Лабораторная работа №5.

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

Выполнил: Михалев Никита Романович М3311.

## 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».

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

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

Мой isu номер – 368535, следовательно ID=35 и буду создавать user-35, который потребляет 50% CPU.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, Шрифт, снимок экрана, алгебра

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

## 2. Ограничение памяти для процесса (cgroups):

* Создайте cgroup для ограничения памяти, потребляемой процессом.
* Запустите процесс и переместите его в созданную вами группу. Пример команды: tail /dev/zero.
* Ограничьте потребление памяти следующим образом: ID\*10 + 500 МБ (например, ID=23 → 730 МБ).
* Проверьте, что при исчерпании памяти процессом он прерывается ОС.

Изображение выглядит как текст, Шрифт, снимок экрана, документ

Автоматически созданное описание

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

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

* Скрипт резервного копирования (backup.sh) перегружает дисковую подсистему.
* Ограничьте его до:
  + Чтение: 1000 + <ID>\*10 IOPS.
  + Запись: 500 + <ID>\*10 IOPS.
* Используйте cgcreate для установки ограничений io.max.
* Протестируйте с помощью fio или dd.

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, документ

Автоматически созданное описание

## 4. Закрепление к определенному ядру процессора для приложения

* Настройте с помощью cgroups процесс команды top за процессором 0.
* Используйте cpuset.cpus в cgroups.
* Проверьте с помощью taskset -p <PID>. (требуется пакет sysstat)

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, Шрифт, снимок экрана, белый

Автоматически созданное описание

## 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.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, алгебра

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, белый

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, алгебра

Автоматически созданное описание

## 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>.

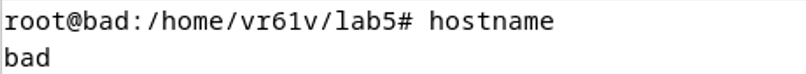
• Проверьте изоляцию:

◦ hostname # Должно отображаться «isolated-student-».

◦ Проверьте имя хоста (в новом терминале): hostname # По-прежнему показывает оригинальное имя хоста.

Изображение выглядит как текст, Шрифт, снимок экрана, белый

Автоматически созданное описание



## 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.

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, документ

Автоматически созданное описание

## 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\_$(whoami) # Запишите в отчет результат.
* Проверка на хосте (в новом терминале): df -h | grep private\_$(whoami).

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

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

* Запустите командный интерпретатор bash без доступа к 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.

Изображение выглядит как текст, снимок экрана, Шрифт, алгебра

Автоматически созданное описание

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

## 10. Создайте и проанализируйте монтирование OverlayFS

1. 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
2. Имитация неполадки и отладка:

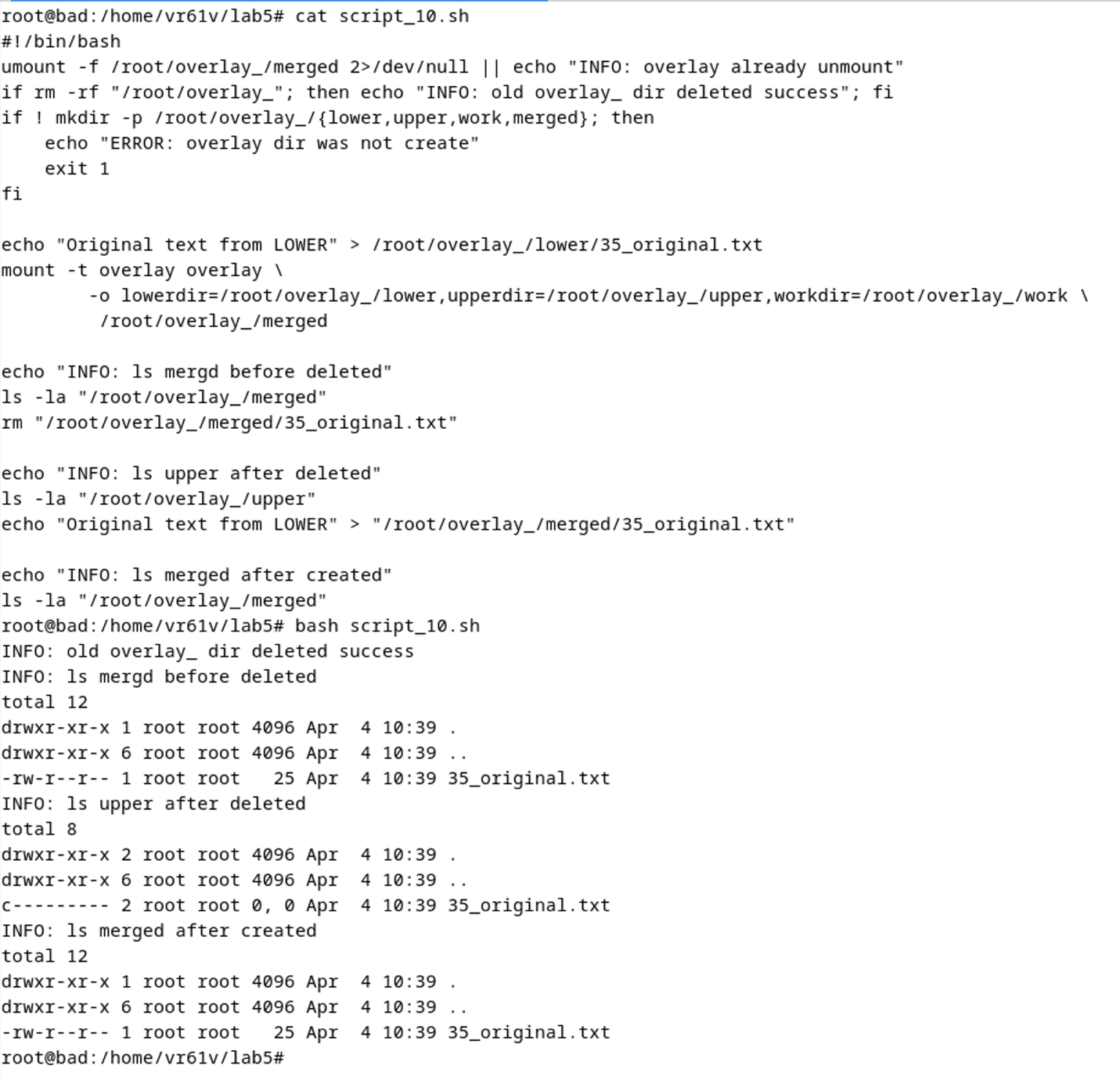
* Удалите файл <ID>\_original.txt из каталога merged.
* Понаблюдайте: Какой файл(ы) появился(ись) в верхнем каталоге? Задокументируйте их имена и содержимое.
* Измените каталог merged, чтобы восстановить <ID>\_original.txt, не размонтируя и не изменяя нижний уровень.

1. Разработайте скрипт, который:

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

1. Ответьте на вопросы:

* Как OverlayFS скрывает файлы из нижнего слоя при удалении в объединенном?
* Если вы удалите рабочий каталог work, сможете ли вы перемонтировать оверлей? Объясните, почему.
* Что произойдет с объединенным слоем, если верхний каталог будет пуст?



Изображение выглядит как текст, снимок экрана, Шрифт, документ

Автоматически созданное описание

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

* OverlayFS использует **whiteouts (белые отметки)**, чтобы скрывать файлы нижнего слоя (lowerdir), если они были "удалены" в объединённом слое (merged view).

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

* Если удалить workdir, то при попытке перемонтировать OverlayFS произойдёт **ошибка монтирования** так как отсутствует обязательный компонент.

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

* **Объединённый слой** будет отражать **только содержимое нижнего слоя**

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

App.py:

from flask import Flask

import socket

import os

app = Flask(\_\_name\_\_)

@app.route('/')

def container\_info():

# Get container IP

hostname = socket.gethostname()

ip\_address = socket.gethostbyname(hostname)

# Get student name from environment variable

student\_name = os.getenv('STUDENT\_NAME', 'Rincewind')

return f"Container IP: {ip\_address} Student: {student\_name}"

if \_\_name\_\_ == '\_\_main\_\_':

app.run(host='0.0.0.0', port=5000)

Исходный Dockerfile:

FROM python:latest

COPY . /app

WORKDIR /app

RUN pip install flask

CMD ["python", "app.py"]

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, документ, Шрифт

Автоматически созданное описание**

**Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание**

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

**Изображение выглядит как текст, снимок экрана, документ, меню

Автоматически созданное описание**

**Изображение выглядит как текст, программное обеспечение, Значок на компьютере, веб-страница

Автоматически созданное описание**