?????????? ? ???????

Работа со службами мониторинга, логирование, отладка

- Journalctl - справочник
- systemctl — справочник
- top и htop
- netstat, ss и ip

Journalctl - ??????????

Journalctl — это утилита для работы с системным журналом **systemd**.
Она объединяет логи всех сервисов и ядра в единое хранилище, доступное через команды фильтрации.
В отличие от классических /var/log/syslog и /var/log/messages, journalctl позволяет быстро находить ошибки, фильтровать логи по времени, сервисам, приоритетам.

???????? ???? ?????

Простая команда для просмотра системного журнала:

```
journalctl
```

Обычно логи будут длинные, поэтому часто используют `-e` (сразу перейти к концу).

```
journalctl -e
```

???????? ????????? ??????? ? ???????? ???????

```
journalctl -f
```

Аналогично `tail -f /var/log/syslog`. Удобно использовать при отладке сервисов.

?????????? ?? ???????

Посмотреть логи конкретного сервиса:

```
journalctl -u nginx
```

В реальном времени:

```
journalctl -u nginx -f
```

?????????? ?? ???????

Примеры:

```
journalctl --since "2025-10-01" --until "2025-10-05"   # за указанный период
journalctl --since "2 hours ago"                        # за последние 2 часа
journalctl --since yesterday                            # за вчерашний день
```

?????????? ?? ??????????? (???????)

Приоритеты: **emerg, alert, crit, err, warning, notice, info, debug**.

```
journalctl -p err        # только ошибки
journalctl -p warning    # предупреждения
journalctl -p info       # информационные сообщения
```

???????? ????? ????????

```
journalctl -b      # текущая загрузка
journalctl -b -1   # предыдущая загрузка
journalctl -b -2   # позапрошлая загрузка
```

Очень полезно для анализа причин падения системы или kernel panic.

???????? ????? ????

```
journalctl -k             # сообщения ядра
journalctl -k -p err      # только ошибки ядра
```

??????????? ?????????? ?????

```
journalctl -n 50          # последние 50 строк
journalctl -n 100 -u ssh  # последние 100 строк сервиса ssh
```

????? ? ????????????? ??? ???????? ???????

```
journalctl -o short       # стандартный вывод
journalctl -o json-pretty # JSON-формат, удобно для анализа
journalctl -o cat         # только текст сообщения
```

???????? ???????? ? ??????? ????????????

- Сервис не запускается — смотрим причину:
  `journalctl -u имя_сервиса -xe`
- Отслеживание ошибок SSH-подключений:
  `journalctl -u ssh -p err -f`
- Анализ падения Nginx:
  `journalctl -u nginx --since "10 min ago"`
- Поиск "OOM-killer" (система убила процесс из-за нехватки памяти):
  `journalctl -k | grep -i oom`
- Выяснить, кто перезагружал систему:
  `journalctl _COMM=systemd-logind | grep "Power key pressed"`

???????? ?????? ? ???????

- **Ошибка:** логи не сохраняются после перезагрузки.
  Решение: включить постоянное хранение:

```
sudo mkdir -p /var/log/journal
sudo systemctl restart systemd-journald
```

- **Ошибка:** слишком большой размер логов.
  Решение: очистка и настройка лимита:

```
sudo journalctl --vacuum-time=7d      # оставить только 7 дней
sudo journalctl --vacuum-size=500M    # ограничить размер 500 МБ
```

systemctl — ??????????

Systemctl — утилита для управления **systemd** — системой инициализации и менеджером служб в Linux. Она отвечает за запуск и контроль демонов, таймеров, точек монтирования и других компонентов, описываемых *unit-файлами*.

??????

systemd управляет набором объектов: сервисами, сокетами, таймерами, монтированиями и т.д. Каждый объект описывается unit-файлом; `systemctl` — основной интерфейс для взаимодействия с этими unit'ами и с самой системой.

??? ????? unit-?????

Unit-файл — это конфигурационный файл, который описывает, как именно systemd должен управлять конкретным компонентом (сервисом, сокетом, таймером и т.д.). Обычно имеют расширение `.service`, `.socket`, `.timer`, `.mount`, `.target` и т.д.

Где хранятся:

- `/usr/lib/systemd/system/` — unit-файлы поставщика (пакетов).
- `/etc/systemd/system/` — локальные или переопределяющие unit-файлы администратора.

Пример простого unit-файла (`nginx.service`):

```
Description=A high performance web server
After=network.target
[Service]
ExecStart=/usr/sbin/nginx -g 'daemon off;'
ExecReload=/bin/kill -s HUP $MAINPID
ExecStop=/bin/kill -s QUIT $MAINPID
Restart=on-failure
[Install]
WantedBy=multi-user.target
```

Пояснение полей:

- `[Unit]` — метаинформация и зависимости (например, `After=`).
- `[Service]` — команды запуска/остановки, поведение при сбое (`Restart`), пользователь и окружение.
- `[Install]` — как unit включается в target'ы (автозапуск).

??????? ??????????

???????? ??????? ???????

Команда показывает состояние (active/failed), PID, путь к unit-файлу и последние строки логов:

```
sudo systemctl status nginx
```

?????? / ????????? / ??????????

Запуск вручную (не влияет на автозапуск при следующей загрузке):

```
sudo systemctl start nginx
```

Остановка:

```
sudo systemctl stop nginx
```

Перезапуск (полностью остановит, затем запустит):

```
sudo systemctl restart nginx
```

Перезапустить только если сервис уже запущен:

```
sudo systemctl try-restart nginx
```

Когда использовать: перезапуск обязателен при изменении конфигурации демона, но перед этим убедитесь, что unit-файл и конфигурация корректны.

???????????? ????????

enable / disable

`enable` создаёт символьные ссылки unit'а в каталогах target'ов — сервис запустится при загрузке системы. `disable` убирает эти ссылки, но не мешает ручному запуску.

```
sudo systemctl enable nginx
sudo systemctl disable nginx
systemctl is-enabled nginx
```

Важно: `enable` не запускает сервис немедленно — для этого используйте `start` или `restart`.

???????????? (mask / unmask)

Иногда нужно гарантированно запретить запуск сервиса — даже случайный или зависимый запуск. Для этого используется `mask`, который перенаправляет unit на `/dev/null`.

```
sudo systemctl mask nginx    # сервис нельзя будет запустить ни вручную, ни зависимостями
sudo systemctl unmask nginx  # снять маску
```

Когда применять `mask`:

- Нужно полностью заблокировать конфликтующий сервис (например, запрещаем start сетевого демона, если используем другой).
- Убрать возможность случайного запуска устаревших, небезопасных демонов (telnet, rsh и т.п.).
- Тестирование: временно запрещаем сервис, чтобы проверить поведение зависимостей.

?????? ? ??????????? unit-??????

После добавления/изменения unit-файла **systemd** нужно пересканировать. Это делается командой:

```
sudo systemctl daemon-reload
```

Дальше обычно перезапускают сервис:

```
sudo systemctl restart имя_сервиса
```

Если забыть `daemon-reload`, systemd будет использовать старую конфигурацию, даже если файл на диске изменён.

Targets (?????? runlevels)

В systemd понятие runlevel заменено на *targets* — наборы unit'ов, которые запускаются вместе.

- `multi-user.target` — многопользовательский режим без графики (аналог runlevel 3).
- `graphical.target` — графический режим (аналог runlevel 5).
- `rescue.target` — аварийный режим.

```
systemctl get-default               # посмотреть текущую цель
sudo systemctl set-default multi-user.target  # установить дефолтную цель
sudo systemctl isolate multi-user.target     # переключиться сразу
```

????: journalctl

systemd использует журнал (journald). Для просмотра логов сервиса применяют:

```
journalctl -u nginx         # все логи для unit
journalctl -u nginx -f      # следить в реальном времени
```

Полезно сочетать с временными фильтрами: `--since`, `--until`, и с `-p` (уровень приоритета).

???????? ??????? ? ??????

```
systemctl list-units --type=service        # активные службы
systemctl list-unit-files --type=service     # все файлы unit (включая disabled)
systemctl is-active ssh                       # быстро узнать, активен ли сервис
systemctl list-dependencies имя_сервиса       # показать зависимости
```

???????? ?????? ? ?? ???????

1) ??????? unit-????, ?? ????????? ?? ???????????

```
sudo systemctl daemon-reload
sudo systemctl restart имя_сервиса
```

Обязательная последовательность: `daemon-reload` → перезапуск/старт.

2) `Failed to start ...`

Проверяем статус и логи:

```
sudo systemctl status имя_сервиса
journalctl -xeu имя_сервиса
```

Частые причины: ошибка в пути в `ExecStart`, неверные права, конфликт портов, отсутствующие зависимости.

3) ?????? ??????????? ???????, ?? ?? ???????? ??? ????????

```
systemctl is-enabled имя_сервиса
sudo systemctl enable имя_сервиса
```

Проверьте также, не мешают ли target'ы или маскировка, и нет ли ошибок в `[Install]` блока unit-файла (например, `WantedBy=`).

4) "Unit not found"

Скорее всего unit-файл отсутствует или не установлен:

```
systemctl list-unit-files | grep имя_сервиса
```

Возможные причины: пакет не установлен, unit называется иначе, или unit расположен в нестандартном каталоге.

5) Unit ??????? ??? `masked`

```
systemctl is-enabled имя_сервиса   # покажет "masked"
sudo systemctl unmask имя_сервиса
```

???? unit-?????? (??????)

- `.service` — сервис/демон.
- `.socket` — сокет-активация; может запускать .service при подключении.
- `.timer` — периодические/отложенные задания (альтернатива cron).
- `.mount` — описание точки монтирования.
- `.target` — агрегатор unit'ов (аналог runlevel).

???????? ???????????? (best practices)

- Редактируйте unit-файлы в `/etc/systemd/system/` (локальные переопределения не затрутся при обновлении пакета).
- После правок — всегда `daemon-reload`, затем `restart` или `try-restart`.
- Используйте `Restart=on-failure` аккуратно; логируйте причины падений (через `StandardOutput`/`StandardError` или journald).
- Для временной блокировки сервиса применяйте `mask`. Для обычного отключения — `disable`.

top ? htop

??? ????? top ? htop?

`top` и `htop` — консольные утилиты для мониторинга процессов в Linux. С их помощью можно в реальном времени отслеживать загрузку процессора, использование памяти, swap, uptime и управлять процессами. Разница: **top** есть по умолчанию в любой системе, **htop** удобнее и нагляднее, но может требовать установки.

????????? htop

```
sudo apt install htop   # Ubuntu/Debian
sudo yum install htop   # CentOS/RHEL
sudo dnf install htop   # Fedora
```

??????

```
top
htop
```

????????? ? ???????

- **top:** `P` — сортировка по CPU, `M` — по памяти, `k` — завершить процесс, `q` — выход.
- **htop:** стрелки — перемещение, `F6` — сортировка, `F9` — убить процесс, `F10` — выход.

??????? ?????????????

```
top -o %MEM             # отсортировать процессы по памяти
htop -u www-data        # показать процессы пользователя www-data
top -n 1 -b > /tmp/top.txt  # вывести результат в файл (batch mode)
```

???????? ????????

- Сервер "подвисает" → проверить процессы с высокой загрузкой CPU.
- Падает сервис → отследить, не уходит ли в swap (**htop** показывает swap в цветах).
- Подозрительная активность → сортировать по PID/памяти и найти виновника.

???????? ?????? ? ???????

- **Ошибка:** htop не найден.
  Решение: установить через пакетный менеджер (см. выше).

- **Ошибка:** невозможно убить процесс.
  Решение: использовать root-права:

  ```
  sudo kill -9 PID
  ```

netstat, ss ? ip

??? ????? netstat, ss ? ip?

Эти утилиты используются для диагностики сетевых подключений и маршрутов:
- **netstat** — старая утилита (в пакете net-tools).
- **ss** — современная замена netstat.
- **ip** — инструмент для управления сетевыми интерфейсами, IP-адресами и маршрутами.

??????? ???????

```
ss -tuln                # список слушающих портов
ss -tulpn | grep 5432   # найти процесс на порту 5432
ip a                    # список интерфейсов и адресов
ip r                    # таблица маршрутов
ip link show up         # активные интерфейсы
netstat -anp            # (устар.) список соединений с PID
```

???????????

- Проверить, слушает ли nginx порт 80 → `ss -tuln | grep :80`
- Кто слушает порт 22 → `ss -tulpn | grep :22`
- Какой маршрут до 8.8.8.8 → `ip route get 8.8.8.8`
- Сбросить интерфейс → `sudo ip link set eth0 down && sudo ip link set eth0 up`

???????? ??????

- **Ошибка:** netstat не найден.
  Решение: установить пакет `net-tools`.
- **Ошибка:** сервис не слушает порт.
  Решение: проверить конфиг сервиса и firewall (`ufw`/`iptables`).