# Btrfs: от новичка до профессионала

Btrfs (B-tree File System) — это современная файловая система Linux с поддержкой атомарных снапшотов, субтомов, встроенного сжатия, RAID и даже дубликатного хранения. В отличие от традиционных ext4 или XFS, она работает по принципу COW (copy-on-write), что позволяет: безболезненно откатывать систему после обновлений или сбоев; создавать резервные копии без остановки служб и сервисов; хранить сразу несколько состояний системы (как в Git); автоматически экономить место за счёт сжатия; использовать один раздел как полноценную систему снапшотов и подкаталогов. В НАЙС.ОС Btrfs используется как корневая файловая система по умолчанию и даёт возможность реализовать атомарные обновления, защиту от сбоев и откаты через GRUB. Эта статья подойдёт как новичкам (поясним всё с нуля), так и профессионалам (оптимизация, производительность, RAID, безопасность).

## 1. Что такое Btrfs — в двух словах

**Btrfs** (B-tree File System) — это современная копирующая при записи ( *Copy-on-Write* , COW) файловая система, разработанная в 2007 году инженерами **Oracle** как ответ на ограничения ext3/ext4 и необходимость встроенных механизмов целостности, снапшотов, управления томами и отказоустойчивости.

В отличие от ext4 или XFS, Btrfs изначально создавалась как **система с расширенными возможностями** , где такие вещи как:

- 📸 **Снапшоты** — моментальные копии системы без копирования данных;
- 🧩 **Субтомы** — логическое разбиение на независимые разделы внутри одной ФС;
- 🌀 **Сжатие** — встроенное, на лету (zstd/lzo/zlib);
- **Встроенный RAID** — без необходимости в mdadm;
- **Целостность данных** — встроенные CRC-контроли и проверки;

- **Дедупликация и балансировка** — управление многотомными хранилищами;

Всё это делает Btrfs универсальным решением для рабочих станций, серверов, CI/CD, дистростроения и защищённых систем.

## 🥿 Чем Btrfs отличается от ext4 на пальцах

- ext4 — это просто журналируемая ФС, без встроенных снапшотов;
- Btrfs — это файловая система + менеджер логических томов + снапшоты + RAID;
- Btrfs не нуждается в LVM — всё уже встроено в неё.

## 🥿 Чем Btrfs отличается от XFS на пальцах

- XFS — очень быстрая, но "жёсткая" система без снапшотов и COW;
- Btrfs — чуть медленнее на запись, но поддерживает снапшоты и сжатие из коробки;
- XFS отлично подходит для баз данных, Btrfs — для гибких, безопасных и откатываемых систем.

## 2. Почему Btrfs снова актуальна

После долгого пути развития, **Btrfs** к 2025 году стал **по-настоящему зрелой и стабильной** файловой системой. Он официально поддерживается в **ядрах Linux 5.10+** и активно используется в нескольких крупных дистрибутивах:

- **SUSE Linux Enterprise** — основная ФС по умолчанию;
- **openSUSE (Leap и MicroOS)** — мощный инструмент для атомарных обновлений;
- **Fedora** — для рабочих станций и серверов с интеграцией snapper;
- **НАЙС.ОС** — для атомарных обновлений, снапшотов и rollback через GRUB;

Сегодня Btrfs предлагает **уникальный баланс** между:

- **Надёжностью** — COW, встроенные контрольные суммы, защита от silent corruption;
- ⚙ **Функциональностью** — снапшоты, субтомы, RAID, сжатие;
- 🚀 **Производительностью** — особенно с включённым zstd-сжатием и SSD;
- 🛠 **Удобством** — простой CLI, хорошая документация и стабильность.

Если раньше Btrfs вызывал сомнения в продакшене, то теперь — это один из **лучших вариантов для новых систем** , требующих отказоустойчивости и гибкости.

## 3. Установка и разметка

Для работы с Btrfs достаточно установить пакет `btrfs-progs` — он предоставляет все необходимые инструменты: от форматирования до управления снапшотами и RAID.

## 🔧 Создание файловой системы

```
mkfs.btrfs -L ROOTFS /dev/sdX
```

Здесь `/dev/sdX` — ваш раздел (например, `/dev/sda2` ), а `-L` задаёт метку тома.

## 📎 Монтирование и fstab

Один из ключевых аспектов — работа с **субтомами** . Стандартная схема в большинстве дистрибутивов:

- `@` — корень системы `/`
- `@home` — домашние каталоги `/home`
- Опционально: `@log` , `@var` , `@snapshots`

Создание субтомов:

```
mount /dev/sdX /mnt
btrfs subvolume create /mnt/@
btrfs subvolume create /mnt/@home
umount /mnt
```

Монтирование с параметрами:

```
mount -o subvol=@,compress=zstd,autodefrag /dev/sdX /mnt
mount -o subvol=@home,compress=zstd,autodefrag /dev/sdX /mnt/home
```

## 🏷️ Btrfs в /etc/fstab: пример для systemd-boot или GRUB

```
UUID=xxxxx /      btrfs defaults,subvol=@,compress=zstd,autodefrag 0 1
UUID=xxxxx /home  btrfs defaults,subvol=@home,compress=zstd,autodefrag 0 2
```

Здесь `UUID=xxxxx` — UUID вашего раздела (можно узнать через `blkid`).

Такая схема позволяет использовать **атомарные обновления**, гибко управлять частями системы, делать снапшоты только нужных субтомов и не затрагивать пользовательские данные при откатах.

## 4. Снапшоты и откат: в чём магия

**Снапшоты** в Btrfs — это моментальные копии состояния субтома. Благодаря технологии *Copy-on-Write*, снапшот не копирует все файлы, а лишь фиксирует текущее дерево данных. Это **мгновенно** и **экономит место**, особенно при сжатии.

## 📄 Read-only vs Read-write

- **Read-only снапшоты** — подходят для отката, резервного копирования, атомарных обновлений;
- **Read-write снапшоты** — можно использовать как рабочую копию или временную среду.

## 🛠️ Основные команды

```
# Создание снапшота:
btrfs subvolume snapshot /mnt/@ /mnt/@_backup_2025_07_22

# Read-only:
btrfs subvolume snapshot -r /mnt/@ /mnt/@_ro_snapshot

# Удаление снапшота:
btrfs subvolume delete /mnt/@_ro_snapshot

# Откат (замена субтома):
btrfs subvolume set-default @backup_2025_07_22
reboot
```

Для ручного отката достаточно выбрать другой снапшот как `default` и перезагрузить систему — **без chroot, без LiveCD** .

## 🔁 Интеграция с утилитами

- **Snapper** — мощный инструмент управления снапшотами с diff, rollback и автосозданием;
- **Timeshift** — GUI-решение для снапшотов пользовательской системы, удобно в десктопах.

НАЙС.ОС использует схему снапшотов при **обновлении системы** (до и после), с возможностью выбора в `GRUB` . Это позволяет откатиться даже если обновление полностью сломало загрузку .

# 5. Субтомы (subvolumes): организация пространства

В **Btrfs** нет понятия «разделов» внутри файловой системы. Вместо этого используются **subvolumes** — логические единицы хранения, которые можно монтировать отдельно, снапшотить, или удалять независимо.

## 📦 Что такое subvolume?

Subvolume — это изолированная директория верхнего уровня , которая ведёт себя как самостоятельный том. Однако физически всё это лежит на одном разделе/устройстве. Они удобны для:

- гибкой разметки диска без переразбивки;
- атомарных обновлений и избирательных снапшотов;
- разделения системных и пользовательских данных.

## 🔧 Управление субтомами

```
# Создание:
btrfs subvolume create /mnt/@
btrfs subvolume create /mnt/@home
btrfs subvolume create /mnt/@var_log

# Список:
```

```
btrfs subvolume list /mnt

# Удаление:
btrfs subvolume delete /mnt/@old_snapshot
```

## 💡 Примеры использования

- `@` — корень системы ( `/` )
- `@home` — пользовательские файлы ( `/home` )
- `@var/log` — системные логи (можно исключить из снапшотов)
- `@snapshots` — каталог с автоснапшотами (например, snapper)

В **fstab** каждое подмонтирование можно настроить отдельно, включая опции сжатия, noatime и даже read-only.

Это даёт системным администраторам гибкость уровня LVM — но без необходимости в партиционировании.

## 6. Сжатие и экономия места

Одно из самых полезных и недооценённых свойств Btrfs — встроенное **прозрачное сжатие данных** . Оно позволяет:

- экономить место на диске (особенно для текстовых/JSON/лог-файлов);
- уменьшить количество I/O операций и ускорить загрузку файлов с SSD;
- в некоторых случаях — повысить производительность.

## 🎁 Поддерживаемые алгоритмы

- **zstd** — современный, быстрый и сбалансированный по компрессии;
- **lzo** — очень быстрый, но слабее по сжатию;
- **zlib** — старый, самый медленный, но максимально совместимый.

## ⚙️ Как включить сжатие

В fstab или при монтировании вручную:

```
mount -o compress=zstd /dev/sdX /mnt
```

Пример строки в `/etc/fstab`:

```
UUID=xxxx / btrfs defaults,subvol=@,compress=zstd 0 1
```

Для эффективной работы сжатия **файлы должны быть заново записаны** на диск. Используйте:

```
btrfs filesystem defragment -r -v -czstd /home
```

## ✔ Как проверить, что сжатие работает

```
btrfs filesystem df /
```

## 🥿 Как узнать, какие файлы действительно сжаты

Утилита `filefrag` может подсказать:

```
filefrag -v somefile | grep -i compressed
```

Также можно использовать `btrfs inspect-internal map-logical` для анализа extent-ов.

## ⊘ Что не стоит сжимать

- Медиафайлы (видео, фото, mp3) — они уже сжаты;
- Базы данных (например, PostgreSQL) — может привести к снижению производительности;
- Кэш браузеров или временные каталоги.

В **НАЙС.ОС** сжатие `zstd` включено по умолчанию для большинства субтомов — как оптимальный компромисс между скоростью и эффективностью.

# 7. RAID и многодисковая конфигурация

**Btrfs** обладает встроенной поддержкой RAID — без необходимости использовать `mdadm` или сторонние средства. Массивы создаются и управляются средствами самой файловой системы, а не на уровне блочных устройств.

## ⚙️ Поддерживаемые уровни RAID

- **RAID 0** — распределение данных (striping), без избыточности;
- **RAID 1** — зеркалирование, копии на двух устройствах;
- **RAID 10** — комбинация RAID 1 + RAID 0 (striped mirrors);
- **RAID 5 / 6** — распределённый паритет, (экспериментальные!)

В отличие от `mdadm`, Btrfs позволяет использовать RAID **на уровне данных и/или метаданных** независимо:

```
mkfs.btrfs -d raid1 -m raid1 /dev/sdX /dev/sdY
```

Также можно добавить устройство в существующий том:

```
btrfs device add /dev/sdZ /mnt
btrfs balance start /mnt
```

## 🔄 Замена и управление

Если нужно заменить диск «на горячую»:

```
btrfs replace start /dev/sdX /dev/sdY /mnt
btrfs replace status /mnt
```

## ⚠️ Ограничения RAID5/6

Поддержка **RAID5 и RAID6 в Btrfs считается экспериментальной** . Есть риск потери данных в некоторых сценариях восстановления, особенно при незавершённой балансировке или сбоях питания.

Рекомендуется использовать **RAID1 или RAID10** для критичных данных. RAID5/6 —

только для экспериментов или невысоких требований к надёжности.

## 🥊 Чем отличается от mdadm?

- **mdadm** работает на уровне устройств, прозрачен для ФС;
- **Btrfs RAID** — часть логики самой ФС, поддерживает гибкое распределение метаданных/данных;
- Btrfs умеет добавлять и заменять устройства без размонтирования.

В **НАЙС.ОС** Btrfs RAID может быть использован при установке вручную или через кастомный инсталлятор. По умолчанию используется одиночный том ( `single` ), но доступны профили RAID1/10.

# 8. Проверка и восстановление

**Btrfs** предоставляет встроенные инструменты для мониторинга и восстановления состояния файловой системы. Это позволяет выявить ошибки, деградацию носителей и предотвратить потерю данных.

## 🔍 Проверка целостности

- `btrfs scrub start /mnt` — фоновая проверка блока данных на ошибки, и при наличии RAID — автоматическое восстановление с зеркала.
- `btrfs scrub status /mnt` — статус выполнения.

## 📊 Статистика и ёмкость

```
btrfs filesystem df /
btrfs device stats /
```

## ⚖️ Балансировка тома

Если вы добавили или удалили диски, или видите разбалансировку, используйте:

```
btrfs balance start /mnt
```

## 🛠 Проверка на ошибки

В отличие от ext4, утилита `btrfs check` не должна использоваться на смонтированной файловой системе!

```
umount /dev/sdX
btrfs check /dev/sdX
```

Для исправления ошибок (только если уверены):

```
btrfs check --repair /dev/sdX
```

⚠️ **ВНИМАНИЕ:** `--repair` может повредить данные. Используйте в крайнем случае и только с бэкапом.

## 🔧 Вспомогательные утилиты

- `btrfs inspect-internal dump-super` — чтение суперблока вручную;
- `btrfs rescue` — инструменты для восстановления повреждённых томов (метаданные, ключи, zero-log).

## 🩹 Как понять, что пошло не так — и что делать

- **Ошибка scrub** : может означать повреждённый блок — попробуйте заменить диск и выполнить `btrfs replace` .
- **Нехватка места, но df показывает «достаточно»** : проверьте `btrfs filesystem df` — данные могли фрагментироваться, выполните `balance` .
- **Система не монтируется** : используйте LiveCD, `btrfs check` без `--repair` и анализируйте.

В **НАЙС.ОС** задачи scrub и balance могут запускаться автоматически по cron или systemd timer. Это повышает стабильность при длительной работе и снижает риск silent errors.

## 9. Btrfs и безопасность

**Btrfs** — это не только удобство и гибкость, но и мощные возможности для повышения безопасности Linux-систем. В сочетании с механизмами мандатного контроля, он становится основой защищённых дистрибутивов.

## 🛡️ Read-only снапшоты

- Любой снапшот можно сделать только для чтения:

```
btrfs property set -ts /mnt/@snapshot ro true
```

Это защищает его от случайных или злонамеренных изменений — идеальный способ зафиксировать «золотое состояние» системы.

## 🔐 Интеграция с IMA и SELinux

- Снапшоты можно использовать совместно с IMA (Integrity Measurement Architecture): хэши файлов фиксируются при установке и проверяются при запуске.
- С помощью SELinux можно ограничить доступ к снапшотам только доверенным процессам, исключая постэксплуатационные атаки.

## 🕵️ Защита от rollback-атак

Система может предотвращать запуск старых снапшотов (например, с известными уязвимостями) через механизм **подписей снапшотов** и контроль целостности.

В **НАЙС.ОС** предусмотрена проверка:

- Целостности снапшота (через AIDE или встроенную подпись);
- Сопоставления снапшота с версией ядра и конфигурацией загрузчика.

## Пример из НАЙС.ОС

В защищённой ОС **НАЙС.ОС** используется Btrfs как корневая ФС по умолчанию. Перед каждым обновлением создаётся read-only снапшот с меткой времени. После успешного обновления он может быть проверен через `IMA + AIDE`, а в случае

компрометации — использован для отката через GRUB.

## 🛌 Как снапшоты помогают при атаке

- Позволяют быстро откатить изменения без LiveCD;
- Сохраняют целостность конфигурации даже при root-доступе злоумышленника;
- В сочетании с SELinux блокируют несанкционированный доступ к критичным данным.

# 10. Btrfs для разработчика

**Btrfs** — не только файловая система для пользователей, но и мощный инструмент в руках DevOps-инженеров, сборочных систем и CI/CD-конвейеров. Благодаря Copy-on-Write, снапшотам и `btrfs send/receive` можно строить гибкие, воспроизводимые среды сборки и доставки ПО.

## 🛠️ Работа со снапшотами в CI/CD

- Быстрое клонирование окружений: один `btrfs subvolume snapshot` и вы получили копию среды без копирования данных.
- Откат окружения в начале каждого теста без потерь времени на пересоздание дисков.
- Изоляция билдов без необходимости использовать полноценные контейнеры.

## 🎁 Создание сборочных слоёв

- Сборка базового слоя (toolchain, deps) — снапшот → сборка основного ПО — снапшот → финальный слой.
- Можно «откатить» до любого слоя и пересобрать только нужные части.

## 📩 Использование `btrfs send/receive`

Позволяет экспортировать снапшот как поток байт (read-only!) и получить минимальный «дельта-образ».

```
btrfs send /mnt/@build-root | lz4 > build-layer.img.lz4
```

```
# а на стороне CI или сборки ISO:
lz4 -d build-layer.img.lz4 | btrfs receive /mnt/
```

- Можно использовать для сборки ISO-образов, атомарных обновлений или отложенной доставки в staging/production.
- Дельты между снапшотами сильно экономят трафик и ускоряют delivery.

## 🦭 Использование в Kubernetes / Podman

- Btrfs позволяет использовать снапшоты в качестве `overlayfs lowerdir`.
- Это снижает накладные расходы при запуске контейнеров, особенно в dev/CI-средах.
- Может использоваться совместно с Podman и CRI-O (опция: `--storage-driver=btrfs`).

## 🥿 Советы:

- Для layer-based CI создавайте сабтомы с понятными именами: `@base-sdk`, `@layer-gost`, `@final`.
- Храните снапшоты в отдельной точке монтирования (`/snapshots`) для централизованного управления.
- Используйте read-only снапшоты для воспроизводимости.

В **НАЙС.ОС** механизм снапшотов Btrfs активно используется в сборочном пайплайне, включая доставку образов, откат на CI-сервере, минимальные обновления и создание доверенных, подписанных ISO.

# 11. Производительность: мифы и реальность

Вокруг **Btrfs** долгое время ходили мифы о «медленной работе», «нестабильности» и «опасности RAID5». Но к 2025 году многое изменилось. Давайте разберёмся, что правда, а что — пережитки прошлого.

## 🐢 Что действительно тормозит Btrfs?

- Отключённый TRIM на SSD — приводит к накоплению мусора и снижению производительности.
- Использование `nodatacow` на сжимаемых файлах — приводит к неэффективному

дисковому IO.
- Старые ядра Linux (< 5.10) — в них отсутствуют важные оптимизации записи и сбалансировки метаданных.
- Некорректно настроенные `btrfs balance` — могут тормозить систему в фоне.

## ⚖ Сравнение с ext4 и XFS

| Операция | Btrfs | ext4 | XFS |
|---|---|---|---|
| Копирование больших файлов | ext4 (с включённым `compress=zstd`) | Очень быстро | Очень быстро |
| Создание снапшота | **Мгновенно** (COW) | ⛔ Не поддерживается | ⛔ Не поддерживается |
| Удаление снапшота | Средне, зависит от размера метаданных | ⛔ | ⛔ |
| Сжатие данных | **Да (zstd/lzo/zlib)** | ⛔ | ⛔ |

## ⊘ Когда Btrfs — не лучший выбор

- **Real-time системы** (RT-патчи, критичная задержка) — COW может мешать предсказуемости записи.
- **Тяжёлые СУБД с высоким IO** (PostgreSQL, Oracle) — возможны накладные расходы при высоком уровне фрагментации.
- **RAID5/6** — всё ещё считается нестабильным и экспериментальным (особенно при сбоях питания).

## 🖋 Рекомендации:

- Для серверов и рабочих станций — используйте Btrfs с `compress=zstd` и сбалансированной политикой снапшотов.
- Для real-time и СУБД — предпочитайте ext4 или XFS с tuned-профилем.
- Используйте ядра Linux  5.10 или дистрибутивы с Btrfs-патчами (например, НАЙС.ОС).

На практике, **Btrfs** обеспечивает отличную производительность в типовых сценариях (рабочие станции, CI/CD, файловые сервера) при грамотной настройке и свежем ядре.

## 12. Советы по эксплуатации

**Btrfs** мощна, но требует внимания. Ниже — базовые советы по обслуживанию, предотвращению проблем и грамотной эксплуатации в реальных системах.

## 🔄 Как часто делать `scrub` и `balance`

- `btrfs scrub` — рекомендуем запускать **раз в месяц** :

  ```
  btrfs scrub start -Bd /
  ```

  Это проверка целостности данных и метаданных, особенно важна при использовании встроенного RAID.
- `btrfs balance` — оптимизирует распределение блоков. Запускайте:
  - Раз в 1–2 месяца;
  - После удаления больших объёмов данных;
  - Только частично:

    ```
    btrfs balance start -dusage=75 -musage=75 /
    ```

## Как не засорить систему снапшотами

- Используйте **read-only снапшоты** по умолчанию — они меньше по размеру.
- Автоматически удаляйте старые снапшоты по числу/возрасту:
  - Вручную или через `cron`
  - Или с помощью `snapper cleanup`
- Храните снапшоты в отдельном subvolume: `@snapshots`

## ⚠ Btrfs в read-only после сбоя

Если система загрузилась с корнем в **read-only** после сбоя питания или ошибки метаданных:

1. Проверьте состояние:

```
dmesg | grep BTRFS
```

2. Запустите `scrub` или `check` (последний — только как крайняя мера):

```
btrfs scrub start -Bd /
```

Если необходимо — загрузитесь с LiveCD и примените:

```
btrfs rescue zero-log /dev/sdXn
```

**Важно:** `btrfs check --repair` использовать только при бэкапе и понимании риска!

## 💾 Резервное копирование через `send | receive`

Это один из лучших способов сделать полную, **атомарную** и **быструю** резервную копию Btrfs-системы:

```
# На исходной машине:
btrfs subvolume snapshot -r /@ /@backup-2025-07-22
btrfs send /@backup-2025-07-22 | ssh backuphost 'btrfs receive /mnt/backup/'
```

- Поддерживает инкрементальные бэкапы;
- Идеально подходит для offline-архивов и air-gapped систем.

## 🗞 Полезный cron для домашнего сервера

```
0 3 * * 7 root btrfs scrub start -Bd /
0 4 * * 7 root snapper cleanup number
```

Следуя этим простым рекомендациям, вы получите от Btrfs стабильность, предсказуемость и безопасность — как на сервере, так и на рабочей станции.

## 13. Дорожная карта и развитие

**Btrfs** активно развивается и получает новые функции, направленные на повышение безопасности, надёжности и применимости в защищённых ОС.

## 🔒 Интеграция с verity и dm-integrity

- **dm-verity** — механизм для создания неизменяемых, проверяемых разделов. Планируется поддержка подписанных снапшотов Btrfs с хеш-деревом.
- **dm-integrity** — проверка целостности данных на уровне блока. Интеграция с ним обеспечит защиту от silent corruption и аппаратных сбоев.
- Это особенно актуально для:
  - систем хранения с требованиями к аудиту;
  - доверенной загрузки (TBoot, SecureBoot);
  - встроенных систем и промышленных контроллеров.

## 🗝 Поддержка пользовательского шифрования (fscrypt)

- fscrypt — современный интерфейс для **прозрачного шифрования каталогов** .
- Btrfs уже поддерживает `fscrypt` в новых ядрах (5.15+), и ведётся работа над улучшением производительности и удобства интеграции.
- Это позволит:
  - шифровать отдельные профили пользователей без LUKS;
  - использовать аппаратные ускорения (AES-NI, Кузнечик);
  - гибко управлять ключами в системах с мандатным доступом.

Btrfs продолжает эволюцию: от просто интересной COW-ФС до полноценной базы для защищённых и надёжных Linux-дистрибутивов.

## 14. Заключение

**Btrfs** — это современная, мощная и уже зрелая файловая система, которую можно (и нужно) использовать в повседневной практике.

- **Это не страшно** — начните с одного снапшота, одной команды `scrub` или одного сабтома `@home` .
- **Подходит как новичкам** , так и **профессионалам** : от простых откатов до автоматизации в CI/CD и дистрострое.
- **Идеальный выбор** для:
  - разработчиков и сборщиков дистрибутивов (атомарные обновления, откат без боли);
  - DevSecOps-инфраструктуры (интеграция с IMA, снапшоты, контроль

целостности);
- домашних и серверных рабочих станций (надёжность + экономия места через сжатие);
- и, конечно, защищённых операционных систем, таких как **НАЙС.ОС** .

Настало время переосмыслить корневую файловую систему. **Btrfs** уже здесь — надёжный, проверенный, гибкий.