Разработка генератора потока команд, управляющих восстановлением графа ресурсов в CRIU.

Сергеев Павел Алексеевич

научный руководитель: Баталов Евгений Александрович

СПб АУ НОЦНТ РАН

14 июня 2016 г.

Введение

CRIU – Checkpoint/Restore in User-space

CRIU (criu.org) – проект с открытым исходным кодом, предоставляющий функциональность по сохранению состояния linux приложения (checkpoint) с целью дальнейшего восстановления (restore).

Введение

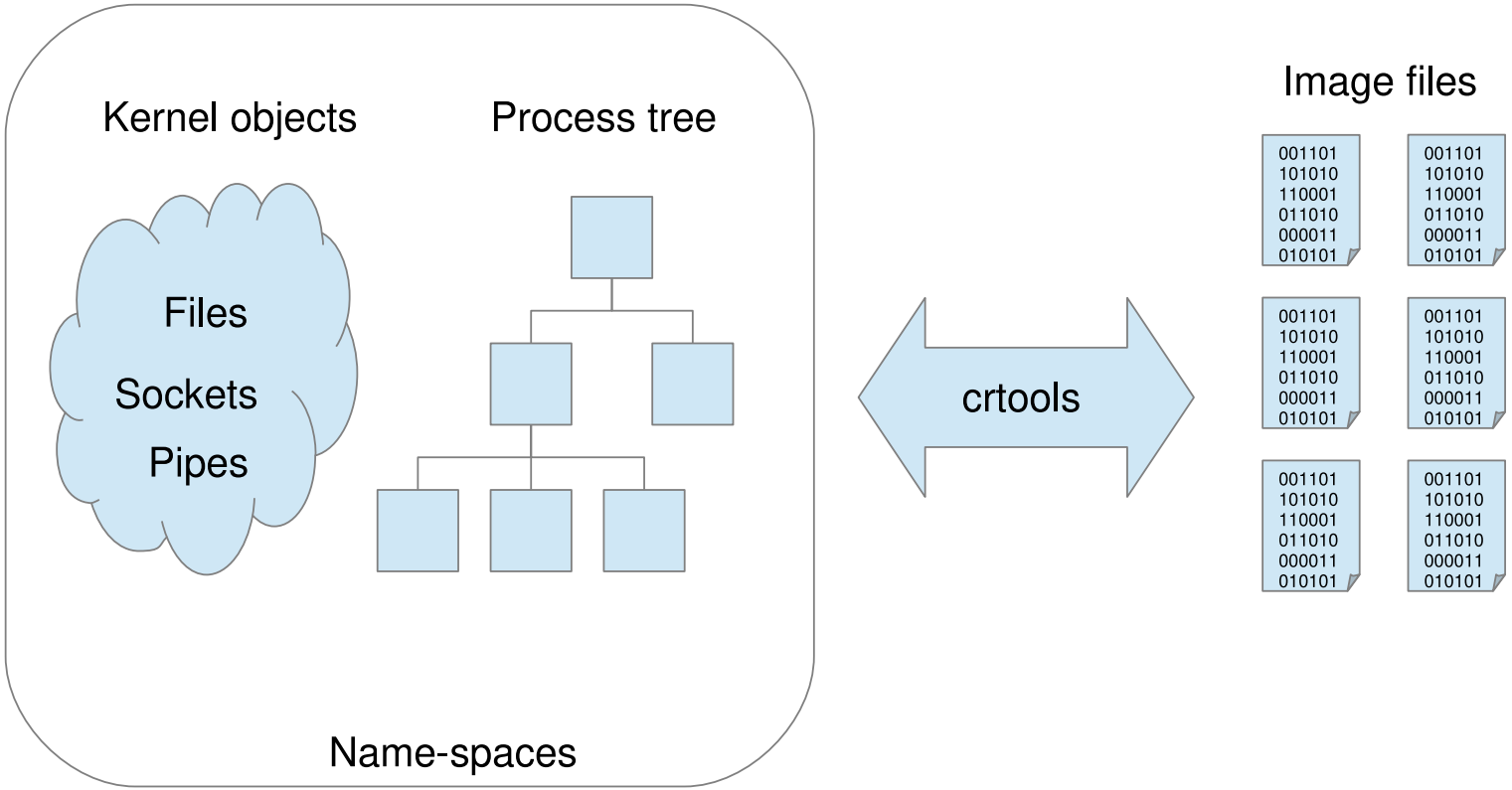
Сценарии использования CRIU

- Восстановление после сбоев
- Живая миграция работающих приложений
- Обновление ядра без перезапуска приложений
- Ускорение запуска медленных приложений
- Миграция контейнеров

Особенности CRIU

- Практически все делается в пространстве пользователя
- Работает на ядре linux 3.11 и старше
- Не требует предварительной модификации приложений

Как это работает?



Проблемы существующего подхода

Предпосылки

- В образах храниться состояние приложения, но не описан способ его получения.
- ОС не предоставляет простого способа для восстановления состояния.
- Логика, ответственная за взаимодействие с ОС, смешана с логикой, ответственной за процесс восстановления состояния.

Проблема

- Сложный код CRIU restorer.

Цель и задачи

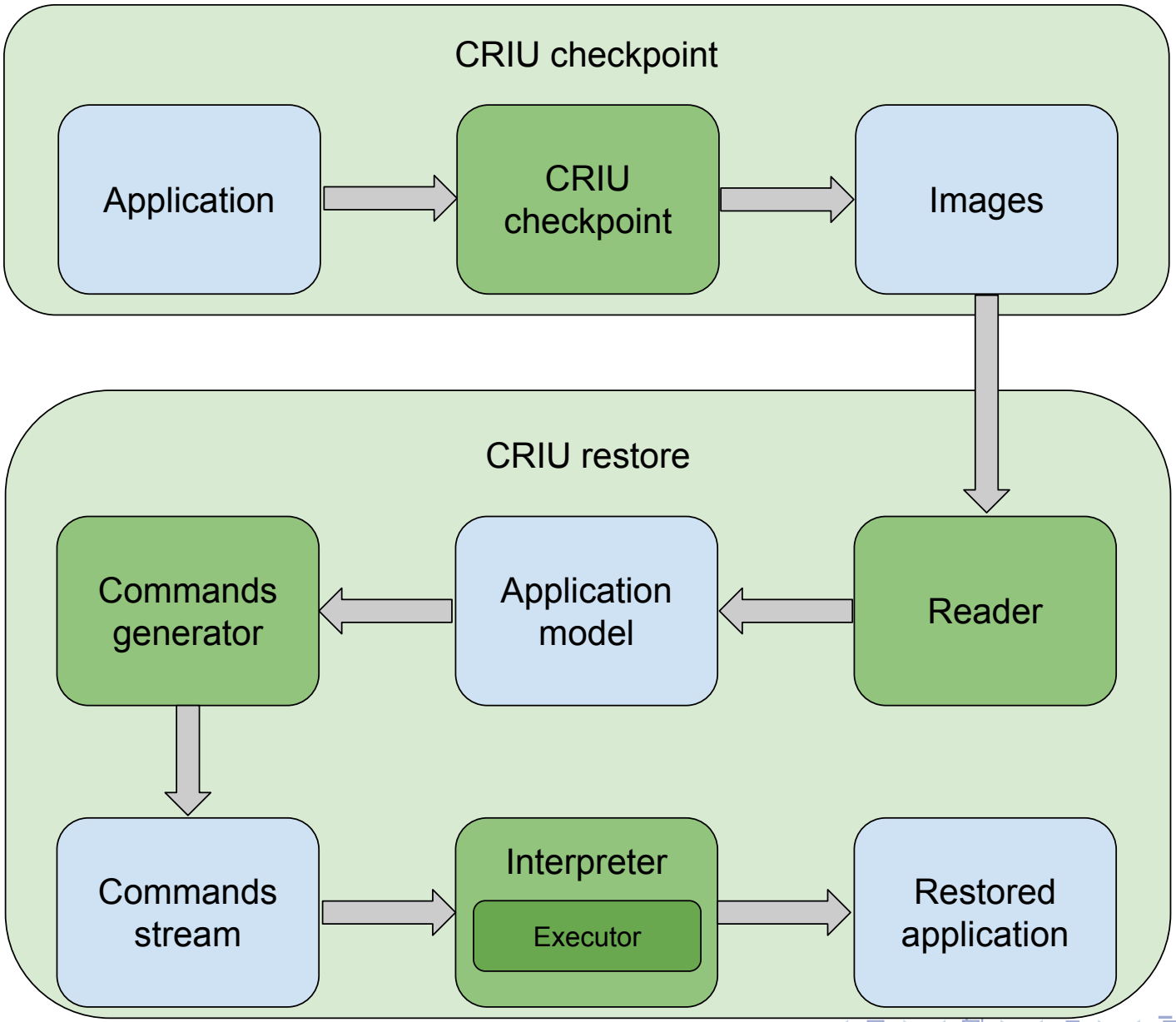
Цель

Разработать и реализовать подход к управлению процессом восстановления приложений в CRIU.

Задачи

- Формализовать модель, описывающую состояние приложения.
- Спроектировать DSL для описания команд восстановления.
- Предложить и реализовать алгоритм, генерирующий по модели приложения поток команд DSL.
- Предоставить набор тестов, проверяющих корректность алгоритма.

Архитектура решения

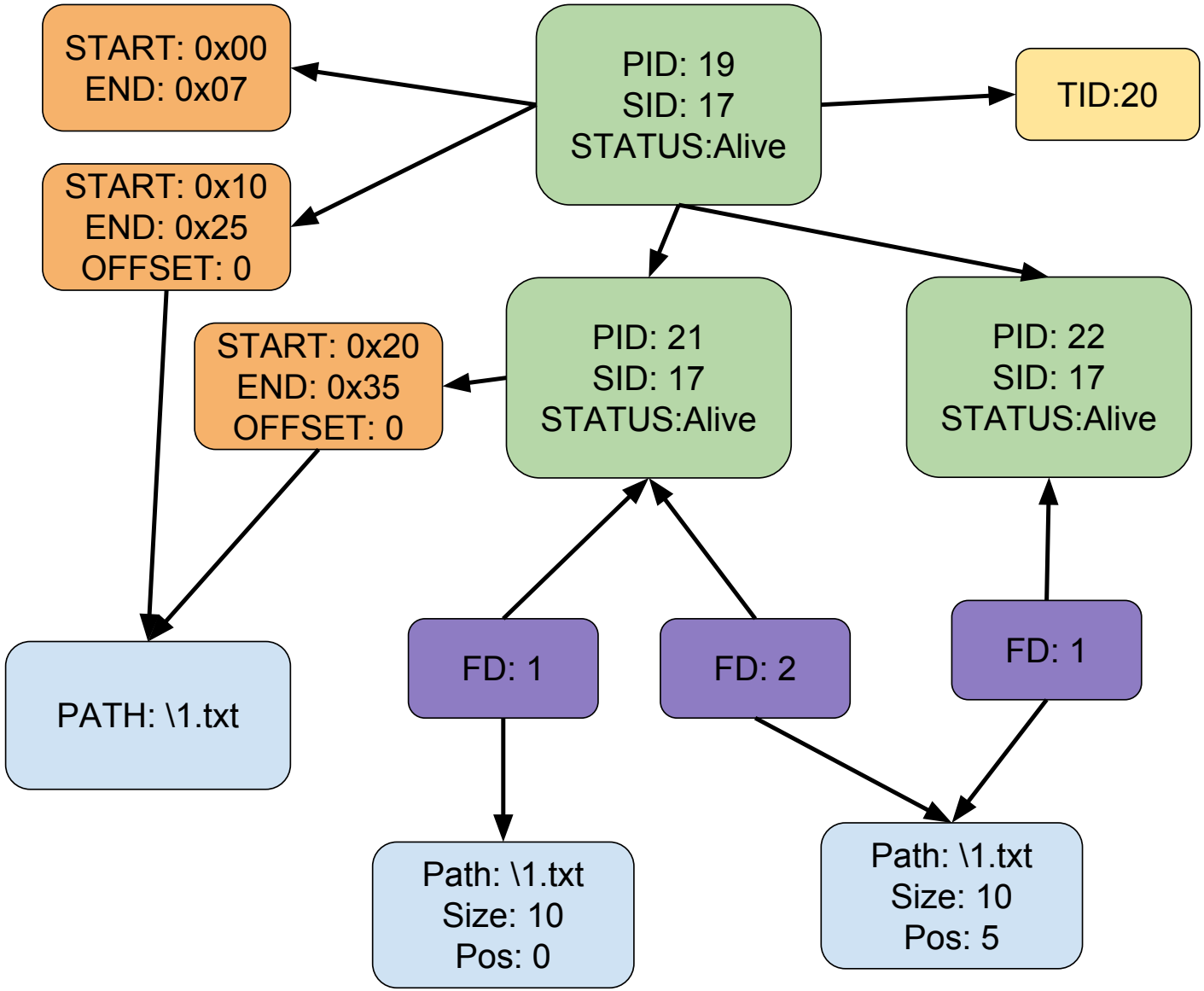


Модель состояния приложения

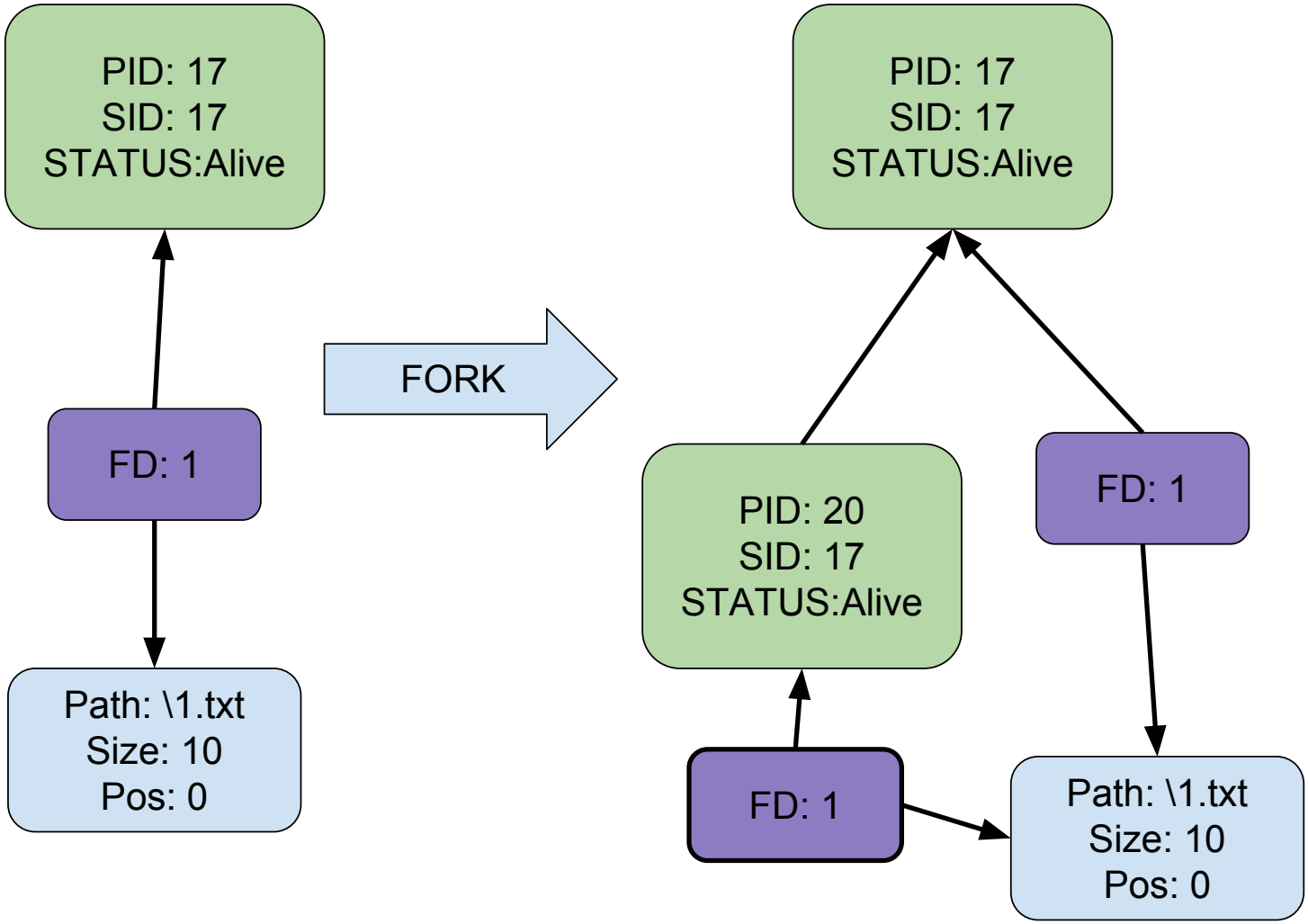
Модель (подмножества состояния) включает в себя:

- Дерево процессов
- Сессии
- Обычные файлы
- Регионы памяти
- Потоки

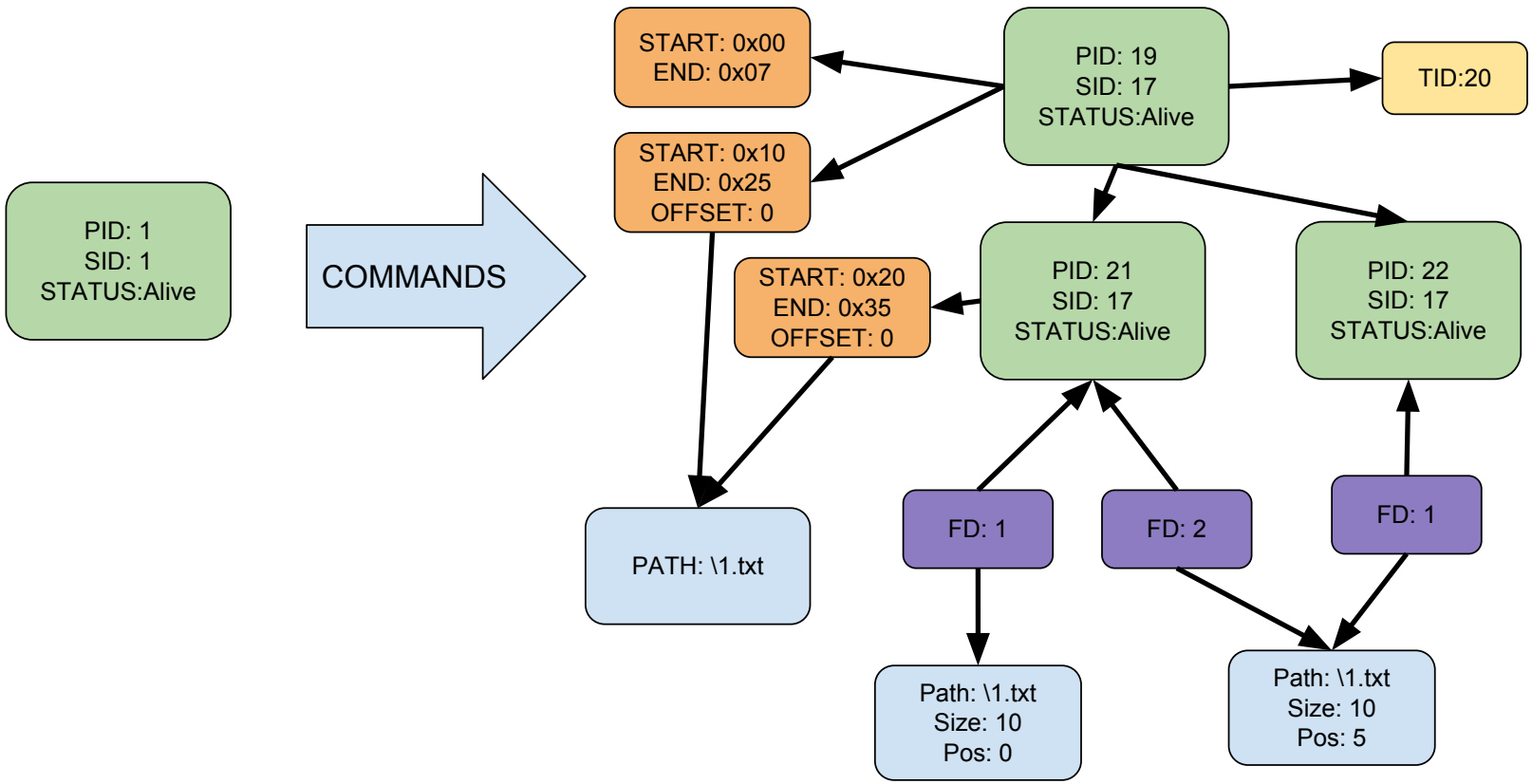
Пример модели



Преобразования модели



Генератор команд восстановления



Генератор команд восстановления

Основные шаги работы генератора

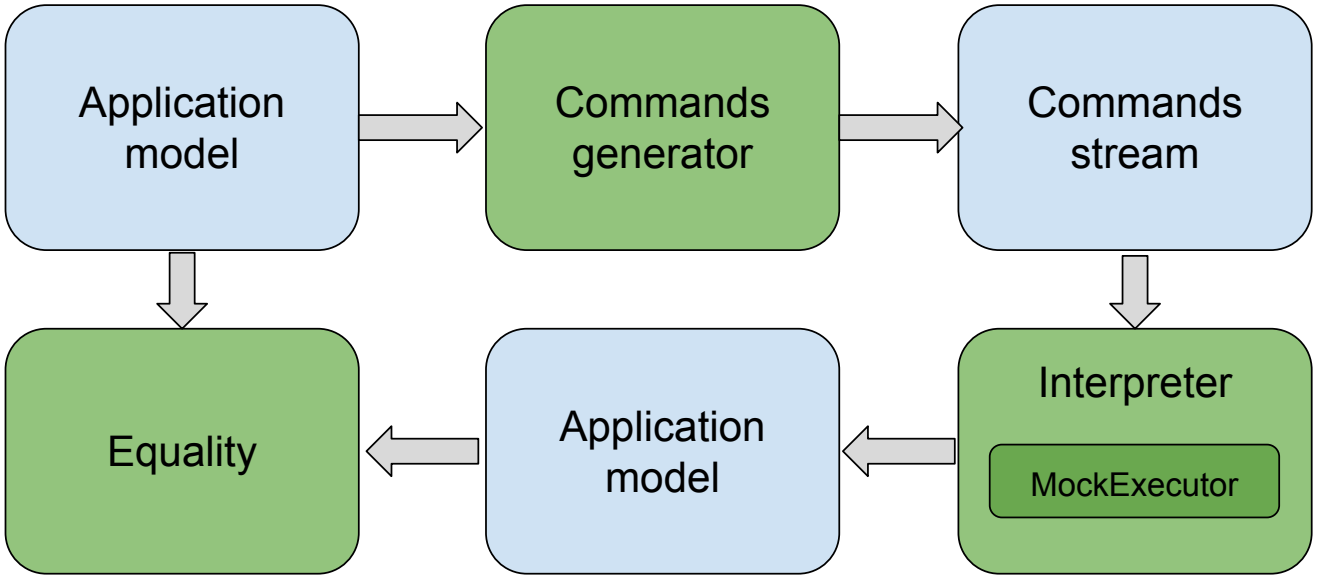
- Для каждого разделяемого ресурса определяется процесс-владелец.
- Для каждого процесса генерируются команды открытия ресурсов и создания потомков.
- Для каждого процесса генерируются команды переупорядочивания ресурсов.
- Генерация команд закрытия лишних ресурсов.

Синтаксис описания преобразований

- Реализовано 16 типов команд (преобразований).
- Синтаксис основан на формате JSON.
- Реализована проверка корректности программ на основе допустимых команд.

```
[
    {
        "command": "FORK",
        "pid": 0,
        "child_pid": 10
    },
    {
        "command": "SET_SID",
        "pid": 10
    }
]
```

Тестирование



Результаты

- Формализовано описание подмножество состояния приложения.
- Спроектирован DSL и реализован интерпретатор команд.
- Реализован генератор команд.
- Предоставлены тесты проверяющие корректность генератора.

Дальнейшие планы

- Реализовать модуль Executor.
- Расширить модель, включив остальные ресурсы.
- На данный момент, предполагается параллельное с CRIU развитье.