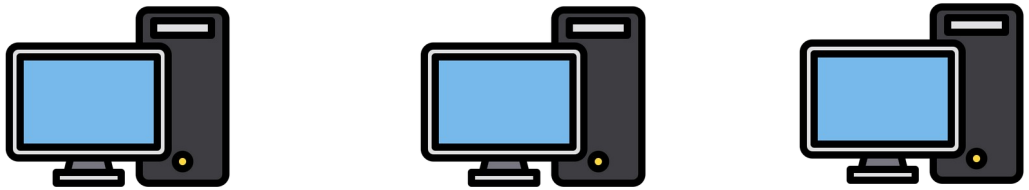
Разработка инфраструктуры программного обеспечения

Контейнеризация: podman (docker)

Лаборатория RISC-V технологий,
2025 г.

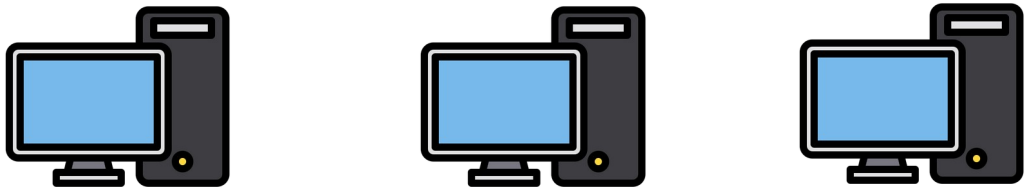
Проблема окружения

- Представьте себе типичную командную разработку, в котором каждый разработчик работает за своей машиной



Проблема окружения

- Представьте себе типичную командную разработку, в котором каждый разработчик работает за своей машиной
- Допустим, у вас есть зависимости по тестированию — googletests, а в ридми вашего проекта примерно такое: `sudo apt install libgtest-dev`



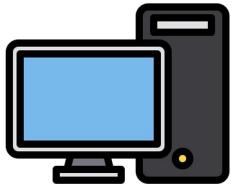
Проблема окружения

- Представьте себе типичную командную разработку, в котором каждый разработчик работает за своей машиной
- Допустим, у вас есть зависимости по тестированию — googletests, а в ридми вашего проекта примерно такое: `sudo apt install libgtest-dev`
- У первого юзера — устанавливается последний на данный момент **libtest/1.16.0**

Проблема окружения

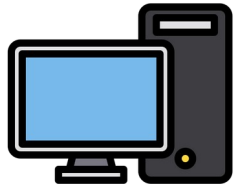
- Представьте себе типичную командную разработку, в котором каждый разработчик работает за своей машиной
- Допустим, у вас есть зависимости по тестированию — googletests, а в ридми вашего проекта примерно такое: `sudo apt install libgtest-dev`
- У первого юзера — устанавливается последний на данный момент **libtest/1.16.0**
- У второго юзера — Ubuntu 18.04 и устанавливается **libgtest/1.12.0**

Ubuntu
24.04 user

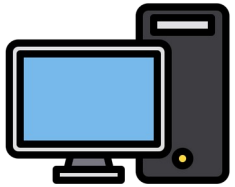


libgtest
1.16.0

Ubuntu
18.04 user



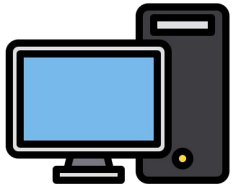
libgtest
1.12.0



Проблема окружения

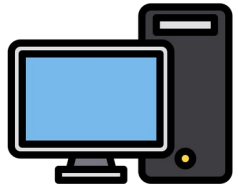
- Представьте себе типичную командную разработку, в котором каждый разработчик работает за своей машиной
- Допустим, у вас есть зависимости по тестированию — googletests, а в ридми вашего проекта примерно такое: `sudo apt install libgtest-dev`
- У первого юзера — устанавливается последний на данный момент **libtest/1.16.0**
- У второго юзера — Ubuntu 18.04 и устанавливается **libgtest/1.12.0**
- У третьего используется арч и вообще другой пакетный менеджер (не apt)

Ubuntu 24.04 user



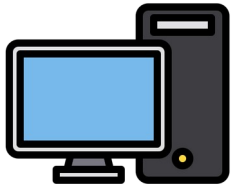
libgtest 1.16.0

Ubuntu 18.04 user



libgtest 1.12.0

BTW arch user

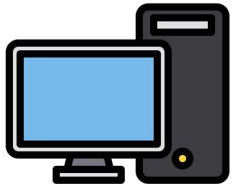


ERROR: apt not found?

Проблема окружения

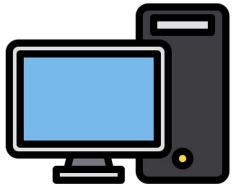
- Что делать arch пользователю?

Ubuntu 24.04 user



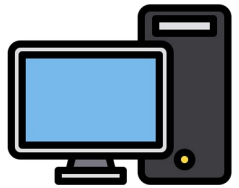
libgtest 1.16.0

Ubuntu 18.04 user



libgtest 1.12.0

BTW arch user

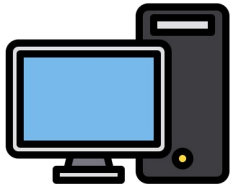


ERROR: apt not found!

Проблема окружения

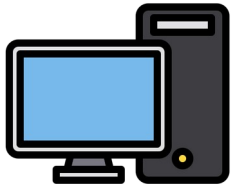
- Что делать arch пользователю?
- Скорее всего, ему придется ставить данную зависимость в систему альтернативными способами: поставить с помощью другого пакетного менеджера, собрать руками, скачать готовые бинарники с каких-нибудь сторонних релизов

Ubuntu 24.04 user



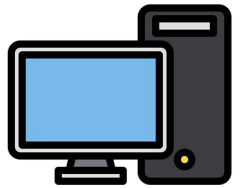
libgtest 1.16.0

Ubuntu 18.04 user



libgtest 1.12.0

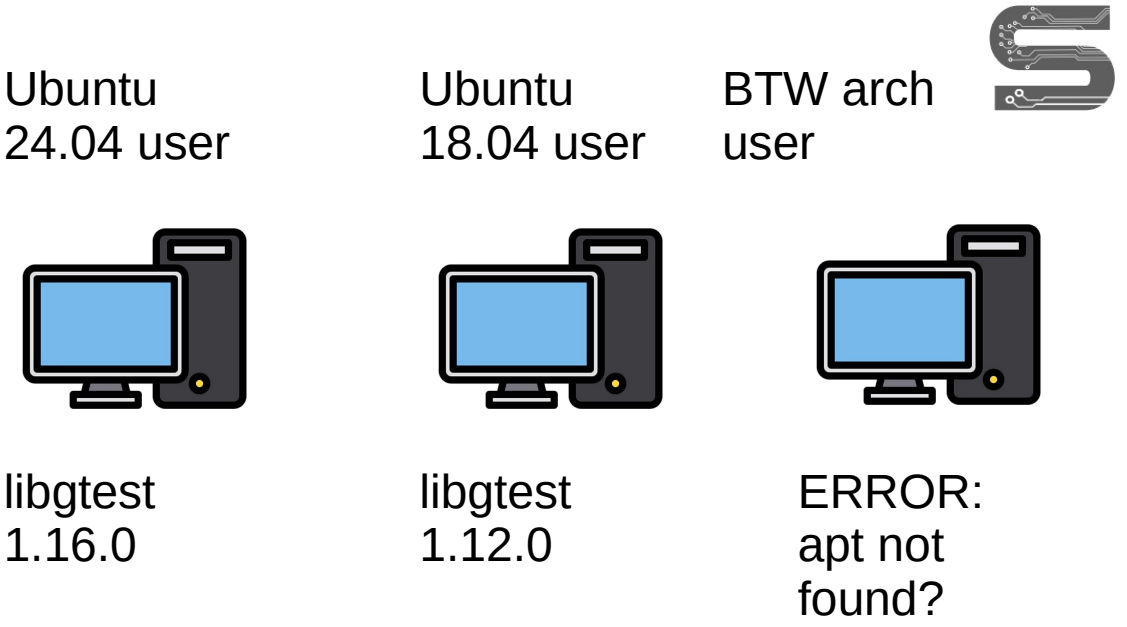
BTW arch user



ERROR: apt not found?

Проблема окружения

- Что делать arch пользователю?
- Скорее всего, ему придется ставить данную зависимость в систему альтернативными способами: поставить с помощью другого пакетного менеджера, собрать руками, скачать готовые бинарники с каких-нибудь сторонних релизов
- В любом случае версия **libgtest** у всех пользователей будет **разная**



Проблема окружения

- К чему могут привести разные версии данной зависимости на различных машинах

Проблема окружения

- К чему могут привести разные версии данной зависимости на различных машинах

```
error: ‘::testing::InitGoogleTest’ has not been declared
```

Проблема окружения

- К чему могут привести разные версии данной зависимости на различных машинах

```
error: ‘::testing::InitGoogleTest’ has not been declared
```

```
/usr/bin/ld: CMakeFiles/test.dir/test.cpp.o: in function `main':
test.cpp:(.text+0x15): undefined reference to `testing::InitGoogleTest'
```

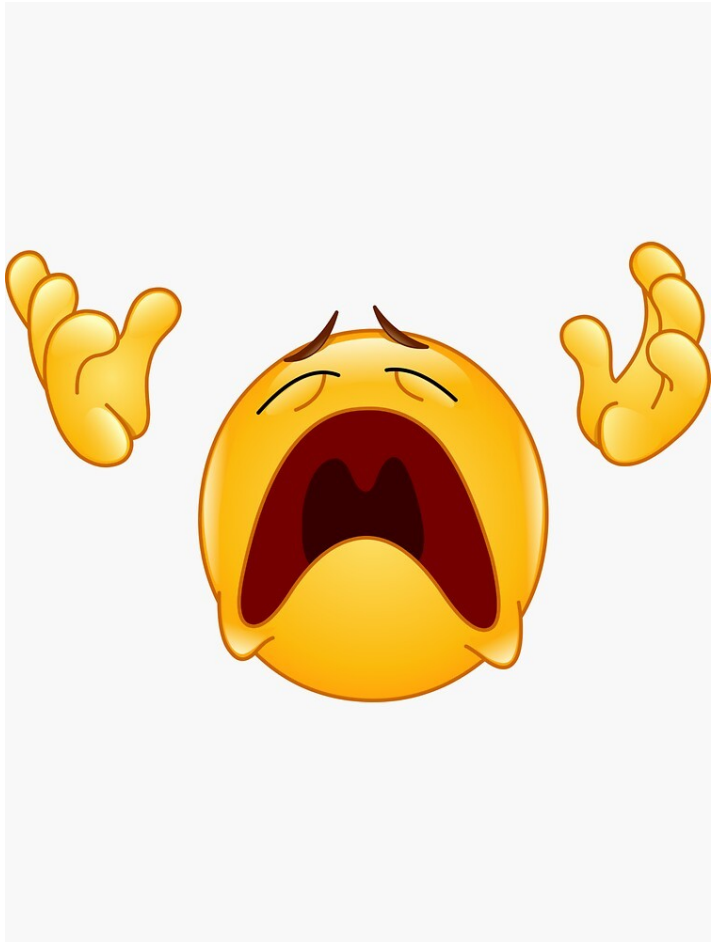
Проблема окружения

- К чему могут привести разные версии данной зависимости на различных машинах

```
error: ‘::testing::InitGoogleTest’ has not been declared
```

```
/usr/bin/ld: CMakeFiles/test.dir/test.cpp.o: in function `main':
test.cpp:(.text+0x15): undefined reference to `testing::InitGoogleTest'
```

```
error: ‘ASSERT_EQ’ does not compile inside a function returning void
```



Проблема окружения

- Теперь давайте масштабируем данные примеры

Проблема окружения

- Теперь давайте масштабируем данные примеры
- Представьте, что проблемная зависимость у вас не одна, а огромное множество

- Например, в крупном проекте с большим количеством зависимостей, при отсутствии статической линковки, динамические библиотеки будут браться из системы

Проблема окружения

- Теперь давайте масштабируем данные примеры
- Представьте, что проблемная зависимость у вас не одна, а огромное множество

- Например, в крупном проекте с большим количеством зависимостей, при отсутствии статической линковки, динамические библиотеки будут браться из системы
- В таком случае сборка вашего проекта может одного человека на машине работать, а у другого — не будет

Проблема окружения

- Теперь давайте масштабируем данные примеры
- Представьте, что проблемная зависимость у вас не одна, а огромное множество

- Например, в крупном проекте с большим количеством зависимостей, при отсутствии статической линковки, динамические библиотеки будут браться из системы
- В таком случае сборка вашего проекта может одного человека на машине работать, а у другого — не будет
- Непонятно, с чем связаны ошибки в вашем проекте — с вашими коммитами или с проблемами в окружении
- **Окружение —** это набор параметров машины , на которой происходит конфигурация, сборка и тестирование, и **которые влияют на эти процессы**

Проблема окружения

- Теперь давайте масштабируем данные примеры
- Представьте, что проблемная зависимость у вас не одна, а огромное множество
- Например, в крупном проекте с большим количеством зависимостей, при отсутствии статической линковки, динамические библиотеки будут браться из системы
- В таком случае сборка вашего проекта может одного человека на машине работать, а у другого — не будет
- Непонятно, с чем связаны ошибки в вашем проекте — с вашими коммитами или с проблемами в окружении
- **Окружение —** это набор параметров машины , на которой происходит конфигурация, сборка и тестирование, и **которые влияют на эти процессы**

system libraries

System calls

Dependencies versions

Shell environment

Контейнеры

- Проблемзного окружения можно решить двумя способами

Контейнеры

- Проблемзного окружения можно решить двумя способами
- Способ 1 — работать всем на одной машине (например, общий сервер)

Контейнеры

- Проблемы разного окружения можно решить двумя способами
- Способ 1 — работать всем на одной машине (например, общий сервер)
- Плюсы: легко подключаться, быстродействие

Контейнеры

- Проблемы разного окружения можно решить двумя способами
- Способ 1 — работать всем на одной машине (например, общий сервер)
- Плюсы: легко подключаться, быстродействия
- Минусы: крайне дорого, необходимо настроить некоторые системы для общего пользования ресурсами (например, **coder**)

Контейнеры

- Проблемы разного окружения можно решить двумя способами
- Способ 1 — работать всем на одной машине (например, общий сервер)
- Плюсы: легко подключаться, быстродействия
- Минусы: крайне дорого, необходимо настроить некоторые системы для общего пользования ресурсами (например, **coder**)
- Способ 2 — использование контейнеров
- Контейнеры — это специальные приложения, которые в некотором виде **виртуализуют окружение и фиксируют его**

Контейнеры

- Проблемы разного окружения можно решить двумя способами
- Способ 1 — работать всем на одной машине (например, общий сервер)
- Плюсы: легко подключаться, быстродействия
- Минусы: крайне дорого, необходимо настроить некоторые системы для общего пользования ресурсами (например, **coder**)
- Способ 2 — использование контейнеров
- Контейнеры — это специальные приложения, которые в некотором виде **виртуализуют окружение и фиксируют его**
- В контейнере содержатся некоторый образ системы (например, ubuntu20, centos7 и т.д)

Контейнеры