Root Filesystem on RAID-1 with NetBSD's RAIDframe

Last updated $Date: 2004/04/17 14:45:01 $

Martti Kuparinen <martti.kuparinen@iki.fi>

This article was contributed to The NetBSD Guide in November 2003.

Abstract

Many users have nowadays big hard drives which contain lots of valuable files such as email archives, pictures from digital cameras, "backup copies" of the latest movies and so on. As the hard drives are so cheap yet so big in capacity it can be a difficult task to take backup of the not-so-important files. At the same time it would be a real pity to lose those files in case of a disk failure.

This article describes a real-life NetBSD installation with RAID-1 protected filesystems. With RAID-1 the server can be fully accessible even if the faulty drive is disconnected and sent back to the manufacturer.

Table of Contents

---

1. Introduction

I wanted to create a home server with failure resistant filesystems to protect myself against disk failures as this is unfortunately not very uncommon these days. I decided to use RAID-1 for everything (/, swap and /home). This article describes how to setup RAID-1 and make the system bootable even if the primary boot device is removed due to failure. The reason for choosing RAID-1 instead of RAID-10 or RAID-5 is the fact that currently NetBSD supports only RAID-1 on root filesystem (/). RAID-0 was not even considered as it provides no redundancy in case of disk

failure.

This article assumes basic knowledge about RAID. In this example we have two identical IDE disks (wd#) which we are going to mirror (RAID-1). These disks are identified as:

```
wd0 at atabus0 drive 0: <IC35L120AVV207-1>
wd0: drive supports 16-sector PIO transfers, LBA48 addressing
wd0: 115 GB, 239340 cyl, 16 head, 63 sec, 512 bytes/sect x 241254720 sectors
wd0: 32-bit data port
wd0: drive supports PIO mode 4, DMA mode 2, Ultra-DMA mode 5 (Ultra/100)
wd0(viaide0:0:0): using PIO mode 4, Ultra-DMA mode 5 (Ultra/100) (using DMA data
transfers)
wd1 at atabus1 drive 0: <IC35L120AVV207-1>
wd1: drive supports 16-sector PIO transfers, LBA48 addressing
wd1: 115 GB, 239340 cyl, 16 head, 63 sec, 512 bytes/sect x 241254720 sectors
wd1: 32-bit data port
wd1: drive supports PIO mode 4, DMA mode 2, Ultra-DMA mode 5 (Ultra/100)
wd1(viaide0:1:0): using PIO mode 4, Ultra-DMA mode 5 (Ultra/100) (using DMA data
transfers)
```

Both disks are masters (drive 0) in separate channels. This is important with IDE disks as you might not be able to boot from a slave disk (drive 1) on older systems. Also performance is better if both disks are on separate channels. SCSI disks (sd#) can be on the same channel as the controller is smart and knows how to get the optimal throughput with multiple disks. In case of SCSI disks we need to replace wd# with sd# in this article.

2. Initial install

We start by installing NetBSD on the first disk (wd0) without any RAID support. We split the disk into three parts (/, swap and /home) as this is what we are going to have in the final RAID-1 configuration. It is also possible to convert an existing system to RAID-1, even without console access. The author has done this over SSH session and there was no need to go to the console. If an existing system is converted it is important to take backup of all important files!

Next we need to make sure we have RAID support in the kernel (which is included in the GENERIC kernel). The kernel configuration file must have the following settings:

```
pseudo-device   raid            8       # RAIDframe disk driver
options         RAID_AUTOCONFIG         # auto-configuration of RAID components
```

The RAID support must be detected by the NetBSD kernel, which can be checked by looking at the output of the dmesg command.

```
# dmesg
...
Kernelized RAIDframe activated
```

3. Setting up the second disk

Next we setup the second disk (wd1). We can find the correct numbers from wd0 and use them with wd1 as the disks are identical. We must remember to mark the

NetBSD partition active or the system will not boot.

```
# fdisk /dev/wd0
Disk: /dev/rwd0d
NetBSD disklabel disk geometry:
cylinders: 16383 heads: 16 sectors/track: 63 (1008 sectors/cylinder)

BIOS disk geometry:
cylinders: 1024 heads: 255 sectors/track: 63 (16065 sectors/cylinder)

Partition table:
0: sysid 169 (NetBSD)
    start 63, size 241254657 (117800 MB, Cyls 0-15956), Active
1: <UNUSED>
2: <UNUSED>
3: <UNUSED>
```

NetBSD is on the 1st partition on wd0 so we use the same number for wd1.

```
# dd if=/dev/zero of=/dev/rwd1d bs=8k count=1
# fdisk -0ua /dev/wd1
```

Both disks should be identical now. We verify this once more.

```
# fdisk /dev/wd0
# fdisk /dev/wd1
```

NetBSD 2.x

Next we configure our newly created partition. In this example we have one RAID slice (a) for all filesystems and swap. Note that there is no swap (b) yet.

```
# disklabel /dev/wd1 > disklabel.wd1
# vi disklabel.wd1

8 partitions:
#        size    offset     fstype  [fsize bsize cpg/sgs]
 a: 241254657        63     RAID
 c: 241254657        63     unused      0     0
 d: 241254720         0     unused      0     0
```

Note how the a and c slices have same values. This is because NetBSD uses the whole disk and we want RAID-1 for everything. Next we install the new disklabel on wd1.

```
# disklabel -R -r /dev/wd1 disklabel.wd1
# disklabel /dev/wd1
```

NetBSD 1.6.x

NetBSD 1.6.x need an additional slice (h) for the boot loader in addition to the RAID slice (a). And as with the NetBSD 2.x example, there is no swap (b) yet.

```
# disklabel /dev/wd1 > disklabel.wd1
# vi disklabel.wd1

8 partitions:
#        size    offset     fstype  [fsize bsize cpg/sgs]
 a: 241250561      4159     RAID
```

```
 c: 241254657        63     unused      0     0
 d: 241254720         0     unused      0     0
 h:      4096        63     4.2BSD   1024  8192  64
```

The sizes are calculated like this:

```
# dc
241254657   # size of c
4096        # size of h (2 MB = 4096 blocks x 512 bytes/blocks)
-p
241250561   # size of a

4096        # size of h
63          # start of h
+p
4159        # start of a
q
```

Next we install the new disklabel on wd1.

```
# disklabel -R -r /dev/wd1 disklabel.wd1
# disklabel /dev/wd1
```

4. Configuring the RAID device

Next we create configuration files for the RAID devices. These files are needed only during the initial setup as we auto-configure the devices later.

```
# cat > /var/tmp/raid0.conf << EOF
START array
1 2 0

START disks
/dev/wd9a
/dev/wd1a

START layout
128 1 1 1

START queue
fifo 100
EOF
```

Note that wd9 is a non-existing disk. There must be, however, a device node for that in the /dev directory. wd9 will be replaced later by wd0.

```
# cd /dev
# sh MAKEDEV wd9
# cd
```

Next we configure the RAID device and initialize the serial number to something unique. In this example we use 2004### (Nth RAID device in 2004). After that we start the initialization process.

```
# raidctl -C /var/tmp/raid0.conf raid0
# raidctl -I 2004001 raid0
# raidctl -i raid0
# raidctl -s raid0
```

5. Setting up filesystems

The RAID device is now configured and available. Now it is the time to think about the filesystems. In this example we use the following layout:

- 8 GB for /
- 1 GB for swap
- everything else for /home

We must next create a disklabel inside the RAID device and format the filesystems. The values in the next example are for NetBSD 2.x. For 1.6.x the size of d is smaller because of the extra slice (h) for the boot-loader in the beginning of the physical disk (wd#h).

```
# disklabel raid0 > disklabel.raid0
# vi disklabel.raid0

8 partitions:
#        size    offset     fstype  [fsize bsize cpg/sgs]
 a:  16777216         0     4.2BSD      0     0     0
 b:   2097152  16777216     swap
 d: 241254528         0     unused      0     0     0
 e: 222380160  18874368     4.2BSD      0     0     0
```

It should be noted that 1 GB is 2*1024*1024=2097152 blocks (1 block is 512 bytes, or 0.5 kilobytes). The sizes and offsets can be calculated like this:

```
# dc
241254528   # size of d
16777216    # size of a (8 GB = 8 x 2 x 1024 x 1024)
-
2097152     # size of b (1 GB = 2 x 1024 x 1024)
-p
222380160   # size of e

16777216    # size of a
2097152     # size of b
+p
18874368    # offset of e
q
```

raid0a will be the root filesystem (/), raid0b the swap and raid0e /home. In this example we do not have separate filesystems for /usr and /var. The next thing is to install the new disklabel for the RAID device and format the filesystems. Note that the swap area is not formated.

```
# disklabel -R -r raid0 disklabel.raid0
# newfs /dev/raid0a
# newfs /dev/raid0e
```

6. Setting up kernel dumps

The normal swap area in our case is on raid0b but this can not be used for crash dumps as process scheduling is stopped when dumps happen. Therefore we must use a real disk device. However, nothing stops us from defining a dump area which overlaps with raid0b. The trick here is to calculate the correct start offset for our crash dump area. This is dangerous and it is possible to destroy valuable data if we

make a mistake in these calculations! Data corruption will happen when the kernel write its memory dump over a normal filesystem. So we must be extra careful here. (The author destroyed his 100+ GB /home with a kernel crash dump!)

First we need to take a look at the disklabel for swap (raid0b) and the real physical disk (wd1).

```
# disklabel /dev/raid0

8 partitions:
#        size    offset     fstype  [fsize bsize cpg/sgs]
 a:  16777216         0     4.2BSD   1024  8192    64
 b:   2097152  16777216     swap
 d: 241254528         0     unused      0     0     0
 e: 222380160  18874368     4.2BSD   1024  8192    64


# disklabel /dev/wd1

8 partitions:
#        size    offset     fstype  [fsize bsize cpg/sgs]
 a: 241254657        63     RAID
 c: 241254657        63     unused      0     0
 d: 241254720         0     unused      0     0
```

Each RAID set has a 64 block reserved area (see RF_PROTECTED_SECTORS in <dev/raidframe/raidframevar.h>) in the beginning of the set to store the internal RAID structures.

```
# dc
63              # offset of wd1a
64              # RF_PROTECTED_SECTORS
+
16777216        # offset of raid0b
+p
16777343        # offset of swap within wd1
q
```

We know now that real offset of the still-nonexisting wd1b is 16777343 and size is 2097152. Next we need to add wd1b to wd1's disklabel.

```
# disklabel /dev/wd1 > disklabel.wd1
# vi disklabel.wd1

8 partitions:
#        size    offset     fstype  [fsize bsize cpg/sgs]
 a: 241254657        63       RAID
 b:   2097152  16777343       swap
 c: 241254657        63     unused      0     0
 d: 241254720         0     unused      0     0
```

Next we install the new disklabel.

```
# disklabel -R -r /dev/wd1 disklabel.wd1
```

Why isn't sizeof(raid0d) == (sizeof(wd1a) - RF_PROTECTED_SECTORS)? Size of raid0d is based on the largest multiple of the stripe size used for a RAID set. As an example, with stripe width of 128, size of raid0d is:

```
# dc
241254657   # size of wd1a
64          # RF_PROTECTED_SECTORS
```

```
-
128         # stripe width
/p
1884801     # number of stripes
128         # number of blocks per stripe
*p
241254528   # size of raid0d
```

7. Moving the existing files into the new filesystems

The new RAID filesystems are now ready for use. We mount them under /mnt and copy all files from the old system.

```
# mount /dev/raid0a /mnt
# dump -0 -f - / | (cd /mnt && restore -x -f -)

# mount /dev/raid0e /mnt/home
# dump -0 -f - /home | (cd /mnt/home && restore -x -f -)
```

The data is now on the RAID filesystems. We need to fix the mount-points in fstab or the system will not come up correctly.

Note that the kernel crash dumps must not be saved on a RAID device but on a real physical disk (wd0b). This dump area was created in the previous chapter on the second disk (wd1b) but we will make wd0 an identical copy of wd1 later so wd0b and wd1b will have the same size and offset. If wd0 fails and is removed from the server wd1 becomes wd0 after reboot and crash dumps will still work as we are using wd0b in /etc/fstab. The only fault in this configuration is when the original, failed wd0 is replaces by a new drive and we haven't initialized it yet with fdisk and disklabel. In this short period of time we can not make crash dumps in case of kernel panic. Note how the dump device has the "dp" keyword on the 4th field.

```
# vi /mnt/etc/fstab

/dev/raid0a     /               ffs     rw      1       1
/dev/raid0b     none            swap    sw      0       0
/dev/raid0e     /home           ffs     rw      1       1
/dev/wd0b       none            swap    dp      0       0
```

The swap should be unconfigured upon shutdown to avoid parity errors on the RAID device. This can be done with a simple, one-line setting in /etc/rc.conf.

```
# cat >> /mnt/etc/rc.conf << EOF
swapoff=YES
EOF
```

Next the boot loader must be installed on wd1. Failure to install the loader will render the system unbootable if wd0 fails. Please note how the boot loader is installed on the small slice (wd1h) which is in the beginning of wd1.

NetBSD 2.x

```
# /usr/sbin/installboot -v /dev/rwd1a /usr/mdec/bootxx_ffsv1
```

```
# newfs /dev/wd1h
# /usr/mdec/installboot -v /usr/mdec/biosboot.sym /dev/rwd1h
```

Finally the RAID sets must be made auto-configurable and the system should be rebooted. After the reboot everything is mounted from the RAID devices.

```
# raidctl -A root raid0
# shutdown -r now
```

8. The first boot with RAID-1

The system should come up now and all filesystems should be on the RAID devices.

```
# df -h
Filesystem  1K-blocks     Used     Avail Capacity  Mounted on
/dev/raid0a   7.9G     570M      6.9G     7%    /
/dev/raid0e   104G     7.4G       92G     7%    /home

# swapctl -l
Device      1K-blocks     Used    Avail Capacity  Priority
/dev/raid0b   1048576        0  1048576     0%    0
```

The RAID devices are not fully functional yet as the (non-existing) drive wd9 has failed.

```
# raidctl -s raid0
Components:
           component0: failed
            /dev/wd1a: optimal
```

9. Adding the first disk

First we need to relabel wd0 to have the same layout as wd1. Then we add wd0 as hot-space and initiate the reconstruction for all RAID devices.

```
# disklabel /dev/wd1 > disklabel.wd1
# disklabel -R -r /dev/wd0 disklabel.wd1
# disklabel /dev/wd0

# raidctl -a /dev/wd0a raid0
/netbsd: Warning: truncating spare disk /dev/wd0a to 241254528 blocks
# raidctl -F component0 raid0
```

Please note that the reconstruction is a slow process and can take several hours to complete.

```
# raidctl -S raid0
Reconstruction is 0% complete.
Parity Re-write is 100% complete.
Copyback is 100% complete.
Reconstruction status:
  17% |******                                | ETA: 01:58:08 -
```

After reconstruction both disks should be "optimal".

```
# raidctl -s raid0
Components:
           component0: spared
            /dev/wd1a: optimal
Spares:
           /dev/wd0a: used_spare
```

When the reconstruction is ready we need to install the boot loader on the first disk (wd0).

NetBSD 2.x

```
# /usr/sbin/installboot -v /dev/rwd0a /usr/mdec/bootxx_ffsv1
```

NetBSD 1.6.x

```
# newfs /dev/wd0h
# /usr/mdec/installboot -v /usr/mdec/biosboot.sym /dev/rwd0h
```

That's it, we have a redundant RAID-1 host. The next thing we should do is to read the raid(4) and raidctl(8) manual pages and educate yourself what to do when (not if) one of the drives fail. Finally the reader must note that RAID systems do not make backups obsolete as they do not protect against rm -rf. It is also important to test the kernel crash dumps so that they work correctly and do not overwrite any important filesystems (like the raid0e filesystem).

Press Ctrl+Alt+Esc to test the kernel crash dump. This will invoke the kernel debugger. Type sync and press Enter. This will save the current kernel memory to the dump area (wd0b) for further analysis. Most likely the offset and/or size of wd0b is wrong if the system will not come up correctly after reboot (unable to mount /home, corrupted super-blocks, etc). It is very important to test this now, not when we have lots of valuable files in /home. As an example, the author destroyed his 100+ GB /home with a kernel crash dump! No real harm was caused by this because of up-to-date backups (backup was made just before converting to RAID-1). One more time: take a backup of all your files before following these instructions!

Did this article help you? Is this missing something? Send your comments to the author.