

ТРУДНОСТИ РАЗРАБОТКИ СВОБОДНОГО СИСТЕМНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ УСТРОЙСТВ НА АРХИТЕКТУРЕ ARM

А. Ю. Тимонин¹, Д. И. Макеев², О. Р. Филиппов³

^{1,2,3}Пензенский государственный университет, Пенза, Россия

¹timonin@pnzgu.ru

²danilmakeev6@gmail.com

³olegfil26@ya.ru

Аннотация. Рассматриваются особенности портирования операционных систем семейства Linux на устройства с архитектурной платформой ARM: ARMv7 и Aarch64. Проведен анализ текущего опыта разработки существующих специализированных дистрибутивов Linux с выделением основных сложностей, возникающих при запуске и функционировании системного ПО на подобных устройствах. Предложены идеи для новой методики, упрощающей и автоматизирующей процесс адаптации ядра Linux под конкретные ARM-устройства.

Ключевые слова: операционные системы, ядро Linux, мобильные устройства, ARM архитектура, свободное программное обеспечение, дерево устройства

DIFFICULTIES IN DEVELOPING FREE SYSTEM SOFTWARE FOR DEVICES BASED ON ARM ARCHITECTURE

A. Y. Timonin¹, D. I. Makeev², O. R. Filippov³

^{1,2,3}Penza State University, Penza, Russia

¹timonin@pnzgu.ru

²danilmakeev6@gmail.com

³olegfil26@ya.ru

Abstract. The article discusses the features of porting Linux-based operating systems to the ARM platform devices (ARMv7 and Aarch64). An analysis of the current experience of developing specialized Linux distributives was carried out. Also, common difficulties of launching and processing system software on ARM devices is highlighted. Some ideas have been proposed for a new technique that simplifies and automates the adaptation the Linux kernel to specific ARM devices.

Keywords: operating system, Linux kernel, mobile devices, ARM platform, free software, device tree

В настоящее время широко развит рынок «умной» потребительской электроники, которая в своем составе содержит полноценные микропроцессорные модули с памятью и многофункциональной шиной ввода-вывода. Такие устройства

обладают свойствами классических IBM-совместимых персональных компьютеров с архитектурой x86-x64. Однако, в целях достижения высокой энергоэффективности и компактности большая часть «умной» электроники основывается на других аппаратных архитектурах, таких как MIPS, RISC-V, ARM и других. Это связано в большей степени с тем, что подобные устройства в основном являются встраиваемыми или мобильными. Большинство смартфонов, планшетов, микрокомпьютеров и компактных ноутбуков основываются на 64 битной платформе компании ARM – Aarch 64. В прошлом десятилетии также были распространены 32 битные платформы – ARMv6 и ARMv7. Количество произведенных устройств с данными архитектурами исчисляется миллиардами [1]. Основными используемыми на них операционными системами являются Google Android, Microsoft Windows, Apple iOS и MacOS, а также специализированные дистрибутивы Linux, наподобие ARMbian и Raspberry Pi OS. [1, 2]

Наибольшее разнообразие представляют мобильные устройства на базе ОС Android. Однако производители не закладывают большой жизненный цикл для таких устройств – обновления системного ПО поставляются максимум 5 лет, но обычно этот срок составляет 1–2 года [1, 2]. Данный факт приводит к тому, что пользоваться такими устройствами становится некомфортно, т. к. новое прикладное ПО адаптируется в первую очередь под наиболее свежие и актуальные версии операционных систем, и небезопасно, так как найденные программные уязвимости никто не устраняет, а в проприетарных операционных системах могут быть «закладки» для удаленного контроля над устройствами третьими лицами. Решением этих проблем занимаются сообщества пользователей-энтузиастов, которые адаптируют новые версии предустановленных операционных систем, либо занимаются разработкой и портированием альтернатив. Здесь можно упомянуть такие проекты, как:

- AOSP – стандартная версия Android от компании Google, свободная от сторонних проприетарных надстроек [3];
- LineageOS – форк Android, развиваемый сообществом, в котором использование сервисов Google в базовой поставке сведено к минимуму [4];
- Ubuntu Touch – адаптированная под мобильные устройства редакция дистрибутива Ubuntu, первоначально развиваемая компанией Canonical, впоследствии — сообществом UBports [5];
- PostmarketOS – наиболее успешный проект по адаптации операционной системы Linux под множество различных ARM-устройств. Основывается на компактном дистрибутиве Alpine Linux, используемом во встраиваемых решениях и системах контейнеризации. Целями проекта является продление срока эксплуатации мобильных устройств после завершения сроков поддержки производителем, а также расширение их функциональных возможностей до уровня полноценного персонального компьютера [6].

Основная сложность портирования системного ПО заключается в отсутствии стандартизации при проектировании ARM-устройств, а также закрытости спецификаций аппаратных компонентов устройств, программного кода драйверов и прочих частей предустановленной операционной системы, поставляемых в виде скомпилированных бинарных файлов (блобов). Даже если производители систем на чипах (SoC) предоставляют исходные коды и полную документацию на свою продукцию, остаются периферийные устройства, такие как сетевые карты и модули камер, прошивки которых закрыты. Также зачастую производители конечных устройств вносят в их архитектуру свои нестандартизированные и часто не документируемые изменения и настройки. Частичным решением данной проблемы является проект Halium [7], унифицирующий уровень аппаратной абстракции для

устройств на базе системы Android. Его минусами являются несовместимость с другими операционными системами, а также ограниченный круг совместимости с аппаратными платформами (устройства, выпущенные до 2017 года, не поддерживаются).

Рассмотрим основные трудности, возникающие при портировании операционных систем семейства Linux на ARM-устройства.

В связи с тем, что у значительной части персональных ARM-устройств присутствует сенсорный экран и отсутствует аппаратная клавиатура, изменяется подход к пользовательскому интерфейсу [8]. Он должен быть адаптирован [8] на работу со стилусом и пальцевому набору с минимумом использования аппаратных кнопок и клавиш. На данный момент существуют ориентированные на мобильные устройства окружения рабочего стола: Phosh, Plasma Mobile, sxmo, Lomiri и другие. Однако большая часть системного и прикладного ПО в Linux создавалось для десктопов (графический интерфейс, GUI) и серверов (текстовый интерфейс, TUI). Решением здесь является создание адаптированных форков программ, либо использование специальных фреймворков, наподобие Kirigami от проекта KDE для приложений написанных на Qt.

Проблема адаптации драйверов периферийных устройств не имеет единого решения. Часть разработчиков проводит реверс-инжиниринг таких устройств, что требует не только знаний программирования, но и серьезного опыта в электронике и электротехнике, а также наличия специального оборудования, такого как осциллограф, паяльная станция, терминал отладки, программатор и т.п. Некоторые организации проводят исследования по возможности применения искусственного интеллекта в разработке программного обеспечения [2, 9], в том числе для автоматизации процесса редактирования исходных кодов системного ПО, такого как драйверы и ядра операционных систем.

Главной задачей разработчиков, переносящих Linux на конкретное ARM-устройство, является адаптация и запуск на нем актуального (mainline) ядра Linux. Использовать поставляемое с этим устройством исходное (downstream) ядро нежелательно, так как в нем присутствуют неисправленные уязвимости и специфический нестандартизированный код, из-за которого система может не запускаться или работать нестабильно. Разработчик при этом сталкивается с рядом проблем [10]:

- загрузчик может быть заблокирован, либо не способен взаимодействовать с актуальным ядром;
- разметка и размеры разделов системного диска могут не совпадать с разметкой портируемой операционной системы;
- необходимость править исходные файлы dts и dtsi «дерева устройства» (device tree), содержащего именованные узлы и свойства, описывающие оборудование, которое невозможно обнаружить путем его опроса [11];
- в случае, если устройство достаточно старое, потребуется создавать дерево устройства с нуля, перенося переменные из board-файлов, находящиеся в каталоге /arch/arm/mach-<тип SoC> исходных кодов нативного ядра данного устройства, причем часть переменных могут быть представлены в ином формате и требуют конвертации;
- часть драйверов компонентов устройства представлены в виде скомпилированных модулей ядра (формат .ko), исходные коды к ним могут быть не предоставлены, что затрудняет возможность их запуска на актуальной версии ядра;
- отладка ядра через удаленный терминал с использованием утилит dmesg, gdb и т.п., поиск контактов последовательного порта UART или JTAG на устройстве и получение с них отладочной информации.

Проблема заблокированного загрузчика решается посредством обращения пользователя к производителю устройства, чтобы тот предоставил необходимые технические средства для разблокировки, либо разблокировал загрузчик удаленно. Однако далеко не все производители разрешают производить манипуляции с загрузчиком, также наличие этой возможности зависит от производителей SoC. Отсутствие поддержки актуального ядра загрузчиком устройства решается за счет сборки из исходных кодов свободных альтернативных загрузчиков: U-Boot [12] или lk2nd [13]. Новый загрузчик помещается в раздел Boot, где находится стандартное запускаемое ядро Linux, либо в раздел Recovery, где располагаются средства по восстановлению системы с отдельным загружаемым ядром.

Разметка разделов меняется либо в Recovery перед установкой портируемой системы, либо с помощью команд загрузчика через терминал отладки.

Проблема создания дерева устройства и обновления нестандартных компонентов ядра на данный момент не имеет универсального решения. Можно предложить следующий способ, использующий машинное обучение и средства визуальной аналитики, для переноса параметров из board-файлов в файл дерева устройства:

1. В исходном коде актуального ядра Linux анализируются dts-файлы поддерживаемых систем на чипе, а также dtsti-файлы уже адаптированных устройств.

2. Осуществляется поиск оригинальных версий ядер для этих устройств. В зависимости от наличия скомпилированного бинарного файла dtb дерева устройства, либо board-файлов, производится сравнение их содержимого с актуальным деревом устройства.

3. По спецификациям исследуемого устройства проводится поиск устройств с аналогичными аппаратными компонентами, исходных кодов их ядер и извлечение частей кода, отвечающих за эти компоненты.

4. Найденные совпадения и различия помечаются, они выступают исходными данными для тренировки нейронной сети.

5. После нахождения достаточного количества исходных данных производится обучение нейронной сети.

6. Обученную нейронную сеть применяют для составления актуального дерева устройства для устройства, еще его не имеющего.

7. При необходимости производится ручная корректировка параметров в исходных кодах дерева устройства и его последующая компиляция в dtb-файл.

8. Осуществляется сборка и установка актуального ядра с внедренным актуальным dtb-файлом.

9. Через UART производится проверка запуска и первичная отладка ядра. В случае возникновения проблем производится возврат на предыдущие шаги.

Данный алгоритм планируется использовать в качестве отправной точки для создания специализированного программного средства, помогающего автоматизировать процесс адаптации и запуска актуальных версий ядра Linux на ARM-устройствах.

Список литературы

1. Шилов А. 842 Chips Per Second: 6.7 Billion Arm-Based Chips Produced in Q4 2020. URL: <https://www.tomshardware.com/news/arm-6-7-billion-chips-per-quarter> (дата обращения: 14.10.2023).

2. Дебелов В. В., Апарин В. А., Иванов В. В. Современные методы отладки электронных устройств, построенных на базе микроконтроллеров с архитектурой ARM // Алгоритмические и программные средства в информационных технологиях, радиоэлектронике и телекоммуникациях : материалы 2-й Междунар. заоч.

науч.-техн. конф. (Тольятти, 9–10 января 2014 г.). Тольятти : Поволжский государственный университет сервиса, 2014. С. 33–38. EDN: YYVTCJ

3. Официальный сайт проекта Android Open Source Project (AOSP). URL: <https://source.android.google.cn/?hl=ru> (дата обращения: 14.10.2023).

4. Официальный сайт проекта LineageOS. URL: <https://www.lineageos.org/> (дата обращения: 14.10.2023).

5. Официальный сайт проекта UBports. URL: <https://ubports.com/ru/> (дата обращения: 14.10.2023).

6. Справочник проекта PostmarketOS. URL: <https://wiki.postmarketos.org/wiki/> (дата обращения: 14.10.2023).

7. Официальный сайт проекта Hailium. URL: <https://hailium.org/> (дата обращения: 14.10.2023).

8. Бурукина И. П., Привалов А. Э. Исследование современных подходов к проектированию цифровых интерфейсов // Известия высших учебных заведений. Поволжский регион. Технические науки. 2022. № 1 (61). С. 78–87. doi: 10.21685/2072-3059-2022-1-7

9. Бевзенко С. А. Применение искусственного интеллекта и машинного обучения в разработке программного обеспечения // Инновации и инвестиции. 2023. № 8. URL: <https://cyberleninka.ru/article/n/primenenie-iskusstvennogo-intellekta-i-mashinnogo-obucheniya-v-razrabotke-programmnogo-obespecheniya> (дата обращения: 14.10.2023).

10. Руководство по портированию актуальной версии ядра Linux на мобильные устройства. URL: https://wiki.postmarketos.org/wiki/Mainlining_Guide (дата обращения: 14.10.2023).

11. Подольская Н. А. Дерево устройств. Лекция и практическое занятие. URL: <http://mech.math.msu.su/~nap/2021/SystProg/DeviceTree.pdf> (дата обращения: 14.10.2023).

12. Документация проекта свободного универсального загрузчика U-Boot. URL: <https://u-boot.readthedocs.io/en/latest/> (дата обращения: 14.10.2023).

13. Проект альтернативного загрузчика lk2nd для устройств на базе платформ Qualcomm MSM8916 MSM8226 MSM8974. URL: <https://www.opensourceagenda.com/projects/lk2nd> (дата обращения: 14.10.2023).

УДК 004.042:004.624

ПОСТАНОВКА ЭКСПЕРИМЕНТА ПО ПОДТВЕРЖДЕНИЮ ГИПОТЕЗЫ О ВРЕМЕНИ ОБРАБОТКИ РАСПРЕДЕЛЕННОЙ ФИЛЬТРАЦИИ СОБЫТИЙ

А. С. Митрохин^{1,2}

¹ЗАО «НВП Болид», Королев, Россия

***²Мытищинский филиал Московского государственного технического
университета им. Н. Э. Баумана (национального исследовательского
университета), Мытищи, Россия***

alexsak2006@mail.ru

Аннотация. Для обеспечения уведомления о событиях системы безопасности разработана настраиваемая администратором система распределенной филь-