

Criando uma imagem Linux com Buildroot para o Raspberry-Pi

Rafael Guterres Jeffman

2019

There is no spoon.

Você sabe o que é uma
distribuição Linux?

Distribuições Linux

- ▶ Filosofia
- ▶ Gerenciamento
- ▶ Pacotes
- ▶ Qual o problema que a sua distribuição resolve?



Back to the future...

Gobolinux



- ▶ Desenvolvida em 2003 por alunos da Unisinos.
- ▶ Modificava a árvore de diretórios do Linux.
- ▶ Criada a partir do Linux From Scratch.
- ▶ Foi a primeira vez que eu vi que isso era legal...

Me respeita que antes isso
aqui era tudo mato...

A gente tentava compilar e
saia dizendo
"EU MATO! EU MATO!"...

Por que criar uma imagem Linux?

- ▶ Porque você aprende sobre como o sistema funciona de verdade.
- ▶ Porque você identifica onde pode melhorar o sistema que você usa.
- ▶ Porque, depois disso, você pode fazer o que quiser quiser...

"O tempo passa e um dia bate a porta um senhor de alta classe com dinheiro na mão..."

Faroeste Caboclo, Legião Urbana

Uso de imagens Linux

- ▶ Instalação de software
- ▶ Testes de diagnóstico
- ▶ Máquinas virtuais *mínimas* para aplicações.
- ▶ Criar um firmware baseado em Linux.

Como inicia o Linux?

- ▶ Um bootloader carrega o kernel em memória e o inicializa.
- ▶ Kernel inicializa o hardware e os subsistemas
- ▶ Kernel dispara o `/sbin/init`
- ▶ `/sbin/init` inicializa as outras aplicações em userspace.

Bootloaders

- ▶ lilo
- ▶ elilo (IPF)
- ▶ yaboot (OpenFirmware PowerPC)
- ▶ grub (x86)
- ▶ uBoot (ARM, MIPS, m68k...)
- ▶ Barebox (ARM, x86, MIPS, PowerPC)

Kernel do Linux

- ▶ Suporta mais de 30 arquiteturas diferentes
- ▶ Originalmente feito para arquiteturas com MMU.
- ▶ Gerencia recursos de hardware e processos.

Quanto tempo ainda temos?

Toolchain

- ▶ Para compilar o kernel e os programas é necessário ter um *toolchain* de compilação.
- ▶ Para compilar para uma arquitetura diferente da de desenvolvimento, precisamos de compilação cruzada.
- ▶ O mínimo que precisamos é de um compilador e uma biblioteca padrão C. Um é o ovo, o outro, a galinha.

Criando o Toolchain

- ▶ Croostool-ng
- ▶ Buildroot
- ▶ OpenEmbedded
- ▶ Yocto Project

Componentes Mínimos de uma imagem

- ▶ Um kernel.
- ▶ As bibliotecas do sistema (libC, por exemplo)
- ▶ Um sistema de arquivos, o *rootfs*.
- ▶ Um programa para executar (o `/sbin/init`).
- ▶ As ferramentas e bibliotecas necessárias para executar o programa...

Busybox

- ▶ **BusyBox** é uma aplicação que traz diversas aplicações *core*, com um *footprint* mínimo.
- ▶ Todo mundo usa.
- ▶ Todo mundo deveria usar.

Buildroot

- ▶ Ferramenta que automatiza o processo de compilar um sistema Linux embarcado, utilizando compilação cruzada.
- ▶ Cria todo o *toolchain* necessário para a compilação, isolando o processo da instalação do usuário.

Filosofia do Buildroot

- ▶ Foco na simplicidade.
- ▶ Minimalista.
- ▶ Baseado em tecnologias existentes (make, kconfig, etc).
- ▶ Comunidade aberta.

A saída do Buildroot

- ▶ Um toolchain para compilação cruzada.
- ▶ Um kernel Linux para a plataforma escolhida.
- ▶ Bootloaders, Busybox e ferramentas.
- ▶ Uma imagem de sistema de arquivos **root** (*root filesystem image*)

Instalando o Buildroot

- ▶ `wget`
`https://buildroot.org/downloads/buildroot-2019.02.2.tar.gz`
- ▶ `tar xzvf buildroot-2019.02.2.tar.gz`
- ▶ `cd buildroot-2019.02.2`
- ▶ `make menuconfig`

2

Buildroot 2019.02.2 Configuration

Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> selects a feature, while <N> excludes a feature. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] feature is selected

Target options --->

Build options --->

Toolchain --->

System configuration --->

Kernel --->

Target packages --->

Filesystem images --->

Bootloaders --->

Host utilities --->

Legacy config options --->

<Select>

< Exit >

< Help >

< Save >

< Load >

Configurando o Buildroot para o Raspberry

- ▶ Arquitetura: ARM (little endian)
- ▶ Variante: arm1176jzf-s
- ▶ ABI: EABI
- ▶ Float-point: VFPv2
- ▶ Instruction Set: ARM

Sempre pode ser mais fácil...

- ▶ Configurações e documentação em `boards/raspberry`
- ▶ `make raspberrypi_defconfig`
- ▶ `make` (and waaaaait...)
- ▶ i3 2.5GHz, 13Gb RAM, HDD: 45 minutos + 52 minutos

Durante o *build*...

- ▶ É necessário conexão com a Internet, pois o buildroot baixa os fontes necessários.
- ▶ O *build* (`make`) não deve ser executado em paralelo (`-jX`).
- ▶ É necessário um bom espaço em disco.

O que o *Buildroot* cria?

- ▶ **build**: o resultado da compilação de todos os pacotes.
- ▶ **target**: o *root file system*, sem dispositivos, que será utilizado na imagem.
- ▶ **host**: *toolchain* para o cross-compile.
- ▶ **images**: kernel, imagens, as coisas que a gente grava no dispositivo.

Configurando a imagem.

- ▶ Crie um diretório com os arquivos a serem instalados.
- ▶ No *menuconfig* configure a opção *System Configuration -> Root Filesystem overlay directories*
- ▶ Todos os arquivos nesse diretório serão copiados para a imagem.
- ▶ Recompile, regrave, reteste.

And it works!

Referências

- ▶ The Buildroot User Manual
- ▶ <https://elinux.org/images/2/2a/Using-buildroot-real-project.pdf>
- ▶ https://events.static.linuxfound.org/sites/events/files/slides/belloni-petazzoni-buildroot-oe_0.pdf
- ▶ <https://cellux.github.io/articles/diy-linux-with-buildroot-part-1/>
- ▶ https://www.inf.pucrs.br/emoreno/undergraduate/CC/progperif/sem15.1/trabalhos/TP1/tutorial_buildroot.pdf

Muito Obrigado!

`mailto:rafasgj@gmail.com`

`https://slides.tchelinux.org`