

A Linux Software RAID Tutorial

March 10, 2006

Dustin C. Kirkland

Copyright (C) International Business Machines Corp., 2006

Table of Contents

<u>An Overview of RAID</u>	1
<u>What is a RAID?</u>	1
<u>Hardware RAID vs. software RAID?</u>	1
<u>How does software RAID work in Linux?</u>	1
<u>What are the differences between the various RAID levels?</u>	2
<u>RAID0 (striping)</u>	2
<u>RAID1 (mirroring)</u>	2
<u>RAID2 (bit striping)</u>	2
<u>RAID3 (byte striping)</u>	2
<u>RAID4 (block striping)</u>	3
<u>RAID5 (block striping with striped parity)</u>	3
<u>Other RAID Levels</u>	3
<u>Installing Linux onto a software RAID</u>	4
<u>The RHEL/Fedora graphical process</u>	5
<u>The SLES/openSUSE graphical process</u>	12
<u>Creating a Software RAID on an Existing Linux System</u>	19
<u>Userspace utilities</u>	19
<u>raidtools</u>	19
<u>mdadm</u>	19
<u>RAID creation on the command line</u>	19
<u>Losing a RAID Component</u>	23
<u>Physically removed disk</u>	23
<u>Corrupted or damaged disk</u>	23
<u>Simulating the loss</u>	23
<u>Recovering and Restoring a RAID</u>	25
<u>RAID recovery on the command line</u>	25
<u>Appendices</u>	29
<u>Appendix A</u>	30
<u>A network installable tree</u>	30
<u>Appendix B</u>	31
<u>Minimally configured NFS server</u>	31
<u>Appendix C</u>	32
<u>Minimally configured DHCP server</u>	32
<u>Appendix D</u>	33
<u>Minimally configured DNS server</u>	33
<u>Appendix E</u>	34
<u>Simple peer-to-peer TCP/IP Linux networking</u>	34

Table of Contents

<u>Appendix F</u>	35
<u>Installation initiation from USB media</u>	35
<u>RHEL/Fedora</u>	35
<u>SLES/openSUSE</u>	35
<u>References</u>	38
<u>About the Author</u>	39
<u>Legal Statement</u>	40

An Overview of RAID

What is a RAID?

The term RAID is an acronym for the phrase, Redundant Array of Independent Disks. RAID is a way of combining the storage available across multiple disks and supplying users a single, unified virtual device. RAID can be used to provide:

- data integrity
- fault tolerance
- improved performance
- greater storage capacity

Hard disks are mechanical devices involving moving parts and unfortunately tend to fail over time. There are also physical limits to the speed at which data can be read and/or written to disks. RAID helps mitigate this risk by protecting data stored on hard disks and improving disk performance by writing the data to multiple physical locations according to several different schemas, known as "RAID Levels". Furthermore, RAID can be provided by either dedicated, specialized hardware or by the operating system at a virtual layer.

Hardware RAID vs. software RAID?

Hardware RAID solutions exist that operate as dedicated devices, usually as PCI expansion cards or directly on the motherboard. The independent disks attach to the hardware interface. In a true hardware RAID, the operating system simply writes data to the hardware RAID controller which handles the multiplicitous reads and writes to the associated disks. Other so-called hardware RAIDs rely on special drivers to the operating system; these act more like software RAIDs in practice. With current technology, hardware RAID configurations are generally chosen for very large RAIDs.

Additionally, some operating systems, including Linux®, provide RAID functionality within a software layer. RAID partitions are logically combined and a virtual device appears to higher layers of the operating system in place of the multiple constituent devices. This solution is often a high-performance and inexpensive alternative available for RAID users.

How does software RAID work in Linux?

Support for software RAID is provided natively within the Linux kernel. A RAID device consists of multiple disk partition block devices grouped together and associated with a RAID level. The kernel creates a virtual block device representing the entire RAID. Thereafter, reads and writes are performed on the RAID virtual block device based on the particular algorithm of the chosen RAID level.

In addition to the code in kernel space that implements reading data from and writing data to the software RAID, a user space component also exists in Linux environments that provides management access to the RAID. Using the mdadm suite, Linux users can create, query, deconstruct, synchronize, and otherwise maintain software RAID devices.

What are the differences between the various RAID levels?

RAID0 (striping)

RAID0 implements *striping*, which is a way of distributing reads and writes across multiple disks for improved disk performance. Striping reduces the overall load placed on each component disk in that different segments of data can be simultaneously read or written to multiple disks at once. The total amount of storage available is the sum of all component disks. Disks of different sizes may be used, but the size of the smallest disk will limit the amount of space usable on all of the disks. Data protection and fault tolerance is not provided by RAID0, as none of the data is duplicated. A failure in any one of the disks will render the RAID unusable and data will have been lost. However, RAID0 arrays are sometimes used for read-only fileserving of already-protected data.

Linux users wishing to concatenate multiple disks into a single, larger virtual device should consider Logical Volume Management (LVM). LVM supports striping and allows dynamically growing or shrinking logical volumes and concatenation of disks of different sizes. LVM is not covered further in this tutorial. [5]

RAID1 (mirroring)

RAID1 is an implementation where all written data is duplicated (or *mirrored*) to each constituent disk, thus providing data protection and fault tolerance. RAID1 can also provide improved performance, as the RAID controllers have multiple disks from which to read when one or more are busy. The total storage available to a RAID1 user, however, is equal to the smallest disk in the set, and thus RAID1 does not provide a greater storage capacity.

An optimal RAID1 configuration will usually have two identically sized disks. A failure of one of the disks will not result in data lost since all of the data exists on both disks, and the RAID will continue to operate (though in a state unprotected against a failure of the remaining disk). The faulty disk can be replaced, the data synchronized to the new disk, and the RAID1 protection restored.

RAID2 (bit striping)

RAID2 stripes data at the bit level across disks and uses a Hamming code for parity. However, the performance of bit striping is abysmal and RAID2 is not practically used.

RAID3 (byte striping)

RAID3 stripes data at the byte level and dedicates an entire disk for parity. Like RAID2, RAID3 is not practically used for performance reasons. As most any read requires more than one byte of data, reads involve operations on every disk in the set. Such disk access will easily thrash a system. Additionally, loss of the parity disk yields a system vulnerable to corrupted data.

RAID4 (block striping)

RAID4 stripes data at the block-level and dedicates an entire disk for parity. RAID4 is similar to both RAID2 and RAID3 but significantly improves performance as any read request contained within a single block can be serviced from a single disk. RAID4 is used on a limited basis due to the storage penalty and data corruption vulnerability of dedicating an entire disk to parity.

RAID5 (block striping with striped parity)

RAID5 implements block level striping like RAID4, but instead stripes the parity information across all disks as well. In this way, the total storage capacity is maximized and parity information is distributed across all disks. RAID5 also supports hot spares, which are disks that are members of the RAID but not in active use. The hot spares are activated and added to the RAID upon the detection of a failed disk. RAID5 is the most commonly used level as it provides the best combination of benefits and acceptable costs.

Other RAID Levels

Other RAID levels exist, such as RAID6, RAID1+0, Nested RAID, and various proprietary RAID implementations. For the practical purposes of this tutorial, detailed treatment is only given to RAID1 and RAID5. [13]

Installing Linux onto a software RAID

RHEL/Fedora and SLES/openSUSE support RAID creation at the time of installation. This is a great way to ensure RAID protection of your data from the beginning. The graphical processes are mostly intuitive, but both involve their own nuances. Screen shots of each with instructional explanations of every step of the processes are found below.

Note: These screen shots actually demonstrate Linux software RAID entirely on a single disk (`/dev/hda`). This does not actually offer RAID's redundancy protection as all of the data is on the same disk! However, the Linux kernel treats the array in the same way, and thus we can sufficiently use this virtualization for the purposes of this demonstration. In a real scenario, you should create partitions on each of a set of disks (for example, `/dev/hda`, `/dev/hdb`, `/dev/hdc`).

Note: Taken literally, the instructions below perform a completely destructive installation. In other words, ***you should NOT perform these examples on any systems having valuable data, as all data will be lost during the partitioning and formatting steps.***

A few key points about software RAID to keep in mind regardless of your choice of distribution:

- RAID0 (striping) does not provide redundant data protection and is not demonstrated by this tutorial.
- RAID1 (mirroring) requires at least two partitions, and your total space available will equal that of the smallest partition. If you are creating more than two partitions, RAID1 is probably not your best option; you should consider RAID5 instead to more efficiently use your available storage.
- RAID5 (redundant striping) requires at least 3 partitions and supports any number of hot spare partitions.
- Optimal chunk size (block size) will vary. This tutorial will not cover this in detail.
- A *persistent superblock* is important RAID configuration data that is written to the beginning of every disk and allows the kernel to initialize the array when the configuration files are actually stored in the array.

Also, these instructions show a basic partitioning setup, with a 100 MB `/boot` partition, a 1 GB swap partition, and either:

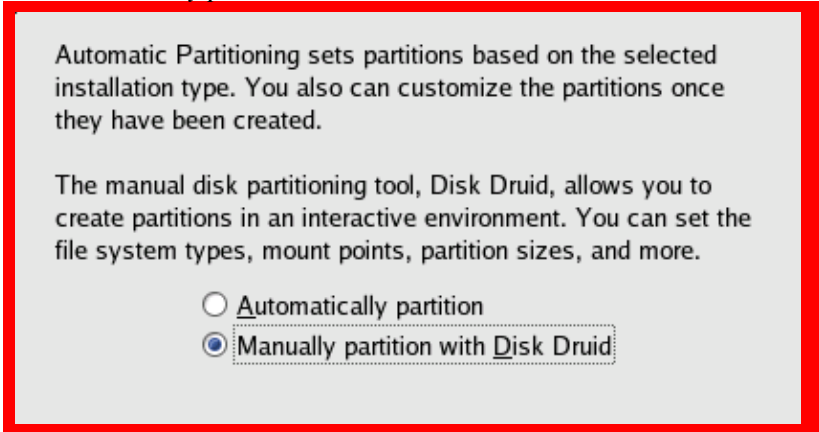
- a pair of 2 GB software RAID partitions combined into a RAID1 mounted at `/`
- or three 2 GB software RAID partitions combined into a RAID5 mounted at `/`

The reasoning for these choices:

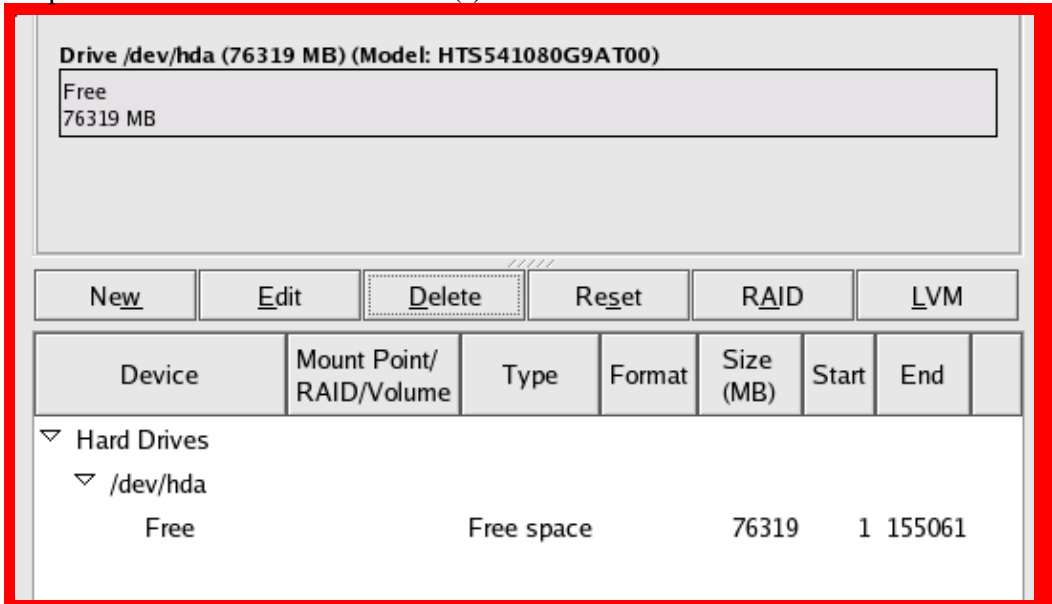
- It is probably a good idea to create a separate boot partition, as some bootloaders are not capable of reading RAID partitions at boot time. That is, software RAID is provided by the Linux kernel, which is not yet running when the bootloader executes.
- Swap space is a special disk partition the kernel uses when caching information from RAM to hard disk to improve performance. Optimally, swap space should be twice the size of your RAM. Swap space should not exist on a RAID, as the Linux kernel optimizes swap reads and writes according to its own algorithm.
- A size of 2 GB per software RAID partition was chosen in this tutorial in the interest of a speedy demonstration. 2 GB is large enough for a basic Linux installation and does not require very long to format and initially synchronize. Your partitions may well be significantly larger.
- If you select `/` (the root directory) as your mount point for the RAID, you will install all of Linux onto the RAID. Alternatively, you might instead choose a different mount point, such as `/data`.

The RHEL/Fedora graphical process

1. When prompted, select *Manually partition with Disk Druid*.



2. Delete all partitions and start with clean disk(s).



A Linux Software RAID Tutorial

3. Create a 100 MB /boot partition formatted ext3.

The screenshot shows the 'Add Partition' dialog box with the following configuration:

- Mount Point:** /boot
- File System Type:** ext3
- Allowable Drives:** A list containing one entry: hda 76319 MB HTS541080G9AT00
- Size (MB):** 100
- Additional Size Options:**
 - Fixed size
 - Fill all space up to (MB): 1
 - Fill to maximum allowable size
- Force to be a primary partition

Buttons: Cancel, OK

4. Create a 1024 MB swap partition.

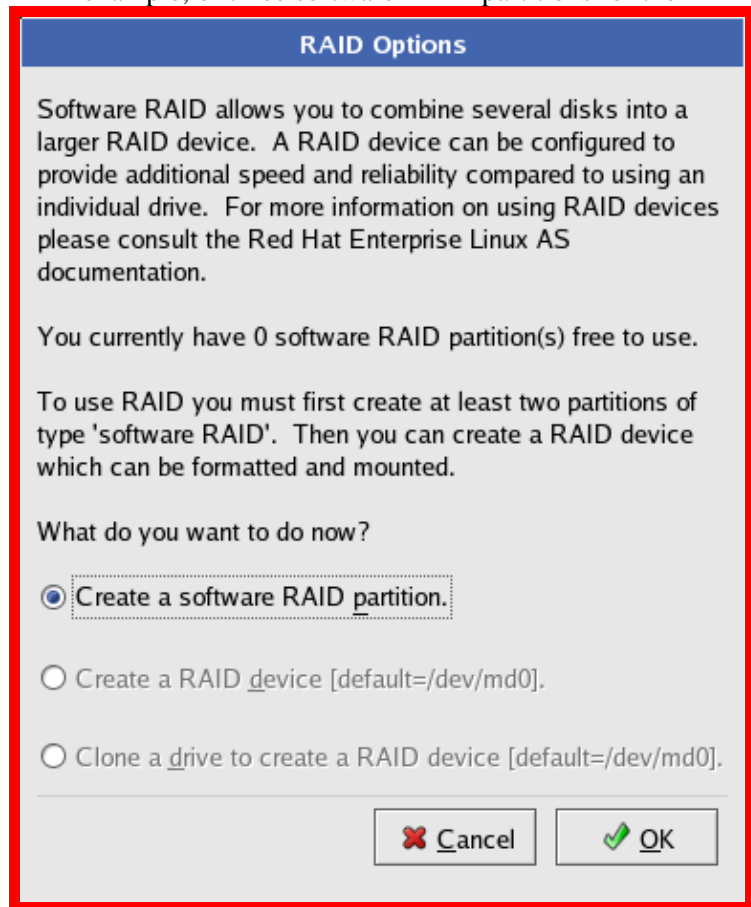
The screenshot shows the 'Add Partition' dialog box with the following configuration:

- Mount Point:** <Not Applicable>
- File System Type:** swap
- Allowable Drives:** A list containing one entry: hda 76319 MB HTS541080G9AT00
- Size (MB):** 1024
- Additional Size Options:**
 - Fixed size
 - Fill all space up to (MB): 1
 - Fill to maximum allowable size
- Force to be a primary partition

Buttons: Cancel, OK

A Linux Software RAID Tutorial

5. Create as many individual RAID partitions as required. In this example, create two software RAID partitions for the RAID1 example, or three software RAID partitions for the RAID5 example.



A Linux Software RAID Tutorial

6. Choose the properties of the partition.
 - ◆ Make sure the *File System Type* selected is *software RAID*.
 - ◆ Choose the physical drive on which the partition will be created.
 - ◆ Define the size of the partition.
 - ◆ You do not need to select the *Force to be a primary partition* box.

The screenshot shows the 'Add Partition' dialog box with the following configuration:

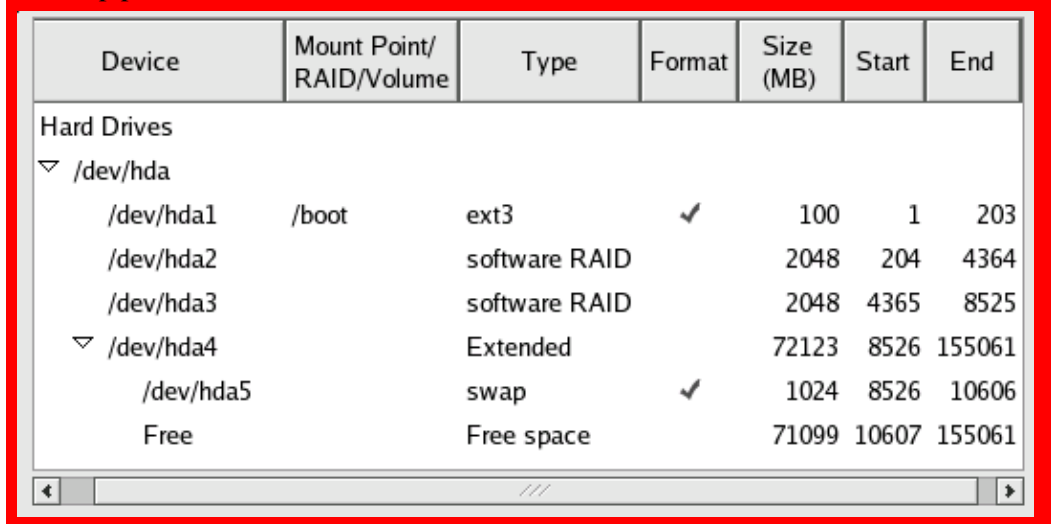
- Mount Point:** <Not Applicable>
- File System Type:** software RAID
- Allowable Drives:** hda 76319 MB HTS541080G9AT00
- Size (MB):** 2048
- Additional Size Options:**
 - Fixed size
 - Fill all space up to (MB): 1
 - Fill to maximum allowable size
- Force to be a primary partition

Buttons:

A Linux Software RAID Tutorial

7. You should see:

- ◆ An ext3 /boot.
- ◆ At least two software RAID partitions.
- ◆ A swap partition.



Device	Mount Point/ RAID/Volume	Type	Format	Size (MB)	Start	End
Hard Drives						
▼ /dev/hda						
/dev/hda1	/boot	ext3	✓	100	1	203
/dev/hda2		software RAID		2048	204	4364
/dev/hda3		software RAID		2048	4365	8525
▼ /dev/hda4		Extended		72123	8526	155061
/dev/hda5		swap	✓	1024	8526	10606
Free		Free space		71099	10607	155061

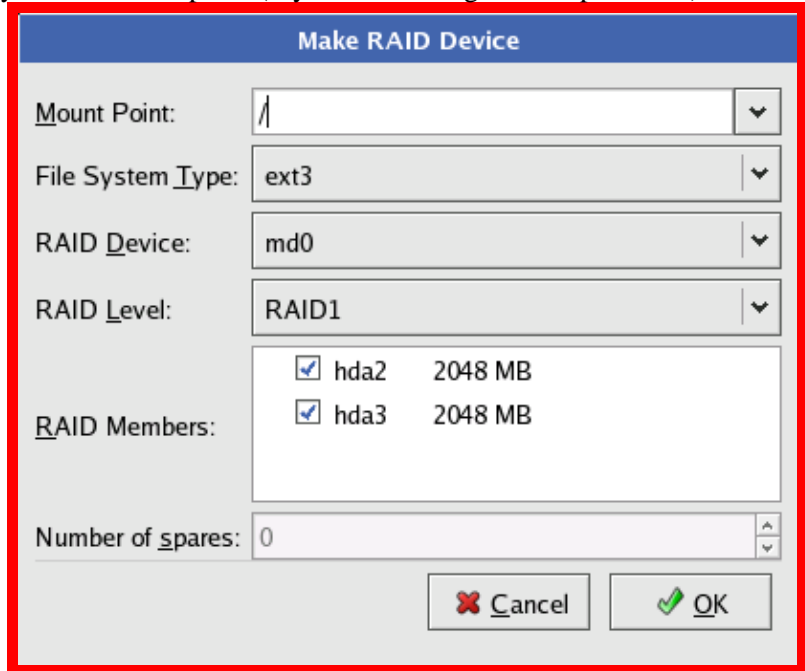
8. You have created at least two software RAID partitions, and now you can assimilate those into a RAID. Instead of creating another partition, *Create a RAID device*.



A Linux Software RAID Tutorial

9. This RAID device will appear as any other filesystem device. Here, you choose the RAID's properties.

- ◆ Choose the mount point. (/ if you want to install to RAID)
- ◆ Choose the file system type. (usually ext3 on RHEL/Fedora)
- ◆ Choose the name of the RAID device. (usually increments from /dev/md0)
- ◆ Choose your RAID level. Consult the documentation above.
- ◆ Choose each of the partitions you want to add to the RAID.
- ◆ You may also indicate spares (if you have enough RAID partitions).



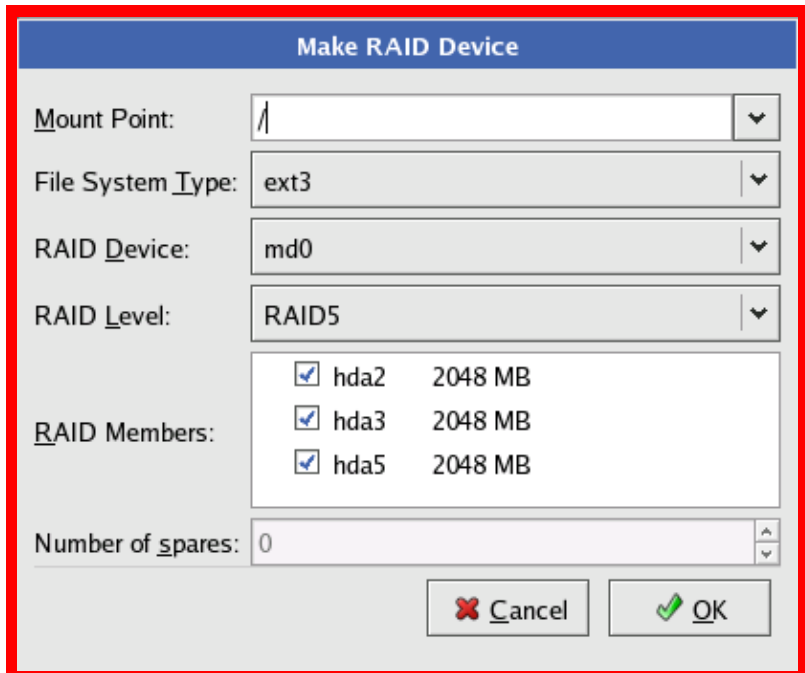
The screenshot shows the 'Make RAID Device' dialog box with the following settings:

- Mount Point: /
- File System Type: ext3
- RAID Device: md0
- RAID Level: RAID1
- RAID Members:

<input checked="" type="checkbox"/>	hda2	2048 MB
<input checked="" type="checkbox"/>	hda3	2048 MB
- Number of spares: 0

Buttons: Cancel, OK

... or ...



The screenshot shows the 'Make RAID Device' dialog box with the following settings:

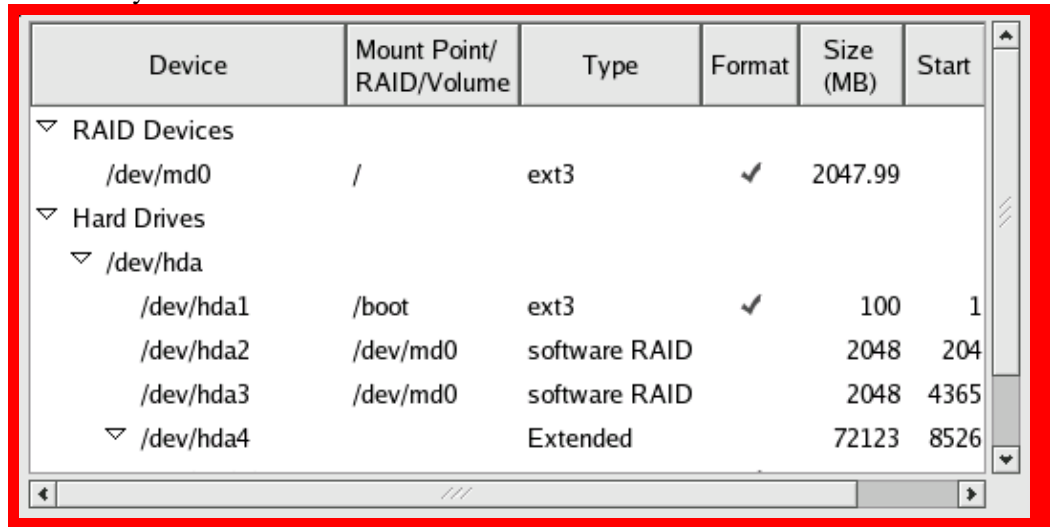
- Mount Point: /
- File System Type: ext3
- RAID Device: md0
- RAID Level: RAID5
- RAID Members:

<input checked="" type="checkbox"/>	hda2	2048 MB
<input checked="" type="checkbox"/>	hda3	2048 MB
<input checked="" type="checkbox"/>	hda5	2048 MB
- Number of spares: 0

Buttons: Cancel, OK

A Linux Software RAID Tutorial

10. You should now see a new device, /dev/md0, in your partition table. Repeating the process above, you can create any number of RAID devices.



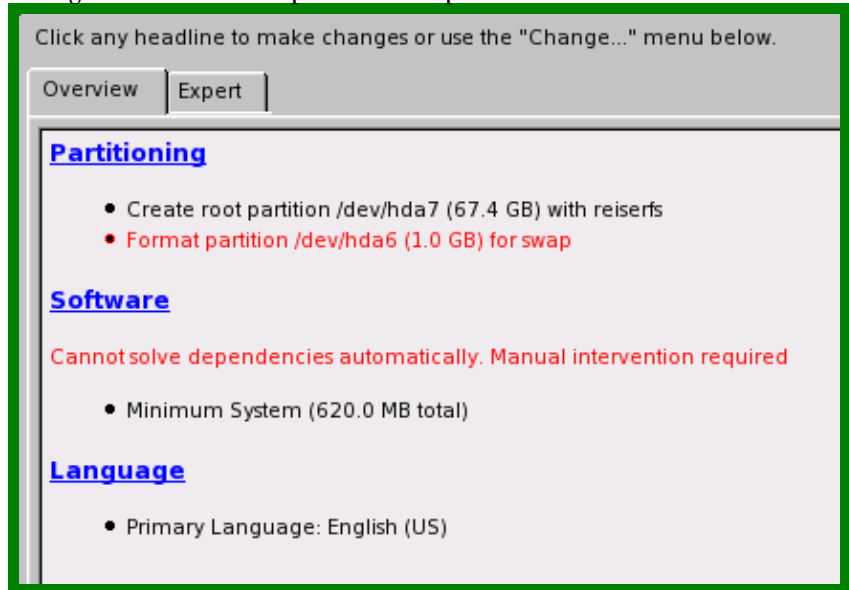
The screenshot shows a partition table with the following data:

Device	Mount Point/ RAID/Volume	Type	Format	Size (MB)	Start
▼ RAID Devices					
/dev/md0	/	ext3	✓	2047.99	
▼ Hard Drives					
▼ /dev/hda					
/dev/hda1	/boot	ext3	✓	100	1
/dev/hda2	/dev/md0	software RAID		2048	204
/dev/hda3	/dev/md0	software RAID		2048	4365
▼ /dev/hda4		Extended		72123	8526

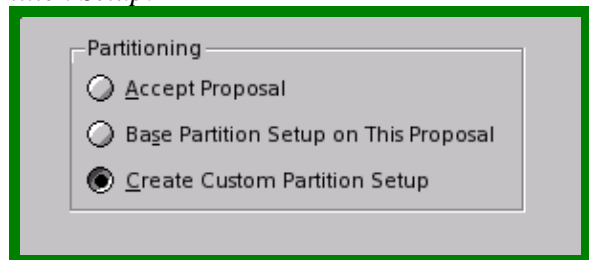
You may now proceed through the rest of the normal RHEL/Fedora installation process.

The SLES/openSUSE graphical process

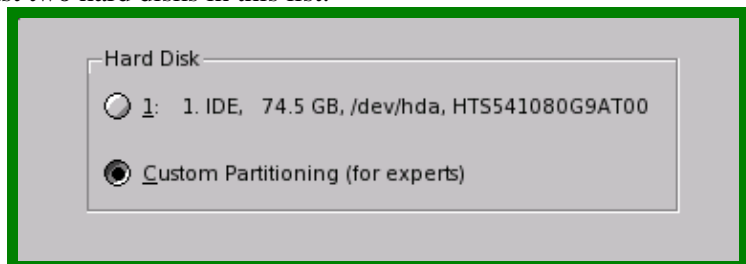
1. Click on *Partitioning* to customize the partition setup.



2. Select *Create Custom Partition Setup*.

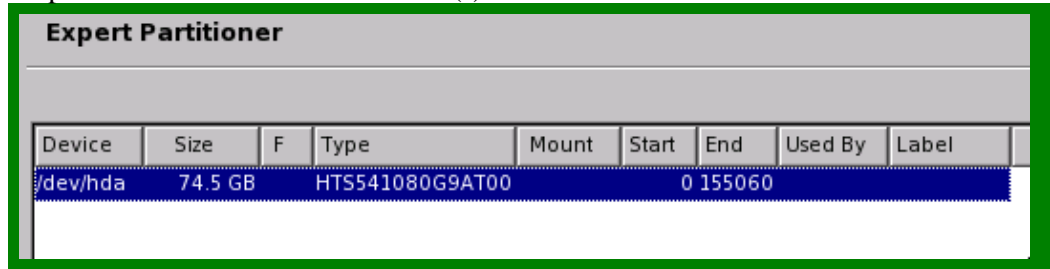


3. Again, choose *Custom Partitioning (for experts)*. Note: In most real RAID configurations, you should probably see at least two hard disks in this list.

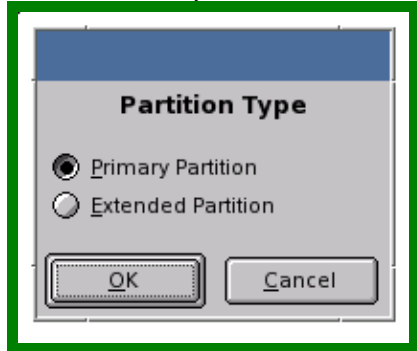


A Linux Software RAID Tutorial

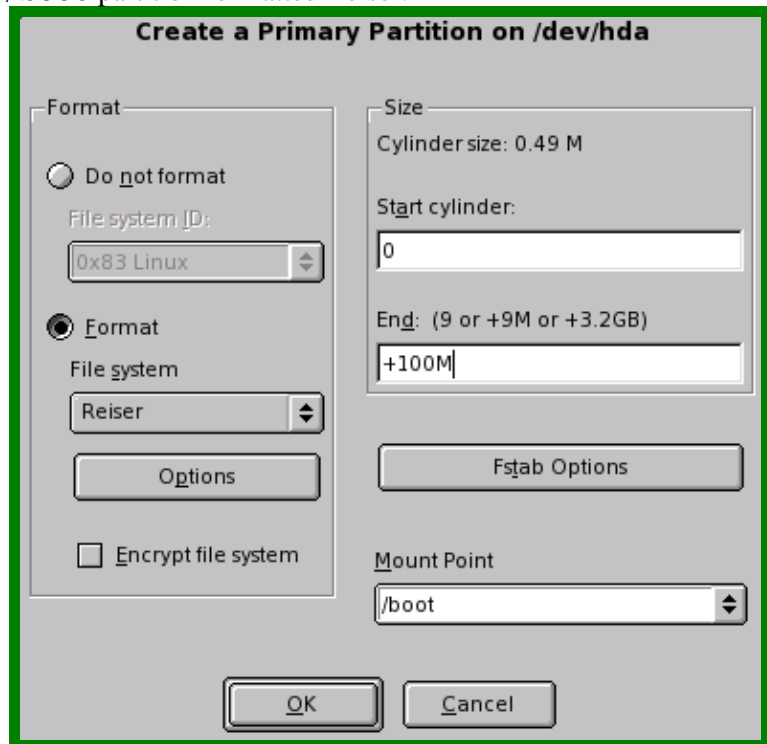
4. Delete all partitions and start with clean disk(s).



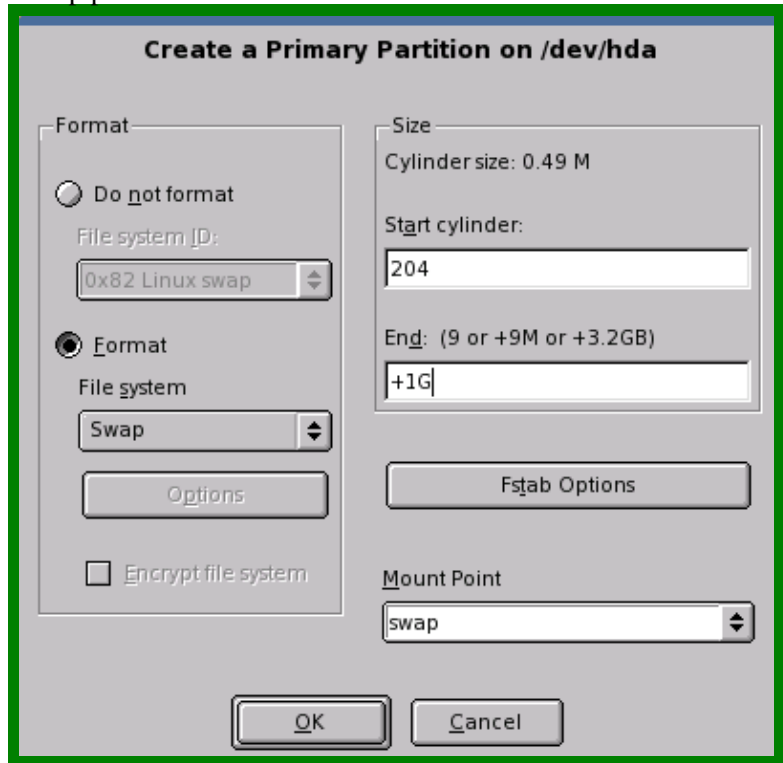
5. The first few partitions you create will be *Primary Partitions*.



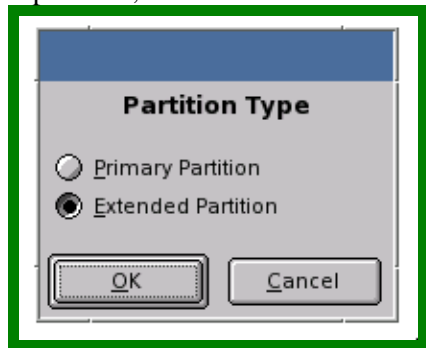
6. Create a 100 MB /boot partition formatted Reiser.



7. Create a 1024 MB swap partition.

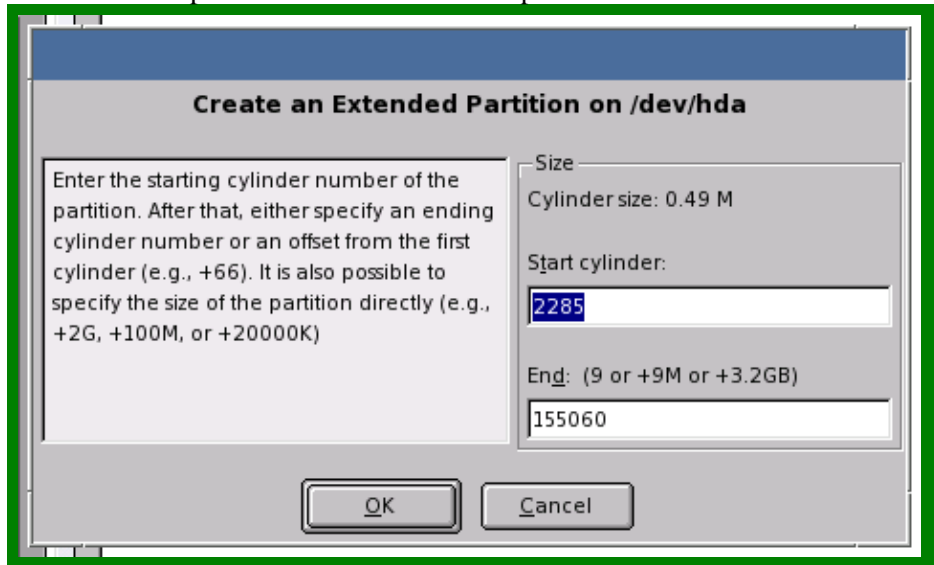


8. You can only have four primary partitions on a hard drive. Although the RHEL/Fedora installer will automatically create extended partitions as needed, in openSUSE this must be done manually. When creating either your third or fourth partition, choose *Extended*.

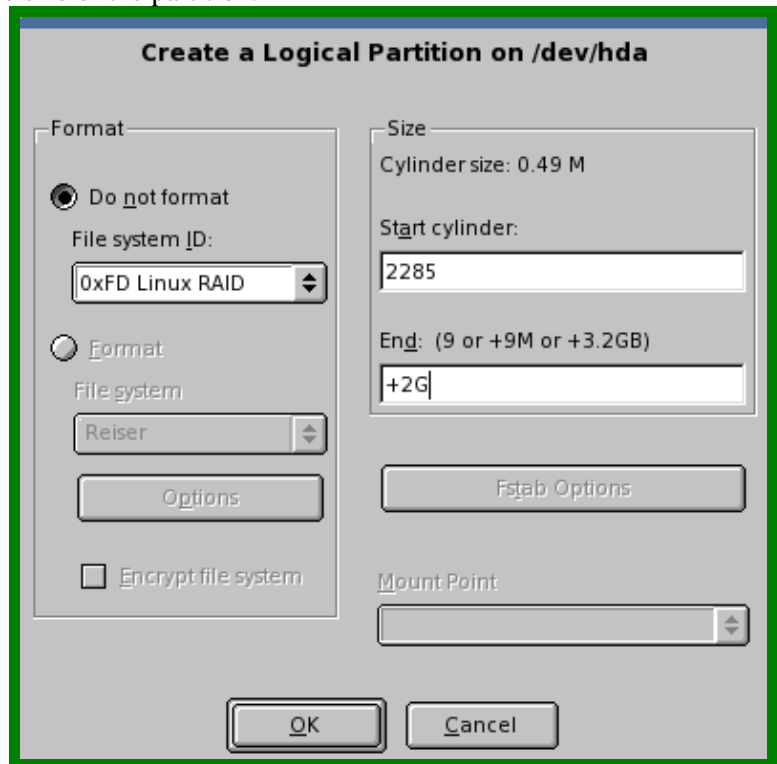


A Linux Software RAID Tutorial

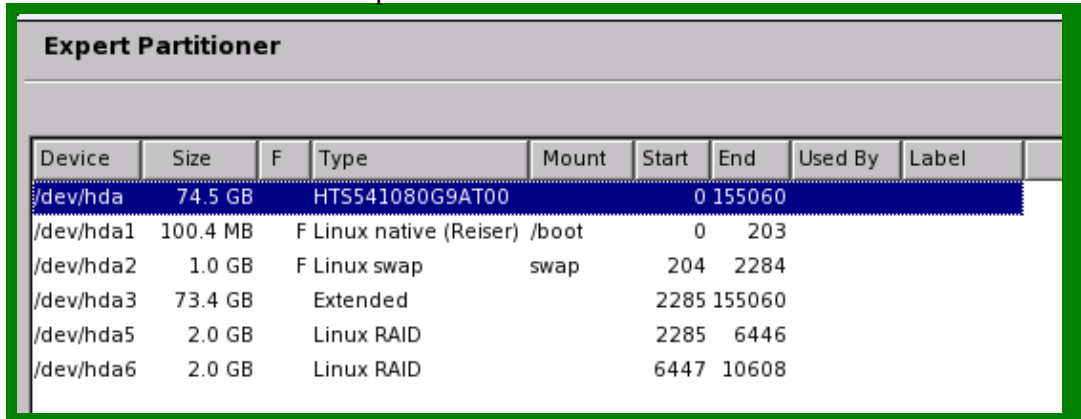
9. Give all of the remaining space on the hard drive to this extended partition. You will carve the rest of your partitions out of this space as additional extended partitions.



10. Create as many individual RAID partitions as required. Remember the requirements of each of the different RAID levels discussed above.
 - ◆ Select *Do not format*.
 - ◆ Set the *File system ID* to *0xFD Linux RAID*.
 - ◆ Choose the physical drive on which the partition will be created.
 - ◆ Define the size of the partition.



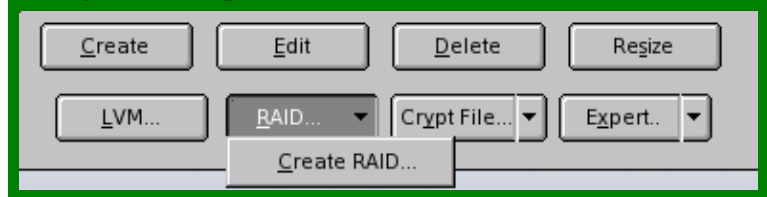
11. You should see:
- ◆ A Reiser/boot partition.
 - ◆ A swap partition.
 - ◆ At least two software RAID partitions.



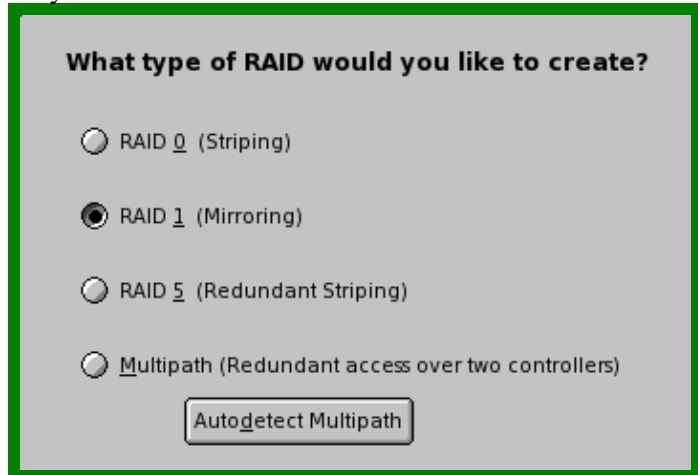
The Expert Partitioner window displays a table of disk partitions. The first row, representing the entire disk, is highlighted in blue. The table columns are Device, Size, F, Type, Mount, Start, End, Used By, and Label.

Device	Size	F	Type	Mount	Start	End	Used By	Label
/dev/hda	74.5 GB		HTS541080G9AT00		0	155060		
/dev/hda1	100.4 MB	F	Linux native (Reiser)	/boot	0	203		
/dev/hda2	1.0 GB	F	Linux swap	swap	204	2284		
/dev/hda3	73.4 GB		Extended		2285	155060		
/dev/hda5	2.0 GB		Linux RAID		2285	6446		
/dev/hda6	2.0 GB		Linux RAID		6447	10608		

12. To create the RAID using the RAID partitions, use the *RAID*→*Create RAID* button.

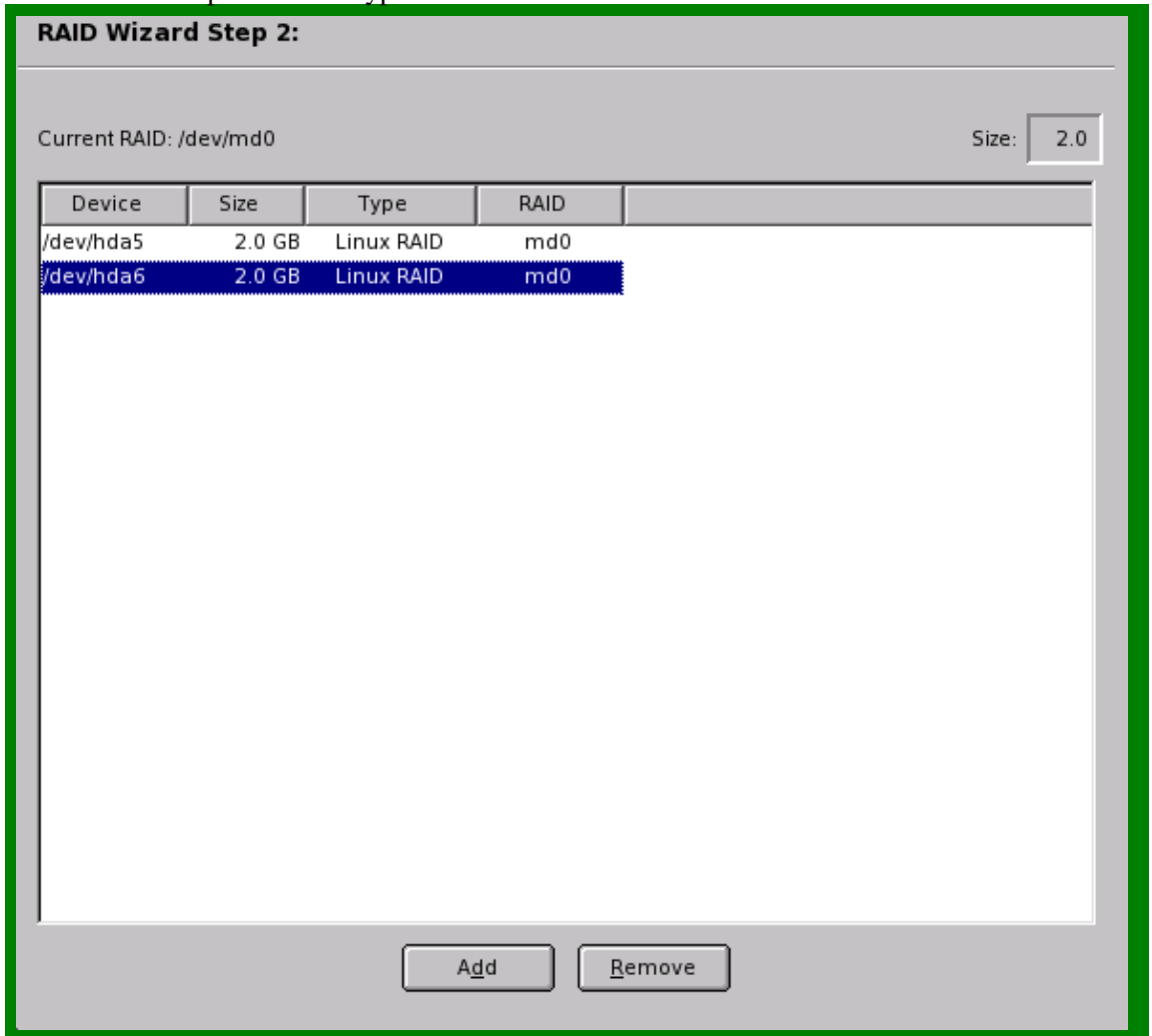


13. Select the type of RAID you want to create.



A Linux Software RAID Tutorial

14. Choose each of the partitions of type RAID and click "Add" to construct the RAID.



A Linux Software RAID Tutorial

15. This RAID device will appear as any other filesystem device. Here, you choose the RAID's properties.

- ◆ Choose how the filesystem should be formatted. The default filesystem in SUSE is usually Reiser.
- ◆ Choose the mount point.

The screenshot shows the 'RAID Wizard Step 3' window. It contains several configuration options:

- Format:** A sub-window with radio buttons for 'Do not format' and 'Format' (selected). Below it is a 'File system' dropdown menu set to 'Reiser' and an 'Options' button.
- Encrypt file system:** A checkbox that is currently unchecked.
- RAID Type:** A dropdown menu set to 'raid1'.
- Chunk size in KB:** A dropdown menu set to '4'.
- Parity algorithm (only for RAID 5):** A dropdown menu set to 'left-asymmetric'.
- Fstab Options:** A button.
- Mount Point:** A dropdown menu set to '/'.

16. You should now see a new device, `/dev/md0`, in your partition table. Using the process above, you can create any number of RAID devices.

The screenshot shows the 'Expert Partitioner' window with a table of partitions. The table has columns for Device, Size, F, Type, Mount, Start, End, Used By, and Label.

Device	Size	F	Type	Mount	Start	End	Used By	Label
/dev/hda	74.5 GB		HTS541080G9AT00		0	155060		
/dev/hda1	100.4 MB	F	Linux native (Reiser)	/boot	0	203		
/dev/hda2	1.0 GB	F	Linux swap	swap	204	2284		
/dev/hda3	73.4 GB		Extended		2285	155060		
/dev/hda5	2.0 GB		Linux RAID		2285	6446	md0	
/dev/hda6	2.0 GB		Linux RAID		6447	10608	md0	
/dev/md0	2.0 GB	F	MD Raid	/	-	-		

You may now proceed through the rest of the normal SLES/openSUSE installation process.

Creating a Software RAID on an Existing Linux System

Although some users are equipped to create a software RAID at the time of installation, it is also both possible and practical to construct a software RAID on a running Linux system. This usually happens when a user adds new storage to an existing system. Sometimes this is possible using hot swappable hard drives (which means that the hard drive(s) can be added without powering the system down). Other times, the system must be powered down and the drive(s) physically installed. Either way, as soon as the Linux kernel is able to see the hard disk device(s), you can partition and create the RAID.

Userspace utilities

`raidtools`

Legacy Linux systems use a suite of utilities collectively known as `raidtools` to manage software RAIDs. This tutorial does not cover `raidtools` as both RHEL/Fedora and SLES/openSUSE distributions use the `mdadm` suite in the latest releases. You will need to seek additional documentation on `raidtools` if you wish to manage software RAIDs using this suite.

`mdadm`

Current Linux distributions contain the `mdadm` package to handle the management of software RAIDs in user space. The `mdadm` manpage [6] contains a far more complete discussion of all of its features. The uses of `mdadm` demonstrated below represent a handy subset and are enough to demonstrate some basic administration of software RAIDs.

RAID creation on the command line

The following example creates a software RAID1 out of two USB flash cards, located at `/dev/sda` and `/dev/sdd`.

You can either use `fdisk` to manually delete each partition, or you can simply zero out the partition table. *n.b.*, This will destroy all data on these devices!

```
# dd if=/dev/zero of=/dev/sda bs=1024 count=1024
1024+0 records in
1024+0 records out
# dd if=/dev/zero of=/dev/sdd bs=1024 count=1024
1024+0 records in
1024+0 records out
```

A Linux Software RAID Tutorial

Display all partitioning information for each of the drives before you begin.

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

Disk /dev/sda: 131 MB, 131072000 bytes
5 heads, 50 sectors/track, 1024 cylinders
Units = cylinders of 250 * 512 = 128000 bytes

Disk /dev/sda doesn't contain a valid partition table

Disk /dev/sdd: 131 MB, 131072000 bytes
5 heads, 50 sectors/track, 1024 cylinders
Units = cylinders of 250 * 512 = 128000 bytes

Disk /dev/sdd doesn't contain a valid partition table
```

Create the software RAID partition on each of the disks. Use the same process for each of the media.

```
# fdisk /dev/sda
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF disklabel
Building a new DOS disklabel. Changes will remain in memory only,
until you decide to write them. After that, of course, the previous
content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)

Command (m for help): n
Command action
  e   extended
  p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-1024, default 1): 1
Last cylinder or +size or +sizeM or +sizeK (1-1024, default 1024):
Using default value 1024

Command (m for help): t
Selected partition 1
Hex code (type L to list codes): fd
Changed system type of partition 1 to fd (Linux raid autodetect)

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
```

A Linux Software RAID Tutorial

Now, the partition tables of each media should each have a Linux RAID partition:

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

Disk /dev/sda: 131 MB, 131072000 bytes
5 heads, 50 sectors/track, 1024 cylinders
Units = cylinders of 250 * 512 = 128000 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1                1         1024      127975   fd  Linux raid autodetect

Disk /dev/sdd: 131 MB, 131072000 bytes
5 heads, 50 sectors/track, 1024 cylinders
Units = cylinders of 250 * 512 = 128000 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sdd1                1         1024      127975   fd  Linux raid autodetect
```

You may need to reboot for the kernel to re-read the updated partition tables. In this case, the Linux kernel does not require a reboot to re-read the partition tables of the USB media.

Add the disks to a new RAID device. *Note:* If you already have RAID(s) on your system, you should increment `/dev/md0` to the next available block device. This is also the point at which you define the RAID level and the number of devices in the array.

```
# mdadm --create /dev/md0 --level=1 --raid-devices=2 /dev/sda1 /dev/sdd1
mdadm: array /dev/md0 started.
```

Format the RAID with the filesystem of your choice.

```
# mkfs.ext3 /dev/md0
mke2fs 1.35 (28-Feb-2004)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
32000 inodes, 127872 blocks
6393 blocks (5.00%) reserved for the super user
First data block=1
Maximum filesystem blocks=67371008
16 block groups
8192 blocks per group, 8192 fragments per group
2000 inodes per group
Superblock backups stored on blocks:
    8193, 24577, 40961, 57345, 73729

Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 25 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```


A Linux Software RAID Tutorial

Mount the RAID device and check to see that the mount worked.

```
# mount /dev/md0 /mnt/
# mount
/dev/hda2 on / type ext3 (rw)
none on /proc type proc (rw)
none on /sys type sysfs (rw)
none on /dev/pts type devpts (rw,gid=5,mode=620)
usbfs on /proc/bus/usb type usbfs (rw)
/dev/hdal on /boot type ext3 (rw)
none on /dev/shm type tmpfs (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
sunrpc on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw)
/dev/md0 on /mnt type ext3 (rw)
```

Test reading from, and writing data to the mounted RAID filesystem.

```
# echo "Howdy, RAID :)" > /mnt/test.txt
# cat /mnt/test.txt
Howdy, RAID :)
```

Update `/etc/mdadm.conf` so that your RAID is started automatically with the rest of your filesystems at boot time.

```
# echo "DEVICE /dev/sda1 /dev/sdd1" > /etc/mdadm.conf
# mdadm --detail --scan | tee -a /etc/mdadm.conf
ARRAY /dev/md0 level=raid1 num-devices=2 UUID=b49c7f86:3898f1c0:e5478f50:aa589542
    devices=/dev/sda1,/dev/sdd1
```

Optional: If RAID1, you can verify that the data is in fact being mirrored. Stop and unmount the RAID, and mount (read-only) each of the component partitions and see that the same data is in place.

```
# mdadm --stop /dev/md0
# mount -o ro /dev/sda1 /mnt
# ls /mnt/
lost+found test.txt
# cat /mnt/test.txt
Howdy, RAID :)
# umount /mnt/
# mount -o ro /dev/sdd1 /mnt
# ls /mnt/
lost+found test.txt
# cat /mnt/test.txt
Howdy, RAID :)
# umount /mnt
```

To restart and remount the RAID:

```
# mdadm --assemble /dev/md0
# mount /dev/md0 /mnt
```

Losing a RAID Component

Although software RAIDs have performance advantages, most users create RAIDs for redundant protection of critical data. This chapter demonstrates how to simulate the loss or destruction of a disk to test the data protection aspect of your RAID. In the following section, you will learn how to restore data from the mirrored disk or parity information.

Physically removed disk

It is quite easy to simulate a damaged disk on systems where the hard drive is easily removable: simply remove or disconnect the disk. The above example with the USB media was designed with this feature in mind—it is very easy to simply remove one of the cards to simulate a faulty disk.

Corrupted or damaged disk

Other examples of this tutorial involve creating software RAIDs using multiple partitions on the same disk. In this case, it is not possible to physically remove the disk to test the operation of the RAID. Instead, you can overwrite one of the RAID partitions with zeros (or random data) to simulate a disk crash, and then reformat that partition and restore the RAID onto it.

Simulating the loss

Before simulating a damaged disk, check the status of the RAID:

```
# cat /proc/mdstat
Personalities : [raid1]
md0 : active raid1 sda1[0] sdd1[1]
      127872 blocks [2/2] [UU]
# mdadm --detail /dev/md0
/dev/md0:
   Version : 00.90.01
  Creation Time : Mon Feb 13 21:09:43 2006
    Raid Level : raid1
    Array Size : 127872 (124.88 MiB 130.94 MB)
   Device Size : 127872 (124.88 MiB 130.94 MB)
    Raid Devices : 2
   Total Devices : 2
 Preferred Minor : 0
  Persistence : Superblock is persistent

   Update Time : Mon Feb 13 21:40:29 2006
         State : clean
   Active Devices : 2
 Working Devices : 2
  Failed Devices : 0
   Spare Devices : 0

   Number   Major   Minor   RaidDevice State
    0         8       1         0     active sync   /dev/sda1
    1         8       49        1     active sync   /dev/sdd1
   UUID : b49c7f86:3898f1c0:e5478f50:aa589542
   Events : 0.34
```

A Linux Software RAID Tutorial

Remove one of the USB media.

Alternatively, you could use `dd` to overwrite one of the RAID partitions with `/dev/zero` or `/dev/random`. *Note*: `dd` should always be used with caution.

Check that you can continue to read/write to/from the RAID.

```
# echo "new data" >> /mnt/test.txt
# cat /mnt/test.txt
Howdy, RAID :)
new data
```

Now check your RAID status. Notice that one of the devices has been marked "faulty" and the removal is detected. Your RAID will continue operating, though in an unprotected manner, until the faulty media is replaced.

```
# mdadm --detail /dev/md0
/dev/md0:
    Version : 00.90.01
  Creation Time : Mon Feb 13 21:09:43 2006
    Raid Level : raid1
    Array Size : 127872 (124.88 MiB 130.94 MB)
    Device Size : 127872 (124.88 MiB 130.94 MB)
    Raid Devices : 2
    Total Devices : 2
Preferred Minor : 0
    Persistence : Superblock is persistent

    Update Time : Mon Feb 13 21:44:14 2006
      State : clean, degraded
Active Devices : 1
Working Devices : 1
Failed Devices : 1
Spare Devices : 0

   Number   Major   Minor   RaidDevice State
     0         8         1         0     active sync   /dev/sda1
     1         0         0        -1     removed
     2         8        49        -1     faulty   /dev/sdd1
    UUID : b49c7f86:3898f1c0:e5478f50:aa589542
    Events : 0.41
# cat /proc/mdstat
Personalities : [raid1]
md0 : active raid1 sda1[0] sdd1[2](F)
      127872 blocks [2/1] [U_]

```

Recovering and Restoring a RAID

Hopefully, you will never need to execute the instructions in this section. However, most any computer user has experienced a hard disk failure. If you go through the trouble to create a software RAID, it is only prudent to understand how to recover your data and restore your RAID after an unfortunate disk failure. The previous section demonstrated a couple of ways to simulate a damaged or lost disk. This section assumes that one of the disks in a RAID has been destroyed in some way.

RAID recovery on the command line

This example assumes that you have created a RAID1 of two USB media (`/dev/sda` and `/dev/sdd`) and removed one of them (`/dev/sdd`) to simulate the loss of a disk (as discussed in the previous two sections).

Insert the new media (which might be the same media you previously removed). Delete the partition table and repartition the media as previously instructed above to simulate a completely new and fresh replacement media.

```
# dd if=/dev/zero of=/dev/sdd bs=1024 count=1024
1024+0 records in
1024+0 records out
# fdisk /dev/sdd
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF disklabel
Building a new DOS disklabel. Changes will remain in memory only,
until you decide to write them. After that, of course, the previous
content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-1024, default 1): 1
Last cylinder or +size or +sizeM or +sizeK (1-1024, default 1024):
Using default value 1024

Command (m for help): t
Selected partition 1
Hex code (type L to list codes): fd
Changed system type of partition 1 to fd (Linux raid autodetect)

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
```

To re-establish the RAID, first unmount and stop the RAID.

```
# umount /mnt
# mdadm --stop /dev/md0
```

A Linux Software RAID Tutorial

Next, restart the RAID. It will be started with one out of two devices.

```
# mdadm --assemble /dev/md0
mdadm: /dev/md0 has been started with 1 drive (out of 2).
# mdadm --detail /dev/md0
/dev/md0:
    Version : 00.90.03
  Creation Time : Wed Feb 15 15:17:39 2006
    Raid Level : raid1
    Array Size : 127872 (124.88 MiB 130.94 MB)
    Device Size : 127872 (124.88 MiB 130.94 MB)
    Raid Devices : 2
    Total Devices : 1
Preferred Minor : 0
    Persistence : Superblock is persistent

    Update Time : Wed Feb 15 15:29:35 2006
      State : clean, degraded
Active Devices : 1
Working Devices : 1
Failed Devices : 0
Spare Devices : 0

    UUID : d3b24ecb:48835afd:655f74bc:1d6ada33
    Events : 0.35

    Number   Major   Minor   RaidDevice State
     0         8       33         0   active sync  /dev/sdc1
     1         0         0         -   removed
```

A Linux Software RAID Tutorial

Now add the new replacement partition to the RAID. It will be resynchronized to the original partition.

```
# mdadm --add /dev/md0 /dev/sdd1
mdadm: hot added /dev/sdd1
# mdadm --detail /dev/md0
/dev/md0:
    Version : 00.90.03
    Creation Time : Wed Feb 15 15:17:39 2006
    Raid Level : raid1
    Array Size : 127872 (124.88 MiB 130.94 MB)
    Device Size : 127872 (124.88 MiB 130.94 MB)
    Raid Devices : 2
    Total Devices : 2
Preferred Minor : 0
    Persistence : Superblock is persistent

    Update Time : Wed Feb 15 15:31:39 2006
    State : clean, degraded, recovering
    Active Devices : 1
    Working Devices : 2
    Failed Devices : 0
    Spare Devices : 1

    Rebuild Status : 31% complete

    UUID : d3b24ecb:48835afd:655f74bc:1d6ada33
    Events : 0.36

    Number   Major   Minor   RaidDevice State
     0         8       33         0   active sync   /dev/sdc1
     1         0         0         -   removed
     2         8       49         1   spare rebuilding /dev/sdd1
# cat /proc/mdstat
Personalities : [raid1]
md0 : active raid1 sdd1[2] sdc1[0]
      127872 blocks [2/1] [U_]
      [=====>.....] recovery = 34.4% (44608/127872) finish=2.9min speed=461K/sec

unused devices:
```

A Linux Software RAID Tutorial

Depending on filesystem size and bus speed, the synchronization could take some time. When the resync completes, your RAID is fully restored.

```
# mdadm --detail /dev/md0
/dev/md0:
  Version : 00.90.03
  Creation Time : Wed Feb 15 15:17:39 2006
  Raid Level : raid1
  Array Size : 127872 (124.88 MiB 130.94 MB)
  Device Size : 127872 (124.88 MiB 130.94 MB)
  Raid Devices : 2
  Total Devices : 2
  Preferred Minor : 0
  Persistence : Superblock is persistent

  Update Time : Wed Feb 15 15:36:25 2006
  State : clean
  Active Devices : 2
  Working Devices : 2
  Failed Devices : 0
  Spare Devices : 0

  UUID : d3b24ecb:48835afd:655f74bc:1d6ada33
  Events : 0.37

  Number   Major   Minor   RaidDevice State
     0         8       33         0   active sync   /dev/sdc1
     1         8       49         1   active sync   /dev/sdd1
```

Appendices

The following appendices are not directly related to Linux software RAID. However, your instructor uses several utilities and services that help ensure that the demonstrations operate efficiently and smoothly; primarily optimizing the RAID installations to perform rapid network installs. In the spirit and vein of open source solution development, these practices are briefly described and interested readers are directed to references with more complete discussions [[1,2,7,8](#)].

Appendix A

A network installable tree

Depending on your network bandwidth, it can be considerably faster to install Linux over a network than using multiple CDROMs. The installation proceeds seamlessly without requiring the user to switch between multiple install media. Furthermore, 10/100Mbps ethernet is faster than almost any CDROM bus transfer rate.

Red Hat® and SUSE install trees are created identically, by copying the contents of all install media into a single directory. Note that it helps to copy the disk contents in reverse order, such that a couple of key files with the same name are finally written with the contents from disk one.

Create a directory for the installable tree:

```
# mkdir -p /install/Fedora
```

If you have physical copies of the media, mount each of the disks in this way:

```
# mount /dev/cdrom /media/cdrom
```

If you have downloaded ISO images, you can mount each of the disks in this way:

```
# mount -o loop /tmp/Fedora-disc4.iso /media/cdrom
```

Copy the contents of each disk (starting with the last disk moving to the first) using rsync, and then unmount:

```
# rsync -av /media/cdrom/ /install/Fedora/  
# umount /media/cdrom
```

You now have a network installable tree.

Appendix B

Minimally configured NFS server

The client you want to install must be able to access the installable tree you created. Red Hat and SUSE support installations over HTTP, FTP, and NFS. Any of these could be easily configured, though NFS is used for these demonstrations.

You need to add the directory containing the install trees to `/etc/exports` and restart the `portmap` and `nfs` daemons:

```
# echo "/install *(ro)" > /etc/exports
# /etc/init.d/portmap restart
# /etc/init.d/nfs restart
# /etc/init.d/iptables stop
```

Note: These instructions also turn off your firewall by stopping `iptables`. Generally, this is not a safe NFS configuration. However, these demonstrations are taking place on a two-computer network, and NFS requires some rather complicated firewall modifications due to the number of ports it uses. There are other references available that describe more secure NFS administration.

Your installable tree is now exported over NFS to any machine on your network.

Appendix C

Minimally configured DHCP server

The installation target machine in these demonstrations uses DHCP to dynamically configure its network connection. To accommodate the client, the server must have `dhcpcd` configured and running.

Configure `dhcpcd` in `/etc/dhcpcd.conf`:

```
# cat /etc/dhcpcd.conf
subnet 192.168.0.0 netmask 255.255.255.0 {
    range 192.168.0.100 192.168.0.200;
    option domain-name-servers 192.168.0.1;
    option domain-name "local";
    host raidtest {
        hardware ethernet 00:0a:63:a0:98:13;
        fixed-address 192.168.0.200;
    }
}
ddns-update-style ad-hoc;
```

This very basic DHCP server configuration specifies that:

- The range of IP addresses from `192.168.0.100` – `192.168.0.200` will be assigned by the server to requesting clients.
- The server will also tell requesting clients that the DNS server for this network is `192.168.0.1` (which also happens to be the address of the DHCP and NFS server).
- And furthermore, if the machine with hardware MAC address of `00:0a:63:a0:98:13` requests an IP, always give it `192.168.0.200`.

Once the configuration file is updated, you can start the DHCP server with:

```
# /etc/init.d/dhcpcd start
```

You now have a DHCP server running and clients attached to the same network as this server can connect and dynamically receive their network configurations.

Appendix D

Minimally configured DNS server

DNS is a surprisingly complicated service to configure. The reasons for running in these demonstrations are rather inane, as no domain name resolution in these installation processes is actually required, except for the reverse name lookup of the client. However, both distribution installation processes take an exceedingly long time searching for valid domain name servers before timing out. Thus, your instructor is running a bare bones, crippled DNS server simply to speed along these network installations.

These instructions will create both forward and reverse name resolution records for a host named "raidtest" with IP address 192.168.0.200.

Configure forward name resolution by adding the following zone to `/etc/named.conf`:

```
zone "0.168.192.in-addr.arpa" IN {
    type master;
    file "named.raidtest";
    allow-update { none; };
};
```

Configure reverse name resolution by creating the file `/var/named/named.raidtest`:

```
$TTL      86400
@         IN      SOA     localhost. root.localhost. (
                                1997022700 ; Serial
                                28800      ; Refresh
                                14400      ; Retry
                                3600000    ; Expire
                                86400 )    ; Minimum

         IN      NS      localhost.
1       IN      PTR     localhost.
200    IN      PTR     raidtest.
```

Start DNS:

```
/etc/init.d/named start
```

You are now running just enough of a DNS server to keep these Linux network installations from hanging while looking for a valid DNS server.

Appendix E

Simple peer-to-peer TCP/IP Linux networking

These demonstrations involve connecting two computers: one client being installed, and one server running NFS, DNS, and DHCP. These simple instructions show how to establish a peer-to-peer TCP/IP network connection using Linux.

Connect the machines using either a crossover networking cable, or a hub/switch.

On the server machine, configure its ethernet interface to use a static IP address:

```
# ifdown eth0  
# ifconfig eth0 192.168.0.1
```

The client machine's installation process will be able to automatically obtain a DHCP address from the server.

```
# ifconfig eth0  
# dhclient eth0
```

Alternatively, if you wanted to connect the client machine to the server using a static IP, set the address and gateway as follows:

```
# ifconfig eth0 192.168.0.100  
# route add default gw 192.168.0.1
```

You now have a TCP/IP connection established between these two computers.

Appendix F

Installation initiation from USB media

Your instructor performs the demo installations on an IBM Thinkpad whose CDROM drive has been replaced by a second hard drive, thus booting from an installable CDROM image is impossible. Instead, he uses USB media which are able to boot the machine and begin the installation.

Note: Your system must be able to boot from USB media (most late models can) and this feature must be enabled in your BIOS.

RHEL/Fedora

RHEL and Fedora ship with just such a bootable image. Look for a file called `images/diskboot.img` on the first disk of an installation set.

Note: Be careful with the disk dump (`dd`) utility. It overwrites data on the destination (`of=...`) without confirming. This is a dangerously easy way to fry a disk if your destination is incorrect!

Use `dd` to write this image to the media from which you want to install:

```
# mount /dev/cdrom /media/cdrom
# dd if=/media/cdrom/images/diskboot.img of=/dev/sda
# umount /media/cdrom
```

Insert the USB device and boot the system to be installed. At the grub boot prompt, enter:

```
# linux askmethod
```

- Navigate the installation prompts until you are asked for the location of the install media.
- Choose "NFS".
- Have your network automatically detected by selecting *Use dynamic IP configuration (BOOTP/DHCP)*.
- Enter the NFS server as `192.168.0.1`.
- Enter the mount point, `/install/RHEL` or wherever you may have put it.

SLES/openSUSE

SLES9 ships with images for three floppy disks, from which one can boot and begin a network installation. Loading 4.5 MB of data over a floppy bus, however, takes some time. It is far more efficient to boot from a USB device, whose contents can be read much faster. Unfortunately, SLES9 does not ship with such an image precompiled. Fortunately, though, openSUSE contains a utility `mkbootimg` that is able to create such images and also is backward compatible with SLES9. The instructions below show how to use this utility to create bootable USB media to install either SLES or openSUSE.

A Linux Software RAID Tutorial

You will need to partition and format your USB media. Insert it and check `dmesg` to find its location. *Note:* Partitioning and formatting will delete all data!

```
# dmesg | tail
SCSI device sda: 256000 512-byte hdwr sectors (131 MB)
sda: Write Protect is off
sda: Mode Sense: 0b 00 00 08
sda: assuming drive cache: write through
sda: sda1
sd 0:0:0:0: Attached scsi removable disk sda
```

Remove the existing partition table by overwriting it with a megabyte of zero's:

```
# dd if=/dev/zero of=/dev/sda bs=1024 count=1024
1024+0 records in
1024+0 records out
```

Partition the disk with one 32-bit File Allocation Table (FAT32) partition of size 12 MB:

```
# fdisk /dev/sda
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF disklabel
Building a new DOS disklabel. Changes will remain in memory only,
until you decide to write them. After that, of course, the previous
content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)

Command (m for help): n
Command action
  e   extended
  p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-1024, default 1):
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-1024, default 1024): +12M

Command (m for help): t
Selected partition 1
Hex code (type L to list codes): b
Changed system type of partition 1 to b (W95 FAT32)

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: If you have created or modified any DOS 6.x
partitions, please see the fdisk manual page for additional
information.
```

Now format the partition as a FAT16 filesystem:

```
# mkdosfs -F 16 /dev/sda1
mkdosfs 2.10 (22 Sep 2003)
```

A Linux Software RAID Tutorial

Next, you need to obtain the `boot/mkbootdisk` utility from a recent openSUSE first disk:

```
# mount /dev/cdrom /media/cdrom
# cp /media/cdrom/boot/mkbootdisk /tmp
# umount /media/cdrom
```

Insert the first disk of the SUSE distribution you will be installing (SLES or openSUSE) and use `mkbootdisk` to install the bootable image to your USB media:

```
# mount /dev/cdrom /media/cdrom
# /tmp/mkbootdisk --32 --partition /dev/sda /media/cdrom
# umount /media/cdrom
```

Insert the USB device and boot the system to be installed. At the boot prompt, enter:

```
# linux install=nfs://192.168.0.1/install/SUSE
```

You can now begin the installation.

References

1. DHCP Howto
<http://www.troubleshooters.com/linux/dhcp.htm>
 2. DNS
<http://www.aboutdebian.com/dns.htm>
 3. Fedora
<http://fedora.redhat.com>
 4. IBM
<http://www.ibm.com>
 5. LVM Howto
<http://www.tldp.org/HOWTO/LVM-HOWTO/>
 6. mdadm Manpage
<http://man-wiki.net/index.php/8:mdadm>
 7. Network Linux Installations
<http://www.tldp.org/HOWTO/Network-Install-HOWTO.html>
 8. NFS Howto
<http://www.faqs.org/docs/Linux-HOWTO/NFS-HOWTO.html>
 9. Novell SUSE
<http://www.novell.com/linux/suse/>
 10. openSUSE
<http://en.opensuse.org/>
 11. Red Hat
<http://www.redhat.com>
 12. Software RAID Howto
<http://www.tldp.org/HOWTO/Software-RAID-HOWTO.html>
 13. Wikipedia: RAID
http://en.wikipedia.org/wiki/Redundant_array_of_independent_disks
-

About the Author

Dustin Kirkland is a software architect and Master Inventor in IBM's Linux Technology Center in Austin, Texas. He worked for the last few years as a security software developer, contributing to Common Criteria EAL2, EAL3, and EAL4 certifications of RHEL and SLES, developing Audit kernel and userspace features, extending SELinux capabilities into the Audit system, and working with static source code analysis tools to identify and fix vulnerabilities in various open source projects.

Dustin spent most of 2005 onsite at Red Hat's Westford, MA campus as a resident IBM LTC developer cooperating on several key feature adds and bug fixes for RHEL4u2 —one of which was improved support for software RAID installation on IBM's POWER5 platform. This involved modifications to both the Anaconda installer and the Yaboot bootloader. It was at this time, while testing these features, that Dustin became intimately familiar with creating, destroying, and restoring systems using software RAID and the motivation for this tutorial surfaced.

Legal Statement

This work represents the view of the author and does not necessarily represent the view of IBM.

IBM, IBM (logo), e-business (logo), pSeries, e (logo) server, and xSeries are trademarks or registered trademarks of International Business Machines Corporation in the United States and/or other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.
