artyomsoft 20 мар в 12:00

Как создать аппаратный эмулятор CD-ROM без паяльника

Средний 19 мин 9.2K

Блог компании RUVDS.com, Системное администрирование*, Разработка под Linux*

Несмотря на то, что постепенно оптические диски уходят в прошлое, использование ISO-образов этих дисков остаётся актуальным. Многие операционные системы поставляются в виде ISO-образов, а администраторам необходимо поддерживать разношёрстный парк старых персональных компьютеров.

Существует множество решений, как можно установить операционную систему с ISO-образа без записи его на оптический носитель. Я уже затрагивал тему ISO-образов в моих статьях: «Раскрываем секреты загрузочных ISO-образов» и «Что вам нужно знать о внешних загрузочных дисках».

В этой статье я хочу рассказать о ещё одном способе, который, как оказывается, вшит в ядро Linux. Если ваш одноплатный компьютер имеет USB OTG-разъём, и на него возможна установка Linux, то вы с большой долей вероятности можете сделать из одноплатника аппаратный эмулятор привода оптических дисков.

Меня этот способ заинтересовал. Я проверил его сам и, получив положительный результат у себя, решил поделиться с вами.

Я сам узнал много интересного, систематизировал свои знания, поэтому надеюсь, что чтение будет познавательно и интересно для вас.

Как всегда, если вы хотите посмотреть, что получится в итоге, уточнить детали, вы всегда можете найти исходный код в моём репозитории на GitHub.

При написании статьи я поставил себе следующие цели:

1. Аппаратный эмулятор CD-ROM должен быть реализован без использования паяльника и макетных плат.
2. Реализация должна быть понятна человеку, имеющему лишь базовые представления о Linux, USB и Bluetooth.

3. Решение должно быть таким, чтобы его можно было с небольшими изменениями реализовать на различных одноплатных компьютерах.
4. Побудить интерес читателя к изучению используемых в статье технологий.
5. Изложить материал, необходимый для решения задачи лаконично и просто. *Не уверен, что у меня это получилось из-за большого объёма темы. Буду признателен, если вы в отзывах напишете своё мнение.*

Оглавление

Суть решения

Решение заключается в том, что, модифицируя операционную систему Linux на одноплатном компьютере (встраиваемой системе), можно получить из него устройство, которое будет распознаваться компьютером как внешний оптический привод USB или флеш-накопитель.

В ядро Linux включена поддержка эмуляции CD-ROM и эмуляции флеш-накопителя. Но это не значит, что любую встраиваемую систему можно превратить в них. Для этого ещё необходимо, чтобы встраиваемая система имела USB OTG-контроллер или USB-контроллер периферийного устройства.

Проверка решения на практике

Я делал эмулятор оптических дисков, используя Raspberry Pi Zero 2 W. Но вы можете использовать и другие одноплатные компьютеры. Естественно, вам тогда придётся самим разбираться с некоторыми проблемами, которые с большой долей вероятностью у вас возникнут. У меня других одноплатных компьютеров кроме Raspberry Pi не было, поэтому привожу алгоритм, как делал я.

1. Скачать образ Raspberry OS Light с сайта raspberrypi.org.
2. Записать образ на SD-карту. Я использовал программу balenaEtcher.
3. Добавить строку `dtoverlay=dwc2` в файле config.txt на SD-карте.
4. Записать файлы ssh и wpa_supplicant.conf на SD-карту. Файл userconf.txt нужен, чтобы установить пароль для пользователя, ssh — чтобы включить SSH, wpa_supplicant.conf — чтобы указать точку доступа и пароль для Wi-Fi.
5. Вставить SD-карту в Raspberry Pi Zero 2 W.
6. Подключить USB-разъём Raspberry Pi Zero 2 W к USB-разъёму компьютера.
7. Подождать, пока выполнится первая загрузка и Raspberry Pi Zero 2 W подключится к Wi-Fi-сети.
8. Найти IP-адрес Raspberry Pi Zero 2 W в Wi-Fi-сети и подключиться к нему по протоколу SFTP. Я использовал приложение WinSCP.

9. Записать ISO-образы, которые вы хотите эмулировать, в файловую систему Raspberry Pi.
10. Подключиться через SSH к Raspberry Pi W 2. Это можно сделать при помощи приложения PuTTY.
11. Для того, чтобы ваш Raspberry Pi Zero 2 W стал внешним USB CD-ROM, ввести команду:

```
$ sudo modprobe g_mass_storage cdrom=y removable=y stall
```

После чего у вас на компьютере распознается внешний USB CD-ROM, в который вставлен диск.

12. Для прекращения эмуляции ввести команду:

```
$ sudo modprobe -r g_mass_storage
```

- Файл userconf.txt

Приведённый файл userconf.txt устанавливает для пользователя pi пароль «raspberry».

- Файл wpa_supplicant.conf

Файл ssh — это пустой файл, который не содержит никаких данных.

От проверки идеи до реализации

Приведённая выше последовательность шагов позволяет вам посмотреть работу эмуляции в действии. Однако это решение обладает рядом недостатков.

1. Чтобы загрузить и выбрать образ для эмуляции, необходимо наличие Wi-Fi.
2. При перезагрузке Raspberry Pi необходимо заново монтировать образ.
3. Если нужно эмулировать образ для загрузки с него операционной системы, понадобится ещё один компьютер для управления Raspberry Pi.
4. Эмулировать можно образ размером максимум 2 Gib.

Если вам интересно, как избавиться от этих недостатков, у вас есть время и интерес разобраться в этом вопросе, то предлагаю продолжить чтение.

Краткое содержание моей реализации следующее:

1. Для общения с Raspberry Pi будет использоваться Bluetooth.
2. Чтобы работа устройства была возможна в автономном режиме (без подключений Wi-Fi и Bluetooth), управляющий скрипт оформляется в виде службы Systemd.
3. Для управления по Bluetooth будет использоваться приложение Serial Bluetooth Terminal с Google Play.
4. Для эмуляции оптических дисков с образов размером больше 2Gib необходимо внести небольшие изменения в модуль ядра Linux и выполнить перекомпиляцию.

Приведу кратко, что вам нужно знать, чтобы лучше понять суть того, что мы будем делать.

Операционная система Linux

Ядро Linux

Ядро Linux содержит в себе абстракции для работы с устройствами там, где оно запускается. Реализуются эти абстракции в специальных программах, называемых драйверами. В ОС Linux драйвер может находиться непосредственно в файле ядра, а может быть оформлен в виде отдельного модуля. В большинстве случаев предпочтителен второй способ, так как модули можно динамически удалять и добавлять. Например, если устройство не подключено к системе или с ним не осуществляется работа, драйвер нам не нужен, и его можно выгрузить из памяти или не загружать вообще.

При загрузке ядра ему необходима информация об устройствах, которые присутствуют в системе, чтобы корректно загрузить драйверы (модули) для них. Эта информация может передаваться из различных источников. Например, в архитектуре x86 это будет ACPI. В архитектуре ARM это Device Tree.

Device Tree и Overlays

Иногда Device Tree нужно модифицировать, чтобы можно было загрузить корректно драйвера для устройств. Делается это при помощи подключения overlays. Они содержат информацию, что необходимо изменить в исходном Device Tree.

Headless-режим работы Raspberry Pi

Очень часто в различных статьях и самоучителях по работе с Raspberry Pi необходимо подключение монитора, клавиатуры и мыши. Но на самом деле есть возможность работать с ним в так называемом headless-режиме. В этом режиме вы работаете с Raspberry Pi при помощи эмулятора терминала. Соединение его с Raspberry Pi может быть UART, USB, Bluetooth, Ethernet, Wi-Fi.

Главная сложность заключается в том, как можно работать в этом режиме с самого начала, сразу после записи образа операционной системы на SD-карту, если у вас нет лишнего монитора и клавиатуры. Как активировать SSH, настроить Wi-Fi на

использование определённой точки доступа?

В Raspberry Pi OS такая возможность есть. Достаточно разместить определённые файлы в разделе FAT32 на SD-карте и загрузиться с неё. Raspberry Pi OS сделает необходимые настройки сама.

Файловые системы, блочные устройства, разделы, монтирование

Меня всегда восхищала идея, что в Linux всё является файлом. Правильно используя средства Linux, можно практически без программирования выполнять сложные задачи.

Блочное устройство — это некоторый файл, в который можно записывать и считывать данные блоками байтов различной длины. Что будет происходить при этом, зависит от того, с чем реально ассоциирован этот файл. Например, если он ассоциирован с жёстким диском, то тогда будут читаться/записываться данные на жёсткий диск, не обращая внимание на разделы и файловые системы. Если он ассоциирован с разделом жёсткого диска, то будут читаться/записываться данные, не обращая внимание на файловую систему.

При помощи команды losetup можно добиться того, что он будет ассоциироваться с обычным файлом на диске, что позволит создавать образы разделов и дисков.

Ещё полезной командой Linux является команда kpartx, которая создаёт блочные устройства из файла образа диска. Каждое из устройств будет ассоциировано с образом раздела, который хранится в этом файле.

Форматирование раздела в Linux выполняется одной командой. В качестве параметра необходимо передать имя файла блочного устройства. Например, для создания файловой системы exFAT на блочном устройстве /dev/mmcblk0p3:

```
$ mkfs.exfat /dev/mmcblk0p3
```

Чтобы можно было работать с файлами файловой системы, размещённой на блочном устройстве, нужно примонтировать файловую систему к корневой файловой системе при помощи команды mount.

```
$ mkdir -p /mnt/data
$ mount -t auto /dev/mmcblk0p3 /mnt/data
```

Обратите внимание, что директория, куда будет производиться монтирование, должна существовать до того, как вы будете монтировать. Если нужно размонтировать файловую систему, используется команда umount.

```
$umount /mnt/data
```

Чтобы посмотреть, какие у вас есть блочные устройства, и куда они примонтированы, можно использовать команду:

```
$ lsblk
```

Systemd

Загрузка Linux происходит в несколько этапов. Сначала загружается ядро операционной системы, затем ядро запускает процесс init. Задача процесса init загрузить и инициализировать процессы пространства пользователя и находиться в памяти до перезагрузки или выключения компьютера (устройства). За долгие годы существования Linux было написано множество реализаций init. На данный момент во многих Linux-дистрибутивах используется реализация, называемая systemd. Её мы и будем использовать.

Минимум команд, которые необходимо знать для работы с systemd.

Команда	Назначение
systemctl start sn.service	запустить службу
systemctl stop sn.service	остановить службу
systemctl status sn.service	посмотреть статус службы
systemctl enable sn.service	включить службу (служба будет автоматически запущена при следующей загрузке Linux)
systemctl disable sn.service	выключить службу
journalctl -u sn.service -b	посмотреть логи службы, начиная с момента последней загрузки Linux

Терминалы, PuTTY, sshd, agetty

Для администрирования ОС Linux из Windows часто используют эмулятор терминала PuTTY. Он позволяет подключаться к компьютеру или устройству с ОС Linux с помощью различных соединений (Ethernet, Wi-Fi, эмулируемого последовательного порта на Bluetooth или USB) и работать удалённо с консолью Linux в Windows. Чтобы такое было возможно, в ОС Linux должна быть запущена специальная программа, которая будет взаимодействовать с PuTTY. Это может быть sshd в случае SSH-соединения или agetty в случае последовательного порта.

При подключении через последовательный порт по умолчанию вы увидите чёрно-белый экран без поддержки манипулятора мышь. Чтобы добавить поддержку мыши и цветного экрана, необходимо

изменить значение переменной окружения TERM в файле /usr/lib/systemd/system/serial-getty@.service.

```
[Service]
Environment=TERM=xterm
```

USB

Чтобы два USB-устройства могли работать друг с другом, необходимо наличие у каждого из них USB-контроллера. USB-контроллер в конкретный момент времени может работать в режиме хоста (Host) или режиме периферийного устройства (Device). Если одно из взаимодействующих устройств работает в режиме Host, то другое должно работать в режиме Device. Существуют следующие виды USB-контроллеров:

- Host — всегда работает в режиме Host.
- Device — всегда работает в режиме Device.
- OTG — может работать или в режиме хоста или в режиме периферийного устройства. Переключение режимов может быть аппаратным (при помощи особой распайки кабеля OTG кабель переводит в контроллер режим хоста) или программным

Режим хоста подразумевает посылку команд, а режим периферийного устройства — их обработку.

OTG USB-контроллер

Возьмём Android-телефон с OTG-контроллером. Это означает, что при сопряжении по USB с компьютером (для записи файлов с компьютера на телефон), он будет играть роль периферийного устройства, а при сопряжении по USB с периферийным устройством (мышью, клавиатурой, сетевой картой, флэш-накопителем, монитором, принтером) телефон будет играть роль хоста.

Обычно USB-контроллер периферийного устройства или USB OTG-контроллер присутствуют во встраиваемых устройствах. Также они могут быть интегрированы в однокристальную систему (SoC). Но по факту на устройстве может отсутствовать физический USB-разъём для подключения.

Например, на всех Raspberry Pi установлена SoC, которая имеет OTG-контроллер, но фактически физический разъём для него есть только в Raspberry Pi Zero (Zero W, Zero 2 W) и в Raspberry Pi 4.

Дескрипторы USB

Каждое USB-устройство имеет дескрипторы. Дескрипторы — это информация о USB-устройстве, которая используется операционной системой для корректного выбора драйвера для устройства. Мне понравилось описание, которое приведено на сайте Microsoft.

Создание USB-устройств в Linux

Ядро Linux содержит модули, которые позволяют создавать виртуальные USB-устройства. Это может быть Mass Storage, последовательный порт, сетевая карта. Загрузив и настроив эти модули, вы можете сделать так, чтобы компьютером ваш одноплатник распознавался одним или несколькими такими устройствами.

Если вам достаточно одного устройства, то вы можете загрузить модуль для этого устройства, опционально передав ему параметры для конфигурации при помощи команды modprobe. Когда отпадёт необходимость в этом устройстве, его можно выгрузить при помощи команды modprobe -r.

Чтобы на одном физическом порту у вас распознавалось несколько устройств одновременно, нужно использовать модуль libcomposite и сконфигурировать эти устройства при помощи создания структур в файловой системе ConfigFS в директории

/sys/kernel/config/usb_gadget.

Такие устройства называются композитными USB-устройствами. Вы, скорее всего, встречались с такими, например, если у вас беспроводная клавиатура и мышь, а для них используется один приёмопередатчик.

В нашем случае мы создадим композитное USB-устройство, которое будет последовательным портом и устройством хранения. Последовательный порт мы будем использовать для подключения к нашему эмулятору оптических дисков через PuTTY. Изначально я хотел, что бы это была сетевая карта и SSH, но карта требует настройки в операционной системе компьютера, поэтому для простоты отказался от этой идеи в пользу последовательного порта.

Создание композитного USB-устройства при помощи ConfigFS

1. Загружаем модуль libcomposite.

```
modprobe libcomposite
```

2. Заполняем дескрипторы для устройства.

```
$ usb_dev=/sys/kernel/config/usb_gadget/cdemu

$ mkdir -p $usb_dev

$ echo 0x0137 > $usb_dev/idProduct
$ echo 0x0100 > $usb_dev/bcdDevice
$ echo 0x0200 > $usb_dev/bcdUSB

$ echo 0xEF  > $usb_dev/bDeviceClass
$ echo 0x02 > $usb_dev/bDeviceSubClass
$ echo 0x01 > $usb_dev/bDeviceProtocol
```

```
$ mkdir -p $usb_dev/strings/0x409

$ echo "abababababababa" > $usb_dev/strings/0x409/serialnumb
$ echo "Linux Foundation" > $usb_dev/strings/0x409/manufactu
$ echo "USB CD-ROM Emulator" > $usb_dev/strings/0x409/produc
```

3. Создаём конфигурацию.

```
mkdir -p $usb_dev/configs/c.1
mkdir -p $usb_dev/configs/c.1/strings/0x409
echo "acm+usb" > $usb_dev/configs/c.1/strings/0x409/configur
echo "0x80" > $usb_dev/configs/c.1/bmAttributes
echo 250 > $usb_dev/configs/c.1/MaxPower
```

4.Создаём и подключаем функцию acm (последовательный порт через USB).

```
$ mkdir -p $usb_dev/functions/acm.usb0
$ ln -s $usb_dev/functions/acm.usb0 $usb_dev/configs/c.1
```

5. Создаём и подключаем функцию mass_storage. Mass_storage в данном случае — это эмуляция CD-ROM для ISO-образа /home/pi/1.iso.

```
$ mkdir -p $usb_dev/functions/mass_storage.usb0/lun.0
$ echo 1 > $usb_dev/functions/mass_storage.usb0/lun.0/cdrom
$ echo 1 > $usb_dev/functions/mass_storage.usb0/lun.0/remova
$ echo 0 > $usb_dev/functions/mass_storage.usb0/lun.0/nofua
$ echo 0 > $usb_dev/functions/mass_storage.usb0/stall
$ echo "/home/pi/1.iso" > $usb_dev/functions/mass_storage.us
$ ln -s $usb_dev/functions/mass_storage.usb0 $usb_dev/config
```

6. Активируем созданное устройство.

```
$ ls /sys/class/udc > $usb_dev/UDC
```

Удаление композитного USB-устройства при помощи ConfigFS

1. Деактивируем устройство.

```
$ usb_dev=/sys/kernel/config/usb_gadget/cdemu
$ echo ""> $usb_dev/UDC
```

2. Удаляем функцию mass_storage.

```
$ rm $usb_dev/configs/c.1/mass_storage.usb0
$ rmdir $usb_dev/functions/mass_storage.usb0
```

3. Удаляем функцию acm.

```
$ rm $usb_dev/configs/c.1/acm.usb0
$ rmdir $usb_dev/functions/acm.usb0
```

4. Удаляем конфигурацию.

```
$ rmdir $usb_dev/configs/c.1/strings/0x409
$ rmdir $usb_dev/configs/c.1
```

5. Удаляем устройство.

```
$ rmdir $usb_dev/strings/0x409
$ rmdir $usb_dev
```

6. Выгружаем загруженные устройством модули.

```
$ modprobe -r usb_f_mass_storage
$ modprobe -r usb_f_acm
$ modprobe -r libcomposite
```

```
pi@raspberrypi:/sys/kernel/config/usb_gadget/cdemu $ tree
.
├── bcdDevice
├── bcdUSB
├── bDeviceClass
├── bDeviceProtocol
├── bDeviceSubClass
├── bMaxPacketSize0
├── configs
│   └── c.1
│       ├── acm.usb0 -> ../../../../usb_gadget/cdemu/functions/acm.usb0
│       ├── bmAttributes
│       ├── mass_storage.usb0 -> ../../../../usb_gadget/cdemu/functions/mass_storage.usb0
│       ├── MaxPower
│       └── strings
│           └── 0x409
│               └── configuration
├── functions
│   ├── acm.usb0
│   │   └── port_num
│   └── mass_storage.usb0
│       ├── lun.0
│       │   ├── cdrom
│       │   ├── file
│       │   ├── inquiry_string
│       │   ├── nofua
│       │   ├── removable
│       │   └── ro
│       └── stall
├── idProduct
├── idVendor
├── max_speed
├── os_desc
│   ├── b_vendor_code
│   ├── qw_sign
│   └── use
├── strings
│   └── 0x409
│       ├── manufacturer
│       ├── product
│       └── serialnumber
└── UDC
```

Структура файловой системы для созданного эмулятора CD-ROM

Bluetooth

Тема Bluetooth очень объёмная, и её невозможно изложить в одной статье, поэтому приведу только тот минимум, который позволяет понять, как мы будем использовать Bluetooth.

Bluetooth — технология, которая позволяет связывать устройства без проводов по радиоканалу. На данный момент существует множество версий спецификации Bluetooth. Спецификация Bluetooth освещает

множество вопросов.

Чтобы передать данные с одного устройства на другое, необходимо наличие на обоих устройствах контроллеров и стеков Bluetooth.

Bluetooth-контроллер — аппаратное устройство, обычно выполненное в виде микросхемы или части более сложной микросхемы, позволяющее получать/передавать данные по радиоканалу в соответствии со спецификацией Bluetooth.

Bluetooth-стек — программная реализация протоколов, описанных в спецификации Bluetooth.

Протоколы Bluetooth, предназначенные для решения определённых задач, группируются в профили Bluetooth.

Мы будем использовать два профиля Bluetooth:

1. Generic Access Profile (GAP), который поддерживается всеми Bluetooth-устройствами.
2. Serial Port Profile (SPP), который подразумевает использование последовательного порта поверх соединения Bluetooth.

Поддержка Bluetooth операционными системами

Bluetooth-контроллеры могут иметь различные аппаратные интерфейсы для доступа. Это может быть UART, USB, PCIe. В случае операционной системы многие детали скрываются, и можно о них не думать. С контроллером можно работать на низком уровне через драйвер или уже используя высокоуровневые библиотеки и приложения, предоставляемые стеком Bluetooth, например, в Linux широко распространён стек BlueZ.

BlueZ

Стек Bluetooth BlueZ состоит из двух частей.

Одна часть представлена модулями ядра Linux, уже включена в ядро. Если она отсутствует, то её нужно включить и перекомпилировать ядро.

Вторая часть представлена приложениями для пространства пользователя. Приложения позволяют конфигурировать и работать со стеком Bluetooth.

На данный момент многие приложения считаются устаревшими, и разработчики BlueZ рекомендуют использовать более новые приложения и интерфейс D-Bus для работы со стеком.

Но, как мне кажется, именно те старые, устаревшие приложения позволяют лучше понять работу Bluetooth, поэтому в учебных целях я буду использовать их, для чего нужно будет инициализировать BlueZ в режиме совместимости.

```
$ bluetoothd --noplugin=sap -C
```

Протоколы Bluetooth

Я не буду утомлять вас различными схемами, диаграммами, которые вы легко можете найти в интернете. Расскажу только о тех протоколах, с которыми нам предстоит столкнуться и нужно будет сконфигурировать.

Service Discovery Protocol (SDP)

При помощи протокола SDP можно определить, какие приложения (сервисы) находятся на хосте, и с ними возможен обмен данными через Bluetooth

Чтобы можно было увидеть сервис с другого устройства, его

необходимо зарегистрировать в SDP database. Например, если мы хотим зарегистрировать службу, представляющую эмуляцию последовательного порта в Bluetooth, это можно сделать следующей командой:

```
$ sdptool add SP
```

Чтобы можно было посмотреть службы, зарегистрированные у вас на хосте, нужно ввести команду:

```
$ sdptool browse local
```

Radio Frequency Communications (RFCOMM)

Протокол RFCOMM позволяет создавать виртуальное соединение по последовательному порту между двумя хостами.

На одном из хостов создаётся сервер, которому выделяется канал RFCOMM, второй из хостов подключается к нему, указывая MAC-адрес и номер канала

Канал RFCOMM немного напоминает порт в UDP или TCP, но если у них и у источника и у получателя есть порты, то у RFCOMM для источника и получателя один и тот же канал. Поэтому невозможно создать несколько подключений на один и тот же канал.

В Linux можно использовать команду rfcomm для создания процесса, который будет слушать определённый канал RFCOMM и при соединении запускать другой процесс.

```
$ rfcomm -r watch hci0 1 /usr/local/bin/cdemu-cmd /dev/rfcor
```

В данном случае на Bluetooth-контроллере hci0 RFCOMM будет

прослушиваться канал 1 и запускаться процесс cdemu-cmd с двумя параметрами командной строки /dev/rfcomm0 и /dev/rfcomm0.

Утилита bluetoothctl

Утилита Bluetoothctl позволяет сопрягать устройство, на котором вы её запустили с другим устройством.

Вы можете сделать устройство видимым для обнаружения другими устройствами, а также найти другое устройство и выполнить с ним сопряжение. Более подробно расписано в документации к утилите, которая доступна по команде:

```
$ man bluetoothctl
```

Serial Bluetooth Terminal

Для отладки приложений, использующих Bluetooth, удобно использовать приложение для Android Serial Bluetooth Terminal. Это приложение позволяет работать с Bluetooth-устройствами, у которых доступен профиль SPP. В нашем случае мы будем использовать его как визуальный интерфейс для работы с нашим эмулятором оптических дисков.

Сборка и модификация дистрибутива Raspberry Pi OS

Чтобы сделать полноценный аппаратный эмулятор оптических дисков, нам придётся немного модифицировать исходный дистрибутив Linux. Это подразумевает перекомпиляцию ядра, изменение нескольких конфигурационных файлов и добавление своего программного кода. Для меня это было удобно сделать при помощи Docker.

Кросс-компиляция ядра Linux

Кросс-компиляция позволяет на компьютере с одной архитектурой

получать исполняемые файлы для другой архитектуры. Мы можем компилировать ядро Linux для Raspberry Pi на Raspberry Pi, а можем, используя кросс-компиляцию, сделать это на обычном компьютере с архитектурой x86, что существенно сократит время компиляции из-за большего быстродействия компьютера. Подробно о том, как выполнять кросс-компиляцию Raspberry Pi OS, можно почитать тут.

Chroot и запуск бинарных файлов другой архитектуры

Команда Linux chroot позволяет запускать процессы с изменённой корневой системой. Это кажется немного запутанным, но суть в следующем. В качестве параметра команде передаётся путь к корневой директории. В результате запуска команды через chroot запущенный процесс будет считать, что корнем файловой системы является та директория, которую передали в качестве параметра.

Применений у команды chroot несколько, например, её можно использовать, чтобы запустить команду apt для Raspberry Pi в Docker-контейнере.

Интересно, что Docker Desktop для Windows позволяет запускать исполняемые файлы для архитектуры ARM. В Linux-версии Docker такое сделать можно, но нужна дополнительная настройка.

Реализация

Созданный мной проект состоит из следующих файлов:

1. Dockerfile и скрипт, который выполняется в Docker-контейнере.
2. Файлы, которые необходимо добавить или обновить в исходном дистрибутиве:

    - cdemu — основная логика работы эмулятора оптических дисков, написанная на языке bash;

- cdemu-cmd — bash-скрипт для обработки команд от пользователя и передачи их эмулятору;
- bash-utils.sh — bash-скрипт со вспомогательными функциями;
- cdemu-bluetooth-ui.service — systemd-служба, которая запускает интерпретатор команд на создаваемом RFCOMM-соединении телефона и Raspberry Pi;
- cdemu.service — systemd-служба, которая запускает эмулятор оптических дисков при загрузке;
- bluetooth.service — изменённая служба systemd для инициализации bluetooth;
- serial-getty@.service — изменённая служба systemd для запуска agetty на создаваемом соединении на последовательном порту;
- firstboot.service — служба systemd для запуска скрипта при первой загрузке операционной системы. Я её позаимствовал из проекта raspberian-firstboot;
- config.txt — изменённый файл конфигурации для загрузки Raspberry Pi. Содержит подключение overlay dwc. Это необходимо, чтобы USB-контроллер мог работать в device mode;
- fistboot.sh — скрипт, который запускается службой systemd firstboot.service;
- userconf.txt — файл, который необходим, чтобы установить пароль для пользователя pi. В последних версиях Raspberry Pi OS пользователь pi не активирован по умолчанию, поэтому необходимо наличие этого файла;
- ssh — файл необходим, чтобы активировать ssh, который отключён по умолчанию;
- wpa_supplicant.conf — файл, необходимый, если вы хотите настроить Raspberry Pi на работу с вашей точкой доступа.

Листинги файлов не привожу, так как это ещё больше раздует и так большую статью.

Ознакомиться вы с ними можете здесь.

Как пользоваться эмулятором

1. Собираем Docker-образ.

```
docker build -t raspi-image .
```

2. Собираем образ RaspberryPi OS.

```
docker run --privileged -v c:\temp:/build --name raspi-image
```

3. Записываем образ на SD-карту. Вставляем её в Raspberry Pi.
4. Подключаем Rapsberry Pi Zero 2 W к компьютеру.
5. Через некоторое время у вас появится съёмный накопитель.
6. На этот съёмный накопитель, содержащий файл Readme.txt, копируем образы, которые хотим эмулировать.
7. Находим виртуальный COM-порт, созданный после подключения Raspberry Pi к компьютеру.
8. Подключаемся к Raspberry Pi через с помощью Putty через виртуальный COM-порт.
9. Запускаем интерактивное приложение для управления эмулятором.

```
$ sudo cdemu-cmd
```

10. Если хотим сделать управление с телефона, то выполняем сопряжение телефона и Raspberry Pi. Для чего вводим в эмуляторе терминала команду:

```
$ sudo bluetoothctl
```

11. Делаем Raspberry Pi доступным для обнаружения:

```
discoverable on
```

12. Находим его на телефоне и выполняем полключение. После чего соглашаемся с PIN-кодом на телефоне и Raspberry Pi.

```
yes
```

13. Выходим из bluetoothctl.

```
exit
```

14. Запускаем на телефоне Serial Bluetooth Terminal и выполняем подключение к Raspberry Pi из него. Теперь можно посылать команды созданному эмулятору CD-ROM.

Команды, которые можно посылать эмулятору:

1. hdd — переключение в режим эмуляции внешних жёстких дисков.
2. cdrom — переключение в режим эмуляции внешних приводов оптических дисков.
3. list — вывести список доступных ISO-образов, которые можно эмулировать.
4. insert <порядковый номер> — поместить ISO-образ для эмуляции.
5. eject — извлечь ISO-образ из эмулятора.
6. help — показать список доступных команд в текущем режиме.

Особенности моего эмулятора

Интересно, но в Linux по умолчанию нельзя эмулировать ISO-образы размером больше 2 Gib. Я просмотрел исходный код драйвера в файле drivers/usb/gadget/function/storage_common.c и предположил, что нет оснований не применять патч к ядру Linux от Adam Bambuch, который просто удаляет одно условие. Образы эмулировались нормально и при снятии ISO-образа с эмулируемого CD-ROM он был идентичен исходному. Поверял по хеш-коду для файла ISO-образа.

Не пойму, почему есть это ограничение в Linux и почему его до сих пор не убрали? Если вы знаете ответ, ответьте в комментариях.

Моя реализация не требует никаких дополнительных деталей. Нужен только Raspberry Pi Zero 2 W, один или два кабеля USB и адаптер питания, если будете использовать два кабеля USB. Один для питания, второй для передачи данных. Хоть и использование дополнительного кабеля и адаптера добавляет громоздкости, это решает проблему перезагрузки Raspberry Pi, если компьютер или ноутбук отключает ненадолго питание при перезагрузке.

Кроме того, я не использую Python, только bash.

Выводы

Полученное программно-аппаратное решение, хоть и обладает рядом недостатков по сравнению с карманом Zalman (не поддерживается USB 3.0, нет интерактивного меню на самом устройстве), позволит вам установить практически любую операционную систему на широкий спектр компьютеров путём простого копирования ISO-образа.

Решение является прототипом, но вместе с тем позволяет углубить знания по многим темам, или получить, если вы были с ними не знакомы.

Так как основной целью была разработка прототипа, я запускал bluetoothd в режиме совместимости, и почти всю логику написал на

bash.

Я хотел показать возможность превратить встраиваемое устройство с операционной системой Linux в аппаратный эмулятор флеш-накопителя или привода оптических дисков, приложив минимум усилий. Надеюсь, что это удалось.

Чтобы уместить всё в одной статье, я лишь поверхностно коснулся тех тем, которые необходимы для понимания. Если вас заинтересовало, вы можете самостоятельно изучить их углублённо.

Объём статьи не позволяет осветить все интересности, с которыми я столкнулся при разработке эмулятора, и решения, которые применял и проверял. Приведу лишь несколько из них.

Например, я долго боролся с зависанием при удалении составного устройства. Помогло использование службы serial-getty вместо getty, хотя во многих статьях упоминалась getty.

Я долго разбирался, как можно сделать сопряжение через Bluetooth между Raspberry и телефоном, использовал команду bt-agent, но потом всё-таки отказался от неё в пользу bluetoothctl.

При переключении эмулятора в режим HDD для записи ISO-образов изначально я открывал для доступа всю SD-карту и хранил ISO-образы в отдельном разделе, но потом посчитал, что для безопасности лучше хранить образ диска с ISO-образами в отдельном файле и открывать доступ только к нему, хоть это и снизило скорость записи, но пользователя не обескураживают появляющиеся несколько дисков.

Разработанный прототип есть куда улучшать. Можно, например, создать более минималистичный дистрибутив Linux, который будет содержать только то, что реально используется для эмуляции, или создать более удобное графическое приложение для Android для работы с эмулятором. А можно упростить работу с Bluetooth, напрямую работая с драйверами bluetooth или используя интерфейс

D-Bus для работы с Bluetooth-устройствами. Или вообще всё-таки взяться за паяльник и сделать устройство, более похожее по функционалу на карман Zalman. Но главное, вы увидели, что это реально сделать, а когда видишь положительный результат, это вдохновляет на большее творчество.

В процессе тестирования и отладки программного кода было замечено, что на Lenovo X1 Extreme Gen 2 эмулятор CD-ROM дисков великолепно определялся в Windows 10, но отказывался определяться в BIOS. Эмпирически было определено, что помогает отключение режима экономии энергии процессора в BIOS. Также ноутбук отключал питание на usb при перезагрузке, поэтому понадобилось дополнительное питание Raspberry Pi.

Интересно, но на ASUS K53E и Gigabyte BRIX всё работает без проблем.

Решение с небольшими модификациями можно реализовать на Raspberry Pi 4. Но если вы поняли суть решения, вы его сможете повторить и на других одноплатных компьютерах, которые имеют выведенные USB-порты для OTG или USB-контроллеры периферийных устройств.

Dockerfile на данный момент только выполняется в Docker Desktop для Windows. В Linux он работать не будет.

В заключение хочу сказать, что существует ещё один способ эмулировать оптические диски, который я не пробовал, но знаю о его существовании из ваших комментариев к одной из моих статей — это программа DriveDroid для Android. Я ей не пользовался, так как для её работы нужно получать права root на телефоне. Но, скорее всего, из-за ограничений в ядре Linux программа поддерживает ISO-образы до 2 Gib и/или работает только с гибридными ISO-образами. Если я не прав, буду рад увидеть ваши опровержения в комментариях.

> **Telegram-канал с розыгрышами призов, новостями IT и постами о ретроиграх**

Теги: linux kernel, usb, bluetooth, bluez, linux modules, эмуляция, iso, cd-rom, systemd, agetty, ruvds_статьи

Хабы: Блог компании RUVDS.com, Системное администрирование, Разработка под Linux, Разработка на Raspberry Pi, DIY или Сделай сам

RUVDS.com

VDS/VPS-хостинг. Скидка 15% по коду **HABR15**

Telegram ВКонтакте Twitter

84 82.1
КармаРейтинг

@artyomsoft
Пользователь

Комментарии 17

Публикации

ЛУЧШИЕ ЗА СУТКИ ПОХОЖИЕ

ИНФОРМАЦИЯ

Сайт	ruvds.com
Дата регистрации	18 марта 2016
Дата основания	27 июля 2015
Численность	11–30 человек
Местоположение	Россия
Представитель	ruvds