

8:mdadm

mdadm - manage MD devices *aka* Linux Software Raid.

```
mdadm [mode] <raiddevice> [options] <component-devices>
```

Contents [1 DESCRIPTION](#) [2 MODES](#) [3 OPTIONS](#) [4 For create or build:](#) [5 For assemble:](#) [6 For Manage mode:](#) [7 For Examine mode:](#) [8 For Misc mode:](#) [9 For Monitor mode:](#) [10 ASSEMBLE MODE](#) [11 BUILD MODE](#) [12 CREATE MODE](#) [13 MANAGE MODE](#) [14 MISC MODE](#) [15 MONITOR MODE](#) [16 GROW MODE](#) [17 EXAMPLES](#) [18 FILES](#) [19 /proc/mdstat](#) [20 /etc/mdadm.conf](#) [20.1 DEVICE NAMES](#) [20.2 NOTE](#) [20.3 RELATED](#) [20.4 CATEGORY](#)

DESCRIPTION

RAID devices are virtual devices created from two or more real block devices. This allows multiple devices (typically disk drives or partitions there-of) to be combined into a single device to hold (for example) a single filesystem. Some RAID levels include redundancy and so can survive some degree of device failure.

Linux Software RAID devices are implemented through the md (Multiple Devices) device driver.

Currently, Linux supports **LINEAR** md devices, **RAID0** (striping), **RAID1** (mirroring), **RAID4**, **RAID5**, **RAID6**, **MULTIPATH**, and **FAULTY**.

MULTIPATH is not a Software RAID mechanism, but does involve multiple devices. For **MULTIPATH** each device is a path to one common physical storage device.

FAULTY is also not true RAID, and it only involves one device. It provides a layer over a true device that can be used to inject faults.

MODES

mdadm has 7 major modes of operation:

Assemble

Assemble the parts of a previously created array into an active array. Components can be explicitly given or can be searched for. **mdadm** checks that the components do form a bona fide array, and can, on request, fiddle superblock information so as to assemble a faulty array.

Build Build an array that doesn't have per-device superblocks. For these sorts of arrays, **mdadm** cannot differentiate between initial creation and subsequent assembly of an array. It also cannot perform any checks that appropriate devices have been requested. Because of this, the **Build** mode should only be used together with a complete understanding of what you are doing.

Create Create a new array with per-device superblocks.

Manage This is for doing things to specific components of an array such as adding new spares and removing faulty devices.

Misc This mode allows operations on independent devices such as examine MD superblocks, erasing old superblocks and stopping active arrays.

Follow or Monitor

Monitor one or more md devices and act on any state changes. This is only meaningful for raid1, 4, 5, 6 or multipath arrays as only these have interesting state. raid0 or linear never have missing, spare, or failed drives, so there is nothing to monitor.

Grow Grow (or shrink) an array, or otherwise reshape it in some way. Currently supported growth options including changing the active size of component devices in RAID level 1/4/5/6 and changing the number of active devices in RAID1.

OPTIONS

Available options are:

-A, --assemble

Assemble a pre-existing array.

-B, --build

Build a legacy array without superblocks.

-C, --create

Create a new array.

-Q, --query

Examine a device to [see\(1\)](#) if it is an md device [and\(2\)](#) if it is a component of an md array. Information about what is discovered is presented.

-D, --detail

Print detail of one or more md devices.

- E, --examine**
Print content of md superbblock on device(s).
- F, --follow, --monitor**
Select **Monitor** mode.
- G, --grow**
Change the size or shape of an active array.
- h, --help**
Display general help message or, after one of the above options, a mode specific help message.
- help-options**
Display more detailed help about command line parsing and some commonly used options.
- V, --version**
Print version information for mdadm.
- v, --verbose**
Be more verbose about what is happening. This can be used twice to be extra-verbose. The extra verbosity currently only affects **--detail --scan** and **--examine --scan**.
- b, --brief**
Be less verbose. This is used with **--detail** and **--examine**. Using **--brief** with **--verbose** gives an intermediate level of verbosity.
- f, --force**
Be more forceful about certain operations. See the various modes of the exact meaning of this option in different contexts.
- c, --config=**
Specify the config file. Default is **/etc/mdadm.conf**. If the config file given is **partitions** then nothing will be read, but **mdadm** will act as though the config file contained exactly **DEVICE partitions** and will read **/proc/partitions** to find a list of devices to scan. If the word **none** is given for the config file, then **mdadm** will act as though the config file were empty.
- s, --scan**
scan config file or **/proc/mdstat** for missing information. In general, this option gives **mdadm** permission to get any missing information, like component devices, array devices, array identities, and alert destination from the configuration file: **/etc/mdadm.conf**. One exception is MISC mode when using **--detail** or **--stop** in which case **--scan** says to get a list of array devices from **/proc/mdstat**.

For create or build:

- c, --chunk=**
Specify chunk size of kibibytes. The default is 64.

--rounding=

Specify rounding factor for linear array (==chunk size)

-l, --level=

Set raid level. When used with **--create**, options are: linear, raid0, 0, stripe, raid1, 1, mirror, raid4, 4, raid5, 5, raid6, 6, multipath, mp, faulty. Obviously some of these are synonymous.

When used with **--build**, only linear, stripe, raid0, 0, raid1, multipath, mp, and faulty are valid.

-p, --parity=

Set raid5 parity algorithm. Options are: left-asymmetric, left-symmetric, right-asymmetric, right-symmetric, la, ra, ls, rs. The default is left-symmetric.

This option is also used to set the failure mode for **faulty**. The options are: write-transient, wt, read-transient, rt, write-persistent, wp, read-persistent, rp, write-all, read-fixable, rf, clear, flush, none.

Each mode can be followed by a number which is used as a period between fault generation. Without a number, the fault is generated once on the first relevant request. With a number, the fault will be generated after that many request, and will continue to be generated every time the period elapses.

Multiple failure modes can be current simultaneously by using the "--grow" option to set subsequent failure modes.

"clear" or "none" will remove any pending or periodic failure modes, and "flush" will clear any persistent faults.

To set the parity with "--grow", the level of the array ("faulty") must be specified before the fault mode is specified.

--layout=

same as --parity

-n, --raid-devices=

Specify the number of active devices in the array. This, plus the number of spare devices (see below) must equal the number of **component-devices** (including "missing" devices) that are listed on the command line for **--create**. Setting a value of 1 is probably a mistake and so requires that **--force** be specified first. A value of 1 will then be allowed for linear, multipath, raid0 and raid1. It is never allowed for raid4 or raid5.

This number can only be changed using **--grow** for RAID1 arrays, and only on kernels which provide necessary support.

-x, --spare-devices=

Specify the number of spare (eXtra) devices in the initial array. Spares can also be added and removed later. The number of component devices listed on the command line must equal the number of raid devices plus the number of spare devices.

-z, --size=

Amount (in Kibibytes) of space to use from each drive in RAID1/4/5/6. This must be a multiple of the chunk size, and must leave about 128Kb of space at the end of the drive for the RAID superblock. If this is not specified (as it normally is not) the smallest drive (or partition) sets the size, though if there is a variance among the drives of greater than 1%, a warning is issued.

This value can be set with **--grow** for RAID level 1/4/5/6. If the array was created with a size smaller than the currently active drives, the extra space can be accessed using **--grow**. The size can be given as **max** which means to choose the largest size that fits on all current drives.

--assume-clean

Tell **mdadm** that the array pre-existed and is known to be clean. This is only really useful for Building RAID1 array. Only use this if you really know what you are doing. This is currently only supported for **--build**.

-R, --run

Insist that **mdadm** run the array, even if some of the components appear to be active in another array or filesystem. Normally **mdadm** will ask for confirmation before including such components in an array. This option causes that question to be suppressed.

-f, --force

Insist that *mdadm* accept the geometry and layout specified without question. Normally *mdadm* will not allow creation of an array with only one device, and will try to create a raid5 array with one missing drive (as this makes the initial resync work faster). With **--force**, *mdadm* will not try to be so clever.

-a, --auto{=no,yes,md,mdp,part,p}{NN}

Instruct *mdadm* to create the device file if needed, possibly allocating an unused minor number. "md" causes a non-partitionable array to be used. "mdp", "part" or "p" causes a partitionable array (2.6 and later) to be used. "yes" requires the named md device to have a from this. See DEVICE NAMES below.

The argument can also come immediately after "-a". e.g. "-ap".

If **--scan** is also given, then any **auto=** entries in the config file will over-ride the **--auto** instruction given on the command line.

For partitionable arrays, *mdadm* will create the device file for the whole array and for the first 4 partitions. A different number of partitions can be specified at the end of this option (e.g. **--auto=p7**). If the device name ends with a digit, the partition names add a 'p', and a number, e.g. "/dev/homelp3". If there is no trailing digit, then the partition names just have a number added, e.g. "/dev/scratch3".

If the md device name is in a 'standard' format as described in DEVICE NAMES, then it will be created, if necessary, with the appropriate number based on that name. If the device name is not in one of these formats, then a unused minor number will be allocated. The minor number will be considered unused if there is no active array for that number, and there is no entry in /dev for that number and with a non-standard name.

For assemble:

-u, --uuid=

uuid of array to assemble. Devices which don't have this uuid are excluded

-m, --super-minor=

Minor number of device that array was created for. Devices which don't have this minor number are excluded. If you create an array as /dev/md1, then all superblocks will contain the minor number 1, even if the array is later assembled as /dev/md2.

Giving the literal word "dev" for **--super-minor** will cause *mdadm* to use the minor number of the md device that is being assembled. e.g. when assembling /dev/md0, will look for super blocks with a minor number of 0.

-f, --force

Assemble the array even if some superblocks appear out-of-date

-R, --run

Attempt to start the array even if fewer drives were given than are needed for a full array. Normally if not all drives are found and **--scan** is not used, then the array will be assembled but not started. With **--run** an attempt will be made to start it anyway.

-a, --auto{=no,yes,md,mdp,part}

See this option under Create and Build options.

-U, --update=

Update the superblock on each device while assembling the array. The argument given to this flag can be one of **sparc2.2**, **summaries**, **resync**, or **super-minor**.

The **sparc2.2** option will adjust the superblock of an array what was created on a Sparc machine running a patched 2.2 Linux kernel. This kernel got the alignment of part of the superblock wrong. You can use the **--examine --sparc2.2** option to **mdadm** to see what effect this would have.

The **super-minor** option will update the **preferred minor** field on each superblock to match the minor number of the array being assembled. This is not needed on 2.6 and later kernels as they make this adjustment automatically.

The **resync** option will cause the array to be marked **dirty** meaning that any redundancy in the array (e.g. parity for raid5, copies for raid1) may be incorrect. This will cause the raid system to perform a "resync" pass to make sure that all redundant information is correct.

The **summaries** option will correct the summaries in the superblock. That is the counts of total, working, active, failed, and spare devices.

For Manage mode:

-a, --add

hotadd listed devices.

-r, --remove

remove listed devices. They must not be active. i.e. they should be failed or spare devices.

-f, --fail, --set-faulty

mark listed devices as faulty.

--clean

mark listed devices as clean. (Only supported by multipath)

--active

mark listed devices as active. (Only supported by multipath)

--inactive

mark listed devices as inactive. (Only supported by multipath)

For Examine mode:

--sparc2.2

If an array was created on a 2.2 Linux kernel patched with RAID support, the superblock will have been created incorrectly, or at least incompatibly with 2.4 and later kernels. Using the **--sparc2.2** flag with **--examine** will fix the superblock before displaying it. If this appears to do the right thing, then the array can be successfully assembled using **--assemble --update=sparc2.2**.

For Misc mode:

-R, --run

start a partially built array.

-S, --stop

deactivate array, releasing all resources.

-o, --readonly

mark array as readonly.

-w, --readwrite

mark array as readwrite.

--zero-superblock

If the device contains a valid md superblock, the block is over-written with zeros. With **--force** the block where the superblock would be is over-written even if it doesn't appear to be valid.

-t, --test

When used with **--detail**, the exit status of **mdadm** is set to reflect the status of the device.

For Monitor mode:

-m, --mail

Give a mail address to send alerts to.

-p, --program, --alert

Give a program to be run whenever an event is detected.

-d, --delay

Give a delay in seconds. **mdadm** polls the md arrays and then waits this many seconds before polling again. The default is 60 seconds.

-f, --daemonise

Tell **mdadm** to run as a background daemon if it decides to monitor anything. This causes it to fork and run in the child, and to disconnect from the terminal. The process id of the child is written to stdout. This is useful with **--scan** which will only continue monitoring if a mail address or alert program is found in the config file.

-i, --pid-file

When **mdadm** is running in daemon mode, write the pid of the daemon process to the specified file, instead of printing it on standard output.

-1, --oneshot

Check arrays only once. This will generate **NewArray** events and more significantly **DegradedArray** and **SparesMissing** events. Running

```
mdadm --monitor --scan -1
```

from a cron script will ensure regular notification of any degraded arrays.

-t, --test

Generate a **TestMessage** alert for every array found at startup. This alert gets mailed and passed to the alert program. This can be used for testing that alert message do get through successfully.

ASSEMBLE MODE

Usage: **mdadm --assemble md-device options-and-component-devices...**

Usage: **mdadm --assemble --scan md-devices-and-options...**

Usage: **mdadm --assemble --scan options...**

This usage assembles one or more raid arrays from pre-existing components. For each array, **mdadm** needs to know the md device, the identity of the array, and a number of component-devices. These can be found in a number of ways.

In the first usage example (without the **--scan**) the first device given is the md device. In the second usage example, all devices listed are treated as md devices and assembly is attempted. In the third (where no devices are listed) all md devices that are listed in the configuration file are assembled.

If precisely one device is listed, but **--scan** is not given, then **mdadm** acts as though **--scan** was given and identify information is extracted from the configuration file.

The identity can be given with the **--uuid** option, with the **--super-minor** option, can be found in the config file, or will be taken from the super block on the first component-device listed on the command line.

Devices can be given on the **--assemble** command line or in the config file. Only devices which have an md superblock which contains the right identity will be considered for any array.

The config file is only used if explicitly named with **--config** or requested with (a possibly implicit) **--scan**. In the later case, **/etc/mdadm.conf** is used.

If **--scan** is not given, then the config file will only be used to find the identity of md arrays.

Normally the array will be started after it is assembled. However if **--scan** is not given and insufficient drives were listed to start a complete (non-degraded) array, then the array is not started (to guard against usage errors). To insist that the array be started in this case (as may work for RAID1, 4, 5 or 6), give the **--run** flag.

If an **auto** option is given, either on the command line (**--auto**) or in the configuration file (e.g. **auto=part**), then **mdadm** will create the md device if necessary or will re-create it if it doesn't look usable as it is.

This can be useful for handling partitioned devices (which don't have a stable device number - it can change after a reboot) and when using "udev" to manage your **/dev** tree (udev cannot handle md devices because of the unusual device initialisation conventions).

If the option to "auto" is "mdp" or "part" or (on the command line only) "p", then mdadm will create a partitionable array, using the first free one that is not in use, and does not already have an entry in /dev (apart from numeric /dev/md* entries).

If the option to "auto" is "yes" or "md" or (on the command line) nothing, then mdadm will create a traditional, non-partitionable md array.

It is expected that the "auto" functionality will be used to create device entries with meaningful names such as "/dev/md/home" or "/dev/md/root", rather than names based on the numerical array number.

When using this option to create a partitionable array, the device files for the first 4 partitions are also created. If a different number is required it can be simply appended to the auto option. e.g. "auto=part8". Partition names are created by appending a digit string to the device name, with an intervening "p" if the device name ends with a digit.

The **--auto** option is also available in Build and Create modes. As those modes do not use a config file, the "auto=" config option does not apply to these modes.

BUILD MODE

Usage: **mdadm --build device --chunk=X --level=Y --raid-devices=Z devices**

This usage is similar to **--create**. The difference is that it creates an array without a superblock. With these arrays there is no difference between initially creating the array and subsequently assembling the array, except that hopefully there is useful data there in the second case.

The level may raid0, linear, multipath, or faulty, or one of their syn-

onyms. All devices must be listed and the array will be started once complete.

CREATE MODE

Usage: **mdadm --create device --chunk=X --level=Y
--raid-devices=Z devices**

This usage will initialise a new md array, associate some devices with it, and activate the array.

If the **--auto** option is given (as described in more detail in the section on Assemble mode), then the md device will be created with a suitable device number if necessary.

As devices are added, they are checked to see if they contain raid superblocks or filesystems. They are also checked to see if the variance in device size exceeds 1%.

If any discrepancy is found, the array will not automatically be run, though the presence of a **--run** can override this caution.

To create a "degraded" array in which some devices are missing, simply give the word **"missing"** in place of a device name. This will cause **mdadm** to leave the corresponding slot in the array empty. For a RAID4 or RAID5 array at most one slot can be **"missing"**; for a RAID6 array at most two slots. For a RAID1 array, only one real device needs to be given. All of the others can be **"missing"**.

When creating a RAID5 array, **mdadm** will automatically create a degraded array with an extra spare drive. This is because building the spare into a degraded array is in general faster than resyncing the parity on a non-degraded, but not clean, array. This feature can be over-ridden with the **--force** option.

The General Management options that are valid with **--create** are:

--run insist on running the array even if some devices look like they might be in use.

--readonly

start the array readonly - not supported yet.

MANAGE MODE

Usage: **mdadm device options... devices...**

This usage will allow individual devices in an array to be failed, removed or added. It is possible to perform multiple operations with on command. For example:

```
mdadm /dev/md0 -f /dev/hda1 -r /dev/hda1 -a /dev/hda1
```

will firstly mark **/dev/hda1** as faulty in **/dev/md0** and will then remove it from the array and finally add it back in as a spare. However only one md array can be affected by a single command.

MISC MODE

Usage: **mdadm options ... devices ...**

MISC mode includes a number of distinct operations that operate on distinct devices. The operations are:

--query

The device is examined to see if it is(1) an active md array, or(2) a component of an md array. The information discovered is reported.

--detail

The device should be an active md device. **mdadm** will display a detailed description of the array. **--brief** or **--scan** will cause the output to be less detailed and the format to be suitable for inclusion in **/etc/mdadm.conf**. The exit status of **mdadm** will normally be 0 unless **mdadm** failed to get useful information about the device(s). However if the **--test** option is given, then the exit status will be:

0 The array is functioning normally.

1 The array has at least one failed device.

2 The array has multiple failed devices and hence is unusable (raid4 or raid5).

4 There was an error while trying to get information about the device.

--examine

The device should be a component of an md array. **mdadm** will read the md superblock of the device and display the contents. If **--brief** is given, or **--scan** then multiple devices that are components of the one array are grouped together and reported in a single entry suitable for inclusion in **/etc/mdadm.conf**.

Having **--scan** without listing any devices will cause all devices listed in the config file to be examined.

--stop The devices should be active md arrays which will be deactivated, as long as they are not currently in use.

--run This will fully activate a partially assembled md array.

--readonly

This will mark an active array as read-only, providing that it is not currently being used.

--readwrite

This will change a **readonly** array back to being read/write.

--scan For all operations except **--examine**, **--scan** will cause the operation to be applied to all arrays listed in **/proc/mdstat**. For **--examine**, **--scan** causes all devices listed in the config file to be examined.

MONITOR MODE

Usage: **mdadm --monitor options... devices...**

This usage causes **mdadm** to periodically poll a number of md arrays and to report on any events noticed. **mdadm** will never exit once it decides that there are arrays to be checked, so it should normally be run in the background.

As well as reporting events, **mdadm** may move a spare drive from one array to another if they are in the same **spare-group** and if the destination array has a failed drive but no spares.

If any devices are listed on the command line, **mdadm** will only monitor those devices. Otherwise all arrays listed in the configuration file will be monitored. Further, if **--scan** is given, then any other md devices that appear in **/proc/mdstat** will also be monitored.

The result of monitoring the arrays is the generation of events. These events are passed to a separate program (if specified) and may be mailed to a given E-mail address.

When passing event to program, the program is run once for each event and is given 2 or 3 command-line arguments. The first is the name of the event (see below). The second is the name of the md device which is affected, and the third is the name of a related device if relevant, such as a component device that has failed.

If **--scan** is given, then a program or an E-mail address must be specified on the command line or in the config file. If neither are available, then **mdadm** will not monitor anything. Without **--scan** **mdadm** will continue monitoring as long as something was found to monitor. If no program or email is given, then each event is reported to **stdout**.

The different events are:

DeviceDisappeared

An md array which previously was configured appears to no longer be configured.

If **mdadm** was told to monitor an array which is RAID0 or Linear, then it will report **DeviceDisappeared** with the extra information **Wrong-Level**. This is because RAID0 and Linear do not support the device-failed, hot-spare and resync operations which are monitored.

RebuildStarted

An md array started reconstruction.

RebuildNN

Where **NN** is 20, 40, 60, or 80, this indicates that rebuild has passed that many percentage of the total.

RebuildFinished

An md array that was rebuilding, isn't any more, either because it finished normally or was aborted.

Fail An active component device of an array has been marked as faulty.

FailSpare

A spare component device which was being rebuilt to replace a faulty device has failed.

SpareActive

A spare component device which was being rebuilt to replace a faulty device as been successfully rebuild and has been made active.

NewArray

A new md array has been detected in the `/proc/mdstat` file.

DegradedArray

A newly noticed array appears to be degraded. This message is not generated when **mdadm** notices a drive failure which causes degradation, but only when **mdadm** notices that an array is degraded when it first sees the array.

MoveSpare

A spare drive has been moved from one array in a **spare-group** to another to allow a failed drive to be replaced.

SparesMissing

If **mdadm** has been told, via the config file, that an array should have a certain number of spare devices, and **mdadm** detects that it has fewer than this number when it first sees the array, it will report a **SparesMissing** message.

TestMessage

An array was found at startup, and the `--test` flag was given.

Only **Fail** , **FailSpare** , **DegradedArray** , and **TestMessage** cause Email to be sent. All events cause the program to be run. The program is run with two or three arguments, they being the event name, the array device and possibly a second device.

Each event has an associated array device (e.g. `/dev/md1`) and possibly

a second device. For **Fail**, **FailSpare**, and **SpareActive** the second device is the relevant component device. For **MoveSpare** the second device is the array that the spare was moved from.

For **mdadm** to move spares from one array to another, the different arrays need to be labelled with the same **spare-group** in the configuration file. The **spare-group** name can be any string. It is only necessary that different spare groups use different names.

When **mdadm** detects that an array which is in a spare group has fewer active devices than necessary for the complete array, and has no spare devices, it will look for another array in the same spare group that has a full complement of working drive and a spare. It will then attempt to remove the spare from the second drive and add it to the first. If the removal succeeds but the adding fails, then it is added back to the original array.

GROW MODE

The GROW mode is used for changing the size or shape of an active array. For this to work, the kernel must support the necessary change. Various types of growth may be added during 2.6 development, possibly including restructuring a raid5 array to have more active devices.

Currently the only support available is to change the "size" attribute for arrays with redundancy, and the raid-disks attribute of RAID1 arrays.

Normally when an array is built the "size" is taken from the smallest of the drives. If all the small drives in an array are, one at a time, removed and replaced with larger drives, then you could have an array of large drives with only a small amount used. In this situation, changing the "size" with "GROW" mode will allow the extra space to start being used. If the size is increased in this way, a "resync" process will start to make sure the new parts of the array are synchronised.

Note that when an array changes size, any filesystem that may be stored in the array will not automatically grow to use the space. The filesystem will need to be explicitly told to use the extra space.

A RAID1 array can work with any number of devices from 1 upwards (though 1 is not very useful). There may be times which you want to increase or decrease the number of active devices. Note that this is different to hot-add or hot-remove which changes the number of inactive devices.

When reducing the number of devices in a RAID1 array, the slots which are to be removed from the array must already be vacant. That is, the devices that were in those slots must be failed and removed.

When the number of devices is increased, any hot spares that are present will be activated immediately.

EXAMPLES

```
mdadm --query /dev/name-of-device
```

This will find out if a given device is a raid array, or is part of one, and will provide brief information about the device.

mdadm --assemble --scan

This will assemble and start all arrays listed in the standard confile file. This command will typically go in a system startup file.

mdadm --stop --scan

This will shut down all array that can be shut down (i.e. are not currently in use). This will typically go in a system shutdown script.

mdadm --follow --scan --delay=120

If (and only if) there is an Email address or program given in the standard config file, then monitor the status of all arrays listed in that file by polling them ever 2 minutes.

mdadm --create /dev/md0 --level=1 --raid-devices=2 /dev/hd[ac]1

Create /dev/md0 as a RAID1 array consisting of /dev/hda1 and /dev/hdc1.

echo 'DEVICE /dev/hd*[0-9] /dev/sd*[0-9]' > mdadm.conf

mdadm --detail --scan >> mdadm.conf

This will create a prototype config file that describes currently active arrays that are known to be made from partitions of IDE or SCSI drives. This file should be reviewed before being used as it may contain unwanted detail.

echo 'DEVICE /dev/hd[a-z] /dev/sd*[a-z]' > mdadm.conf

mdadm --examine --scan --config=mdadm.conf >> mdadm.conf This will find what arrays could be assembled from existign IDE and SCSI whole drives (not partitions) and store the information is the format of a config file. This file is very likely to contain unwanted detail, particularly the **devices=** entries. It should be reviewed and edited before being used as an actual config file.

mdadm --examine --brief --scan --config=partitions

mdadm -Ebsc partitions

Create a list of devices by reading **/proc/partitions**, scan these for RAID superblocks, and printout a brief listing of all that was found.

mdadm -Ac partitions -m 0 /dev/md0

Scan all partitions and devices listed in **/proc/partitions** and assemble **/dev/md0** out of all such devices with a RAID superblock with a minor number of 0.

mdadm --monitor --scan --daemonise > /var/run/mdadm

If config file contains a mail address or alert program, run mdadm in the background in monitor mode monitoring all md devices. Also write pid of mdadm daemon to **/var/run/mdadm**.

mdadm --create --help

Providew help about the Create mode.

mdadm --config --help

Provide help about the format of the config file.

mdadm --help

Provide general help.

FILES /proc/mdstat

If you're using the **/proc** filesystem, **/proc/mdstat** lists all active md devices with information about them. **mdadm** uses this to find arrays when **--scan** is given in Misc mode, and to monitor array reconstruction on Monitor mode.