Root RAID HOWTO cookbook

Table of Contents

Table of Contents

Root RAID HOWTO cookbook

Michael A. Robinton, michael@bzs.org

v1.13, July 17, 2000

---

*This document only applys to the **OLD** raidtools, versions 0.50 and under. The workarounds and solutions addressed in this write up have largely been made obsolete by the vast improvment in the 0.90 raidtools and accompanying kernel patch to the 2.0.37, 2.2x and 2.3x series kernels. You may find the detailed descriptions useful, particularly if you plan to run root raid or use initrd. Check these links for a reference to set up of Boot Root Raid using conventional LILO and accompanying initrd working scripts. What follows is the description of the now **OBSOLETE Root RAID HOWTO**. This document was originally written to provide a cookbook for creating a root mounted raid filesystem and companion fallback rescue system using linux initrd. There are complete step−by−step instruction for both raid1 and raid5 md0 devices. Each step is accompanied by an explanation of it's purpose. Included with this revision is a generic **linuxrc** initrd file which may be configured with a single three line /etc/raidboot.conf file for raid1 and raid5 configurations.*

---

4. *initrd* Cookbook for root mounted RAID

5. Configuring the Production RAID system.

6. Building the RAID file system.

7. One last thought.

8. Appendix A. – Bohumil Chalupa's md0 shutdown

9. Appendix B. – Sample SHUTDOWN scripts

10.Appendix C. − other setup files

11.Appendix D. − obsolete linuxrc and shutdown scripts

12.Appendix E. − Gadi's raid stop patch for the linux kernel

13.Appendix F. − rc.raidown

14.Appendix G. − linuxrc theory of operation

15.Appendix H. Setting up ROOT RAID on RedHat

1.Introduction

The reader is assumed to be familiar with the various types of raid implementations, their advantages and drawbacks. This is not a tutorial, just a set of instructions on how to implement root mounted raid on a linux system. All of the information necessary to become familiar with linux raid is listed here directly or by reference, please read it before send e−mail questions.

1.1 Where to get Up−to−date copies of this document.

Click here to browse the author's latest version of this document. Corrections and suggestions welcome!

Root−RAID−HOWTO −− OBSOLETE

Available in LaTeX (for DVI and PostScript), plain text, and HTML.

> http://www.linuxdoc.org/HOWTO/Root−RAID−HOWTO.html

Available in SGML and HTML.

> ftp.bizsystems.net/pub/raid/

1.2 More up−to−date Boot Root Raid with LILO howto

Available in LaTeX (for DVI and PostScript), plain text, and HTML.

> http://www.linuxdoc.org/HOWTO/Boot+Root+Raid+LILO.html

Available in SGML and HTML.

> ftp.bizsystems.net/pub/raid/

1.3 Bugs

As of this writing, the problem of stopping a root mounted RAID device has not yet been solved in a satisfactory way. A work−around proposed by Ed Welbon and implemented by Bohumil Chalupa is incorporated into this document which eliminates the need for a long ckraid at each boot for raid1 and raid5 devices. Without the workaround, it is necessary to **ckraid** the **md** device each time the system is re−booted. On a large array this can cause a severe availability performance degradation. On my 6 gig RAID1 device running on a Pentium 166 with 128 megs of ram, it takes well over half an hour to ckraid :−(after each re−boot. It takes over an hour on my 13 gig RAID5 array with a 20mb/sec scsi adaptor.

The workaround stores the status of the array at shutdown on the **real** boot device and compares it to a reference status placed there when the system is first built. If the status's match at reboot, the superblock on the array is rebuilt on the next boot, otherwise the operator is notified of the status error and the rescue system is left running with all the raid tools available.

Rebuilding the superblock causes the system to ignore that the array was powered down without mdstop by marking all the drives as **OK**, as if nothing happened. This only works if all the drives are OK at shutdown. If the array was operating with a bad drive, the operator must remove the bad drive prior to restarting the md device or the data can be corrupted.

None of this applies to raid0 which does not have to be mdstopped before shutdown.

Final proposed solutions to this problem include a **finalrd** similar to **initrd**, and **mdrootstop** which writes the **clean** flags to the array during shutdown when it is mounted read only. I am sure there are others.

In the mean time, the problem has been by−passed for now Please let me know when this problem is solved more cleanly!!!

1.4 Acknowledgements

The writings and e−mail from the following individuals helped to make this document possible. Many of the ideas were *stolen* from the helpful work of others, I have just tried to put it all in **COOKBOOK** form so that it is straightforward to use. My thanks to:

- Linas Vepstas *for the RAID howto that explained most of this to me.*
- Gadi Oxman *for answering my dumb 'newbie' questions.*
- Ed Welbon *for the execellent **initrd.md** package that inspired me to write this.*
- Bohumil Chalupa *for implementing the re−boot 'workaround' that allows***root−mounted−raid** *to work in a production environment.*
- Keith W. *for his explaination of setting up root raid with* **RedHat**.

- and many others who contributed to this work in one way or another.

1.5 Copyright Notice

2.What you need BEFORE YOU START

The packages you need and the documentation that answers the most common questions about setting up and running raid are listed below. Please review them throughly.

2.1 Required Packages

You need to obtain the most recent versions of these packages.

- a linux kernel that supports raid, initrd and /dev/loopx

  I used linux−2.0.33 from sunsite

* raid145−971022−2.0.31 patch adds support for raid1/4/5
* raidtools−pre3−0.42 tools to create and maintain raid devices (documentation too).
* Gadi's raid stop patch in Appendix E.
* linuxthreads−0.71 required threads package. Use ftp, browser doesn't work ftp.inria.fr/INRIA/Projects/cristal/Xavier.Leroy
* A Linux distribution, ready to install.

  I used Slackware−3.4

Helpful but not required

* raidboot−0.01.tar.gz pre−built raid rescue/boot system.

The detailed instructions in this document are based on the above packages. If the packages have been updated or you use a different linux distribution, you may have to modify the procedures you find here.

The patches, tool assortment, etc... may vary with 2.1 kernels. Please check the most recent documentation at:

    ftp.kernel.org/pub/linux/daemons/raid/

2.2 Other similar implementations.

I chose to include in the kernel all of the pieces necessary to run from boot without loading any modules. My kernel image is a little over 300k compressed.

Take a look at Ed Welbon's**initrd.md.tar.gz** for another way to make a bootable raid device. He uses loadable modules. A look at his concise scripts will show you how it is done if you need a very small kernel with modules.

    http://www.realtime.net/~welbon/initrd.md.tar.gz

2.3 Documentation −− Recommended Reading

Please read:

  /usr/src/linux/Documentation/initrd.txt

as well as the documentation and man pages that accompany the raidtools set. In particular, read **man mdadd** as well as the **QuickStart.RAID** document included in the raidtools package.

You may also wish to review:

- BootPrompt−HOWTO
- **man lilo**
- **man lilo.conf**

2.4 RAID resources

- www.linas.org/linux/Software−RAID/Software−RAID.html
- www.ssc.com/lg/issue17/raid.html
- linas.org/linux/raid.html
- ftp.kernel.org/pub/linux/daemons/raid/
- www.realtime.net/~welbon/initrd.md.tar.gz
- luthien.nuclecu.unam.mx/~miguel/raid/

Mailing lists can be joined at:

- majordomo@nuclecu.unam.mx *send a message to***subscribe raiddev**

  send mail to: raiddev@nuclecu.unam.mx
- majordomo@vger.rutgers.edu *send a message to***subscribe linux−raid**

  send mail to: linux−raid@vger.rutgers.edu*(this seems to be the most active list)*

3.Quick Start for ROOT RAID

If you use **RedHat**, see the Howto set up RedHat section in Appendix H. I have not tried this. If you use it successfully, please let me know so I can update this document.

If you don't want to try and build and debug the rescue system, you can get a generic one created from Slackware−3.4 from:

ftp.bizsystems.com/pub/raid/raidboot−0.01.tar.gz

Perform the following steps:

- Compile the raid enabled kernel with built in support for your disk subsystem
- Test that the raid array will configure and mount correctly
- Build your OS on the raid system
- Correct the entries in **fstab** to show **/dev/md0** as the root device. Make sure that the partition(s) you use for booting are included in **fstab**.
- Modify your shutdown halt and reboot script(s) (mine is /etc/rc.d/rc.6) as shown in Modifying the rc−scripts for SHUTDOWN
- Copy the following from you development filesystem to the rescue system AND the new raid system

```
        cd /root/raidboot
        mkdir mnt
        gzip -d rescue.clean
        losetup /dev/loop0  rescue.clean
        mount /dev/loop0    mnt

copy these files

        cp -p /etc/*         mnt/etc
        cp -p /etc/rc.d/*    mnt/etc/rc.d
                {or as appropriate for your system}
        cp -a /lib/modules/* mnt/lib/modules
```

Some Linux distributions include a test for the **ro/rw** status of the root file system. The **rc startup** files need to have this test removed for the initrd rescue system. See the instructions in the section on Correctons for Rescue System.

Correct the entries in **fstab** to show **/dev/md0** as the root device. Make sure that the partition(s) you use for booting is included in **fstab**.

Create **/etc/raidboot.conf** which describes the raid boot configuration. This file may **NOT** contain comments in the first three lines, after that it doesn't matter.

raidboot.conf

```
        /dev/sda1 /dev/sda2
        raidboot
        raid5.conf
# comments may only be placed 'after' the three
# configuration lines.
#
# This is '/etc/raidboot.conf'
#
# line one, the partition(s) containing the 'initrd' raid-rescue system
#       It is not necessary to boot from these partitions, however,
#       since the rescue system will not fit on floppy, it is necessary
#       to know which partitions are to be used to load the rescue system
#
# line two, the path to the raidboot config information
#       Where the shutdown status, etc... is located at boot time
#       It does NOT include the mount point information, only 'path'
#       /mntpoint/'path'
#
# line -3-, name of the raid configuration file
#       Current raid configuration file i.e. raid1.conf, raid5.conf
```

A few more things to do and the raid systems is ready to boot.

Create rc.raidown, as described in Appendix F, and copy it to /etc/rc.d on the rescue, development, and raid system. Unmount the rescue system and zip it.

```
        umount mnt
        losetup -d /dev/loop0
        mv rescue.clean rescue
        gzip rescue
```

Copy the rescue file to the raidboot partitions.

```
        cp rescue.gz /mnt_point(1)/raidboot
        cp rescue.gz /mnt_point(2)/raidboot
```

Activate the raid array.

```
mdadd -ar
```

Save the **good** reference status to the raidboot partition

```
cat /proc/mdstat | grep md0 > /mnt_point(1)/raidboot/raidgood.ref
cat /proc/mdstat | grep md0 > /mnt_point(1)/raidboot/raidgood.ref
```

Lastly, configure the boot program as outlined in Boot Time Configuration Parameters and reboot your system onto the raid array.

4.*initrd* Cookbook for root mounted RAID

This is the procedure to make an 'initrd' ramdisk with rescue tools for raid.

Specifically, this document referrs to RAID1 and RAID5 implementations.

4.1 Security Reminder

The rescue file system may be used stand alone. Should your raid array fail to mount, you are left with the rescue system mounted and running. TAKE THE APPROPRIATE SECURITY PRECAUTIONS!!!

4.2 Build the Kernel and Raid Tools

The first thing that must be done is to patch and build your kernel and become familiar with the raid tools. Make sure and include Gadi's raid stop patch in Appendix E. Configure, mount and test your raid device(s). The details of how to do this are included in the **raidtools** package and briefly reviewed later in this document.

4.3 Build the *initrd* Rescue and Boot filesystem

I used the **Slackware−3.4** distribution to build both the Rescue/Boot filesystem and the filesystem for the production machine. Any linux distribution should work fine. If you use a different distribution, review the Slackware specific portion of this procedure and modify it to suit your needs.

I use loadlin to boot the kernel image and ramdisk from a dos partition simply because there are oddball devices in my system that have dos configuration software. Lilo will work just as well and a small linux partition can be used instead containing only the raid/boot files and the **lilo** record.

For the raid boot/rescue system, I chose to create a minimum ramdisk system using the Slackware 'setup' script followed by installing the 'linuxthreads' package and 'raidtools' over the clean Slackware installation on my ramdisk. I used the ***identical*** procedure to build the production system. So the rescue and production systems are very similar.

This installation process gives me a 'bare' system (save a copy of the file) to which I overlay

```
/lib/modules/2.x.x......
/etc .... with a modified fstab, mdtab, raidX.conf, raidboot.conf
/etc/rc.d
/dev/md*
```

from my current system to customize it for the particular kernel and machine that it is/will−be running on.

This makes the boot/rescue system the same system that is running on the root mounted raid device, just skinnyed down a bit, while allowing the library, etc... revisions to always be current.

4.4 Start the STEP by STEP instructions

From the root home directory (/root):

```
cd /root
mkdir raidboot
cd raidboot
```

Create a mountpoints to work on

```
mkdir mnt
mkdir mnt2
```

Make a file large enough to do the file system install. This will be a lot larger than the final rescue file system. I chose 24 megs since 16 megs is not large enough

```
dd if=/dev/zero of=build bs=1024k count=24
```

associate the file with a loop device and generate an ext2 file system on the file

```
losetup /dev/loop0 build
mke2fs −v −m0 −L initrd /dev/loop0
mount /dev/loop0 mnt
```

4.5 Install the distribution − Slackware Specific

...skip Slackware Specific stuff and go to next section.

Now that an empty filesystem is created and mounted, run "setup".

```
Specify         /root/raidboot/mnt
```

as the **'target'**. The source is whatever you normally install from. Select the packages you wish to install and proceed but **DO NOT** configure.

Choose 'EXPERT' prompting mode.

I chose 'A', 'AP, and 'N' installing only the minimum to run the system plus an editor I am familiar with (vi, jed, joe) that is reasonably compact.

```
lqqqqqqqq SELECTING PACKAGES FROM SERIES A (BASE LINUX SYSTEM) qqqqqqqqk
x lqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqk x
x x   [X] aaa_base  Basic filesystem, shell, and utils - REQUIRED    x x
x x   [X] bash      GNU bash-1.14.7 shell - REQUIRED                 x x
x x   [X] devs      Device files found in /dev - REQUIRED            x x
x x   [X] etc       System config files & utilities - REQUIRED       x x
x x   [X] shadow    Shadow password suite - REQUIRED                 x x
x x   [ ] ide       Linux 2.0.30 no SCSI (YOU NEED 1 KERNEL)         x x
x x   [ ] scsi      Linux 2.0.30 with SCSI (YOU NEED 1 KERNEL)       x x
x x   [ ] modules   Modular Linux device drivers                     x x
x x   [ ] scsimods  Loadable SCSI device drivers                     x x
x x   [X] hdsetup   Slackware setup scripts - REQUIRED               x x
x x   [ ] lilo      Boots Linux (not UMSDOS), DOS, OS/2, etc.        x x
x x   [ ] bsdlpr    BSD lpr - printer spooling system                x x
x x   [ ] loadlin   Boots Linux (UMSDOS too!) from MS-DOS            x x
x x   [ ] pnp       Plug'n'Play configuration tool                   x x
x x   [ ] umsprogs  Utilities needed to use the UMSDOS filesystem    x x
x x   [X] sysvinit  System V-like INIT programs - REQUIRED           x x
x x   [X] bin       GNU fileutils 3.12, elvis, etc. - REQUIRED       x x
x x   [X] ldso      Dynamic linker/loader - REQUIRED                 x x
x x   [ ] ibcs2     Runs SCO/SysVr4 binaries                         x x
x x   [X] less      A text pager utility - REQUIRED                  x x
x x   [ ] pcmcia    PCMCIA card services support                     x x
x x   [ ] getty     Getty_ps 2.0.7e - OPTIONAL                       x x
x x   [X] gzip      The GNU zip compression - REQUIRED               x x
x x   [X] ps        Displays process info - REQUIRED                 x x
x x   [X] aoutlibs  a.out shared libs - RECOMMENDED                  x x
x x   [X] elflibs   The ELF shared C libraries - REQUIRED            x x
x x   [X] util      Util-linux utilities - REQUIRED                  x x
x x   [ ] minicom   Serial transfer and modem comm package           x x
x x   [ ] cpio      The GNU cpio backup/archiving utility            x x
x x   [X] e2fsbn    Utilities for the ext2 file system               x x
x x   [X] find      GNU findutils 4.1                                x x
x x   [X] grep      GNU grep 2.0                                     x x
x x   [ ] kbd       Change keyboard mappings                         x x
x x   [X] gpm       Cut and paste text with your mouse               x x
x x   [X] sh_utils  GNU sh-utils 1.16 - REQUIRED                     x x
x x   [X] sysklogd  Logs system and kernel messages                  x x
x x   [X] tar       GNU tar 1.12 - REQUIRED                          x x
x x   [ ] tcsh      Extended C shell version 6.07                    x x
x x   [X] txtutils  GNU textutils-1.22 - REQUIRED                    x x
x x   [ ] zoneinfo  Configures your time zone                        x x
x mqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqj x
```

From the 'AP series, I use only 'JOE', and editor I like, and 'MC' a small and useful file management tool. You choose the utilities you will need on your system.

```
lqqqqqqqqq SELECTING PACKAGES FROM SERIES AP (APPLICATIONS) qqqqqqqqqk
x x     [ ] ispell    The International version of ispell          x x
x x     [ ] jove      Jonathan's Own Version of Emacs text editor  x x
x x     [ ] manpgs    More man pages (online documentation)        x x
x x     [ ] diff      GNU diffutils                                x x
x x     [ ] sudo      Allow special users limited root access      x x
x x     [ ] ghostscr  GNU Ghostscript version 3.33                 x x
x x     [ ] gsfonts1  Ghostscript fonts (part one)                 x x
x x     [ ] gsfonts2  Ghostscript fonts (part two)                 x x
x x     [ ] gsfonts3  Ghostscript fonts (part three)               x x
```

```
x x     [ ] jed       JED programmer's editor                      x x
x x     [X] joe       joe text editor, version 2.8                 x x
x x     [ ] jpeg      JPEG image compression utilities             x x
x x     [ ] bc        GNU bc - arbitrary precision math language   x x
x x     [ ] workbone  a text-based audio CD player                 x x
x x     [X] mc        The Midnight Commander file manager          x x
x x     [ ] mt_st     mt ported from BSD - controls tape drive     x x
x x     [ ] groff     GNU troff document formatting system         x x
x x     [ ] quota     User disk quota utilities                    x x
x x     [ ] sc        The 'sc' spreadsheet                         x x
x x     [ ] texinfo   GNU texinfo documentation system             x x
x x     [ ] vim       Improved vi clone                            x x
x x     [ ] ash       A small /bin/sh type shell - 62K             x x
x x     [ ] zsh       Zsh - a custom *nix shell                    x x
x mqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqj x
```

From the 'N' package I only loaded TCPIP. This isn't really necessary, but is very handy and allows access to the network while working on a repair or update with the root raid array dismounted. TCPIP also contains 'biff' which is used by some of the applications in 'A'. If you don't install 'N' you might want to install the biff package anyway.

```
lqqqq SELECTING PACKAGES FROM SERIES N (NETWORK/NEWS/MAIL/UUCP) qqqqqk
x lqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqk x
x x    [ ] apache    Apache WWW (HTTP) server                      x x
x x    [ ] procmail  Mail delivery/filtering utility               x x
x x    [ ] dip       Handles SLIP/CSLIP connections                x x
x x    [ ] ppp       Point-to-point protocol                       x x
x x    [ ] mailx     The mailx mailer                              x x
x x    [X] tcpip     TCP/IP networking programs                    x x
x x    [ ] bind      Berkeley Internet Name Domain server          x x
x x    [ ] rdist     Remote file distribution utility              x x
x x    [ ] lynx      Text-based World Wide Web browser             x x
x x    [ ] uucp      Taylor UUCP 1.06.1 with HDB && Taylor configs x x
x x    [ ] elm       Menu-driven user mail program                 x x
x x    [ ] pine      Pine menu-driven mail program                 x x
x x    [ ] sendmail  The sendmail mail transport agent             x x
x x    [ ] metamail  Metamail multimedia mail extensions           x x
x x    [ ] smailcfg  Extra configuration files for sendmail        x x
x x    [ ] cnews     Spools and transmits Usenet news              x x
x x    [ ] inn       InterNetNews news transport system            x x
x x    [ ] tin       The 'tin' news reader (local or NNTP)         x x
x x    [ ] trn       'trn' for /var/spool/news                     x x
x x    [ ] trn-nntp  'trn' for NNTP (install 1 'trn' maximum)      x x
x x    [ ] nn-spool  'nn' for /var/spool/news                      x x
x x    [ ] nn-nntp   'nn' for NNTP (install 1 'nn' maximum)        x x
x x    [ ] netpipes  Network pipe utilities                        x x
x mqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqj x
```

With the installation complete, say no to everything else (no to all configuration requests) and exit the script.

4.6 Install linux pthreads

Now you must install the 'linuxthreads−0.71' library. I have included this diff for the linuxthreads Makefile rather than explain the details of the installation by hand. Save the original Makefile, apply the diff and then:

```
        cd /usr/src/linuxthreads-0.71
  patch
        make
```

```
        make install

-------------------diff Makefile.old  Makefile.raid-----------------
2a3,13
> # If you are building "linuxthreads" for installation on a mount
> # point which is not the "root" partition, redefine 'BUILDIR' to
> # the mount point to use as the "root" directory
> # You may wish to do this if you are building an 'initial ram disk'
> # such as used with bootable root raid devices.
> # REQUIRES ldconfig version 1.9.5 or better
> # do ldconfig -v to check
> #
> BUILDIR=/root/raidboot/mnt
> #BUILDIR=
>
81,82c92,93
<       install pthread.h $(INCLUDEDIR)/pthread.h
<       install semaphore.h $(INCLUDEDIR)/semaphore.h
---
>       install pthread.h $(BUILDIR)$(INCLUDEDIR)/pthread.h
>       install semaphore.h $(BUILDIR)$(INCLUDEDIR)/semaphore.h
84c95
<       test -f /usr/include/sched.h || install sched.h $(INCLUDEDIR)/sched.h
---
>       test -f $(BUILDIR)/usr/include/sched.h || install sched.h $(BUILDIR)$(INCLUDEDIR)/sched.h
86,89c97,103
<       install $(LIB) $(LIBDIR)/$(LIB)
<       install $(SHLIB) $(SHAREDLIBDIR)/$(SHLIB)
<       rm -f $(LIBDIR)/$(SHLIB0)
<       ln -s $(SHAREDLIBDIR)/$(SHLIB) $(LIBDIR)/$(SHLIB0)
---
>       install $(LIB) $(BUILDIR)$(LIBDIR)/$(LIB)
>       install $(SHLIB) $(BUILDIR)$(SHAREDLIBDIR)/$(SHLIB)
>       rm -f $(BUILDIR)$(LIBDIR)/$(SHLIB0)
>       ln -s $(SHAREDLIBDIR)/$(SHLIB) $(BUILDIR)$(LIBDIR)/$(SHLIB0)
> ifneq ($(BUILDIR),)
>       ldconfig -r ${BUILDIR} -n $(SHAREDLIBDIR)
> else
91c105,106
<       cd man; $(MAKE) MANDIR=$(MANDIR) install
---
> endif
>       cd man; $(MAKE) MANDIR=$(BUILDIR)$(MANDIR) install
```

4.7 Install Raid Tools

The next step is the installation of the raid tools. raidtools−0.42

You must run the "configure" script to point the Makefile at the build directory for the ramdisk files

```
  cd /usr/src/raidtools-0.42
  configure --sbindir=/root/raidboot/mnt/sbin --prefix=/root/raidboot/mnt/usr
  make
  make install
```

Now!! the Makefile for install is not quite right so do the following to clean up. This will be fixed in future releases so that the re−linking will not be necessary.

Fix the make install error

The file links specified in the Makefile at 'LINKS' must be removed and re−linked to operate properly.

```
cd /root/raidboot/mnt/sbin
ln −fs mdadd mdrun
ln −fs mdadd mdstop
```

4.8 Remove un−needed directories and files from new filesystem.

Delete the following directories from filesystem (CAUTION DON'T DELETE FROM YOUR RUNNING SYSTEM) it's easy to do, guess how I found out!!!

```
cd /root/raidboot/mnt
rm −r home/ftp/*
rm −r lost+found
rm −r usr/doc
rm −r usr/info
rm −r usr/local/man
rm −r usr/man
rm −r usr/openwin
rm −r usr/share/locale
rm −r usr/X*
rm −r var/man
rm −r var/log/packages
rm −r var/log/setup
rm −r var/log/disk_contents
```

4.9 Create /dev/md*x*

The last step simply copies the /dev/md* devices from the current file system onto the rescue file system. You could create these with mknode.

```
cp −a /dev/md* /root/raidboot/mnt/dev
```

4.10 Create a bare filesystem suitable for *initrd*

Now you have a clean re−useable filesystem ready for customization. Once customized, this file system can be used for rescue should the raid device(s) become corrupted and the raid tools needed to fix them. It will also be used to boot and root−mount the raid device by adding the linuxrc file which will be discussed next.

Copy the file system to a smaller device for the initrd file, 16 megs should be large enough.

Create the smaller file system and mount it

```
cd /root/raidboot
dd if=/dev/zero of=bare.fs bs=1024k count=16
```

associate the file with a loop device and generate a ext2 file system on the file

```
losetup /dev/loop1 bare.fs
mke2fs -v -m0 -L initrd /dev/loop1
mount /dev/loop1 mnt2
```

Copy the 'build' file system to 'bare.fs'

```
cp -a mnt/* mnt2
```

Save the 'bare.fs' system before customization so later update is easy. The 'build' file system is no longer needed and may be deleted.

```
cd /root/raidboot
umount mnt
umount mnt2
losetup -d /dev/loop0
losetup -d /dev/loop1
rm build
cp bare.fs rescue
gzip -9 bare.fs
```

Create the BOOT/RESCUE *initrd* filesystem

Now copy the system dependent items that match the kernel from the development platform, or you can manually modify the files in the rescue file system to match your target system.

```
losetup /dev/loop0 rescue
mount /dev/loop0 mnt
```

Make sure your etc directory is clean of `*~`, core and log files. The next 2 commands creates some warning messages, ignore them.

```
cp -dp /etc/* mnt/etc
cp -dp /etc/rc.d/* mnt/etc/rc.d

mkdir  mnt/lib/modules
cp -a  /lib/modules/2.x.x mnt/lib/modules <--- your current 2.x.x
```

Corrections for the Rescue System

Edit the following files to correct them for your rescue system. Some file names listed below are Slackware specific but have equivalents in other distributions.

```
        cd mnt

Non-network
        etc/fstab
        etc/mdtab       should work OK
Network
        etc/hosts
        etc/resolv.conf
        etc/hosts.equiv         and related files
        etc/rc.d/rc.inet1       correct ip#, mask, gateway, etc...
        etc/rc.d/rc.S           remove entire section on file system status
                from:
                        # Test to see if the root partition is read-only
                to but not including:
                        # remove /etc/mtab* so that mount will .....
                                This avoids the annoying warning that
```

```
                              the ramdisk is mounted rw.
        etc/rc.d/rc.xxxxx       others as required, see later on in this doc
        root/.rhosts            if present
        home/xxxx/xxxx          others as required

    WARNING:    The above procedure moves your password and shadow
                files onto the rescue disk!!!!!

    WARNING:    You may not wish to do this for security reasons.
```

Create any directories for mounting /dev/disk... as may be required that are unique to your system. These are the mountpoints for booting the system (boot partition and backup boot partition). My system boot from dos using **loadlin**, however linux partition(s) and lilo will work fine. My system uses:

```
        cd /root/raidboot/mnt           <--- initrd root
        mkdir dosa                      dos partition mount point
        mkdir dosb                      dos mirror mount point
```

The rescue file system is complete!

You will note upon examination of the files in the rescue file system, that there are still many files that could be deleted. I have not done this since it would overly complicate this procedure and most raid systems have adequate disk and memory. If you wish to skinny down the file system, go to it!

4.11 Making 'initrd' boot the RAID device − linuxrc

To make the rescue disk boot the raid device, you need only copy the executable script file:

linuxrc

to the root of the device.

The theory of operation for this **linuxrc** file is discussed in Appendix G, linuxrc theory of operation.

A very simple and much easier to understand (working) linuxrc is included in Appendix D, *obsolete linuxrc and shutdown scripts*. Copy the following text to **linuxrc** and save in your development area.

```
 −−−−−−−−−−−−−−−−−−−− linuxrc −−−−−−−−−−−−−−−−−−−−−−
#!/bin/sh
# ver 1.13 3−6−98
#
################# BEGIN 'linuxrc' ##################
#                DEFINE FUNCTIONS                    #
####################################################
# Define 'Fault' function in the event something
# goes wrong during the execution of 'linuxrc'
#
FaultExit () {
# correct fstab to show '/dev/ram0' for rescue system
    /bin/cat /etc/fstab | {
    while read Line
    do
        if [ −z "$( echo ${Line} | /usr/bin/grep md0 )" ]; then
```

```
            echo ${Line}
        else
            echo "/dev/ram0 / ext2 defaults 1 1"
        fi
    done
    } > /etc/tmp.$$
    /bin/mv /etc/tmp.$$ /etc/fstab
#       point root at /dev/ram0 (the rescue system)
        echo 0x100>/proc/sys/kernel/real-root-dev
        /bin/umount /proc
        exit
}

# Define 'Warning' procdure to print banner on boot terminal
#
Warning () {
    echo '**********************************'
    echo -e " $*"
    echo '**********************************'
}

# Define 'SplitKernelArg' to help extract 'Raid' related kernel arguments
SplitKernelArg () { eval $1='$( IFS=,; echo $2)' }

#Define 'SplitConfArgs' to help extract system configuration arguments
SplitConfArgs () {
    RaidBootType=$1
    RaidBootDevice=$2
    RaidConfigPath=$3
}
########################################################
################## MAIN linuxrc ######################
########################################################
# mount the proc file system
/bin/mount /proc

# Get the boot partition and configuration location from command line
CMDLINE=`/bin/cat /proc/cmdline`
for Parameter in $CMDLINE; do
    Parameter=$( IFS='='; echo ${Parameter} )
    case $Parameter in
        Raid*) SplitKernelArg $Parameter;;
    esac
done

# check for 'required raid boot'
if [ -z "${Raid_Conf}" ]; then
    Warning Kernel command line \'Raid_Conf\' missing
    FaultExit
fi
SplitConfArgs $Raid_Conf

# tmp mount the boot partition
/bin/mount -t ${RaidBootType} ${RaidBootDevice} /mnt

# get etc files from primary raid system
pushd /etc

# this will un-tar into 'etc' (see rc.6)
if [ ! -f /mnt/${RaidConfigPath}/raidboot.etc ]; then
# bad news, this file should be here
    Warning required file \'raidboot.etc\' \
```

```
    missing from ${RaidBootDevice}/${RaidConfigPath} \\n \
    \\tUsing rescue system defaults
else
    /bin/tar -xf /mnt/${RaidConfigPath}/raidboot.etc
fi
# get 'real' raidboot device for this boot
# status path, and name of raidX.conf
if [ ! -f /mnt/${RaidConfigPath}/raidboot.cfg ]; then
# bad news, this file should be here
    Warning required file 'raidboot.cfg' \
    missing from ${RaidBootDevice}/${RaidConfigPath}\\n \
    \\tUsing rescue system defaults
# Get the first raidX.conf file name in $RArg1
    RaidBootDevs=$RaidBootDevice
    RaidStatusPath=$RaidConfigPath
    for RaidConfigEtc in $( ls raid*.conf )
    do break; done
else
    {
    read RaidBootDevs
    read RaidStatusPath
    read RaidConfigEtc
    } < /mnt/${RaidConfigPath}/raidboot.cfg

fi
popd
/bin/umount /mnt

# Set a flag in case the raid status file is not found
#
RAIDOWN="raidboot.ro not found"
RAIDREF="raidgood.ref not found"
echo "Reading md0 shutdown status."

# search for raid shutdown status
for Device in ${RaidBootDevs}
do
#   these filesystem types should be in 'fstab' since
#   the partitions must be mounted for a clean raid shutdown
    /bin/mount ${Device} /mnt
    if [ -f /mnt/${RaidStatusPath}/raidboot.ro ]; then
        RAIDOWN=`/bin/cat /mnt/${RaidStatusPath}/raidboot.ro`
        RAIDREF=`/bin/cat /mnt/${RaidStatusPath}/raidgood.ref`
        /bin/umount /mnt
        break
    fi
    /bin/umount /mnt
done
# Test for a clean shutdown with array matching reference
if [ "${RAIDOWN}" != "${RAIDREF}" ]; then
    Warning shutdown ERROR ${RAIDOWN}
    FaultExit
fi

# The raid array is clean, remove shutdown status files
for Device in ${RaidBootDevs}
do
    /bin/mount ${Device} /mnt
    /bin/rm -f /mnt/${RaidStatusPath}/raidboot.ro
    /bin/umount /mnt
done
```

```
# Write a clean superblock on all raid devices

echo "write clean superblocks"
/sbin/mkraid -f --only-superblock /etc/${RaidConfigEtc}

# Activate raid array(s)
if [ -z "$Raid_ALT" ]; then
    /sbin/mdadd -ar
else
    /sbin/mdadd $Raid_ALT
fi

#  If there are errors - BAIL OUT and leave rescue running
if [ $? -ne 0 ]; then
   Warning some RAID device has errors
   FaultExit
fi

# Everything is fine, let the kernel mount /dev/md0
# tell the kernel to switch to /dev/md0 as the /root device
# The 0x900 value is the device number calculated by:
#  256*major_device_number + minor_device number
echo "/dev/md0 mounted on root"
echo 0x900>/proc/sys/kernel/real-root-dev
# umount /proc to deallocate initrd device ram space
/bin/umount /proc
exit
#------------------ end linuxrc ----------------------
```

Add 'linuxrc' to initrd boot device

```
        cd /root/raidboot
        chmod 777 linuxrc
        cp -p linuxrc mnt
```

4.12 Modifying the rc−scripts for SHUTDOWN

To complete the installation, modify the rc scripts to save the md status to the real root device when shutdown occurs.

```
In slackware this is rc.0 -> rc.6
In debian 'bo' this is in both 'halt' and 'reboot'

If you implement this in another distribution, please e-mail
the instructions and sample files so they can be included here.
```

I have modified Bohumil Chalupa's raid stop work−around slightly. His original solution is presented in Appendix A.

Since there are no linux partitions left on the production system except **md0**, the boot partitions are used to store the **raidOK readonly** status. I chose to write a file to each of the duplicate boot partitions containing the status of the md array at shutdown and signifying that the md device has been remounted RO. This allows the system to be fail safe when any of the hard drives die.

The shutdown script is modified to call rc.raidown which saves the necessary information to successfully reboot and mount the raid device. Examples of shutdown scripts for various linux distributions are shown in Appendix B.

To capture the raid array shutdown status insert a call to rc.raidown after any **case** statements (if present) but before the actual shutdown (kills, status saves, etc...) begins and before the file systems are dismounted.

```
############ Save raid boot and status info ##############
#
  if [ -x /etc/rc.d/rc.raidown ]; then
    /etc/rc.d/rc.raidown
  fi
################## end raid boot #########################
```

After all the file systems are dismounted (the root file system 'will not' dismount) but before any powerfail status check add:

```
################ for raid arrays #########################
# Stop all known raid arrays (except root which won't stop)
  if [ -x /sbin/mdstop ]; then
    echo "Stopping raid"
    /sbin/mdstop -a
  fi
##########################################################
```

This will cleanly stop all raid devices except root. Root status is passed to the next boot in **raidstat.ro**.

Copy the rc file to your new raid array, the rescue file system that is still mounted on **/root/raidboot/mnt** and the development system if it is on the same machine.

Modify rescue **etc/fstab** as needed and make sure rescue **mdtab** is correct.

Now copy the rescue disk to your dos partition and everything should be ready to boot the raid device as root.

```
        umount mnt
        losetup -d /dev/loop0
        gzip -9 rescue
```

Copy rescue.gz to your boot partitions.

All that remains is to creat the configuration file **raidboot.conf** and test the new file system by rebooting.

4.13 Configuring RAIDBOOT − raidboot.conf

The comments following the example configuration file explain each of the three lines. This example file is for a 4 drive raid5 scsii array with duplicate boot partitions on drives sda1 and sdb1. Put the paramaters descriptive of your file systems here instead.

```
  /dev/sda1 /dev/sdb1
  linux
  raid5.conf
# comments may only be placed 'after' the three
# configuration lines.
#
# This is 'raidboot.conf'
#
# line one, the partition(s) containing the 'initrd' raid-rescue system
#       It is not necessary to boot from these partitions, however,
#       since the rescue system will not fit on floppy, it is necessary
#       to know which partitions are to be used to load the rescue system
#
```

```
# line two, the path to the raidboot config information
#       Where the shutdown status, etc... is located at boot time
#       It does NOT include the mount point information, only 'path'
#       /mntpoint/'path'
#
# line -3-, name of the raid configuration file
#       Current raid configuration file i.e. raid1.conf, raid5.conf
```

4.14 Kernel 'loadlin and lilo' variables for RESCUE and RAID

There are two kernel variables for the RESCUE and RAID system, only the first need be specified.

- Raid_Conf=msdos,/dev/sda1,raidboot

  This variable points to raid boot device and configuration file. For floppy rescue boot, you may want to specify this on the kernel command line or in the loadlin or lilo boot file

  format:
  `'filesystem-type,device,path-to-config-from-mountpoint'`
- Raid_ALT=–r,–p5,/dev/md0,/dev/sda3 /dev/sdb3 /dev/sdc3 /dev/sdd3

  Alternate mdadd parameters necessary when booting with non–redundant raid array. These are the comma separated command line parameters for **mdadd**. Unless they are needed to start a failed/non–redundant array, COMMENT OUT OR SPECIFY WITH A 'NULL'.

  i.e. Raid_ALT=

Either of these parameters may be specified in the lilo or loadlin boot parameter file or on the loadlin kernel command line. Care must be taken that the maximum line length is not exceeded, however, if the command line is used (128 characters).

When booting with **lilo**, the parameters are included in the lilo config file in the form:

```
append="Raid_Conf=msdos,/dev/sda1,raidboot"
append="Raid_ALT=-r,-p5,/dev/md0,/dev/sda3 /dev/sdb3 /dev/sdc3 /dev/sdd3"
```

See **man lilo.conf** for more detailed information.

Since I have some hardware that requires DOS configuration utilities, I have a small dos partition on the system. Therefore, I used loadlin to boot the raid5 system from the dos partition with a mirror (copy) on the companion disk. An identical method is used for the raid1 system. The example below uses loadlin, but the procedure is very similar for lilo.

My dos root system contains a small editor among the utilities so I can modify the boot parameters of loadlin if necessary, allowing me to reboot the linux system on my swap disk while testing.

The dos system contains this tree for linux"

```
c:\raidboot.bat
c:\raidboot\loadlin.exe
c:\raidboot\zimage
c:\raidboot\rescue.gz
c:\raidboot\raidboot.cfg
c:\raidboot\raidboot.etc
c:\raidboot\raidgood.ref
c:\raidboot\raidstat.ro (only at shutdown)
```

linux.bat contains:

```
---------------------- linux.bat ---------------------------
echo "Start the LOADLIN process:"
c:\raidboot\loadlin @c:\raidboot\boot.par
-------------------- end linux.bat -------------------------
```

boot.par contains:

```
# loadlin boot parameter file
#
# version 1.02 3-6-98

# linux kernel image
c:\linux\zimage

# target root device
root=/dev/md0
#root=/dev/ram0
#root=/dev/sdc5

# mount root device as 'ro'
ro

# size of ram disk
ramdisk_size=16384

# initrd file name
initrd=c:\raidboot\rescue.gz
#noinitrd

# memory ends here
mem=131072k

# points to raid boot device, configuration file
# for floppy rescue boot, you may want to specify
# this on the command line instead of here
# format 'filesystem-type,device,path-to-config-frm_mntpnt'
Raid_Conf=msdos,/dev/sda1,raidboot

# Alternate mdadd parameters
# necessary when boot with non-redundant raid
# otherwise, COMMENT OUT OR SPECIFY 'NULL'
#Raid_ALT=-r,-p5,/dev/md0,/dev/sda3 /dev/sdb3 /dev/sdc3 /dev/sdd3

# ethernet devices
ether=10,0x300,eth0
```

```
***** >> NOTE!! the only difference between forcing the rescue system to
            run and the raid device mounting, is the loadlin parameter
```

```
root=/dev/ram0          for the rescue system
root=/dev/md0           for RAID

With root=/dev/ram0 the RAID device will not mount
and the rescue system will run unconditionally.
```

If the RAID array fails, the rescue system is left mounted and running.

5.Configuring the Production RAID system.

5.1 System specs. Two systems with identical motherboards were configured.

```
                                 Raid-1          Raid-5
Motherboard:    Iwill P55TU     dual ide        adaptec scsi
Processor:      Intel P200
Disks:                          2ea  7 gig      4 ea Segate 4.2 gig
                                Maxtors         wide scsii
```

The disk drives are designated by linux as 'sda' through 'sdd' on the raid5 system and 'hda' and 'hdc' on the raid1 system.

5.2 Partitioning the hard drives.

Since testing a large root mountable RAID array is difficult because of the ckraid re−boot problem, I re−partitioned my swap space to include a smaller RAID partition for testing purposes, sda6,sdb6,sdc6,sdd6, and a small root and /usr/src partition pair for developing and testing the raid kernel and tools. You may find this helpful.

```
        <bf/DEVELOPMENT SYSTEM - RAID5/
   Device       System          Size    Purpose

  /dev/sda1     dos boot        16 meg  boot partition
* /dev/sda2     extended        130 meg (see below)
  /dev/sda3     linux native    4 gig   primary raid5-1
----------------------sda2----------------------------
* /dev/sda5     linux swap      113 meg SWAP space
* /dev/sda6     linux native    16 meg  test raid5-1
========================================================
  /dev/sdb1     dos boot        16 meg  boot partition duplicate
* /dev/sdb2     extended        130 meg (see below)
  /dev/sdb3     linux native    4 gig   primary raid5-2
----------------------sdb2----------------------------
* /dev/sdb5     linux swap      113 meg SWAP space
* /dev/sdb6     linux native    16 meg  test raid5-2
========================================================
* /dev/sdc2     extended        146 meg (see below)
  /dev/sdc3     linux native    4 gig   primary raid5-3
```

```
----------------------sdc2-----------------------------
* /dev/sdc5     linux swap      130 meg development root partition
* /dev/sdc6     linux native    16 meg  test raid5-3
========================================================
* /dev/sdd2     extended        146 meg (see below)
  /dev/sdd3     linux native    4 gig   primary raid5-4
----------------------sdd2-----------------------------
* /dev/sdd5     linux swap      130 meg development /usr/src
* /dev/sdd6     linux native    16 meg  test raid5-4


        <bf/DEVELOPMENT SYSTEM - RAID1/
   Device       System          Size    Purpose

  /dev/hda1     dos             16meg   boot partition
* /dev/hda2     extended        126m    (see below)
  /dev/hda3     linux           126m    development root partition
  /dev/hda4     linux           6+gig   raid1-1
----------------------hda2-----------------------------
* /dev/hda5     linux            26m    test raid1-1
* /dev/hda6     linux swap      100m
========================================================

  /dev/hdc1     is simply an exact copy of hda1 so the
                partion can be made active if hda fails
* /dev/hdc2     extended        126m    (see below)
  /dev/hdc3     linux           126m    development /usr/src
  /dev/hdc4     linux           6+gig   raid1-2
----------------------hdc2-----------------------------
* /dev/hdc5     linux            26m    test raid1-2
* /dev/hdc6     linux swap      100m
```

The sdx2 and hdx3 partitions were switched to 'swap' after developing this utility. I could have done it on another machine, however, the libraries and kernels are all about a year or more out of date on my other linux boxes and I preferred to build it on the target machine.

The partitioning scheme was chosen so that in the event that any one of the drives fails catastrophically, the system will continue to run and be bootable with minimum effort and NO data loss.

- If any single hard drive fails, the boot will abort, and the rescue system will run. Examination of the screen message or /dos*x*/raidboot/raidstat.ro will tell the operator the status of the failed array.
- If sda1 (raid5) or hda1 (raid1) fails, the dos backup boot partition must be made 'active' and the bios must recognize the new partition as the boot device or it must be physically be moved to the *x*da position. Alternatively, the system could be booted from a floppy disk using the initrd image on the remaining backup boot drive. The raid system can then be made active again by issuing:

  ```
  "/sbin/mkraid /etc/raid<it/x/.conf -f --only-superblock"
  ```
  to rebuild the remaining superblock(s).
- Once this is done, then

  ```
  mdadd -ar
  ```
- Examine the status of the array to verify that everything is OK then replace the good array reference with the current status until the failed disk can be repaired or replaced.

  ```
  cat /proc/mdstat | grep md0 > /dosx/raidboot/raidgood.ref

  shutdown -r now
  ```
  to do a clean reboot, and the system is up again.

6.Building the RAID file system.

This description is for my RAID systems described in the system specs. Your system may have a different RAID architecture, so modify as appropriate. Please read the man pages and QuickStart.RAID that come with the raidtools−0.42

6.1 /etc/raid5.conf

```
# raid-5 configuration
raiddev                 /dev/md0
raid-level              5
nr-raid-disks           4
chunk-size              32

# Parity placement algorithm
parity-algorithm        left-symmetric

# Spare disks for hot reconstruction
#nr-spare-disks         0

device                  /dev/sda3
raid-disk               0

device                  /dev/sdb3
raid-disk               1

device                  /dev/sdc3
raid-disk               2

device                  /dev/sdd3
raid-disk               3
```

6.2 /etc/raid1.conf

```
# raid-1 configuration
raiddev                 /dev/md0
raid-level              1
nr-raid-disks           2
nr-spare-disks          0

device                  /dev/hda4
raid-disk               0

device                  /dev/hdc4
raid-disk               1
```

6.3 Step by Step procedures for building production RAID file system.

For my RAID5 system I did a complete install of:

```
Slackware-3.4           any current distribution should work OK
linuxthreads-0.71
raidtools-0.42
linux-2.0.33 with raid145 patch and Gadi's patch
```

Create and format the raid device.

```
mkraid /etc/raid5.conf
mdcreate raid5 /dev/md0 /dev/sda3 /dev/sdb3 /dev/sdc3 /dev/sdd3
mdadd -ar
mke2fs /dev/md0
mkdir /md
mount -t ext2 /dev/md0 /md
```

Create the reference files that reboot will use, this may be different on your system.

```
cat /proc/mdstat | grep md0 > /dosa/raidboot/raidgood.ref
cat /proc/mdstat | grep md0 > /dosb/raidboot/raidgood.ref
```

Use Slackware−3.4 or another distribution to build your OS

```
setup
```

Specify '/md' as the target, and the source whatever your normally use. Select and install the disksets of interest except for the kernel. Configure the system, but skip the section on lilo and kernel booting. Exit setup.

Install 'pthreads'

```
cd /usr/src/linuxthreads-0.71
```

edit the Makefile and specify

```
BUILDIR=/md

make
make install
```

Install 'raidtools'

```
cd /usr/src/raidtools-0.42
configure --sbindir=/md/sbin --prefix=/md/usr
```

fix the raidtools make install error

```
cd /md/sbin
rm mdrun
rm mdstop
ln -s mdadd mdrun
ln -s mdadd mdstop
```

Create /dev/mdx

```
cp -a /dev/md* /md/dev
```

Add the system configuration from the current system (ignore errors).

```
cp -dp /etc/* mnt/etc
cp -dp /etc/rc.d/* mnt/etc/rc.d         (include the new rc.6)
mkdir  mnt/lib/modules
cp -a  /lib/modules/2.x.x mnt/lib/modules <--- your current 2.x.x
```

Edit the following files to correct them for your file system

```
        cd /md

Non-network
        etc/fstab       correct for real root and raid devices.
```

```
        etc/mdtab       should work OK
Network
        etc/hosts
        etc/resolv.conf
        etc/hosts.equiv         and related files
        etc/rc.d/rc.inet1       correct ip#, mask, gateway, etc...
        etc/rc.d/rc.S           remove entire section on file system status
                from:
                        # Test to see if the root partition isread-only
                to but not including:
                        # remove /etc/mtab* so that mount will .....
                                This avoids the annoying warning that
                                the ramdisk is mounted rw.
        etc/rc.d/rc.xxxxx       others as required
        root/.rhosts            if present
        home/xxxx/xxxx          others as required

    WARNING:    The above procedure moves your password and shadow
                files onto the new file system!!!!!

    WARNING:    You may not wish to do this for security reasons.
```

Create any directories for mounting /dev/disk... as may be required that are unique to your system. Mine need:

```
        cd /md          <--- new file system root
        mkdir dosa                      dos partition mount point
        mkdir dosb                      dos mirror mount point
```

The new file system is complete. Make sure and save the md reference status to the 'real' root device and you are ready to boot.

mount the dos partitions on dosa and dosb

```
        cat /proc/mdstat | grep md0 > /dosa/raidboot/raidgood.ref
        cat /proc/mdstat | grep md0 > /dosb/raidboot/raidgood.ref

        mdstop /dev/md0
```

7.One last thought.

Remember that an expert is someone who knows at least 1% more than you do about a subject. Bear this in mind when you e−mail me for help. I'll try, but I've only done this once for raid1 and once for raid5!

Michael Robinton Michael@bzs.org

8.Appendix A. − Bohumil Chalupa's md0 shutdown

Bohumil Chalupa's post to the linux raid list on the work around for the raid1 + 5 mdstop problem. His solution does not address the possibility of the raid device being corrupt at shutdown. So I have added a simple status comparison to a good reference status at boot. This allows the operator to intervene if something is wrong with a disk in the array. The description of this is in the main body of this document.

```
> From: Bohumil Chalupa <bochal@apollo.karlov.mff.cuni.cz>
>
> I can now boot initrd and use linuxrc to start the RAID1 array,
> then successfully switch root to /dev/md0.
>
> I don't know, however, any way how to cleanly _stop_ the array.

Well. I have to answer myself :-)

> Date: Mon, 29 Dec 1997 02:21:38 -0600 (CST)
> From: Edward Welbon <welbon@bga.com>
> Subject: Re: dismounting root raid device
>
> For md devices other than raid0, there is probably state that needs to
> be saved that is only known once all writes have completed.  Such state
> of course can't be saved to root once it is mounted readonly.  In that
> case, you would have to be able to mount a writeable filesystem "X"
> on the readonly root and be able to write to "X" (I recall doing this
> during "rescue" operations, but not as an automated procedure).
>
> The filesystem "X" would presumably be a boot device from which the raid
> (during linuxrc exection via initrd) would pickup it's initial state from.
> Fortunately raid0 isn't required to write out any state (though it would
> be pleasant to be able to write the check sums to mdtab after an mdstop).
> Eventually, I will fiddle with this but it doesn't seem difficult though
> the "devil" is always in the "details".

Yes, that's it.
I had this idea in mind for some time already, but had no time to try it.
Yesterday I did, and it works.

With my RAID1 (mirror), I don't save any checksums or raid superblock data.
I only save an information on the "real" boot partition, that the root md
volume was remounted readonly during shutdown. Then, during boot, the
linuxrc script runs mkraid --only-superblock  when it finds this
information; otherwise, it runs ckraid.
This means, that the raid superblock information is not updated during
shutdown; it's updated at the boot time.
It is not very clean, I'm afraid,  :-(   but it works.

I'm using Slackware and initrd.md by Edward Welbon to boot the root raid
device.
As far as I remember now, the only modified files are
mkdisk and linuxrc, and /etc/rc.d/rc.6 shutdown script.
And lilo.conf, of course.

I'm appending the important parts.

Bohumil Chalupa

--------------- my.linuxrc follows -----------------
#!/bin/sh
# we need /proc
/bin/mount /proc
# start up the md0 device. let the /etc/rc.d scripts get the rest of them
# we should do as little as possible here
# ________________________________________
# root raid1 shutdown test & recreation
# /start must be created on the rd image in my.mkdisk
echo "preparing md0: mounting /start"
/bin/mount /dev/sda2 /start -t ext2
echo "reading saved md0 state from /start"
```

```
if [ -f /start/root.raid.ok ]; then
 echo "raid ok, modyfying superblock"
 rm /start/root.raid.ok
 /sbin/mkraid /etc/raid1.conf -f --only-superblock
else
 echo "raid not clean, runing ckraid --fix"
 /sbin/ckraid --fix /etc/raid1.conf
fi
echo "unmounting /start"
/bin/umount /start
# __________________________________________
#
echo "adding md0 for root file system"
/sbin/mdadd /dev/md0 /dev/sda1 /dev/sdb1
echo "starting md0"
/sbin/mdrun -p1 /dev/md0
# tell kernel we want to switch to /dev/md0 as root device, the 0x900 value
# is arrived at via 256*major_device_number + minor_device number.
echo "setting real-root-dev"
/bin/echo 0x900>/proc/sys/kernel/real-root-dev
#  unmount /proc so that the ram disk can be deallocated.
echo "unmounting /proc"
/bin/umount /proc
/bin/echo "We are hopefully ready to mount /dev/md0 (major 9, minor 0) as
root"
exit
--------------- end of my.linuxrc ----------------------------------


----------- extract from /etc/rc.d/rc.6 follows -----------------
  # Turn off swap, then unmount local file systems.
  echo "Turning off swap."
  swapoff -a
  echo "Unmounting local file systems."
  umount -a -tnonfs
  # Don't remount UMSDOS root volumes:
  if [ ! "`mount | head -1 | cut -d ' ' -f 5`" = "umsdos" ]; then
    mount -n -o remount,ro /
  fi

  # Save raid state
  echo "Saving RAID state"
  /bin/mount -n /dev/sda2 /start -t ext2
  touch /start/root.raid.ok
  /bin/umount -n /start

-------------- end of excerpt from rc.6 ------------------------


------------------ part of my.mkdisk follows ----------------------
#
#  now we have the filesystem ready to be populated, we need to
#  get a few important directories.  I had endless trouble till
#  I created a pristine mtab.  In my case, it is convenient that
#  /etc/mdtab is copied over, this way I can activate md with
#  a simple "/sbin/mdadd -ar" in linuxrc.
#
cp -a $ROOT/etc $MOUNTPNT 2>cp.stderr 1>cp.stdout
rm -rf $MOUNTPNT/etc/mtab
rm -rf $MOUNTPNT/etc/ppp*
rm -rf $MOUNTPNT/etc/termcap
rm -rf $MOUNTPNT/etc/sendmail*
```

```
rm -rf $MOUNTPNT/etc/rc.d
rm -rf $MOUNTPNT/etc/dos*
cp -a $ROOT/sbin $ROOT/dev $ROOT/lib $ROOT/bin $MOUNTPNT 2>>cp.stderr
1>>cp.stdout
# ______________________________________________________________________
#  RAID: will need mkraid and ckraid
cp -a $ROOT/usr/sbin/mkraid $ROOT/usr/sbin/ckraid $MOUNTPNT/sbin
2>>cp.stderr 1>>cp.stdout
# ----------------------------------------------------------------------
#  it seems that init wont come out to play unless it has utmp.   this can
#  probably be pruned back alot.  no telling what the real bug was 8-).
#
mkdir $MOUNTPNT/var $MOUNTPNT/var/log $MOUNTPNT/var/run $MOUNTPNT/initrd
touch $MOUNTPNT/var/run/utmp $MOUNTPNT/etc/mtab
chmod a+r $MOUNTPNT/var/run/utmp $MOUNTPNT/etc/mtab
ln -s /var/run/utmp $MOUNTPNT/var/log/utmp
ln -s /var/log/utmp $MOUNTPNT/etc/utmp
ls -lstrd $MOUNTPNT/etc/utmp $MOUNTPNT/var/log/utmp $MOUNTPNT/var/run/utmp
#
#  since I wanted to change the mount point, I needed this though
#  I suppose that I could have done a "mkdir /proc" in linuxrc.
#
mkdir $MOUNTPNT/proc
chmod 555 $MOUNTPNT/proc
#
#  -------------------------------------------------------
#  we'll mount the real boot device to /start temporarily
#  to check the root raid state saved at shutdown time
#
mkdir $MOUNTPNT/start
#  --------------------------------------------------------
#
#  need linuxrc  (it is, after all, the point of this exercise).
#
if [ -x ./my.linuxrc ]; then
  cp -a ./my.linuxrc $MOUNTPNT/linuxrc
  chmod 777 $MOUNTPNT/linuxrc
else
   ln -s /bin/sh $MOUNTPNT/linuxrc
fi
#
----------------- part of my.mkdisk ends -----------------
```

9.Appendix B. – Sample SHUTDOWN scripts

- Slackware
- Debian

9.1 Slackware – /etc/rc.d/rc.6

```
#! /bin/sh
#
# rc.6          This file is executed by init when it goes into runlevel
#               0 (halt) or runlevel 6 (reboot). It kills all processes,
#               unmounts file systems and then either halts or reboots.
#
# Version:      @(#)/etc/rc.d/rc.6      1.50    1994-01-15
#
# Author:       Miquel van Smoorenburg <miquels@drinkel.nl.mugnet.org>
# Modified by:  Patrick J. Volkerding, <volkerdi@ftp.cdrom.com>
#
# Modified by:  Michael A. Robinton < michael@bizsystems.com >
#               to add call to rc.raidown
  # Set the path.
  PATH=/sbin:/etc:/bin:/usr/bin

  # Set linefeed mode to avoid staircase effect.
  stty onlcr

  echo "Running shutdown script $0:"

  # Find out how we were called.
  case "$0" in
        *0)
                message="The system is halted."
                command="halt"
                ;;
        *6)
                message="Rebooting."
                command=reboot
                ;;
        *)
                echo "$0: call me as \"rc.0\" or \"rc.6\" please!"
                exit 1
                ;;
  esac

############ Save raid boot and status info ##############
#
if [ -x /etc/rc.d/rc.raidown ]; then
   /etc/rc.d/rc.raidown
fi
################## end raid boot ########################

  # Kill all processes.
  # INIT is supposed to handle this entirely now, but this didn't always
  # work correctly without this second pass at killing off the processes.
  # Since INIT already notified the user that processes were being killed,
  # we'll avoid echoing this info this time around.
  if [ "$1" != "fast" ]; then # shutdown did not already kill all processes
    killall5 -15
    killall5 -9
  fi

  # Try to turn off quota and accounting.
  if [ -x /usr/sbin/quotaoff ]
  then
        echo "Turning off quota."
```

```
        /usr/sbin/quotaoff -a
  fi
  if [ -x /sbin/accton ]
  then
        echo "Turning off accounting."
        /sbin/accton
  fi

  # Before unmounting file systems write a reboot or halt record to wtmp.
  $command -w

  # Save localtime
  [ -e /usr/lib/zoneinfo/localtime ] && cp /usr/lib/zoneinfo/localtime /etc

  # Asynchronously unmount any remote filesystems:
  echo "Unmounting remote filesystems."
  umount -a -tnfs &

  # Turn off swap, then unmount local file systems.
  echo "Turning off swap."
  swapoff -a
  echo "Unmounting local file systems."
  umount -a -tnonfs
  # Don't remount UMSDOS root volumes:
  if [ ! "`mount | head -1 | cut -d ' ' -f 5`" = "umsdos" ]; then
    mount -n -o remount,ro /
  fi

################ for raid arrays ########################
# Stop all known raid arrays (except root which won't stop)
if [ -x /sbin/mdstop ]; then
  echo "Stopping raid"
  /sbin/mdstop -a
fi
########################################################

  # See if this is a powerfail situation.
  if [ -f /etc/powerstatus ]; then
    echo "Turning off UPS, bye."
    /sbin/powerd -q
    exit 1
  fi

  # Now halt or reboot.
  echo "$message"
  [ ! -f /etc/fastboot ] && echo "On the next boot fsck will be FORCED."
  $command -f
############### end rc.6 #################################
```

9.2 Debian bo − /etc/init.d/halt and /etc/init.d/reboot

The modifications shown here for Debian bo halt and reboot files are NOT TESTED. When you test this, please e−mail me so I can remove this comment.

/etc/init.d/halt

```
#! /bin/sh
#
# halt          The commands in this script are executed as the last
#               step in runlevel 0, ie halt.
#
# Version:      @(#)halt  1.10  26-Apr-1997  miquels@cistron.nl
#

PATH=/sbin:/bin:/usr/sbin:/usr/bin

############ Save raid boot and status info ##############
#
if [ -x /etc/rc.d/rc.raidown ]; then
   /etc/rc.d/rc.raidown
fi
################## end raid boot ########################

# Kill all processes.
echo -n "Sending all processes the TERM signal... "
killall5 -15
echo "done."
sleep 5
echo -n "Sending all processes the KILL signal... "
killall5 -9
echo "done."

# Write a reboot record to /var/log/wtmp.
halt -w

# Save the random seed between reboots.
/etc/init.d/urandom stop

echo -n "Deactivating swap... "
swapoff -a
echo "done."

echo -n "Unmounting file systems... "
umount -a
echo "done."

mount -n -o remount,ro /

################ for raid arrays ########################
# Stop all known raid arrays (except root which won't stop)
if [ -x /sbin/mdstop ]; then
  echo "Stopping raid"
  /sbin/mdstop -a
fi
##########################################################

# See if we need to cut the power.
if [ -x /etc/init.d/ups-monitor ]
then
        /etc/init.d/ups-monitor poweroff
fi

halt -d -f
############# end halt ####################
```

/etc/init.d/reboot

```
#! /bin/sh
#
# reboot        The commands in this script are executed as the last
#               step in runlevel 6, ie reboot.
#
# Version:      @(#)reboot  1.9  02-Feb-1997  miquels@cistron.nl
#

PATH=/sbin:/bin:/usr/sbin:/usr/bin

############ Save raid boot and status info ##############
#
if [ -x /etc/rc.d/rc.raidown ]; then
   /etc/rc.d/rc.raidown
fi
################## end raid boot #########################

# Kill all processes.
echo -n "Sending all processes the TERM signal... "
killall5 -15
echo "done."
sleep 5
echo -n "Sending all processes the KILL signal... "
killall5 -9
echo "done."

# Write a reboot record to /var/log/wtmp.
halt -w

# Save the random seed between reboots.
/etc/init.d/urandom stop

echo -n "Deactivating swap... "
swapoff -a
echo "done."

echo -n "Unmounting file systems... "
umount -a
echo "done."

mount -n -o remount,ro /

################ for raid arrays #########################
# Stop all known raid arrays (except root which won't stop)
if [ -x /sbin/mdstop ]; then
  echo "Stopping raid"
  /sbin/mdstop -a
fi
##########################################################

echo -n "Rebooting... "
reboot -d -f -i
```

10.Appendix C. – other setup files

10.1 linuxrc linuxrc file

10.2 loadlin –– linux.bat file – boot.par linux.bat file – boot.par

10.3 linuxthreads Makefile.diff linuxthreads Makefile.diff

10.4 raid1.conf raid1.conf

10.5 raid5.conf raid5.conf

10.6 raidboot.conf raidboot.conf

10.7 rc.raidown rc.raidown

11.Appendix D. – obsolete linuxrc and shutdown scripts

11.1 Obsolete working – linuxrc

This linuxrc file works fine with the shutdown procedure in the next subsection.

```
 ---------------------- linuxrc --------------------
#!/bin/sh
# ver 1.07 2-12-98
# linuxrc - for raid1 using small dos partition and loadlin
#

# mount the proc file system
/bin/mount /proc

# This may vary for your system.
```

```
# Mount the dos partitions, try both
# in case one disk is dead
/bin/mount /dosa
/bin/mount /dosc

# Set a flag in case the raid status file is not found
# then check both drives for the status file
RAIDOWN="raidstat.ro not found"
/bin/echo "Reading md0 shutdown status."
if [ -f /dosa/raidboot/raidstat.ro ]; then
  RAIDOWN=`/bin/cat /dosa/raidboot/raidstat.ro`
  RAIDREF=`/bin/cat /dosc/raidboot/raidgood.ref`
else
  if [ -f /dosc/raidboot/raidstat.ro ]; then
    RAIDOWN=`/bin/cat /dosc/raidboot/raidstat.ro`
    RAIDREF=`/bin/cat /dosc/raidboot/raidgood.ref`
  fi
fi

# Test for a clean shutdown with all disks operational
if [ "${RAIDOWN} != ${RAIDREF}" ]; then
  echo "ERROR ${RAIDOWN}"
#  Use the next 2 lines to BAIL OUT and leave rescue running
   /bin/echo 0x100>/proc/sys/kernel/real-root-dev
   exit                 # leaving the error files in dosa/raidboot,etc...
fi

# The raid array is clean, proceed by removing
# status file and writing a clean superblock
/bin/rm /dosa/raidboot/raidstat.ro
/bin/rm /dosc/raidboot/raidstat.ro
/sbin/mkraid /etc/raid1.conf -f --only-superblock

/bin/umount /dosa
/bin/umount /dosc

# Mount raid array
echo "Mounting md0, root filesystem"
/sbin/mdadd -ar

#  If there are errors - BAIL OUT and leave rescue running
if [ $? -ne 0 ]; then
   echo "RAID device has errors"
#  Use the next 3 lines to BAIL OUT
   /bin/rm /etc/mtab            # remove bad mtab
   /bin/echo 0x100>/proc/sys/kernel/real-root-dev
   exit
fi

# else tell the kernel to switch to /dev/md0 as the /root device
# The 0x900 value the device number calculated by:
#  256*major_device_number + minor_device number
/bin/echo 0x900>/proc/sys/kernel/real-root-dev

# umount /proc to deallocate initrd device ram space
/bin/umount /proc
/bin/echo "/dev/md0 mounted as root"
exit
#----------------- end linuxrc ---------------------
```

11.2 Obsolete working − shutdown scripts

This shutdown procedure works fine with the preceeding **linuxrc**

To capture the raid array shutdown status, just before the file systems are dismounted insert:

```
        RAIDSTATUS=`/bin/cat /proc/mdstat | /usr/bin/grep md0`
```

After all the file systems are dismounted (the root file system 'will not' dismount) add:

```
        # root device remains mounted RO
        # mount dos file systems RW
        mount −n −o remount,ro /
        echo "Writing RAID read−only boot FLAG(s)."
        mount −n /dosa
        mount −n /dosc
        # create raid mounted RO flag in duplicate
        # containing the shutdown status of the raid array
        echo ${RAIDSTATUS} > /dosa/raidboot/raidstat.ro
        echo ${RAIDSTATUS} > /dosc/raidboot/raidstat.ro

        umount −n /dosa
        umount −n /dosc

        # Stop all the raid arrays (except root)
        echo "Stopping raid"
        mdstop −a
```

This will cleanly stop all raid devices except root. Root status is passed to the next boot in **raidstat.ro**.

The complete shutdown script from my old raid1 Slackware system follows, I have switched raid1 to the new procedure with the /etc/raidboot.conf file.

```
#! /bin/sh
#
# rc.6          This file is executed by init when it goes into runlevel
#               0 (halt) or runlevel 6 (reboot). It kills all processes,
#               unmounts file systems and then either halts or reboots.
#
# Version:      @(#)/etc/rc.d/rc.6      1.50    1994−01−15
#
# Author:       Miquel van Smoorenburg <miquels@drinkel.nl.mugnet.org>
# Modified by:  Patrick J. Volkerding, <volkerdi@ftp.cdrom.com>
# Modified by:  Michael A. Robinton, <michael@bzs.org> for RAID shutdown

  # Set the path.
  PATH=/sbin:/etc:/bin:/usr/bin

  # Set linefeed mode to avoid staircase effect.
  stty onlcr

  echo "Running shutdown script $0:"

  # Find out how we were called.
  case "$0" in
        *0)
                message="The system is halted."
                command="halt"
                ;;
        *6)
                message="Rebooting."
```

```
                command=reboot
                ;;
        *)
                echo "$0: call me as \"rc.0\" or \"rc.6\" please!"
                exit 1
                ;;
esac

# Kill all processes.
# INIT is supposed to handle this entirely now, but this didn't always
# work correctly without this second pass at killing off the processes.
# Since INIT already notified the user that processes were being killed,
# we'll avoid echoing this info this time around.
if [ "$1" != "fast" ]; then # shutdown did not already kill all processes
  killall5 -15
  killall5 -9
fi

# Try to turn off quota and accounting.
if [ -x /usr/sbin/quotaoff ]
then
      echo "Turning off quota."
      /usr/sbin/quotaoff -a
fi
if [ -x /sbin/accton ]
then
      echo "Turning off accounting."
      /sbin/accton
fi

# Before unmounting file systems write a reboot or halt record to wtmp.
$command -w

# Save localtime
[ -e /usr/lib/zoneinfo/localtime ] && cp /usr/lib/zoneinfo/localtime /etc

# Asynchronously unmount any remote filesystems:
echo "Unmounting remote filesystems."
umount -a -tnfs &

# you must have issued
# 'cat /proc/mdstat | grep md0 > {your boot vol}/raidboot/raidgood.ref'
# before linuxrc will execute properly with this info
RAIDSTATUS=`/bin/cat /proc/mdstat | /usr/bin/grep md0 # capture raid status`

# Turn off swap, then unmount local file systems.
# clearing mdtab as well
echo "Turning off swap."
swapoff -a
echo "Unmounting local file systems."
umount -a -tnonfs

# Don't remount UMSDOS root volumes:
if [ ! "`mount | head -1 | cut -d ' ' -f 5`" = "umsdos" ]; then
  mount -n -o remount,ro /
fi

# root device remains mounted
# mount dos file systems RW
echo "Writing RAID read-only boot FLAG(s)."
mount -n /dosa
mount -n /dosc
```

```
# create raid mounted RO flag in duplicate
# containing the shutdown status of the raid array
echo ${RAIDSTATUS} > /dosa/raidboot/raidstat.ro
echo ${RAIDSTATUS} > /dosc/raidboot/raidstat.ro

umount -n /dosa
umount -n /dosc

# Stop all the raid arrays (except root)
echo "Stopping raid"
mdstop -a

# See if this is a powerfail situation.
if [ -f /etc/power_is_failing ]; then
  echo "Turning off UPS, bye."
  /sbin/powerd -q
  exit 1
fi

# Now halt or reboot.
echo "$message"
[ ! -f /etc/fastboot ] && echo "On the next boot fsck will be FORCED."
$command -f
```

12.Appendix E. − Gadi's raid stop patch for the linux kernel

```
--- linux/drivers/block/md.c.old        Fri Nov 21 13:37:11 1997
+++ linux/drivers/block/md.c    Sat Dec  6 13:34:28 1997
@@ -622,8 +622,13 @@
       return do_md_run (minor, (int) arg);

     case STOP_MD:
-      return do_md_stop (minor, inode);
-
+      err = do_md_stop(minor, inode);
+      if (err) {
+        printk("md: enabling auto mdstop for %s\n",
         devname(inode->i_rdev));
+        md_dev[minor].auto_mdstop = 1;
+      }
+      return err;
+
     case BLKGETSIZE:   /* Return device size */
     if  (!arg)  return -EINVAL;
     err=verify_area (VERIFY_WRITE, (long *) arg, sizeof(long));
@@ -692,6 +697,10 @@

   sync_dev (inode->i_rdev);
   md_dev[minor].busy--;
+  if (!md_dev[minor].busy && md_dev[minor].auto_mdstop) {
+       do_md_stop(minor, inode);
+       md_dev[minor].auto_mdstop = 0;
+  }
 }
```

```
 static int md_read (struct inode *inode, struct file *file,
--- linux/include/linux/md.h~   Fri Nov 21 13:29:14 1997
+++ linux/include/linux/md.h    Fri Nov 21 13:29:14 1997
@@ -260,6 +260,7 @@
   int                   repartition;
   int                   busy;
   int                   nb_dev;
+  int                   auto_mdstop;
   void                  *private;
 };
```

13.Appendix F. – rc.raidown

Copy the following text into the script file **rc.raidown** and save it in **/etc/rc.d**.

```
#! /bin/sh
#
# rc.raidown    This file is executed by init when it goes into runlevel
#               0 (halt) or runlevel 6 (reboot). It saves the status of
#               a root mounted raid array for subsequent re-boot
#
# Version:      1.08    3-25-98 Michael A. Robinton < michael@bizsystems.com >
#
############ Save raid boot and status info ##############
if [ -f /etc/raidboot.conf ]
then
  {
  read RaidBootDevs
  read RaidStatusPath
  read RaidConfigEtc
  } < /etc/raidboot.conf

# you must have issued
#       cat /proc/mdstat | grep md0 >
#               {your boot vol mnt(s)}/{RaidStatusPath}/raidgood.ref
# before linuxrc will execute properly with this info
#
#       capture raid status
  RAIDSTATUS=`/bin/cat /proc/mdstat | /usr/bin/grep md0`
  mkdir /tmp/raid$$
  echo "Writing RAID read-only boot FLAG(s)."
  for Device in ${RaidBootDevs}
  do
# get mount point for raid boot device or use tmp
    RBmount=$( cat /proc/mounts | /usr/bin/grep ${Device} )
    if [ -n ${RBmounts} ]; then
      RBmount=$( echo ${RBmount} | cut -f 2 -d ' ' )
    else
      RBmount="/tmp/raid$$"
      mount ${Device} ${RBmount}
    fi
  if [ -d ${RBmount}/${RaidStatusPath} ]; then
# Create raid mounted RO flag = shutdown status of raid array
    echo ${RAIDSTATUS} > ${RBmount}/${RaidStatusPath}/raidboot.ro
# Don't propagate 'fstab' from ramdisk
    if [ -f /linuxrc ]; then
      FSTAB=
    else
```

```
      FSTAB=fstab
    fi
    pushd /etc
# Save etc files for rescue system
    /bin/tar --ignore-failed-read \
        -cf ${RBmount}/${RaidStatusPath}/raidboot.etc \
        raid*.conf mdtab* ${FSTAB} lilo.conf
    popd
# Create new raidboot.cfg
    {
    /bin/echo ${RaidBootDevs}
    /bin/echo ${RaidStatusPath}
    /bin/echo ${RaidConfigEtc}
    } > ${RBmount}/${RaidStatusPath}/raidboot.cfg
    /bin/umount ${RBmount}
  fi
  done
  rmdir /tmp/raid$$
  echo "Raid boot armed"
fi
################## end raid boot #########################
```

14.Appendix G. – linuxrc theory of operation

This is the complex form of the linuxrc file for root mounted raid. It must be processed with 'bash' or another shell that recognizes shell functions.

The advantage is that it is generic and is not dependent on startup files and parameters located in the **initrd** image.

A **Raid_Conf** parameter passed to **linuxrc** by the kernel at boot from lilo or loadlin contains a pointer to the boot devices and location the of initial 2 raidboot files needed by **linuxrc** (*raidboot.etc and raidboot.cfg placed by the shutdown script*).

raidboot.etc containing the 'tar'ed files:

```
raid*
mdtab*
fstab
lilo.conf               ( if applicable )
```

from the primary system that are transferred to the initrd **/etc**etc directory at startup. With care, this file may be edited if necessary when your system 'really' crashes.

raidboot.cfg contains the name of the boot partition in use and applicable backup(s) as well as the path to the rest of the raid start up file used by **linuxrc**. This file is normally created by the shutdown file and may be created manually if necessary.

raidboot.cfg is of the form, 3 lines – no comments

```
/dev/bootdev1 /dev/bootdev2 [/dev/bootdev3 ... and so on]
raid-status/path
name_of_raidX.conf_file
```

the **raid−status/path** does not include the name of the mountpoint

the **raidX.conf** filename is that one found in /etc and normally used for **ckraid** and **mkraid**.

The following additional files reside on the permanent raid boot partitions. This is usually the same as above, but in emergency situations may be loaded from anywhere they are available, such as a floppy boot disk.

- **raidgood.ref** created by the command cat /proc/mdstat | grep md0 > /{raid_status_path}/raidgood.ref

  See the shutdown scripts for saving this file and the next

- **raidstat.ro** created at each shutdown by the shutdown rc file, saving the exit status of the raid array.

15.Appendix H. Setting up ROOT RAID on RedHat

From the linux−raid@vger.rutgers.edu mail list.

```
!    Has anyone figured out how to do root-mounted RAID (as per
!    the Root-RAID HOWTO) using RedHat? The problem is that there
!    is no equivalent of Slackware's setup to install the root
!    filesystem to the RAID device. All RedHat installs have to
!    run from the install floppy, which makes it almost
!    impossible to get at the md devices and utilities during the
!    install.
!
!    I think it's much easier to go out of the distribution and do it by
!    hand!!

Assuming you have enough RAM (or a spare hard disk), install a minimal
system onto what will be your swap space (or onto your spare hard disk)
and/or /boot.  Now do your mkraid, your mke2fs, mdrun, and mount.  Next, do:

        tar clf - / | tar xpfC - /mnt/raidwasmountedhere

(you may want a "v" in the second tar's flags)
Once this is done, you can set up lilo (or whatever) so that the new
raid partition is root.  Then go in with RPM and/or glint (I hate
glint's behavior in the face of failed dependencies, which was fixed
but they broke it again for RH5.0 plus you can go back and forth
forever between an old and a new version of a package without
realizing the other version is installed) and install what you
really wanted.

All this assuming you couldn't sneak in at some point in the install
and do your mkraid then at the VC with the shell prompt...

!    I'm building a server at the moment and I think it would be tidier
!    and less likely to cause problems in the future if I start with
!    glibc2, rather than move to it later.
!
!    Me too.
!
!    The reason I'd like to be able to use RedHat is that they
```

```
!    are the only major distribution that I know of with a
!    glibc2-based release.
!
!    Debian works fine with me. There isn't a CD yet, but you can grab the
!    distribution by ftp.

I avoided root-raid like the plague, largely because initrd is an
extra, very fragile step (having to rdev, and having lilo depend on
the bios' ID number to find the kernel's partition, are bad enough!).
However, Red Hat does have a nice mkinitrd script, needed since they
left all their SCSI drivers modular.  Hack that to include your
raid utils, make sure your mdadd -ar is in the right spot in
/etc/rc.d/rc.sysinit (before any fscking) and make sure mdstop -a is
in /etc/rc.d/init.d/halt after the RO-remount of /, and go for it!
```

Keith kwrohrer@enteract.com
