NAME

gjournal **-** control utility for journaled devices

SYNOPSIS

gjournal label [**-cfhv**] [**-s** *jsize*] *dataprov* [*jprov*]
gjournal stop [**-fv**] *name ...*
gjournal sync [**-v**]
gjournal clear [**-v**] *prov ...*
gjournal dump *prov ...*
gjournal list
gjournal status
gjournal load
gjournal unload

DESCRIPTION

The **gjournal** utility is used for journal configuration on the given GEOM provider. The Journal and data may be stored on the same provider or on two separate providers. This is block level journaling, not file system level journaling, which means everything gets logged, e.g. for file systems, it journals both data and metadata. The **gjournal** GEOM class can talk to file systems, which allows the use of **gjournal** for file system journaling and to keep file systems in a consistent state. At this time, only UFS file system is supported.

To configure journaling on the UFS file system using **gjournal**, one should first create a **gjournal** provider using the **gjournal** utility, then run newfs(8) or tunefs(8) on it with the **-J** flag which instructs UFS to cooperate with the **gjournal** provider below. There are important differences in how journaled UFS works. The most important one is that sync(2) and fsync(2) system calls do not work as expected anymore. To ensure that data is stored on the data provider, the **gjournal sync** command should be used after calling sync(2). For the best performance possible, soft-updates should be disabled when **gjournal** is used. It is also safe and recommended to use the **async** mount(8) option.

When **gjournal** is configured on top of gmirror(8) or graid3(8) providers, it also keeps them in a consistent state, thus automatic synchronization on power failure or system crash may be disabled on those providers.

The **gjournal** utility uses on-disk metadata, stored in the provider's last sector, to store all needed information. This could be a problem when an existing file system is converted to use **gjournal**.

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

label Configures **gjournal** on the given provider(s). If only one provider is given, both data and journal are stored on the same provider. If two providers are given, the first one will be used as data provider and the second will be used as the journal provider.

Additional options include:

-c Checksum journal records.

-f May be used to convert an existing file system to use **gjournal**, but only if the journal will be configured on a separate provider and if the last sector in the data provider is not used by the existing file system. If **gjournal** detects that the last sector is used, it will refuse to overwrite it and return an error. This behavior may be forced by using the **-f** flag, which will force **gjournal** to overwrite the last sector.

-h Hardcode provider names in metadata.

-s *jsize* Specifies size of the journal if only one provider is used for both data and journal. The default is one gigabyte. Size should be chosen based on provider’s load, and not on its size; recommended minimum is twice the size of the physical memory installed. It is not recommended to use **gjournal** for small file systems (e.g.: only few gigabytes big).

clear Clear metadata on the given providers.

stop Stop the given provider.

Additional options include:

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

sync Trigger journal switch and enforce sending data to the data provider.

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

Create a **gjournal** based UFS file system and mount it:

```
gjournal load
gjournal label da0
newfs -J /dev/da0.journal
mount -o async /dev/da0.journal /mnt
```

Configure journaling on an existing file system, but only if **gjournal** allows this (i.e., if the last sector is not already used by the file system):

```
umount /dev/da0s1d
gjournal label da0s1d da0s1e && \
    tunefs -J enable -n disable da0s1d.journal && \
    mount -o async /dev/da0s1d.journal /mnt || \
    mount /dev/da0s1d /mnt
```

SYSCTLS

Gjournal adds the sysctl level kern.geom.journal. The string and integer information available is detailed below. The changeable column shows whether a process with appropriate privilege may change the value.

sysctl name	Type	Changeable
debug	integer	yes
switch_time	integer	yes
force_switch	integer	yes
parallel_flushes	integer	yes
accept_immediately	integer	yes
parallel_copies	integer	yes
record_entries	integer	yes

optimize	integer	yes

debug Setting a non-zero value enables debugging at various levels. Debug level 1 will record actions at a journal level, relating to journal switches, metadata updates, etc. Debug level 2 will record actions at a higher level, relating to the numbers of entries in journals, access requests, etc. Debug level 3 will record verbose detail, including insertion of I/Os to the journal.

switch_time
The maximum number of seconds a journal is allowed to remain open before switching to a new journal.

force_switch
Force a journal switch when the journal uses more than N% of the free journal space.

parallel_flushes
The number of flush I/O requests to be sent in parallel when flushing the journal to the data provider.

accept_immediately
The maximum number of I/O requests accepted at the same time.

parallel_copies
The number of copy I/O requests to send in parallel.

record_entries
The maximum number of record entries to allow in a single journal.

optimize
Controls whether entries in a journal will be optimized by combining overlapping I/Os into a single I/O and reordering the entries in a journal. This can be disabled by setting the sysctl to 0.

cache

The string and integer information available for the cache level is detailed below. The changeable column shows whether a process with appropriate privilege may change the value.

sysctl name	Type	Changeable
used	integer	no
limit	integer	yes
divisor	integer	no
switch	integer	yes

misses	integer	yes
alloc_failures	integer	yes

used
: The number of bytes currently allocated to the cache.

limit
: The maximum number of bytes to be allocated to the cache.

divisor
: Sets the cache size to be used as a proportion of kmem_size. A value of 2 (the default) will cause the cache size to be set to 1/2 of the kmem_size.

switch
: Force a journal switch when this percentage of cache has been used.

misses
: The number of cache misses, when data has been read, but was not found in the cache.

alloc_failures
: The number of times memory failed to be allocated to the cache because the cache limit was hit.

stats

The string and integer information available for the statistics level is detailed below. The changeable column shows whether a process with appropriate privilege may change the value.

sysctl name	Type	Changeable
skipped_bytes	integer	yes
combined_ios	integer	yes
switches	integer	yes
wait_for_copy	integer	yes
journal_full	integer	yes
low_mem	integer	yes

skipped_bytes
: The number of bytes skipped.

combined_ios
: The number of I/Os which were combined by journal optimization.

switches
: The number of journal switches.

wait_for_copy
: The number of times the journal switch process had to wait for the previous journal copy to

complete.

journal_full
: The number of times the journal was almost full, forcing a journal switch.

low_mem
: The number of times the low_mem hook was called.

SEE ALSO

geom(4), geom(8), mount(8), newfs(8), tunefs(8), umount(8)

HISTORY

The **gjournal** utility appeared in FreeBSD 7.0.

AUTHORS

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