Разработка DRM-совместимых дисплейных драйверов для микроядерной ОС

Молодяков Д.С. Denis.Molodyakov@kaspersky.com

АО «Лаборатория Касперского», Россия, Москва, 125212, Ленинградское шоссе, д.39А с1

В процессе разработки дисплейных драйверов для операционных систем общего назначения возникает множество сложностей, среди которых – определение универсального интерфейса для взаимодействия с вышестоящим уровнем (инфраструктура, прикладное ПО). Среди ОС с к открытым исходным кодом можно выделить Linux, Android и Fuchsia. ОС Fuchsia имеет относительно небольшое количество инсталляций на платформы, ее технические решения не отработаны и трудно оценить, насколько они являются удачными. Linux и Android представляют больший интерес. Модель Android имеет существенный минус – закрытые компоненты рендерной части. В KasperskyOS в качестве пробной попытки был выбран интерфейс DRM (Direct Rendering Manager) из Linux. В докладе отражен опыт разработки и портирования видео-драйверов в KasperskyOS под этот интерфейс.

Общие сведения о DRM/KMS

За доступ к ресурсам видеокарты в Linux-системах отвечает компонент ядра DRM. DRM (графический стек в целом) условно можно поделить на две функционально разных части – рендерная (render path) и часть, отвечающая за вывод картинки и управление устройством отображения (display path). Render path останется за рамками, а display path посвящен настоящий доклад, куда входят дисплейные драйвера и та часть монолитных драйверов, которая отвечает за работы с дисплеями.

KMS device model – подсистема DRM, определяющая модель конвейера отображения (display pipeline). Эту модель должны имплементировать все DRM-совместимые драйвера, которые предоставляют возможность управления режимами отображения. Planes, CRTCs (Cathode-Ray Tube Controller), Encoders, Connectors, и т.д. – основные логические блоки, определяющие структуру конвейера отображения. Ниже будет показано как реализации дисплейных драйверов в KasperskyOS под различные устройства имплементируют эту модель.

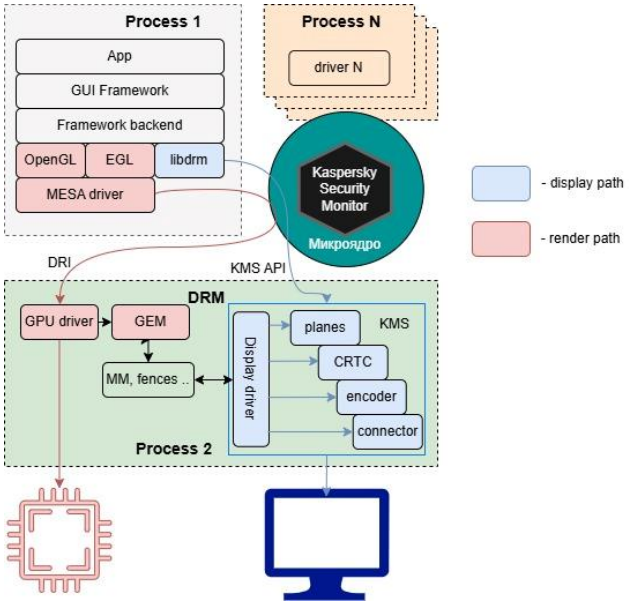

Рис.1 Реализация графического стека в KasperskyOS

Примеры реализация дисплейных драйверов в KasperskyOS

- Intel UHD Graphics. Intel во многом определил облик графического стека в Linux, в т.ч. модель конвейере отображения. Аппаратные блоки дисплейного контроллера практически полностью соотносятся с логическими блоками модели KMS. В докладе отражена архитектура аппаратуры дисплейной части на примере одной из моделей графического ускорителя intel.
- BGA (Bochs Graphics Adapter). Простейший вариант контроллера дисплея. По умолчанию эмулируется qemu для платформы x86_64. Предоставляет доступ к фреймбуферу (LFB - Linear FrameBuffer). Практически все логические блоки модели KMS эмулируют работу аппаратуры. В докладе представлена архитектура программной реализации драйвера.
- Virtio GPU. Драйвер паравиртуализованной видеокарты. Используется для работы ОС в качестве «гостя» под эмулятором (например, qemu). Для вывода изображения и рендеринга используется видеокарта хоста. Данные для вывода на экран кодируются в драйвере гостевой ОС и декодируются на стороне эмулятора в соответствии с спецификацией virtio. В докладе представлена структурная схема работы графического стека с virtio GPU драйвером.
- MediaTek. Аппаратный конвейер отображения довольно сложен. Простейший путь вывода изображения на экран включает аппаратные блоки: Overlayer, блоки коррекции изображения (Color correction, Gamma correction и пр.), RDMA, DSI. В докладе представлена архитектуры аппаратной части и программной реализации.

Особенности дисплейных драйверов для микроядерной архитектуры

Все компоненты системы изолированы, взаимодействие осуществляется через механизм IPC, что накладывает определённые ограничения:

- Временной лаг на взаимодействие с компонентами платформы.
- Передача параметров by-value. В KasperskyOS нет механизмов copy-from-user/copy-to-user. Передача параметров происходит по значению,
- Обработка прерываний происходит в user space. Особенно это важно для обработки событий hot-plug – подключение/отключение дисплея.
- Контроль за трансфером данных, в т.ч. объектов видеопамяти, между драйвером и клиентами берет на себя монитор безопасности KasperskyOS,
- Управление доступом клиентов к функциональным возможностям драйверов можно регулировать посредством политик безопасности.

Выводы

Разработка драйверов под интерфейс графического стека Linux дает огромные преимущества в плане совместимости с прикладным ПО – фреймворки, композиторы и пр. Однако модель KMS в значительной мере ориентирована на аппаратную архитектуру intel и на рассмотренных платформах, за исключением intel, затрудняет построение «чистой» архитектуры драйвера, требуя эмуляцию работы некоторых блоков конвейера отображения.

Функционирование драйверов в пользовательском пространстве и изоляция компонентов накладывают ограничения на взаимодействие с прикладным ПО – существуют дополнительные временные штрафы на коммуникацию компонентов в момент инициализации и смены режимов отображения. Поскольку инициализация происходит однократно, а смена режимов не является часто выполняемой операцией в прикладных сценариях, можно говорить о незначительном влиянии таких временных штрафов на функционирование системы. Кроме того, изоляция компонентов позволяет более гранулярно контролировать потоки данных и регулировать права доступа к ресурсам/сервисам через политики безопасности.