Implementing UFS Journaling on a Desktop PC

Abstract

A journaling file system uses a log to record file system updates and preserves consistency in the event of a system crash or power failure. Although unsaved changes to files may still be lost, journaling greatly reduces the risk of file system corruption caused by an unclean shutdown and significantly shortens recovery time. While the UFS file system employed by FreeBSD does not implement journaling as an inherent on-disk feature, FreeBSD provides journaling support through file system-level mechanisms (Soft Updates with journaling) as well as through the GEOM framework (`gjournal`). This article describes the available UFS journaling mechanisms and explains their appropriate use on modern FreeBSD systems. *The content has been reviewed and updated for FreeBSD versions 13 through 15*.

Table of Contents

1. Introduction

While professional servers are usually well protected from unforeseen shutdowns, the typical desktop is at the mercy of power failures, accidental resets, and other user related incidents that can lead to unclean shutdowns and leave a file system in an inconsistent state. Traditionally, this required running `fsck`, which on large file systems could take a significant amount of time. On rare occasions, file system corruption reaches a point where user intervention is required and data may be lost.

FreeBSD provides two distinct journaling mechanisms for the UFS file system:

- Soft Updates with journaling
- GEOM journaling (`gjournal`)

These mechanisms operate at different layers of the system and have different performance and semantic characteristics.

For most modern FreeBSD systems, including desktops and general-purpose servers, **Soft Updates with journaling** is the recommended solution.

GEOM journaling is a legacy and specialized mechanism and should only be used when its specific semantics are required.

This article explains both mechanisms, their differences, and correct modern usage.

Read this chapter to learn:

- The journaling mechanisms available for the UFS file system in FreeBSD and how they differ.
- When to use file system-level Soft Updates journaling and when GEOM journaling may be appropriate.
- How to enable or disable Soft Updates journaling on existing UFS file systems.

- How to configure GEOM journaling on new or existing partitions, including required kernel support and journal sizing considerations.
- What configuration changes are required to mount journaled file systems and how journaling affects system behavior.
- How to diagnose and resolve common issues related to UFS journaling.

Before reading this article:

- Understand basic UNIX® and FreeBSD concepts.
- Be familiar with the FreeBSD installation process using bsdinstall.
- Have basic knowledge of disk partitioning and file systems, including tools such as gpart(8), newfs(8), and mount(8).

Some of the procedures described in this article involve modifying file system or disk configuration and may require unmounting file systems or changing on-disk metadata. Before making such changes on a system in production, ensure that reliable *backups* of all important data are available.

Enabling file system-level Soft Updates journaling is generally safe and does not require disk repartitioning. However, configuring GEOM journaling involves low-level disk operations and should be attempted only by experienced administrators who fully understand the implications.

2. Understanding Journaling in FreeBSD

2.1. Soft Updates with Journaling (Recommended)

Soft Updates is the default UFS consistency mechanism in FreeBSD. When combined with journaling, it provides **metadata journaling** within the file system itself.

Advantages of Soft Updates with journaling include:

- Very fast crash recovery (typically seconds)
- Normal `sync(2)` and `fsync(2)` semantics
- No additional partitions or GEOM layers
- Simple configuration and maintenance

From `tunefs(8)`:

> Enabling journaling reduces the time spent by fsck_ffs(8) cleaning up a file system after a crash to a few seconds from minutes to hours.

This mechanism is suitable for nearly all UFS-based systems and is the default recommendation.

2.2. GEOM Journaling (`gjournal`)

GEOM journaling operates at the block level, below the file system. It journals all block writes, including **both metadata and file data**.

Key characteristics of GEOM journaling:

- Implemented as a GEOM class (`geom_journal`)
- Journals all block I/O, not just metadata
- Requires explicit cooperation from UFS using the `-J` flag
- Disables Soft Updates
- Alters the semantics of `sync(2)` and `fsync(2)`

On GEOM-journaled file systems, `sync(2)` and `fsync(2)` do not guarantee that data has been committed to stable storage. To ensure persistence, `gjournal sync` must be used.

Because of these differences, GEOM journaling is not recommended for general-purpose desktop or server use.

3. Choosing the Appropriate Journaling Method

The following table summarizes recommended usage:

Use case	Recommended mechanism
Desktop or laptop system	Soft Updates with journaling
General-purpose server	Soft Updates with journaling
Legacy UFS systems	Soft Updates with journaling
File system-independent journaling requirements	GEOM journaling
Specialized block-level logging needs	GEOM journaling

4. Using Soft Updates with Journaling

Soft Updates with journaling provide file system-level journaling for UFS. This mechanism preserves file system consistency while retaining the allocation and ordering optimizations of Soft Updates. Unlike GEOM journaling, no separate journal device or partition is required.

4.1. Overview

Soft Updates with journaling record file system metadata changes in a journal stored inside the file

system itself, implemented as a hidden file named .sujournal in the root of the file system. In the event of a crash or power failure, pending metadata operations are replayed from the journal, allowing the file system to be mounted quickly without a full consistency check.

This form of journaling applies to file system metadata only. File data integrity remains the responsibility of applications.

4.2. Enabling Journaling on an Existing File System

The file system must be unmounted or mounted read-only before changing journaling settings.

```
# umount /usr
# tunefs -n enable -j enable /dev/ada0p2
# mount /usr
```

No additional configuration is required. Standard mount options may be used.

4.3. Journal Storage and Size

A default journal size is selected automatically when journaling is enabled. In most cases, the default size is sufficient and does not require adjustment.

If necessary, the journal size in bytes can be specified explicitly using the `-S` option to `tunefs(8)`:

```
# tunefs -S 64000000 /dev/ada0p2
```

Journal size tuning is rarely required and should be considered only for file systems with unusually high metadata update rates.

4.4. Checking Soft Updates with Journaling

The journaling status of a UFS file system using Soft Updates can be verified with `tunefs(8)` or `dumpfs(8)`.

To display the current file system settings, run:

```
# tunefs -p /dev/ada0p2 | grep -i journal
tunefs: soft updates journaling: (-j)   enabled
tunefs: gjournal: (-J)                  disabled
```

Look for the following indicators in the output:

- Soft Updates enabled
- Soft Updates journaling enabled

Alternatively, dumpfs(8) can be used to inspect the file system superblock:

```
# dumpfs /dev/ada0p2 | grep -i journal
flags   soft-updates+journal
```

These commands are read-only and do not modify file system state.

4.5. File System Checks and Maintenance

Although journaling dramatically reduces recovery time after a crash, it does not eliminate the need for periodic full file system checks.

- Journaling guarantees consistency, not correctness
- Media errors are not repaired by journaling
- Periodic fsck should still be scheduled
- Background fsck can be used on live file systems

5. Using GEOM Journaling (Advanced)

This section is intended for advanced users who understand the implications of block-level journaling and altered sync semantics.

5.1. Overview

This functionality is provided by loading the geom_journal.ko module into the kernel (or building it into a custom kernel) and using the gjournal command to configure the file systems. In general, you would like to journal large file systems, like /usr. You will need however (see the following section) to reserve some free disk space.

GEOM journaling is implemented by the geom_journal kernel module and is configured using the gjournal(8) utility. It provides block-level journaling below the file system layer and requires explicit cooperation from UFS.

When GEOM journaling is used, some disk space is needed to keep the journal itself. The provider that contains the file system data is referred to as the *data provider*, while the provider that stores the journal is referred to as the *journal provider*.

When journaling an existing (non-empty) file system, the data and journal providers must be separate. When journaling a new, empty file system, a single provider may be used to store both data and journal information. In both cases, gjournal(8) combines the data and journal providers to create a new journaled provider, which is then mounted by the file system. For example:

- The /usr file system resides on /dev/ada0p2 and already contains data.
- Free disk space has been allocated in a separate partition, /dev/ada0p4, to hold the journal.

- After configuring GEOM journaling, a new provider `/dev/ada0p2.journal` is created. This journaled provider combines `/dev/ada0p2` as the data provider and `/dev/ada0p4` as the journal provider and is used for all subsequent file system operations.

The amount of disk space required for the journal depends primarily on the write workload of the file system rather than on the size of the data provider. Systems with sustained or bursty write activity require larger journals to avoid excessive journal switching or write throttling.

From the man page `gjournal(8)`:

- The default journal size is 1 GB.
- The recommended minimum journal size is twice the amount of installed physical memory.
- The journal size should be chosen based on expected write load, not file system size.

An undersized journal may result in degraded performance or forced journal switches under heavy write load.

For more information about journaling, please read the manual page of gjournal(8).

5.2. Steps During the Installation of FreeBSD

This section describes optional installation-time preparation for systems that are configured to use GEOM journaling. The goal is to reserve disk space for journal providers that will later be associated with the `/usr` and `/var` file systems.

5.2.1. Reserving Space for Journaling

On a typical system with a single disk, the operating system, installed software, and user data all reside on the same device. The default automatic partitioning performed by the FreeBSD installer allocates most available space to `/usr`, with smaller partitions for `/var` and other mount points.

By default, the installer allocates all available disk space to file systems and does not leave unused space. When GEOM journaling is planned for existing (non-empty) file systems, additional disk space must be reserved to hold journal providers. Each file system that will be journaled requires its own journal provider.

Since `/usr` typically occupies the largest portion of the disk, it is usually the most practical partition to reduce slightly in order to make room for journal providers.

In our example GEOM journaling is planned for both `/usr` and `/var`.

5.2.2. Partitioning Strategy

During installation, use the manual partitioning mode provided by the installer. Create any supported file system type for `/`, and create standard UFS partitions for `/usr` and `/var`, but reduce the size of `/usr` to leave sufficient unallocated space at the end of the disk.

From the remaining unallocated space, create two additional partitions that will be used exclusively as journal providers:

- One partition for the `/usr` journal
- One partition for the `/var` journal

These partitions must not be assigned mount points and must not be used for swap. They will remain unused until they are explicitly associated with their corresponding data providers using `gjournal(8)` after installation.

A typical layout may look like the following:

Table 1. Partitions Reserved for GEOM Journaling

Provider	Intended Use	Notes
/dev/ada0p2	/var	UFS file system (data provider)
/dev/ada0p3	/usr	UFS file system (data provider)
/dev/ada0p4	Journal for /var	Unformatted, unused partition
/dev/ada0p5	Journal for /usr	Unformatted, unused partition

The exact device names will vary depending on the disk layout and partitioning scheme. It is strongly recommended to record the provider names during installation, as they will be required during journal configuration.

5.2.3. Journal Size Considerations

Journal size depends on the expected write workload rather than on the size of the file system. For general-purpose systems, journal sizes of 1–2 GB are commonly sufficient. Systems with sustained or bursty write activity may require larger journals.

Allocate journal space conservatively during installation, as resizing partitions later may require additional downtime.

5.2.4. Completing the Installation

After reserving the required partitions, complete the FreeBSD installation normally. It is recommended to postpone the installation of third-party software and additional system configuration until GEOM journaling has been fully configured.

At this stage, the system will boot using the standard UFS file systems. The reserved journal partitions will remain unused until journaling is explicitly enabled.

5.2.5. First Boot After Installation

After the first successful boot, no immediate configuration changes are required. The system should be verified to boot and operate normally before proceeding.

Once the base system is confirmed to be functional, the next steps involve:

- Loading or enabling GEOM journaling support
- Associating journal providers with their corresponding data providers

- Updating `/etc/fstab` to mount the journaled devices

These steps are described in the following section.

5.3. Setting Up Journaling

5.3.1. Enabling GEOM Journaling on Existing File Systems

This section describes how to enable GEOM journaling on the `/usr` and `/var` file systems using the partitions prepared during installation.

Once the required journal providers have been reserved, GEOM journaling can be configured. Because the file systems to be journaled must not be mounted, the system must be switched to single-user mode.

Log in as `root` and enter:

```
# shutdown now
```

Press `Enter` to obtain a root shell. Unmount the file systems that will be journaled. In this example, `/usr` and `/var`:

```
# umount /usr /var
```

Load the GEOM journaling kernel module:

```
# gjournal load
```

Next, associate each data provider with its corresponding journal provider. Use your notes to determine which partition will be used for each journal.

In this example:

- `/usr` resides on `/dev/ada0p3` and its journal provider is `/dev/ada0p5`
- `/var` resides on `/dev/ada0p2` and its journal provider is `/dev/ada0p4`

Create the journaled providers using `gjournal(8)`:

```
# gjournal label ada0p3 ada0p5
GEOM_JOURNAL: Journal 2948326772: ada0p5 contains journal.
GEOM_JOURNAL: Journal 2948326772: ada0p3 contains data.
GEOM_JOURNAL: Journal ada0p3 clean.

# gjournal label ada0p2 ada0p4
GEOM_JOURNAL: Journal 3193218002: ada0p4 contains journal.
GEOM_JOURNAL: Journal 3193218002: ada0p2 contains data.
```

```
GEOM_JOURNAL: Journal ada0p2 clean.
```

If `gjournal` reports that the last sector of a provider is in use, the command will fail. On freshly created or unused partitions, it is safe to retry the operation with the `-f` flag to force initialization:

```
# gjournal label -f /dev/ada0p2 /dev/ada0p4
```

After labeling, two new journaled providers are created:

- `/dev/ada0p3.journal` for `/usr`
- `/dev/ada0p2.journal` for `/var`

These devices will replace the original data providers when mounting the file systems.

Before mounting, we must however set the journal flag on them and clear the Soft Updates flag:

```
# tunefs -J enable -n disable -j disable /dev/ada0p3.journal
tunefs: soft updates journaling cleared but soft updates still set
tunefs: remove .sujournal to reclaim space
tunefs: gjournal set
tunefs: soft updates cleared

# tunefs -J enable -n disable -j disable /dev/ada0p2.journal
tunefs: soft updates journaling cleared but soft updates still set
tunefs: remove .sujournal to reclaim space
tunefs: gjournal set
tunefs: soft updates cleared
```

Now, mount the new devices manually at their respective places to verify correct operation (note that we can now use the `async` mount option) and remove .sujournal to reclaim space:

```
# mount -o async /dev/ada0p2.journal /var
# rm /var/.sujournal
# mount -o async /dev/ada0p3.journal /usr
# rm /usr/.sujournal
```

Edit /etc/fstab and update the entries for /usr and /var:

```
/dev/ada0p3.journal    /usr    ufs     rw,async      2       2
/dev/ada0p2.journal    /var    ufs     rw,async      2       2
```

Make sure the above entries are correct, or you will have trouble starting up normally after you reboot!

Finally, edit /boot/loader.conf and add the following line so the gjournal(8) module is loaded at every boot:

```
geom_journal_load="YES"
```

Congratulations! Your system is now set for journaling. You can either type `exit` to return to multi-user mode, or reboot to test your configuration (recommended).

```
# shutdown -r now
```

During the boot you will see messages like the following:

```
# dmesg | grep GEOM_JOURNAL
GEOM_JOURNAL: Journal 2948326772: ada0p4 contains journal.
GEOM_JOURNAL: Journal 3193218002: ada0p5 contains journal.
GEOM_JOURNAL: Journal 2948326772: ada0p2 contains data.
GEOM_JOURNAL: Journal ada0p2 clean.
GEOM_JOURNAL: Journal 3193218002: ada0p3 contains data.
GEOM_JOURNAL: Journal ada0p3 clean.
```

After an unclean shutdown, the messages will vary slightly, i.e.:

```
GEOM_JOURNAL: Journal ada0p2 consistent.
```

This usually means that gjournal(8) used the information in the journal provider to return the file system to a consistent state.

5.3.2. Enabling GEOM Journaling on Newly Created Partitions

The procedure described earlier is required when enabling GEOM journaling on file systems that already contain data. When journaling a newly created, empty partition, the process is simpler because both the data and journal providers can reside within the same partition.

This approach is typically used for new disks or newly allocated partitions that have not yet been formatted with a file system.

While the above procedure is necessary for journaling partitions that already contain data, journaling an empty partition is somewhat easier, since both the data and the journal provider can be stored in the same partition. For example, assume a new disk was installed, and a new partition /dev/ada1p1 was created. Creating the journal would be as simple as:

```
# gjournal label /dev/ada1p1
```

The journal size will be 1 GB by default. You may adjust it by using the `-s` option. The value can be

given in bytes, or appended by `K`, `M` or `G` to denote Kilobytes, Megabytes or Gigabytes respectively. Note that `gjournal` will not allow you to create unsuitably small journal sizes.

For example, to create a 2 GB journal, you could use the following command:

```
# gjournal label -s 2G /dev/ada1p1
```

You can then initialyze a UFS file system on your new partition, enable journaling using `newfs(8)` on the `.journal` device, and disable Soft Updates using `tunefs(8)`:

```
# newfs -J /dev/ada1p1.journal
# tunefs -n disable -j disable /dev/ada1p1.journal
```

After the file system has been created, it can be mounted normally. For example, to mount it at `/data`:

```
# mount /dev/ada1p1.journal /data
```

To ensure the file system is mounted automatically at boot, add an entry to /etc/fstab:

```
/dev/ada1p1.journal    /data   ufs     rw,async      2       2
```

5.3.2.1. Notes on Usage

When using a single provider for both data and journal storage:

- The partition must be empty before labeling
- The journal space is allocated from within the partition
- The file system size will be reduced by the size of the journal

5.4. Building Journaling into Your Custom Kernel

If you do not wish to load `geom_journal` as a module, you can build its functions right into your kernel. Edit your custom kernel configuration file, and make sure it includes these two lines:

```
options UFS_GJOURNAL ①

options GEOM_JOURNAL ②
```

① Note: This is already in GENERIC

② You will have to add this one

Rebuild and reinstall your kernel following the relevant instructions in the FreeBSD Handbook.

Do not forget to remove the relevant "load" entry from /boot/loader.conf if you have previously used it.

5.5. Checking GEOM Journaling

GEOM journaling status can be verified by inspecting active GEOM providers.

To list all active GEOM journal devices, run:

```
# gjournal list
```

Or to view a summary of journal status:

```
# gjournal status
```

Journaled file systems are identified by device names ending in `.journal`. The presence of these providers indicates that GEOM journaling is active.

Additional information about GEOM device stacking can be obtained with:

```
# geom -t
```

Alternatively, `dumpfs(8)` can be used to inspect the file system superblock:

```
# dumpfs /dev/ada0p2.journal | grep -i journal
flags   gjournal
```

6. Troubleshooting Journaling

The following section covers frequently asked questions regarding problems related to journaling.

6.1. I am experiencing kernel panics during periods of high disk activity. Is this related to GEOM journaling?

On modern FreeBSD systems, kernel panics caused solely by an undersized GEOM journal are rare.

Under sustained or burst-heavy write workloads, an undersized journal typically results in write throttling or temporary I/O stalls while the journal is flushed to the data provider. A kernel panic may occur only in exceptional cases, usually in combination with additional problems such as I/O errors or resource exhaustion, where continuing operation could risk file system integrity.

If panics are observed during heavy write activity, increasing the size of the journal provider for the affected file system is recommended.

6.2. I made some mistake during configuration, and I cannot boot normally now. Can this be fixed some way?

You either forgot (or misspelled) the entry in /boot/loader.conf, or there are errors in your /etc/fstab file. These are usually easy to fix. During system startup, when the boot process stops and prompts for single-user mode (for example after a failed mount or `fsck` error), press `Enter` to get to the default single user shell. Then locate the root of the problem:

```
# cat /boot/loader.conf
```

Ensure that the following line exists and is spelled correctly:

```
geom_journal_load="YES"
```

If the `geom_journal_load` entry is missing or misspelled, the journaled devices are never created. Load the module manually, mount all partitions, and continue with multi-user boot:

```
# gjournal load

GEOM_JOURNAL: Journal 2948326772: ada0p4 contains journal.
GEOM_JOURNAL: Journal 3193218002: ada0p5 contains journal.
GEOM_JOURNAL: Journal 2948326772: ada0p2 contains data.
GEOM_JOURNAL: Journal ada0p2 clean.
GEOM_JOURNAL: Journal 3193218002: ada0p3 contains data.
GEOM_JOURNAL: Journal ada0p3 clean.

# mount -a
# exit
(boot continues)
```

If, on the other hand, this entry is correct, have a look at /etc/fstab. You will probably find a misspelled or missing entry. In this case, mount all remaining partitions by hand and continue with the multi-user boot.

6.3. Can I remove journaling and return to my standard file system with Soft Updates?

Sure. Use the following procedure, which reverses the changes. The partitions you created for the journal providers can then be used for other purposes, if you so wish.

Login as `root` and switch to single user mode:

```
# shutdown now
```

Unmount the journaled partitions:

```
# umount /usr /var
```

Synchronize the journals:

```
# gjournal sync
```

Stop the journaling providers:

```
# gjournal stop ada0p2.journal
# gjournal stop ada0p3.journal
```

Next step should be done with unloaded gjournal kernel module. If the following command fails:

```
...
# gjournal unload
...
```

ensure that the following line exists in /boot/loader.conf and is spelled correctly:

```
geom_journal_load="NO"
```

Reboot your desktop without gjournal kernel module:

```
# shutdown -r now
```

Boot into single-user mode again. Clear the journaling metadata from all the devices used:

```
# gjournal clear ada0p2
# gjournal clear ada0p3
# gjournal clear ada0p4
# gjournal clear ada0p5
```

Clear the file system journaling flag, and restore the Soft Updates flags:

```
# tunefs -J disable -n enable -j enable ada0p2
tunefs: soft updates journaling set
tunefs: gjournal cleared
```

```
tunefs: soft updates set

# tunefs -J disable -n enable -j enable ada0p3
tunefs: soft updates journaling set
tunefs: gjournal cleared
tunefs: soft updates set
```

Remount the old devices by hand:

```
# mount -o rw /dev/ada0p2 /var
# mount -o rw /dev/ada0p3 /usr
```

Edit /etc/fstab and restore it to its original state:

```
/dev/ada0p2     /usr            ufs     rw      2       2
/dev/ada0p3     /var            ufs     rw      2       2
```

Finally, edit /boot/loader.conf, remove the entry that loads the `geom_journal` module (or re-enable it if gjournal is still required for other partitions), and reboot.

6.4. How do I temporarily boot without GEOM journaling?

To boot the system without GEOM journaling, prevent the journal module from loading during startup.

Comment out or remove the following line from /boot/loader.conf and reboot:

```
geom_journal_load="YES"
```

After booting without GEOM journaling, journaled devices (`*.journal`) will not be created. File Systems must be mounted using their original (non-journaled) providers, or mounted manually as needed for recovery or maintenance.

This method does not modify on-disk journal metadata and can be safely used for troubleshooting.

6.5. Can I resize a GEOM journal after it has been created?

No. A GEOM journal cannot be resized in place after it has been created.

The journal size is fixed at the time the journal is labeled. To change the journal size, GEOM journaling must be removed and reconfigured with a new journal provider of the desired size. For file systems containing data, this requires unmounting the file system and recreating the journal

using appropriately sized providers, as described earlier in this article.

Plan journal sizes conservatively to accommodate peak write activity, as resizing requires reinitialization of the journal configuration.

6.6. What happens after an unclean shutdown or power failure?

After an unclean shutdown or power failure, GEOM journaling and Soft Updates with journaling replay the pending journal records during the next boot to return the file system to a consistent state.

If the journal replay completes successfully, the file system is mounted normally and no full file system check is required. Journal replay typically takes only a few seconds and depends on the amount of recent write activity rather than the size of the file system.

A full file system check may still be required in rare cases, such as when underlying storage errors are detected or the journal itself is inconsistent.

6.7. How does GEOM journaling interact with fsck?

GEOM journaling and Soft Updates with journaling interact with `fsck_ffs(8)` in different ways, reflecting their different integration levels in the FreeBSD storage stack.

For GEOM journaling, crash recovery is performed by the GEOM framework itself. During system startup, the `geom_journal` class detects journal metadata and replays any pending journal records while the provider is being tasted by GEOM, before the journaled device becomes visible to higher layers of the system. If journal replay completes successfully, the file system is returned to a consistent state and the UFS superblock is updated to indicate that no file system check is required. When `fsck_ffs(8)` is later invoked during boot, it reads this state from the superblock and skips the file system check.

For Soft Updates with journaling, recovery is initiated by `fsck_ffs(8)` itself. During the file system check phase at boot time, `fsck_ffs(8)` detects that the files ystem supports Soft Updates with journaling and invokes the journal replay logic. If journal replay reports that the file system is consistent, `fsck_ffs(8)` terminates early without performing a full check. If journal replay cannot complete successfully, a traditional file system check is performed.

As a result, the recovery path during system startup depends on the journaling mechanism in use, as illustrated by the following boot sequence:

```
Firmware / Boot Loader
        |
        v
Kernel initialization
        |
        v
GEOM framework (provider tasting)
```

```
        |
        +--> GEOM journaling (gjournal)
        |       - Journal replay occurs here
        |       - Filesystem consistency is restored
        |       - Superblock is updated
        |
        v
Filesystem check phase (fsck_ffs)
        |
        +--> Soft Updates with journaling
        |       - Journal replay initiated by fsck
        |       - fsck exits early if consistent
        |
        v
Filesystem mount
        |
        v
Normal system operation
```

`fsck_ffs(8)` is still used when journal replay cannot be completed, when underlying storage errors are detected, or during periodic manual or scheduled file system checks. GEOM journaling reduces recovery time after crashes but does not replace `fsck_ffs(8)` entirely.

6.8. Why does `gjournal clear` report "Operation not permitted" even in single-user mode?

In GEOM, the same underlying storage can be exposed through multiple provider names at the same time. For example, a partition may be visible both under its disk-based name (e.g. `vtbd0p7`) and under a GPT label (e.g. `/dev/gpt/labelname`). These are different GEOM providers referring to the same physical storage.

When `gjournal stop` or `gjournal clear` is executed on one provider name, the GEOM framework may immediately detect the journal metadata again through another provider name during the tasting process. As a result, the journal is re-attached automatically, and attempts to clear its metadata fail with an "Operation not permitted" error.

It is technically possible to suppress this behavior by disabling GEOM tasting or by preventing the creation of alternative provider names, but doing so is strongly discouraged. Such changes may negatively affect other GEOM classes and system components that rely on normal provider discovery.

For this reason, the recommended and safe procedure is to unload the `geom_journal` module entirely before removing journal metadata, as described earlier in this article. With the module unloaded, no tasting occurs and the journal cannot be re-attached, allowing `gjournal clear` to complete successfully.

6.9. Can GEOM journaling be used on the root file system?

Yes. GEOM journaling can be used on the root file system, but it requires additional care during system configuration.

The GEOM journal module must be loaded early during the boot process so that the journaled root device is available before the root file system is mounted. Misconfiguration may prevent the system from booting normally, so this setup is generally recommended only for experienced users who fully understand the boot sequence and recovery procedures.

For these reasons, GEOM journaling is more commonly applied to non-root file systems such as `/usr` and `/var`.

6.10. What are the performance implications of using GEOM journaling?

GEOM journaling introduces additional write I/O because all write operations are first recorded in the journal before being committed to the data provider. This results in increased write latency and higher overall write amplification compared to non-journaled file systems.

Read performance is typically unaffected. The performance impact is most noticeable on write-heavy workloads and depends on journal size, placement, and the speed of the underlying storage. Using a sufficiently sized journal and fast storage for the journal provider minimizes performance penalties.

6.11. Can GEOM journaling be used together with other GEOM classes?

Yes. GEOM journaling can be stacked with other GEOM classes.

When used together, GEOM journaling typically operates on top of providers created by other GEOM classes such as `gmirror(8)` or `graid3(8)`. The order of stacking is important and should follow the documented layering requirements of the involved GEOM classes.

GEOM journaling maintains file system consistency across the stacked providers, but it does not replace redundancy or error detection provided by other GEOM classes.

6.12. Can GEOM journaling and Soft Updates be enabled at the same time?

No. GEOM journaling and Soft Updates must not be enabled at the same time on the same file system.

GEOM journaling requires exclusive control over write ordering and consistency guarantees. If Soft Updates are enabled together with GEOM journaling, the file system may behave unpredictably and

data integrity cannot be guaranteed.

When using GEOM journaling, Soft Updates must be disabled on the affected file system, as described earlier in this article.

6.13. What is the difference between Soft Updates and Soft Updates with Journaling?

Both Soft Updates and Soft Updates with Journaling are designed to maintain file system consistency, but they use different mechanisms and behave differently after a system crash.

Soft Updates ensure consistency by carefully ordering metadata writes in memory so that only safe states are written to disk. If a crash occurs, any metadata updates that have not yet been written are simply lost, leaving the file system in a consistent but possibly older state. However, resource accounting may be temporarily inaccurate, and a full `fsck_ffs(8)` run is required to reclaim unused blocks and inodes.

Soft Updates with Journaling build on top of *Soft Updates* by adding a metadata journal stored within the file system. Metadata changes are first recorded in the journal and are replayed automatically during the next boot after an unclean shutdown. This allows the file system to be restored to a consistent state quickly, usually without requiring a full `fsck_ffs(8)` run.

In short, *Soft Updates* provide consistency without a journal, while *Soft Updates with Journaling* add fast and automatic recovery after crashes at the cost of additional disk space and minor performance overhead.

7. Further Reading

Journaling is a fairly new feature of FreeBSD, and as such, it is not very well documented yet. You may however find the following additional references useful:

- A section on journaling is now part of the FreeBSD Handbook.
- This post in FreeBSD-CURRENT mailing list by gjournal(8)'s developer, `Paweł Jakub Dawidek <pjd@FreeBSD.org>`.
- This post in FreeBSD general questions mailing list by `Ivan Voras <ivoras@FreeBSD.org>`.
- Journaled Soft-Updates, Dr. Kirk McKusick, BSDCan 2010 on YouTube
- GEOM - in Infrastructure We Trust, Pawel Jakub Dawidek, AsiaBSDCon 2008 on YouTube
- The manual pages of gjournal(8), geom(8), and tunefs(8).