/sbin/lilo** and reboot.

If you disabled RAID on OS disks and then later changed your mind, then you have to back up data and reinstall Linux. If you change mind this way and want your RAID back on your Linux OS disks, you may have to sacrifice your current data.

10. Tips and Important Notes

- Dont bother using KDE / GNOME or any other X window system to finish RAID installation as explained above. Use simple text terminal.
- You can press **Cntrl−ALT−Fj** to switch to `ttyj` where `j=1,2,..6`
- If you do start KDE/GNOME or any other X window system, it will run on `tty7`, which you can access by pressing **Cntrl−ALT−F7**
- If you decide to use Promise supplied SCSI emulation driver for FastTrack RAID, note that you are stuck at the default kernel version 2.4.7−10. Because, no source code for FastTrack is available.
- During RH 7.2 installation, choose ext3 and swap file types only.
- When you remove Promise SCSI emulation and setup Native Linux RAID, linux lets you access the hard disks either by mirror name: `d0` or individual disks themselves: `hde2`, `hdg3` etc. It is very important that: YOU NEVER access the hard disks directly by their name, instead access the corresponding mirror partition. Example: Use `/dev/ataraid/d0p3` instead of `/dev/hde3` or `/dev/hdg3`

---

11. For more information

For more information, please check the following resources

- Quasi−Mini−HowTo at http://www.geocities.com/ender7007/
- LhD Product Page at http://lhd.datapower.com/db/dispproduct.cgi?DISP?2751. You can also go to LhD Main Page and search for »Promise FastTrack«.
- ATA RAID help document by RedHat, Inc. You can also join their ataraid mailing list, or atleast search the mailing list archive.
- Promise Technology's support web site at http://support.promise.com. Also check out this page.
- Promise FastTrack help for Linux Mandrake is available at http://www.magic−lamp.org.

A. How is this document generated

So you want to know how I generated this HOWTO? Or, did you download SGML version of this document, modified some portions in it and now want to know how you can create HOWTO?

I made sure there are no errors in my SGML by first typing the command **nsgmls −s ataraid.sgml**. I created a script called `makehowto` and ran the command **./makehowto ataraid.sgml**. Here is the my `makehowto` script:

```
#!/bin/bash
#
# makehowto by Murty Rompalli
# (c) All Rights Reserved
# Free for non commercial use. All other uses must be authorized explicitly
# by the creator. Contact me for more info. murty@solar.m u r t y.net
#

function maketut() {
echo;echo Creating Tutorial Files ...
jade \
    -t sgml \
    -d /usr/lib/sgml/stylesheets/nwalsh-modular/html/ldp.dsl\#html \
    $1.sgml
}

function makehtml {
echo;echo Creating html file: $1.html ...
jade \
    -t sgml \
    -d /usr/lib/sgml/stylesheets/nwalsh-modular/html/docbook.dsl \
    -V nochunks \
    $1.sgml > $1.html
}

function maketxt {
        if [ -f $1.html ]
        then
                echo;echo Creating text file: $1.txt ...
                lynx -dump $1.html > $1.txt
        else
                echo;echo $1.html not found, creating ...
                makehtml $1
                maketxt $1
        fi
}

function makepdf {
        [ -f $1.ps ] && gzip $1.ps

        if [ -f $1.ps.gz ]
        then
                echo;echo Creating pdf file: $1.pdf ...
                gzip -dc $1.ps.gz |
                gs -q -dNOPAUSE -dBATCH -sDEVICE=pdfwrite -sOutputFile=$1.pdf -
        else
                echo;echo $1.ps.gz not found creating ...
                makeps $1
                makepdf $1
        fi
```

```
}

function maketex {
echo;echo Creating TeX file $1.tex ...
jade \
    -t tex \
    -d /usr/lib/sgml/stylesheets/cygnus-both.dsl\#print \
    $1.sgml
gzip $1.tex
echo $1.tex gzipped to $1.tex.gz
}

function makedvi {
echo;echo Creating DVI file $1.dvi ...
db2dvi $1.sgml  >/dev/null 2>&1
echo See $1.log for errors
gzip $1.dvi
echo $1.dvi gzipped to $1.dvi.gz
}

function makeps {
echo;echo Creating PS file $1.ps ...
db2ps $1.sgml  >/dev/null 2>&1
echo See $1.log for errors
gzip $1.ps
echo $1.ps gzipped to $1.ps.gz
}

#### Begin Main Program

echo "
makehowto utility for generating HOWTO from SGML file.
(c) Murty Rompalli
"

[ x$1 = x ] &&
echo "Error. Usage: $0 abc.sgml '{tut|html|pdf|tex|dvi|ps|all}'

Option 'all' is default if sgml file is the only option supplied.

Options:
-------
tut     Make complete tutorial, i.e., generate necessary html files
html    Make a printable single HTML file
pdf     Make a PDF file
tex     Make a TeX source file, gzipped
dvi     Make a DVI file, gzipped
ps      Make a PostScript file, gzipped
all     Generate all possible formats.

" && exit

file="`echo $1|sed 's/\.sgml$//'`"

[ x$file = x ] &&
echo Error. Usage: $0 abc.sgml '{tut|html|tex|dvi|ps|all}' && exit

[ -f $file.sgml ] || {
echo Error. $file.sgml does not exist
exit
}
```

```
[ -r $file.sgml ] || {
echo Error. $file.sgml not readable
exit
}


if [ x$2 = x ]
then
        action=all
else
        action=$2
fi


case $action in
tut|tutorial) maketut $file
                ;;
html|htm) makehtml $file
                ;;
tex|latex) maketex $file
                ;;
dvi) makedvi $file
                ;;
ps) makeps $file
                ;;
text|txt) maketxt $file
                ;;
pdf) makepdf $file
                ;;
all)    maketut $file
        makehtml $file
        maketex $file
        makedvi $file
        makeps $file
        maketxt $file
        makepdf $file
        ;;
*) echo error
        ;;
esac

\rm -f $file.aux
\rm -f $file.tex
\rm -f $file.dvi

echo;echo makehowto: Finished
echo You can review $file.log and delete it.
echo Thank you for using makehowto.
echo
```

You can also just type **./makehowto ataraid.sgml pdf**, for example, if you just want create PDF version. Just type **./makehowto** to get more help on using the script. Click here to download this makehowto script.