**Министр науки и высшего образования Российской̆ Федерации**

**Федеральное государственное автономное образовательное учреждение высшего образования**

**«Национальный исследовательский университет ИТМО»**

Факультет информационных технологий и программирования

Лабораторная работа № 5

Управление ресурсами и изоляция с помощью cgroups, namespaces. Контейнеры.

Выполнил студент группы № M3311

Ершова Мария

**Задачи:**

1. Научиться создавать, изменять и удалять лимиты и квоты для пользователей и процессов с помощью Control groups (cgroups V1/V2).
2. Научиться настраивать изоляцию различных ресурсов ОС с помощью namespaces.
3. Научиться работать с контейнерами и понимать, как данная технология работает «под капотом».

**Задание:** Создайте текстовый файл, в котором запишете последовательность команд для выполнения каждого из нижеследующих заданий. Для команд, имеющих интерактивный интерфейс, опишите последовательность выбора управляющих команд и их параметров. Если решение заключается в изменении конфигурационного файла, укажите название файла и вносимые правки. Во многих заданиях будет фигурировать ID – это последние 2 цифры вашего номера студента в ИСУ.

**1. Квоты на процессор для конкретного пользователя (cgroups v2)**

* Создайте пользователя: user-ID (например, user-72).

useradd user-16

* Назначьте квоту процессора на основе номера пользователя:

◦ Если имя пользователя заканчивается на 0-30: 30% CPU.

◦ Если имя пользователя заканчивается на 31-70: 50% CPU.

◦ Если имя пользователя заканчивается на 71-99: 70% CPU

Назначим квоту процессора 30% CPU:

Создаем cgroup для пользователя, чтобы задать ограничения и изолировать ресурсы пользователя:

* user.slice: службы пользовательского режима. Каждому вошедшему в систему пользователю назначается неявный срез.

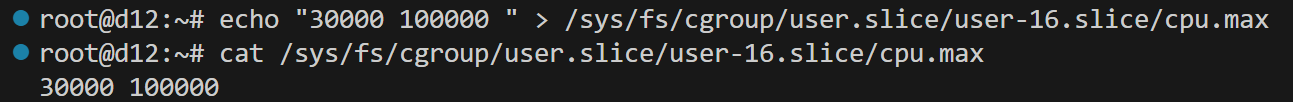
Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

В cgroups v2 параметр cpu.max ожидает два числа: квота и период.

30000 100000 означает 30% CPU:

echo "30000 100000 " > /sys/fs/cgroup/user.slice/user-16.slice/cpu.max



**2. Ограничение памяти для процесса (cgroups)**

* Создайте cgroup для ограничения памяти, потребляемой процессом.

apt install cgroup-tools – качаем.

cgcreate -g memory:memlimit – создали cgroup (-g <контроллеры>:<путь>).

* Ограничьте потребление памяти следующим образом: ID\*10 + 500 МБ (например, ID=23 → 730 МБ).

16\*10 + 500 = 660M

cgset -r memory.max=660M memlimit

* Запустите процесс и переместите его в созданную вами группу. (Пример команды: tail /dev/zero.)

cgexec -g memory:memlimit tail /dev/zero

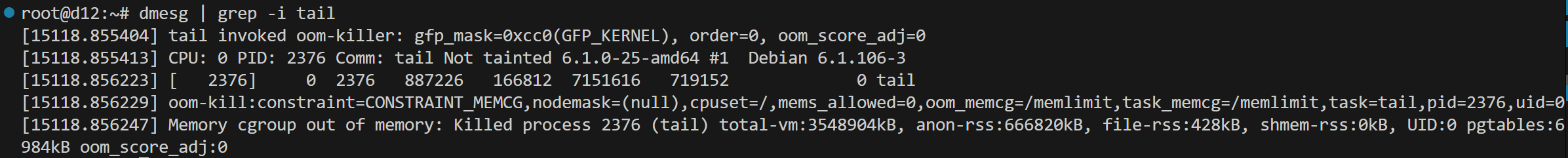
Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Убили, ибо лимит был превышен.

* Проверьте, что при исчерпании памяти процессом он прерывается ОС

dmesg | grep -i tail



Проверили, и все хорошо, ура ура.

**3. Ограничение дискового ввода-вывода для сценария резервного копирования (cgroups)**

* Скрипт резервного копирования (backup.sh) перегружает дисковую подсистему.

apt install fio – качаем чтобы тестировать.

* Ограничьте его до: Чтение: 1000 + ID\*10 IOPS. Запись: 500 + ID\*10 IOPS.

Чтение: 1000 + 16\*10 = 1160.

Запись: 500 + 16\*10 = 660.

* Используйте cgcreate для установки ограничений io.max.

cgcreate -g io:backuplimits

echo "8:0 riops=1160 wiops=660" > /sys/fs/cgroup/backuplimits/io.max

Но в итоге путь вышел длиннее двух команд:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

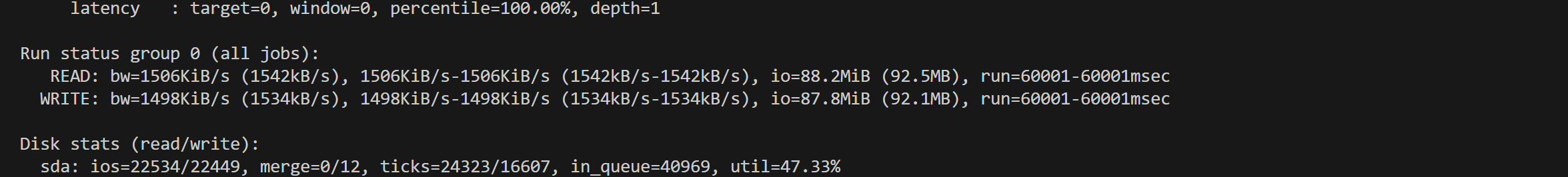
Контент, сгенерированный ИИ, может содержать ошибки.

* Протестируйте с помощью fio или dd

Для чтения:

Изображение выглядит как текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.



**4. Закрепление к определенному ядру процессора для приложения**

• Настройте с помощью cgroups процесс команды top за процессором 0.

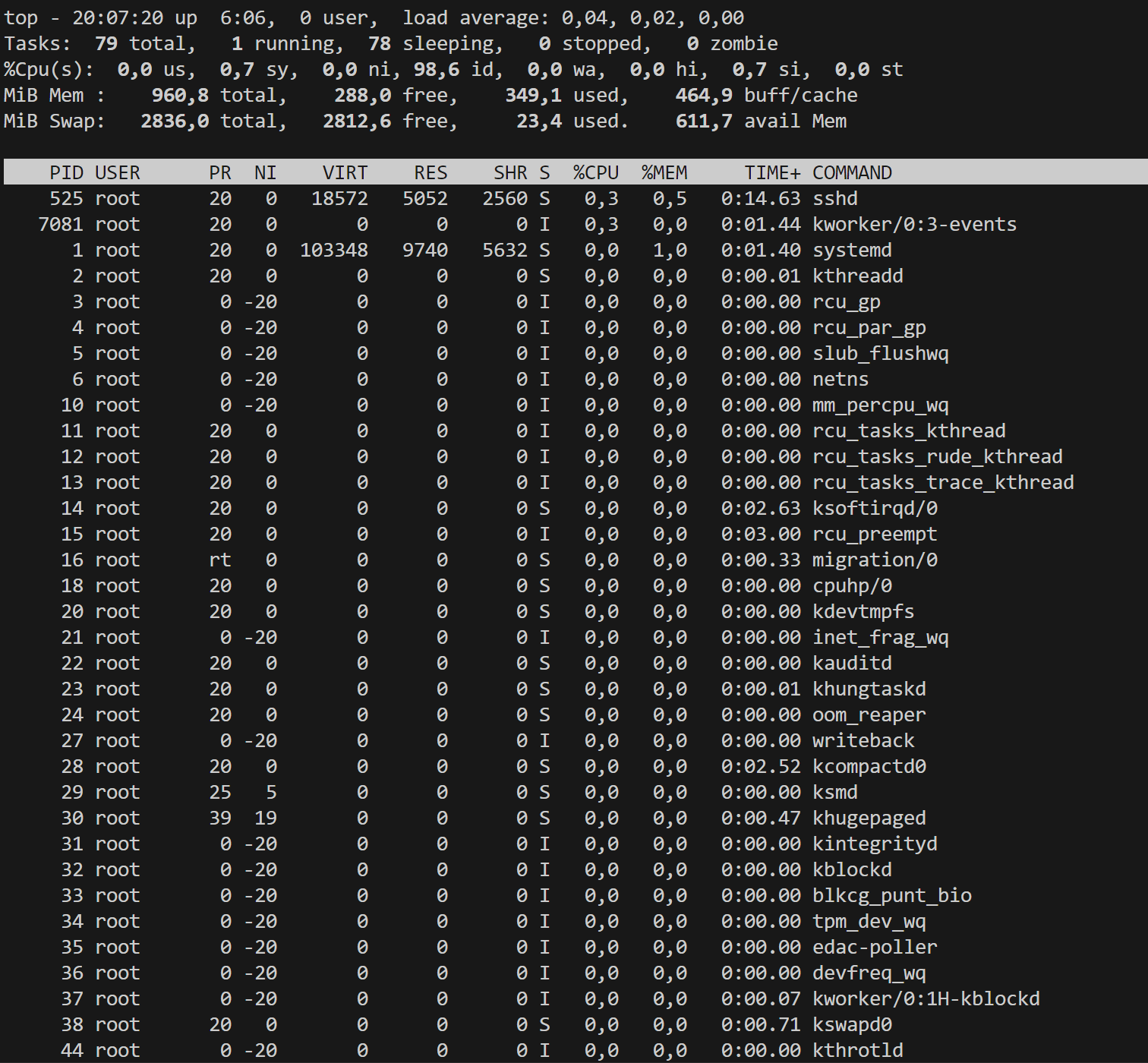
Создание cgroup для cpuset:

cgcreate -g cpuset:top

cgset -r cpuset.cpus=0 top

Запуск top в cgroup:

cgexec -g cpuset:top top



• Используйте cpuset.cpus в cgroups - Done!

• Проверьте с помощью taskset -p . (требуется пакет sysstat)

pgrep top – чтобы узнать PID процесса.

taskset -p 7317

Изображение выглядит как текст, Шрифт, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

affinity mask представляет из себя 32 бита, определяет набор процессоров, на которых может выполняться определённый процесс.

**5. Динамическая корректировка ресурсов (cgroups)**

* Напишите сценарий для мониторинга нагрузки по CPU и динамического изменения cpu.max определенного процесса (его идентификатор задается как входной параметр скрипта).
* Квота ЦП для процесса должна регулироваться в зависимости от общей нагрузки на систему:
* Низкая нагрузка (CPU < 20%): 80% CPU.
* Высокая нагрузка (CPU > 60%): 30% CPU.

apt install sysstat bc – устанавливаем.

Изображение выглядит как текст, Шрифт, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

В dynamic\_cpu\_adjustment.sh пишем скриптик:

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.



chmod +x dynamic\_cpu\_adjustment.sh – устанавливаем флаг "исполняемый" для файла.

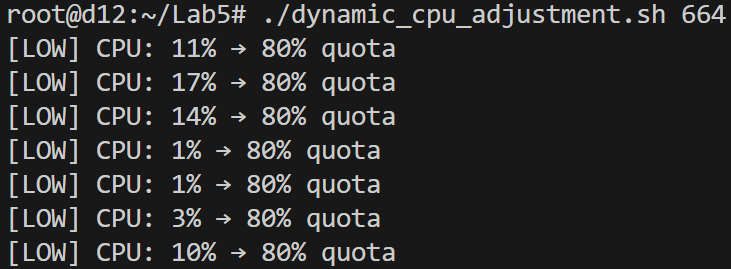
Напишем top, чтобы узнать PID какого-нибудь процесса:

Изображение выглядит как текст, снимок экрана, меню

Контент, сгенерированный ИИ, может содержать ошибки.

Я взяла процесс с PID 664.

./dynamic\_cpu\_adjustment.sh 664 – запускаем скрипт:



Чтобы завершить – ctrl+c.

Все работает.

**6. Создание изолированного имени хоста (пространство имен UTS)**

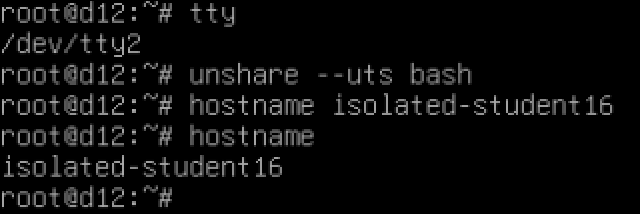
* Запустите оболочку (shell/bash) в отдельном namespace, в которой можно изменить имя хоста, не затрагивая хост.
* Измените имя хоста внутри пространства имен на isolated-student-<ID>.
* Проверьте изоляцию:
  + hostname # Должно отображаться «isolated-student-<ID>».
  + Проверьте имя хоста (в новом терминале): hostname # По-прежнему показывает оригинальное имя хоста.

Идём во второй терминал и запускаем оболочки в новом UTS namespace:

unshare --uts bash

Задаём имя хоста:

hostname isolated-student16



Видим, что все изменилось как надо.

Переходим в первый терминал и проверяем:

Изображение выглядит как текст, Шрифт, снимок экрана, черный

Контент, сгенерированный ИИ, может содержать ошибки.

Видим, что hostname по-прежнему показывает оригинальное имя хоста.

**7. Изоляция процессов (пространство имен PID)**

* Создайте пространство имен, в котором процессы хоста будут невидимы:
  + unshare --pid --fork bash.
* Смонтируйте новый каталог /proc:
  + mount -t proc proc /proc.
* Проверьте процессы:
  + ps aux # Показывает только 2-3 процесса (например, bash, ps).
* Сравните с хостом (в новом терминале):
  + ps aux # Показывает все процессы хоста

unshare --pid --fork bash.

mount -t proc proc /proc.

ps aux

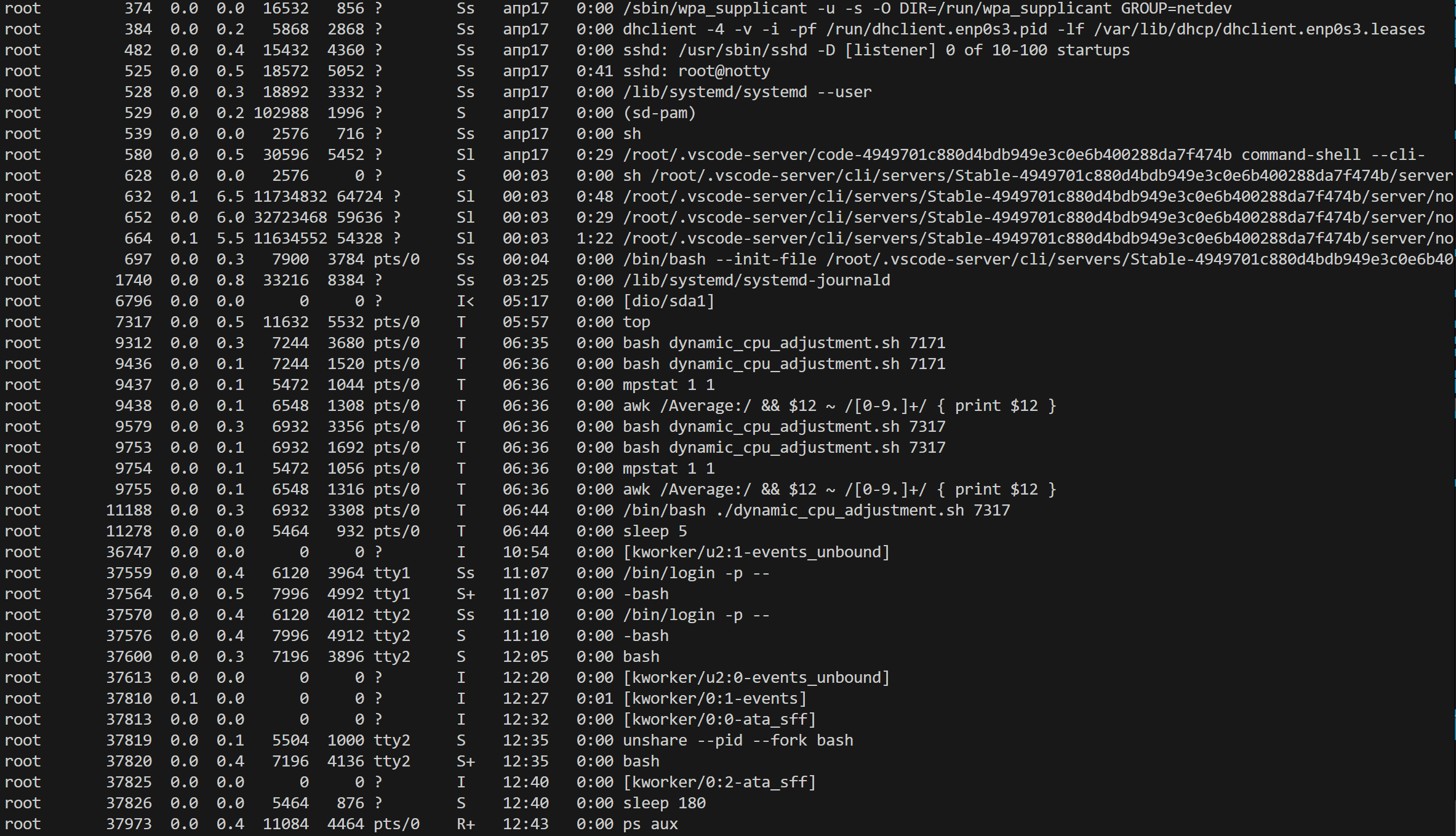
Изображение выглядит как текст, снимок экрана, Шрифт, черный

Контент, сгенерированный ИИ, может содержать ошибки.

Заходим в старый терминал и видим:

Изображение выглядит как текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.



Вывод:

В обычной среде показываются вообще все процессы, а в изолированном окружении мы видим только процессы текущего окружения.

**8. Изолированная файловая система (пространство имен Mount)**

* Создайте каталог, видимый только в пространстве имен:
  + unshare --mount bash.
* Создайте приватный каталог:
  + mkdir /tmp/private\_$(whoami).
* Смонтируйте временную файловую систему:
  + mount -t tmpfs tmpfs /tmp/private\_$(whoami).
* Проверьте изоляцию:
  + df -h | grep private\_$(whoami) # Запишите в отчет результат.

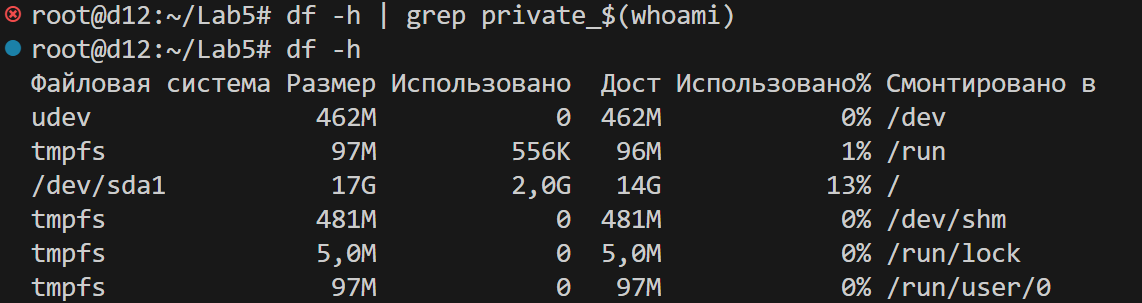
Изображение выглядит как текст, снимок экрана, Шрифт, черный

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Контент, сгенерированный ИИ, может содержать ошибки.

* Проверка на хосте (в новом терминале):
  + df -h | grep private\_$(whoami)

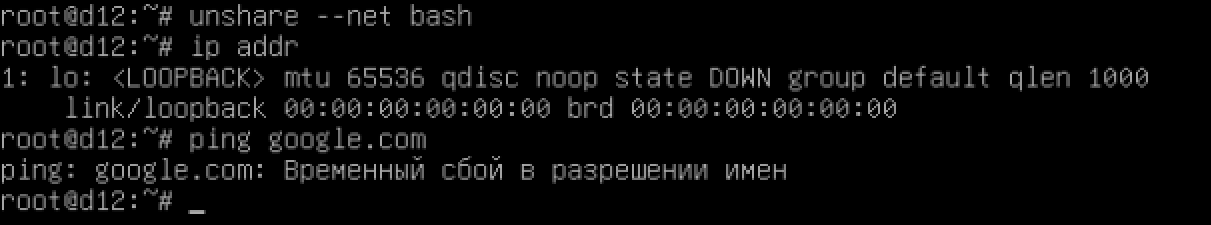


**9. Отключение доступа к сети (пространство имен Network)**

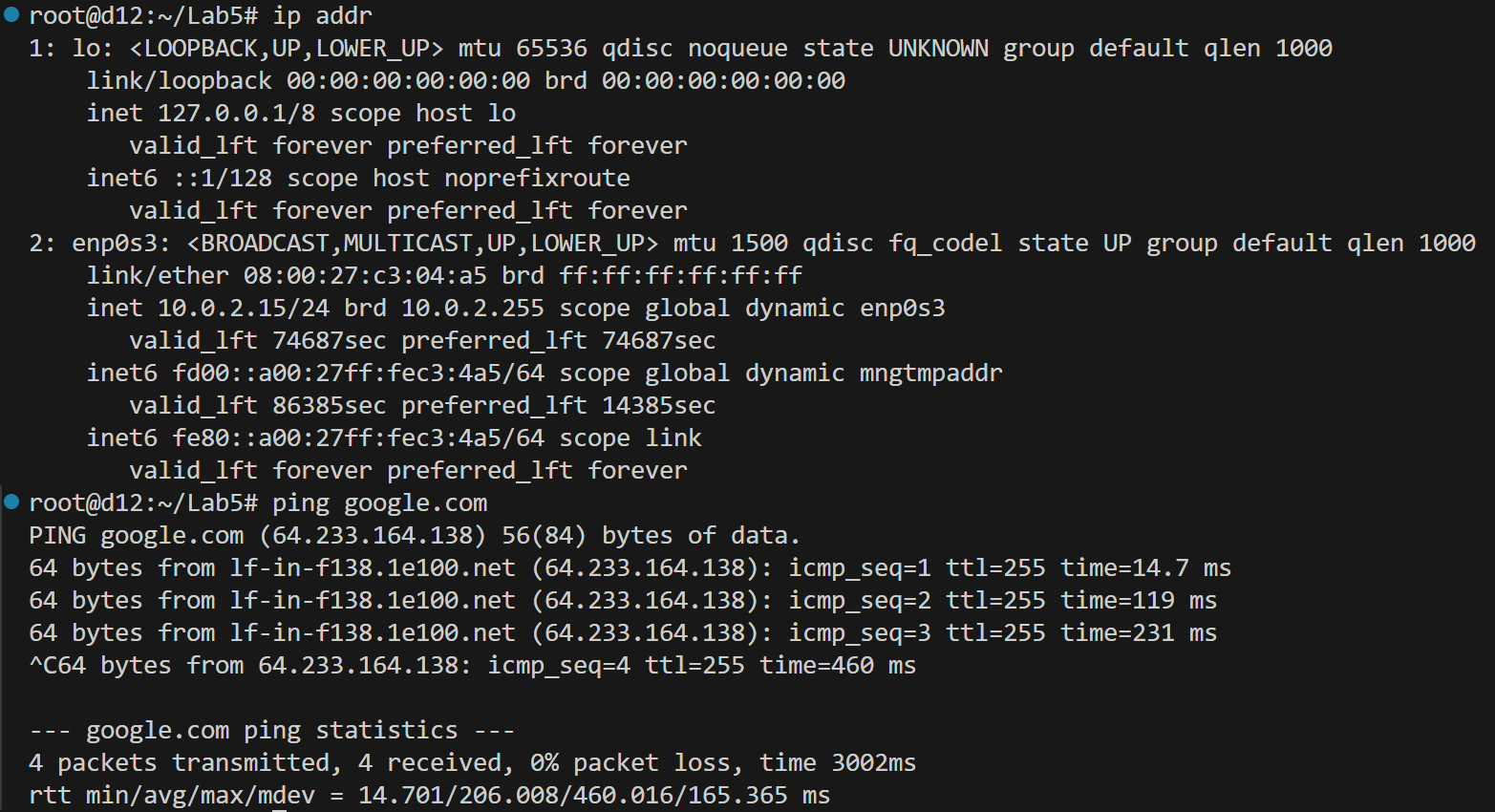
* Запустите командный интерпретатор bash без доступа к сети.

unshare --net bash – запуск без сети.

* Проверьте сетевые интерфейсы:
  + ip addr # Запишите в отчет, что показывает команда.
* Проверьте подключение:
  + ping google.com.
* Сравните с хостом (в новом терминале):
  + ping google.com



Пробуем то же самое в новом терминале:



**10. Создайте и проанализируйте монтирование OverlayFS Шаги:**

a. Первоначальная настройка:

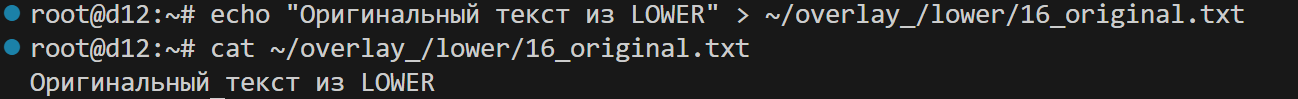
* Создайте каталоги: mkdir -p ~/overlay\_/{lower,upper,work,merged}
* В каталоге lower создайте файл с именем <ID>\_original.txt с содержанием: Оригинальный текст из LOWER
* Смонтируйте OverlayFS: mount -t overlay overlay -o lowerdir=~/overlay\_/lower,upperdir=~/overlay\_/upper,workdir=~/overlay\_/work ~/overlay\_/merged

mkdir -p ~/overlay\_/{lower,upper,work,merged}

Изображение выглядит как текст, Шрифт, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

echo "Оригинальный текст из LOWER" > ~/overlay\_/lower/16\_original.txt

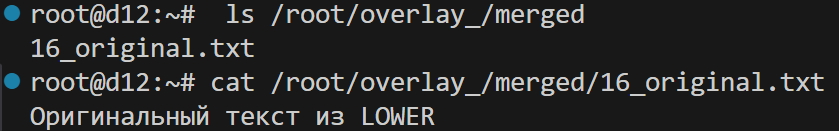


mount -t overlay overlay -o lowerdir=/root/overlay\_/lower,upperdir=/root/overlay\_/upper,workdir=/root/overlay\_/work /root/overlay\_/merged

Прописали команду выше и вот результат, видим, что все смонтировалось:

Изображение выглядит как снимок экрана, текст

Контент, сгенерированный ИИ, может содержать ошибки.

Если заглянем в /root/overlay\_/merged мы наткнемся на файл, созданный в директории /lower:  
  


lower/ - содержит оригинальные файлы

upper/ - изначально пуст, записываются все изменения

work/ - служебный, технический буфер, который позволяет OverlayFS корректно работать с изменениями, копированием и удалениями. Без него OverlayFS не сможет монтироваться вообще.

merged/ - сюда монтируется результат

b. Имитация неполадки и отладка:

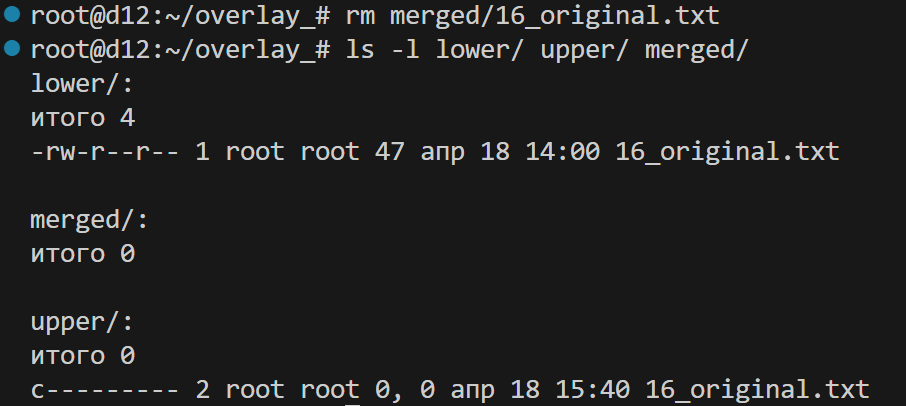
* Удалите файл \_original.txt из каталога merged.
* Понаблюдайте: Какой файл(ы) появился(ись) в верхнем каталоге? Задокументируйте их имена и содержимое.
* Измените каталог merged, чтобы восстановить \_original.txt, не размонтируя и не изменяя нижний уровень.

Посмотрим содержимое трёх директорий с подробной информацией (размер, права, дата, имя файлов).

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Выполним rm merged/27\_original.txt и получим:



* c в начале строки указывает на символьное устройство.
* 0, 0 — major и minor номера.

Это значит, что это whiteout, в контексте OverlayFS это удалённый файл.

Чтобы восстановить файл необходимо удалить его whiteout версию в upper:

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

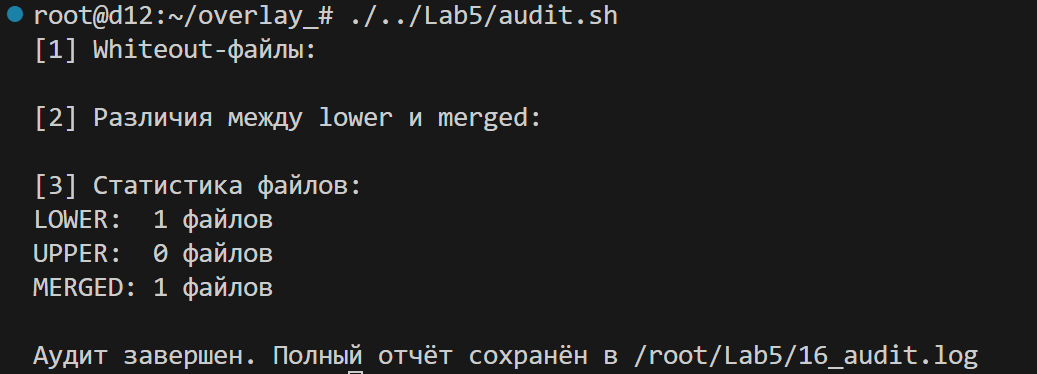
c. Разработайте скрипт, который:

* Обнаруживает все whiteout файлы в верхнем каталоге upper.
* Сравнивает содержимое нижнего и объединенного для выявления несоответствий.
* Выводит отчет с именем \_audit.log.

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Видим, что все работает:



Смотрим отчет:

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

d. Ответьте на вопросы:

* Как OverlayFS скрывает файлы из нижнего слоя при удалении в объединенном?

При удалении в merged слое, в слое lower файлы не удаляются. В upper добавляется специальный whiteout(с правами доступа c---…) файл, который прячет от merged слоя файл из lower слоя.

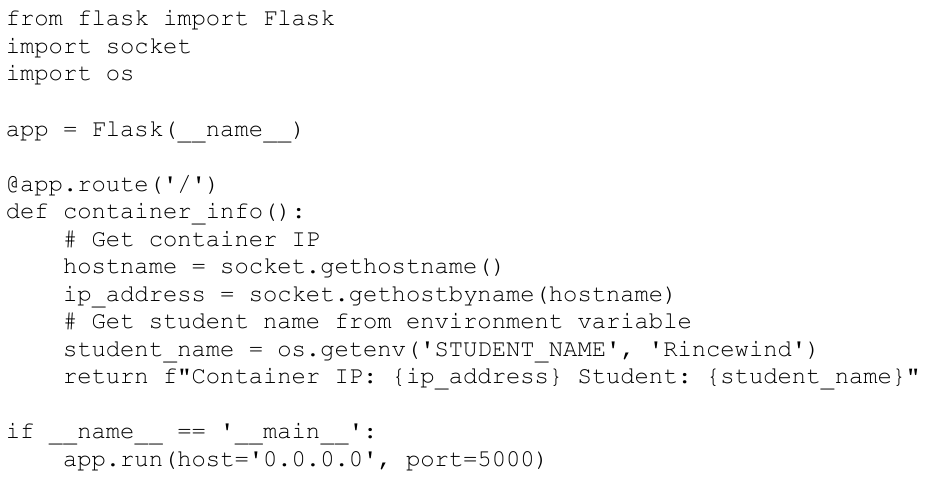
* Если вы удалите рабочий каталог work, сможете ли вы перемонтировать оверлей? Объясните, почему.

Если мы удалим work (вспомогательный каталог), то мы не сможем перемонтировать оверлей, так как он требует наличие lower, upper, work для инициализации и простых операций по типу переименования. При попытке монтирования без него будет ошибка.

* Что произойдет с объединенным слоем, если верхний каталог будет пуст?

Если каталог upper пуст ⇒ нет whiteout файлов ⇒ merged будет точной копией lower.

**11. Оптимизируйте Dockerfile для приведенного ниже приложения app.py**



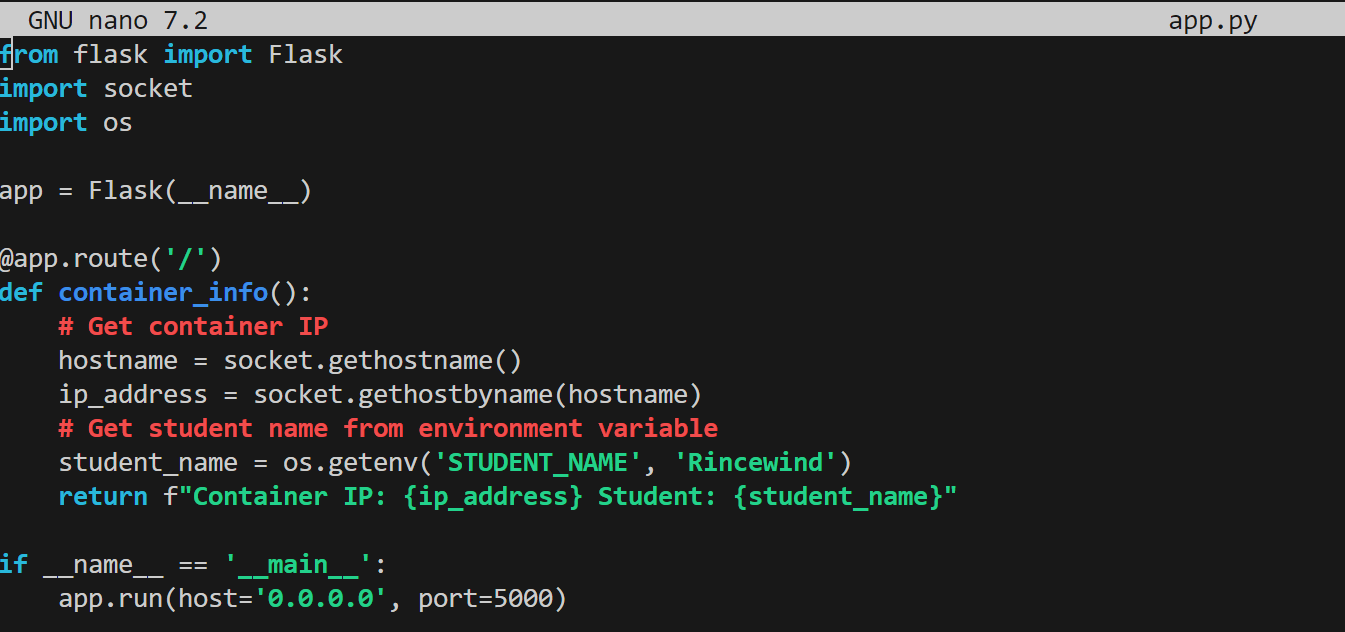
Исходный Dockerfile:

Изображение выглядит как текст, Шрифт, чек, белый

Контент, сгенерированный ИИ, может содержать ошибки.

Улучшите Dockerfile с учетом лучших практик:

* Используйте меньший базовый образ.
* Зафиксируйте версию образа.
* Запуск от имени пользователя, не являющегося root.
* Используйте кэширование слоев для зависимостей.
* Добавьте файл .dockerignore

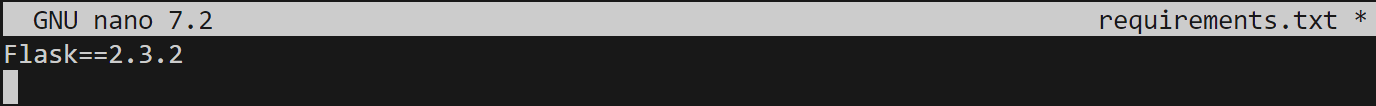


Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, Шрифт, программное обеспечение, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.



**12. Установка платформы публикации WordPress с помощью Docker Compose**

Задача:

Создать docker-compose.yml для запуска WordPress и MySQL/MariaDB с сохранением состояния при перезапуске контейнеров.

Используйте:

* Порт +2000 для WordPress (например, ID = 65 → port = 2065).
* Пароль базы данных: [ваше\_имя]\_db\_pass.
* Том с именем [ваше\_имя]-wp-data для WordPress

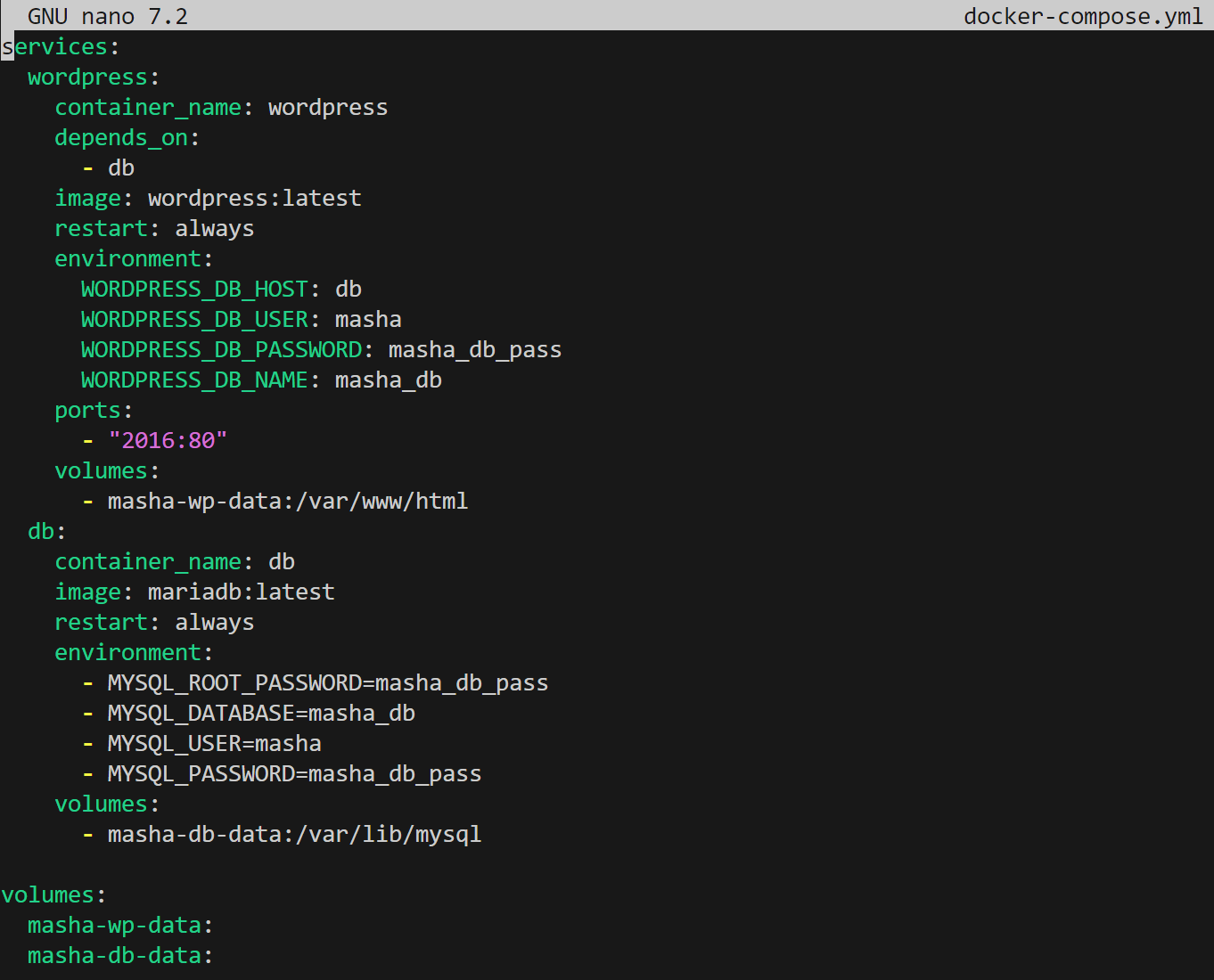
Скачиваем:

apt install curl

apt install docker-compose

Создаем файл docker-compose.yml и заполняем:

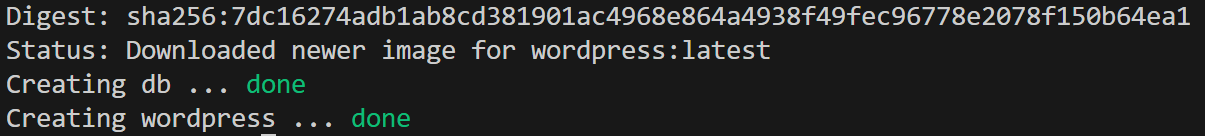
nano docker-compose.yml



docker-compose up -d

Изображение выглядит как текст, снимок экрана, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.



curl -I http://localhost:2016

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.