NAME

gmirror - control utility for mirrored devices

SYNOPSIS

To compile GEOM_MIRROR into your kernel, add the following lines to your kernel configuration file:

```
options GEOM_MIRROR
```

Alternatively, to load the GEOM_MIRROR module at boot time, add the following line to your loader.conf(5):

```
geom_mirror_load="YES"
```

Usage of the **gmirror** utility:

gmirror label [**-Fhnv**] [**-b** *balance*] [**-s** *slice*] *name prov ...*
gmirror clear [**-v**] *prov ...*
gmirror create [**-Fnv**] [**-b** *balance*] [**-s** *slice*] *name prov ...*
gmirror configure [**-adfFhnv**] [**-b** *balance*] [**-s** *slice*] *name*
gmirror configure [**-v**] **-p** *priority name prov*
gmirror rebuild [**-v**] *name prov ...*
gmirror resize [**-v**] [**-s** *size*] *name*
gmirror insert [**-hiv**] [**-p** *priority*] *name prov ...*
gmirror remove [**-v**] *name prov ...*
gmirror activate [**-v**] *name prov ...*
gmirror deactivate [**-v**] *name prov ...*
gmirror destroy [**-fv**] *name ...*
gmirror forget [**-v**] *name ...*
gmirror stop [**-fv**] *name ...*
gmirror dump *prov ...*
gmirror list
gmirror status
gmirror load
gmirror unload

DESCRIPTION

The **gmirror** utility is used for mirror (RAID1) configurations. After a mirror's creation, all components are detected and configured automatically. All operations like failure detection, stale component detection, rebuild of stale components, etc. are also done automatically. The **gmirror** utility uses on-disk metadata (stored in the provider's last sector) to store all needed information. Since the last sector is used for this purpose, it is possible to place a root file system on a mirror.

The first argument to **gmirror** indicates an action to be performed:

label Create a mirror. The order of components is important, because a component's priority is based on its position (starting from 0 to 255). The component with the biggest priority is used by the **prefer** balance algorithm and is also used as a master component when resynchronization is needed, e.g. after a power failure when the device was open for writing.

Additional options include:

-b *balance* Specifies balance algorithm to use, one of:

load Read from the component with the lowest load. This is the default balance algorithm.

prefer Read from the component with the biggest priority.

round-robin Use round-robin algorithm when choosing component to read.

split Split read requests, which are bigger than or equal to slice size on N pieces, where N is the number of active components.

-F Do not synchronize after a power failure or system crash. Assumes device is in consistent state.

-h Hardcode providers' names in metadata.

-n Turn off autosynchronization of stale components.

-s *slice* When using the **split** balance algorithm and an I/O READ request is bigger than or equal to this value, the I/O request will be split into N pieces, where N is the number of active components. Defaults to 4096 bytes.

clear Clear metadata on the given providers.

create Similar to **label**, but creates mirror without storing on-disk metadata in last sector. This special "manual" operation mode assumes some external control to manage mirror detection after reboot, device hot-plug and other external events.

configure Configure the given device.

Additional options include:

-a Turn on autosynchronization of stale components.

-b *balance* Specifies balance algorithm to use.

-d Do not hardcode providers’ names in metadata.

-f Synchronize device after a power failure or system crash.

-F Do not synchronize after a power failure or system crash. Assumes device is in consistent state.

-h Hardcode providers’ names in metadata.

-n Turn off autosynchronization of stale components.

-p *priority* Specifies priority for the given component *prov*.

-s *slice* Specifies slice size for **split** balance algorithm.

rebuild Rebuild the given mirror components forcibly. If autosynchronization was not turned off for the given device, this command should be unnecessary.

resize Change the size of the given mirror.

Additional options include:

-s *size* New size of the mirror is expressed in logical block numbers. This option can be omitted, then it will be automatically calculated to maximum available size.

insert Add the given component(s) to the existing mirror.

Additional options include:

-h Hardcode providers’ names in metadata.

-i Mark component(s) as inactive immediately after insertion.

-p *priority* Specifies priority of the given component(s).

remove Remove the given component(s) from the mirror and clear metadata on it.

activate Activate the given component(s), which were marked as inactive before.

deactivate Mark the given component(s) as inactive, so it will not be automatically connected to the mirror.

destroy Stop the given mirror and clear metadata on all its components.

Additional options include:

-f
Stop the given mirror even if it is opened.

forget Forget about components which are not connected. This command is useful when a disk has failed and cannot be reconnected, preventing the **remove** command from being used to remove it.

stop Stop the given mirror.

Additional options include:

-f
Stop the given mirror even if it is opened.

dump Dump metadata stored on the given providers.

list See geom(8).

status See geom(8).

load See geom(8).

unload See geom(8).

Additional options include:

-v

Be more verbose.

EXIT STATUS

Exit status is 0 on success, and 1 if the command fails.

EXAMPLES

Use 3 disks to setup a mirror. Choose split balance algorithm, split only requests which are bigger than or equal to 2kB. Create file system, mount it, then unmount it and stop device:

```
gmirror label -v -b split -s 2048 data da0 da1 da2
newfs /dev/mirror/data
mount /dev/mirror/data /mnt
...
umount /mnt
gmirror stop data
gmirror unload
```

Create a mirror on disk with valid data (note that the last sector of the disk will be overwritten). Add another disk to this mirror, so it will be synchronized with existing disk:

```
gmirror label -v -b round-robin data da0
gmirror insert data da1
```

Create a mirror, but do not use automatic synchronization feature. Add another disk and rebuild it:

```
gmirror label -v -n -b load data da0 da1
gmirror insert data da2
gmirror rebuild data da2
```

One disk failed. Replace it with a brand new one:

```
gmirror forget data
gmirror insert data da1
```

Create a mirror, deactivate one component, do the backup and connect it again. It will not be resynchronized, if there is no need to do so (there were no writes in the meantime):

```
gmirror label data da0 da1
gmirror deactivate data da1
dd if=/dev/da1 of=/backup/data.img bs=1m
```

```
gmirror activate data da1
```

SYSCTL VARIABLES

The following sysctl(8) variables can be used to configure behavior for all mirrors.

kern.geom.mirror.debug
: Control the verbosity of kernel logging related to mirrors. A value larger than 0 will enable debug logging.

kern.geom.mirror.timeout
: The amount of time, in seconds, to wait for all copies of a mirror to appear before starting the mirror. Disks that appear after the mirror has been started are not automatically added to the mirror.

kern.geom.mirror.idletime
: The amount of time, in seconds, which must elapse after the last write to a mirror before that mirror is marked clean. Clean mirrors do not need to be synchronized after a power failure or system crash. A small value may result in frequent overwrites of the disks’ metadata sectors, and thus may reduce the longevity of the disks.

kern.geom.mirror.disconnect_on_failure
: Determine whether a disk is automatically removed from its mirror when an I/O request to that disk fails.

kern.geom.mirror.sync_requests
: The number of parallel I/O requests used while synchronizing a mirror. This parameter may only be configured as a loader.conf(5) tunable.

kern.geom.mirror.sync_update_period
: The period, in seconds, at which a synchronizing mirror’s metadata is updated. Periodic updates are used to record a synchronization’s progress so that an interrupted synchronization may be resumed starting at the recorded offset, rather than at the beginning. A smaller value results in more accurate progress tracking, but also increases the number of non-sequential writes to the disk being synchronized. If the sysctl value is 0, no updates are performed until the synchronization is complete.

NOTES

Doing kernel dumps to **gmirror** providers is possible, but some conditions have to be met. First of all, a kernel dump will go only to one component and **gmirror** always chooses the component with the highest priority. Reading a dump from the mirror on boot will only work if the **prefer** balance algorithm is used (that way **gmirror** will read only from the component with the highest priority). If you use a different balance algorithm, you should create an rc(8) script that sets the balance algorithm to **prefer**, for example with the following command:

```
gmirror configure -b prefer data
```

Make sure that rcorder(8) schedules the new script before savecore(8). The desired balance algorithm can be restored later on by placing the following command in rc.local(8):

```
gmirror configure -b round-robin data
```

The decision which component to choose for dumping is made when dumpon(8) is called. If on the next boot a component with a higher priority will be available, the prefer algorithm will choose to read from it and savecore(8) will find nothing. If on the next boot a component with the highest priority will be synchronized, the prefer balance algorithm will read from the next one, thus will find nothing there.

SEE ALSO

geom(4), dumpon(8), geom(8), gvinum(8), mount(8), newfs(8), savecore(8), sysctl(8), umount(8)

HISTORY

The **gmirror** utility appeared in FreeBSD 5.3.

AUTHORS

Pawel Jakub Dawidek <*pjd@FreeBSD.org*>

BUGS

There should be a way to change a component’s priority inside a running mirror.

There should be a section with an implementation description.