Мониторинг состояния ФС

1. Введение

В операционных системах Linux файловая система играет ключевую роль в стабильности, производительности и долговечности системы. Её износ, переполнение, фрагментация или ошибки могут не только снизить скорость работы, но и привести к потере данных.

⚠️ **Если вы не мониторите состояние ФС — вы не знаете, когда она выйдет из строя.**

🔍 Почему важно мониторить файловую систему

- 📉 **Износ SSD** — важно контролировать количество циклов перезаписи и своевременно применять TRIM.
- ⛔ **Переполнение разделов** — приводит к сбоям сервисов, невозможности логгирования, падению баз данных.
- **Фрагментация** — снижает производительность, особенно на HDD и при больших файлах на ext4.
- 🐢 **Просадки по I/O** — могут указывать на перегрузку, неэффективный доступ или умирающее устройство.

🏢 Что можно и нужно контролировать

- **Свободное место и inode** (через `df`, `du`);
- **Нагрузку на диск** (через `iostat`, `iotop`, `dstat`);
- **Износ и состояние SSD** (через `smartctl`, `lsblk --discard`);
- **Фрагментацию и дефрагментацию** (через `e4defrag`, `filefrag`);
- **Работу TRIM** (через `fstrim` и `systemd-timer`);
- **Файлы, которые нельзя увидеть напрямую** (удалённые, но занятые — через `lsof`);

📜 Когда мониторинг особенно важен

- 🖥️ **На серверах** — где переполнение раздела может остановить критически

важные процессы;
- 📦 **В контейнерах** — ограниченные ресурсы требуют жёсткого контроля использования места;
- ⚡ **На SSD и NVMe** — где важен TRIM, износ и равномерное распределение нагрузки;
- **В виртуальных машинах** — особенно с thin provisioning, где видимое пространство не отражает реальность;
- **На десктопах** — для избежания фрагментации и потери данных.

💡 **Вывод:** грамотный мониторинг ФС — это не только диагностика, но и профилактика. Чем раньше вы замечаете проблему, тем дешевле и проще её устранить.

2. Использование `df` и `du`: сколько занято и где

Одни из самых базовых и полезных инструментов в Linux — это `df` и `du`. Они позволяют быстро понять, сколько места занято и какими каталогами. Однако они показывают информацию по-разному и предназначены для разных задач.

🧮 `df` — информация о файловых системах

Показывает использование места **на уровне точек монтирования** и устройств:

```
df -h
```

Показывает размеры в человекочитаемом виде (GB, MB)

```
df -i
```

Отображает использование **inode** — критично, если файлов очень много, даже при наличии свободного места.

📂 `du` — анализ содержимого каталогов

Подсчитывает **фактический размер каталогов и подкаталогов** на уровне файловой структуры:

```
du -sh *
```

Показывает, какие директории в текущем каталоге занимают больше всего места.

```
sudo du -xh / --max-depth=2 | sort -h
```

Рекурсивный просмотр с сортировкой по размеру.

💡 **Совет:** используйте `ncdu` (если установлен) для интерактивного анализа:

```
sudo ncdu /
```

🔍 Разница между `df` и `du`

- **`df`** читает информацию из `/proc/mounts` и показывает размеры томов;
- **`du`** обходит реальные файлы и папки, рассчитывая их размеры по факту;
- `df` включает скрытые и удалённые, но ещё открытые файлы (например, журналы, занятые процессами);
- `du` не увидит эти файлы, т.к. они уже не видны в файловом дереве.

⚠️ **Если `df` показывает занятое место, а `du` — нет**, скорее всего, виноваты удалённые, но не освобождённые файлы. Найти их можно командой:

```
lsof +L1
```

3. Мониторинг ввода-вывода: `iostat`, `iotop`, `dstat`

Мониторинг I/O (ввода-вывода) — важная часть наблюдения за состоянием файловой системы. Он помогает выявить медленные диски, «тяжёлые» процессы и узкие места в производительности.

📊 `iostat` — статистика по устройствам

Утилита из пакета `sysstat` показывает производительность дисковых устройств, включая загрузку, задержки и throughput:

```
iostat -xz 1
```

- `%util` — насколько устройство занято (если 100% — диск перегружен);

- await — средняя задержка всех операций (чем выше, тем хуже);
- r/s, w/s — количество чтений и записей в секунду;
- svctm — среднее время выполнения операций устройством.

⚠ **Если %util 100% при больших await** — диск является «бутылочным горлышком».

👁 iotop — кто грузит диск прямо сейчас

Интерактивная утилита, показывающая процессы, активно работающие с диском:

```
sudo iotop
```

- Отображает **PID, процесс, скорость чтения/записи** в реальном времени;
- Удобно для обнаружения утечек логов, неэффективных баз данных и бэкапов;
- Необходимо иметь права root.

🏢 dstat — много метрик в одной строке

dstat — универсальный инструмент для мониторинга I/O, CPU, памяти, сети в реальном времени:

```
dstat -cdngytl
```

Каждая колонка — отдельный ресурс: диск, сеть, память, процессы, задержки.

💡 **Альтернатива:** если dstat не установлен, попробуйте atop или glances для более детального real-time обзора.

Интерпретация показателей

- **Высокий %util + высокий await** диск перегружен (HDD, плохой SSD, или много IOPS);
- **Низкий %util, но медленный доступ** проблемы на уровне ФС, контроллера или блокировок;
- **Большие объёмы записи** — возможно, неочищаемые логи или тяжёлые базы.

Комбинируйте: iostat для диагностики устройства, iotop — для поиска виновников, dstat — для общей картины.

4. Мониторинг файловой активности: `inotify`, `auditd`, `lsof`

Помимо мониторинга места и I/O, в Linux важно следить за **активностью файлов**: кто и когда их создаёт, удаляет, изменяет или использует. Это полезно для отладки, безопасности и устранения проблем с «невидимыми» файлами.

🔍 `lsof +L1` — поиск удалённых, но занятых файлов

Иногда файлы уже удалены, но всё ещё занимают место, потому что открыты каким-то процессом. `lsof` позволяет их найти:

```
sudo lsof +L1
```

- Показывает открытые файлы с нулевыми линками (уже удалённые);
- Часто виноваты журналы, временные файлы, бэкапы и базы данных;
- Можно освободить место, перезапустив процесс или закрыв дескриптор.

⚠️ **df может показывать, что диск забит, но `du` — нет.** Это классический признак удалённого, но занятого файла.

🕵️ `inotifywait` — отслеживание изменений в каталогах

Инструмент из пакета `inotify-tools` позволяет «подслушивать» файловые события в реальном времени:

```
inotifywait -m -r /var/log
```

- `-m` — режим постоянного наблюдения;
- `-r` — рекурсивно по подкаталогам.

Вы увидите события вроде `CREATE`, `DELETE`, `MODIFY`, `OPEN`, `CLOSE`.

Пример: слежение за появлением новых файлов в `/tmp`:

```
inotifywait -m /tmp -e create
```

🛡️ `auditd` — мощный аудит действий на ФС

Демон `auditd` позволяет логировать доступ к файлам, изменения и нарушения политик безопасности. Используется в защищённых системах (в т.ч. с SELinux).

```
sudo auditctl -w /etc/passwd -p war -k passwd-watch
```

- `-w` — путь к файлу;
- `-p` — типы доступа: `w`rite, `a`ttribute change, `r`ead, `x`execute;
- `-k` — метка (ключ) для фильтрации.

Просмотр логов:

```
sudo ausearch -k passwd-watch
```

💡 **auditd** подходит не только для безопасности, но и для диагностики: кто, когда и как менял конфигурационные файлы или запускал удаление.

5. TRIM и износ SSD: `fstrim`, `smartctl`, `lsblk`

В отличие от жёстких дисков, **SSD имеют ограниченное число циклов записи** на ячейку памяти. Для поддержания скорости и срока службы важно регулярно использовать команду **TRIM**, которая сообщает контроллеру, какие блоки больше не используются.

✂️ Что такое TRIM и зачем он нужен

- Позволяет контроллеру SSD «освобождать» блоки после удаления файлов;
- Увеличивает производительность и срок службы накопителя;
- Особенно важен для систем с высокой скоростью записи (журналы, БД, временные файлы).

✔ Проверка и запуск TRIM вручную

Можно вручную инициировать TRIM на смонтированном разделе:

```
sudo fstrim -v /
```

Вывод покажет, сколько байт было освобождено.

🕒 Проверка таймера TRIM (если используется systemd)

Большинство дистрибутивов используют `systemd`-таймер для периодического запуска TRIM:

```
systemctl status fstrim.timer
```

Чтобы вручную запустить его немедленно:

```
sudo systemctl start fstrim.service
```

▦ Проверка износа SSD через `smartctl`

Инструмент `smartctl` из пакета `smartmontools` позволяет узнать состояние SSD, включая wear-leveling и число перезаписей:

```
sudo smartctl -a /dev/sdX | grep -i wear
```

Другие важные метрики, которые стоит искать:

- `Wear_Leveling_Count`
- `Media_Wearout_Indicator`
- `Percent_Lifetime_Remain` — особенно на Intel, Samsung, Kingston

⚠ **Если износ превышает 90%** или `Wear_Leveling_Count` падает к нулю — стоит задуматься о замене SSD.

🗃 Проверка поддержки TRIM устройством

Не все устройства и файловые системы поддерживают TRIM. Проверить можно так:

```
lsblk --discard
```

Интересуют столбцы:

- `DISC-GRAN` — минимальный размер TRIM-операции;
- `DISC-MAX` — максимальный размер;
- Если значения равны 0 — TRIM **не поддерживается**.

💡 **Поддержка TRIM** зависит от:

- Типа накопителя (HDD □ SSD);
- Файловой системы (ext4, XFS, Btrfs — поддерживают);
- Флагов монтирования (`discard` или периодический `fstrim`).

6. Фрагментация: `e4defrag`, `filefrag`, `xfs_db`

Несмотря на распространённое мнение, **фрагментация файлов** существует и в Linux, особенно на **HDD** и при частом создании/удалении больших файлов. Это может привести к снижению производительности чтения и записи, особенно при доступе к большим логам, базам данных и образам VM.

🔎 Проверка фрагментации с помощью `filefrag`

Утилита `filefrag` показывает, из скольких физических фрагментов состоит файл:

```
filefrag -v /path/to/file
```

- Ищите строку `extent` — чем меньше, тем лучше (идеально — 1);
- Подходит для `ext4`, `XFS`, `Btrfs` и других ФС с поддержкой extent;
- Работает с правами пользователя, но лучше запускать от root.

Дефрагментация `ext4` с `e4defrag`

Стандартный инструмент дефрагментации в ext4:

```
sudo e4defrag /
```

Можно дефрагментировать не весь раздел, а конкретную директорию или файл.

Пример:

```
sudo e4defrag /var/log
```

🔧 Для XFS: `xfs_db` и `xfs_fsr`

- `xfs_db` — интерактивный отладочный инструмент (не для новичков);
- `xfs_fsr` — автоматизированная утилита дефрагментации XFS:

```
sudo xfs_fsr -v /dev/sdX1
```

Работает в фоновом режиме, дефрагментируя файлы на лету.

⚠ **XFS и ext4 не дефрагментируют автоматически!** Особенно на старых системах

и HDD это может привести к падению производительности.

💡 Почему фрагментация — это не миф

- На HDD считывание данных с фрагментами частые перемещения головки медленная работа;
- На ext4 без `lazy_itable_init` и при постоянном перезаписывании больших файлов (бэкапы, базы) фрагментация растёт быстро;
- SSD менее чувствительны, но фрагментация влияет на wear-leveling и TRIM.

💡 **Рекомендация:** Проверяйте и дефрагментируйте хотя бы периодически, особенно на лог-серверах, базах данных, видеонаблюдении и архивных HDD.

7. Контроль квот: `quota`, `repquota`, `xfs_quota`, `btrfs qgroup`

Контроль дисковых квот позволяет ограничивать использование файловой системы по пользователям, группам и подтомам. Это особенно важно в мультипользовательских и серверных системах, где необходимо избежать переполнения ресурсов отдельными пользователями или сервисами.

💻 Включение квот в `ext4` и `XFS`

Для **ext4** нужно включить квоты в `/etc/fstab`:

```
/dev/sdX1  /home  ext4  defaults,usrquota,grpquota  0 2
```

Затем инициализировать файлы квот:

```
sudo mount -o remount /home
sudo quotacheck -cum /home
sudo quotaon /home
```

Для **XFS** поддержка встроенная, но также нужно указать флаги в fstab:

```
/dev/sdX1  /data  xfs  defaults,uquota,gquota  0 2
```

💡 **XFS** поддерживает квоты из коробки и может управлять ими на лету через `xfs_quota`.

📊 Утилиты для работы с квотами

- `quota -u username` — показать квоту пользователя;
- `repquota /mountpoint` — отчёт по всем пользователям;
- `edquota username` — установить лимиты вручную;
- `xfs_quota -x -c 'report -h' /mountpoint` — для XFS.

🎁 Квоты в Btrfs: подтомы и qgroup

Btrfs использует механизм **qgroup (quota groups)** для ограничения подтомов.

Включение квот:

```
sudo btrfs quota enable /mnt
```

Посмотреть текущие квоты:

```
sudo btrfs qgroup show -pcre /mnt
```

Установка лимита на подтом:

```
sudo btrfs qgroup limit 5G /mnt/@home
```

Btrfs qgroup работает на уровне подтомов и даёт гибкий контроль, включая учёт снапшотов.

📌 Выводы

- **ext4**: простые user/group квоты, но не поддерживает project-квоты;
- **XFS**: поддерживает user/group/project-квоты, включая гибкое управление;
- **Btrfs**: гибкая квотируемая модель с поддержкой подтомов и снапшотов;
- Для **ReiserFS, exFAT** и других — квоты, как правило, отсутствуют.

8. Наблюдение за файловыми системами в реальном времени

Регулярный мониторинг — хорошо, но бывают ситуации, когда нужно **отслеживать состояние дисков прямо сейчас**. Это особенно полезно при устранении проблем, подозрении на утечку места или отладке активности приложений.

🕰 `watch df -h` — базовый real-time монитор

Отображает занятое место с обновлением каждые 2 секунды по умолчанию:

```
watch df -h
```

Можно задать частоту обновления:

```
watch -n 5 df -i
```

Хорошо подходит для отслеживания резкого роста логов, временных файлов и inode.

🏢 `ncdu`, `gdu` — интерактивный анализ места

Эти утилиты позволяют в интерактивном режиме просматривать, какие каталоги занимают больше всего места.

- `ncdu /` — наглядная древовидная структура, управление стрелками;
- `gdu /` — более быстрая альтернатива на Go (особенно для больших томов);
- Работают даже на сервере по SSH.

```
sudo ncdu /var/log
```

💡 **Полезно:** сразу видно каталоги, занимающие десятки или сотни ГБ, даже если они "глубоко спрятаны".

📊 `glances` и `htop` с колонками I/O

`glances` — комплексная панель мониторинга ресурсов:

- Показывает загрузку CPU, RAM, дисков, сети, температуру и т.д.;
- Поддерживает web-интерфейс и API;
- Запуск: `glances` (требует установки пакета).

`htop` можно настроить для отображения активности I/O по процессам:

- Нажмите F2 → **Columns** → добавьте `IO read rate`, `IO write rate`;
- Удобно отслеживать, кто нагружает диск в данный момент.

Совет: Используйте `ncdu` для анализа, `watch df` — для наблюдения, `htop/glances` — для общего контроля состояния.

9. Интеграция с мониторингом и алертами

Важно не только наблюдать за файловой системой вручную, но и **автоматизировать мониторинг** с оповещениями. Это позволяет предупреждать аварии до того, как они приведут к простоям или потере данных.

📡 `node_exporter` + Prometheus

Один из самых популярных инструментов — **node_exporter** от Prometheus. Он собирает множество системных метрик, включая:

- `node_filesystem_avail_bytes` — доступное место;
- `node_filesystem_size_bytes` — общий размер ФС;
- `node_filesystem_files_free` — свободные inode;
- `node_filesystem_readonly` — признак Read-Only монтирования.

Пример выражения Prometheus:

```
100 * (node_filesystem_avail_bytes{mountpoint="/"} / node_filesystem_size_bytes{mountpoint="/"}) < 10
```

Алерт при остатке меньше 10% на `/`

🚨 Примеры алертов

- **Диск почти заполнен**:
  ```
  node_filesystem_avail_bytes / node_filesystem_size_bytes < 0.1
  ```
- **Inode почти закончились**:
  ```
  node_filesystem_files_free / node_filesystem_files < 0.05
  ```
- **Файловая система в RO**:
  ```
  node_filesystem_readonly == 1
  ```

💡 **Совет:** интегрируйте с `Alertmanager` для отправки уведомлений в Telegram, Slack, email и т.д.

🛠️ **Альтернативы и готовые решения**

- **Netdata** — визуальный real-time мониторинг с auto-оповещениями;
- **Zabbix** — классическая система с агентами и встроенными шаблонами;
- **Grafana + Telegraf + InfluxDB** — мощная система визуализации с лёгким сборщиком данных;
- **Checkmk, Icinga, Nagios** — для корпоративных или гибридных решений.

⚙️ **Вывод:** если у вас есть более одного сервера или важные сервисы — настройте алерты. Лучше узнать о проблеме заранее, чем от пользователей.

10. Заключение

Эффективное использование дискового пространства и предотвращение сбоев напрямую зависит от **регулярного мониторинга файловой системы**. Даже простые инструменты дают ценные данные, если применять их системно.

- 🔁 **Регулярный мониторинг = стабильность**: вовремя замеченные проблемы позволяют избежать аварий и простоев.
- **Комбинируйте инструменты**:
  - `df` + `du` — анализ места;
  - `iostat` + `smartctl` — диагностика I/O и износа;
  - `fstrim` + `e4defrag` — оптимизация SSD и ext4.
- 💾 **Учитывайте носитель:** для SSD критичен TRIM и контроль записи, для HDD — фрагментация и seek time.
- **Автоматизируйте:** используйте `systemd`-таймеры, `cron`, алерты в Prometheus или Zabbix.

✅ **Помните:** поддерживать файловую систему в хорошем состоянии — это не разовая задача, а постоянный процесс.