

Software-Raid1 Root in Woody

Source: <http://linux.derkeiler.com/Mailing-Lists/Debian/2003-08/1532.html>

From: giuseppe bonacci (*g.bonacci_at_libero.it*)

Date: 08/08/03

Date: Fri, 08 Aug 2003 17:12:17 +0200

To: debian-user@lists.debian.org

Hi all.

This document briefly describes the steps needed to install a Debian Woody GNU/Linux system with root on a software raid1 device.

It took me a fair amount of trials and errors before I got it right, so I would like to share my current knowledge.

I'm not subscribed to this list, so if you want to send me corrections/improvements/whatever, please CC my email address.

NO WARRANTY:

This is the procedure that worked for me.

Despite my efforts to be accurate, there is no warranty that it will work for you, or even that it will not damage your system.

Be warned that unless you know what you are doing and you are extremely careful, you seriously risk to loose all the contents of your hard disks. So I strongly advice you not to play with PCs that contain valuable information.

Most important: whatever damage you may do by applying the following procedure, it's YOUR FAULT, and I take no responsibility.

If you do not agree, stop reading now.

Well, now revert to the real stuff:

Since this is a test setup, the environment is going to be small, and supported by the debian vanilla kernel:

- Pentium III
- SCSI disk 0:0, sda, 160 Mb
- SCSI disk 0:1, sdb, 160 Mb

I used SCSI disks because I had them easily available, but the procedure should work for IDE disks on (e.g.) hda and hdc as well.

1. Install Debian Woody.

Debian–User: Software–Raid1 Root in Woody

I started from floppies (vanilla kernel) + network, and partitioned the disks as follows:

```
# fdisk -l /dev/sda
```

```
Disk /dev/sda: 64 heads, 32 sectors, 160 cylinders
Units = cylinders of 2048 * 512 bytes
```

```
Device Boot Start End Blocks Id System
/dev/sda1 * 1 160 163824 83 Linux
```

```
# fdisk -l /dev/sdb
```

```
Disk /dev/sdb: 64 heads, 32 sectors, 160 cylinders
Units = cylinders of 2048 * 512 bytes
```

```
Device Boot Start End Blocks Id System
/dev/sdb1 * 1 160 163824 fd Linux raid autodetect
```

Then I made an ext2 filesystem and installed a minimal system on /dev/sda1. Better avoid installing further packages until the raid is set up, while it might be a good idea to update the system with

```
# apt-get update
# apt-get upgrade
```

2. Switch to kernel 2.4.

I prefer to use raidtools2 with a stock 2.4 series kernel. Moreover, despite later versions of LILO being easier to setup, I decided to stick on GRUB. So the next steps were:

```
# dpkg -P lilo
# apt-get install grub kernel-image-2.4.18-686
# grub-install '(hd0)'
```

Notice that for processors different from Pentium III you should pick a flavour different from '-686'.

Create /boot/grub/menu.lst like this:

```
# cat /boot/grub/menu.lst
title Debian
    root (hd0,0)
    kernel /vmlinuz root=/dev/sda1 ro
    initrd /initrd.img
```

WARNING: Don't install devfsd. Device paths change and initrd-tools get confused.

Reboot.

Software–Raid1 Root in Woody

3. Create and populate the Raid 1 structure in degraded mode.

Install raidtools2:

```
# modprobe -k raid1 # needed to prevent preinst from complaining
# apt-get install raidtools2
```

Create /etc/raidtab like this:

```
# cat /etc/raidtab
raiddev /dev/md0
    nr-raid-disks 2
    raid-level 1
    persistent-superblock 1
    chunk-size 4

    device /dev/sdb1
    raid-disk 0

    device /dev/sda1
    raid-disk 1

    failed-disk 1
```

Notice the failed–disk directive at the end of the file. It specifies that the current root disk, /dev/sda1, is to be ignored for the moment, and marked `failed' in the raid superblock.

Now make up the raid:

```
# mkraid /dev/md0
```

If this is your n–th trial with n>1, you might be forced to use the '-f' option to mkraid. Read carefully the warning before proceeding.

You can check the status by looking at /proc/mdstatus

Build an ext2 filesystem on /dev/md0 and copy the whole system on it (that's why it's useful to keep it minimal)

```
# mke2fs -O sparse_super,filetype /dev/md0
# mount /dev/md0 /mnt
# find / -xdev -depth | cpio -pmdu /mnt
```

Modify /mnt/etc/fstab and /mnt/boot/grub/menu.lst to mention md0 or sdb in place of sda1:

```
# diff /etc/fstab /mnt/etc/fstab
4c4
< /dev/sda1 / ext2 errors=remount-ro 0 1
```

Debian-User: Software-Raid1 Root in Woody

```
---
> /dev/md0      /                ext2      errors=remount-ro      0 1
# diff /boot/grub/menu.lst /mnt/boot/grub/menu.lst
2,3c2,3
<      root      (hd0,0)
<      kernel    /vmlinuz root=/dev/sdal ro
---
>      root      (hd1,0)
>      kernel    /vmlinuz root=/dev/md0 ro
Now build an initrd for the new setup:
# mkinitrd -o /mnt/boot/initrd.img-2.4.18-"your flavour" -k -r /dev/md0
The option '-k' instructs mkinitrd to keep the expanded image under
/tmp/mkinitrd.XXX/initrd instead of deleting it.
Notice that /mnt/initrd.img -> /boot/initrd.img-YYYY, so don't try to be
too smart.
4. Reboot with root on /dev/md0 and synchronize the disks.
Reboot. So far, the system is still configured to boot from the first
disk and leave the second alone.
Now we have to manually interrupt grub's boot sequence in order to
launch the second disk instead of the first:
Ask grub for a command line (press 'c') and at the 'grub>' prompt
issue the command:
grub> configfile (hd1,0)/boot/grub/menu.lst
You should get the same menu item as before, but meaning a different sequence
(use 'e' to examine it, then ESC to return to the menu). Press RETURN
to boot.
You should end up with linux running and /dev/md0 mounted as /.
Now it's time to change the partition type of the old root disk from 83 to FD:
# fdisk -l /dev/sda
Disk /dev/sda: 64 heads, 32 sectors, 160 cylinders
Units = cylinders of 2048 * 512 bytes
   Device Boot   Start     End  Blocks  Id System
/dev/sdal *      1       160   163824  fd  Linux raid autodetect
Now modify /etc/raidtab to remove the last line ('failed-disk...'), and
attach the old root partition as a plex of the raid1 structure:
# raidhotadd /dev/md0 /dev/sdal
The md driver starts synchronization, that can be checked by looking at
/proc/mdstat.
Rebuild the initial ramdisk with a clean raidtab (paranoid mode on):
# mkinitrd -k -o /initrd.img
Setup grub on both disks:
# (echo 'root (hd1,0)'; echo 'setup (hd1)'; echo quit) | grub
# (echo 'root (hd0,0)'; echo 'setup (hd0)'; echo quit) | grub
Done.
Now you can reboot (just to test the setup works), play with dselect
to your pleasure, etc.
Again: if you happen to install devfsd you might end up with an unbootable
system, and no way to recover.
As an exercise to the reader, try disabling one of the disks and test
your ability to recover... If you find a way out of the Kernel panic,
I'll be glad to know. ;-)
best regards
-- gb
--
To UNSUBSCRIBE, email to debian-user-request@lists.debian.org
with a subject of "unsubscribe". Trouble? Contact listmaster@lists.debian.org
```