|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | | | |
| Институт информационных технологий (ИТ) | |
| Кафедра инструментального и прикладного программного обеспечения (ИиППО) | |

|  |  |
| --- | --- |
| **ОТЧЁТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №11 - 13** | |
| **по дисциплине** | |
| «Системное программное обеспечение»  по теме: «Изоляция процессов на уровне ОС \*.nix» | |
|  | |
| Выполнил студент группы ИКБО-16-18 | Пономарев А.Р. |
| Принял преподаватель | Волков М.Ю. |

|  |  |  |
| --- | --- | --- |
| Работа выполнена | «\_\_»\_\_\_\_\_\_\_2021 г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_2021 г. | *(подпись руководителя)* |

Москва 2021

**Цель работы**

Основной целью данной практической работы является знакомство с программно-аппаратными решениями в области изоляции процессов и контейнеризации (контейнерной изоляции) на уровне операционных систем семейства \*.nix.

**Задание**

Использую сеть Интернет и другие открытые источники, необходимо найти и дать обобщенную характеристику средствам, позволяющие изолировать процессы на уровне операционной системы (в данном случае конкретная операционная система не играет роли). Так можно рассмотреть такие средства, как Sandboxie, [LXC](https://en.wikipedia.org/wiki/LXC), [OpenVZ](https://en.wikipedia.org/wiki/OpenVZ), [lmctfy](https://en.wikipedia.org/wiki/Lmctfy), Docker, IBM WPAR. Sandbox в windows 10 (начиная с обновлений 2019 года) и т.д.

**Ход работы**

**Docker**

Docker — программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации, контейнеризатор приложений. Позволяет «упаковать» приложение со всем его окружением и зависимостями в контейнер, который может быть развёрнут на любой Linux-системе с поддержкой cgroups в ядре, а также предоставляет набор команд для управления этими контейнерами. Изначально использовал возможности LXC, с 2015 года начал использовать собственную библиотеку, абстрагирующую виртуализационные возможности ядра Linux — libcontainer. С появлением Open Container Initiative начался переход от монолитной к модульной архитектуре.

Разрабатывается и поддерживается одноимённой компанией-стартапом, распространяется в двух редакциях — общественной (*Community Edition*) по лицензии Apache 2.0 и для организаций (*Enterprise Edition*) по проприетарной лицензии. Написан на языке Go.

Программное обеспечение функционирует в среде Linux с ядром, поддерживающим контрольные группы и изоляцию пространств имён (*namespaces*); существуют сборки только для платформ x86-64 и ARM. Начиная с версии 1.6 (апрель 2015 года) возможно использование в операционных системах семейства Windows.

Для экономии пространства хранения проект использует файловую систему Aufs с поддержкой технологии каскадно-объединённого монтирования: контейнеры используют образ базовой операционной системы, а изменения записываются в отдельную область. Также поддерживается размещение контейнеров в файловой системе Btrfs с включённым режимом копирования при записи.

В состав программных средств входит демон — сервер контейнеров (запускается командой docker -d), клиентские средства, позволяющие из интерфейса командной строки управлять образами и контейнерами, а также API, позволяющий в стиле REST управлять контейнерами программно.

Демон обеспечивает полную изоляцию запускаемых на узле контейнеров на уровне файловой системы (у каждого контейнера собственная корневая файловая система), на уровне процессов (процессы имеют доступ только к собственной файловой системе контейнера, а ресурсы разделены средствами libcontainer), на уровне сети (каждый контейнер имеет доступ только к привязанному к нему сетевому пространству имён и соответствующим виртуальным сетевым интерфейсам).

Набор клиентских средств позволяет запускать процессы в новых контейнерах (docker run), останавливать и запускать контейнеры (docker stop и docker start), приостанавливать и возобновлять процессы в контейнерах (docker pause и docker unpause). Серия команд позволяет осуществлять мониторинг запущенных процессов (docker ps по аналогии с ps в Unix-системах, docker top по аналогии с top и другие). Новые образы возможно создавать из специального сценарного файла (docker build, файл сценария носит название Dockerfile), возможно записать все изменения, сделанные в контейнере, в новый образ (docker commit). Все команды могут работать как с docker-демоном локальной системы, так и с любым сервером Docker, доступным по сети. Кроме того, в интерфейсе командной строки встроены возможности по взаимодействию с публичным репозиторием Docker Hub, в котором размещены предварительно собранные образы приложений, например, команда docker search позволяет осуществить поиск образов среди размещённых в нём, образы можно скачивать в локальную систему (docker pull), возможно также отправить локально собранные образы в Docker Hub (docker push).

Также Docker имеет пакетный менеджер Docker Compose, позволяющий описывать и запускать многоконтейнерные приложения. Конфигурационные файлы Compose описываются на языке YAML.

**WSL**

Windows Subsystem for Linux (WSL) — слой совместимости для запуска Linux-приложений (двоичных исполняемых файлов в формате ELF) в ОС Windows 10. В рамках сотрудничества компаний Майкрософт и Canonical стало возможным использовать оригинальный образ ОС Ubuntu 14.04 для непосредственного запуска поверх WSL множества инструментов и утилит из этой ОС без какой-либо виртуализации. WSL предоставляет интерфейсы, во многом совместимые с интерфейсами ядра Linux; однако подсистема WSL была полностью разработана корпорацией Майкрософт и не содержит в себе каких-либо исходных кодов ядра Linux. WSL запускает многие немодифицированные приложения, работающие в пространстве пользователя, в частности, оболочку bash, утилиты sed, awk, интерпретаторы языков программирования Ruby, Python, и т. д.

По состоянию на 2018 год, подсистема частично поддерживает запуск графических приложений Linux (использующих графические пользовательские интерфейсы GUI на базе X11) и не поддерживает запуск приложений, требующих нереализованных интерфейсов ядра Linux. Графические приложения работают только с внешним сервером оконной системы X11, например VcXsrv или Xming (с ограничениями). Возможен запуск среды Unity и программ из пакета OpenOffice, также работают браузеры (например, Firefox). WSL использует меньше ресурсов, чем полная виртуализация, и стала наиболее простым путем запуска многих Linux-приложений на ОС Windows. Приложения Windows и Linux, запущенные через WSL, имеют доступ ко всем файлам пользователя.

Подсистема WSL доступна только на 64-битных редакциях Windows 10 и может быть активирована на версиях Windows 10 Anniversary Update и более поздних. Подсистема начала разрабатываться в рамках неизданного проекта Astoria, который позволял бы запускать некоторые приложения Android на ОС Windows 10 Mobile. WSL был впервые представлен в Insider Preview Windows 10 build 14316.

Корпорация Microsoft позиционирует WSL в первую очередь как инструмент для разработчиков, веб-разработчиков и тех, кто работает над или с приложениями с открытым исходным кодом.

**WSL 2**

Windows Subsystem for Linux 2 (WSL 2) — это вторая версия подсистемы Windows для Linux. Новая архитектура WSL 2 обеспечивает иной способ взаимодействия дистрибутивов Linux с Windows. WSL 2 использует технологию виртуализации и ядро Linux для реализации новых возможностей.

Подсистема WSL 2 доступна только в Windows 10 версии 2004, сборки 19041 или выше.

Основные приоритеты WSL 2:

* увеличение производительности файловой системы;
* добавление полной совместимости системных вызовов.

Простыми словами Windows Subsystem for Linux 2 работает значительно быстрее чем предыдущая версия подсистемы Linux.

Сравнение возможностей

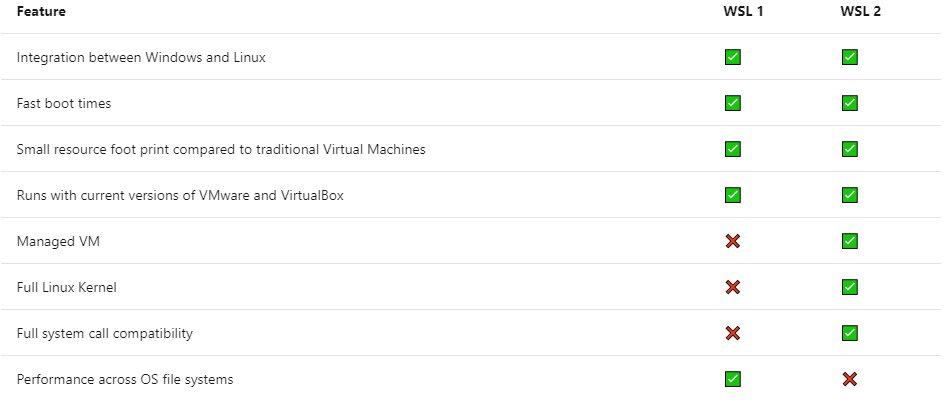


Рис.1 – Сравнение возможностей WSL и WSL 2.

Каждый дистрибутив Linux может переключаться между WSL 1 или WSL 2 в любое время.

Несмотря на то, что WSL 2 гораздо производительнее, есть два случая, когда рекомендуется использовать WSL 1:

* Файлы проекта должны храниться в файловой системе Windows.

Если вы будете использовать дистрибутив Linux WSL для доступа к файлам проекта в файловой системе Windows, и эти файлы не могут храниться в файловой системе Linux, вы получите более высокую производительность в файловых системах ОС, используя WSL 1.

* Проект, для которого требуется перекрестная компиляция с использованием средств Windows и Linux на одних и тех же файлах.

Операции с файлами в операционных системах Windows и Linux выполняются быстрее в WSL 1, чем на WSL 2. Поэтому если вы используете приложения Windows для доступа к файлам Linux, в настоящее время вы получите более высокую производительность при использовании WSL 1.

**Вывод**

В ходе практической работы мы ознакомились с программно-аппаратными решениями в области изоляции процессов и контейнеризации (контейнерной изоляции) на уровне операционных систем семейства \*.nix.

**Список использованных источников**

1. Пошаговое руководство. Создание классического оконного приложение [Электронный ресурс]. URL: https://docs.microsoft.com/ru-ru/cpp/windows/walkthrough-creating-windows-desktop-applications-cpp?view=msvc-160
2. Администрирование в Windows 10. [Электронный ресурс] : [https://docs.microsoft.com/ru-ru/windows/client-management/administrative-tools-in-windows-10 /](https://docs.microsoft.com/ru-ru/windows/client-management/administrative-tools-in-windows-10%20/) Ричард Ворд 2020 — 100 с.
3. «Тайм-менеджмент для системных администраторов». / Лимончелли Т.А. М.: Символ-плюс 2007 — 247 с. ISBN:978-5-93286-090-8
4. Википедия. Docker. [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Docker