Ensino & Informação
Prof.: Altamir A. R. Araldi
Revista®

Anatomia de um Sistema Linux embarcado

O artigo **Anatomia de um Sistema Linux embarcado** foi publicado originalmente no *site* **Embarcados** e é de autoria de **Diego Sueiro.** É um ótimo artigo introdutório sobre esse assunto.

==========

Anatomia de um Sistema Linux embarcado

Um sistema Linux embarcado não se difere, no quesito sistêmico, de um sistema Linux *desktop*. A mesma estrutura e conceitos são aplicados em ambos os domínios. A principal diferença está nos requisitos de processamento, armazenamento, consumo de energia e confiabilidade. Na maioria dos sistemas Linux embarcado, os recursos disponíveis são limitados e muitas vezes a interface com usuário é bastante limitada ou simplesmente não existe.

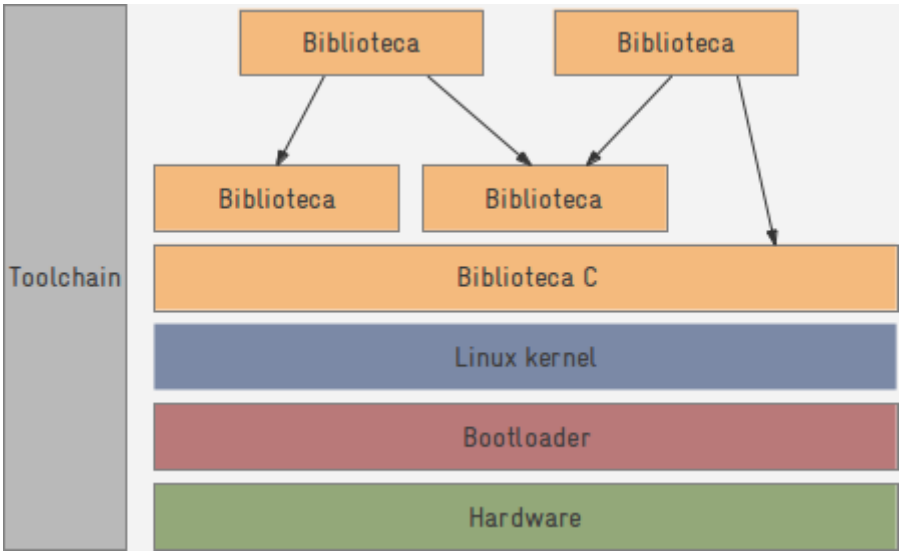
Seguem alguns exemplos de sistemas embarcados que utilizam o Linux como sistema operacional:

- Roteador;
- Setup-box;
- Smart TV;
- Controlador Lógico Programável (CLP);
- Câmera Digital.

.

Arquitetura Básica

Na arquitetura básica de um sistema Linux podemos identificar cinco componentes básicos:

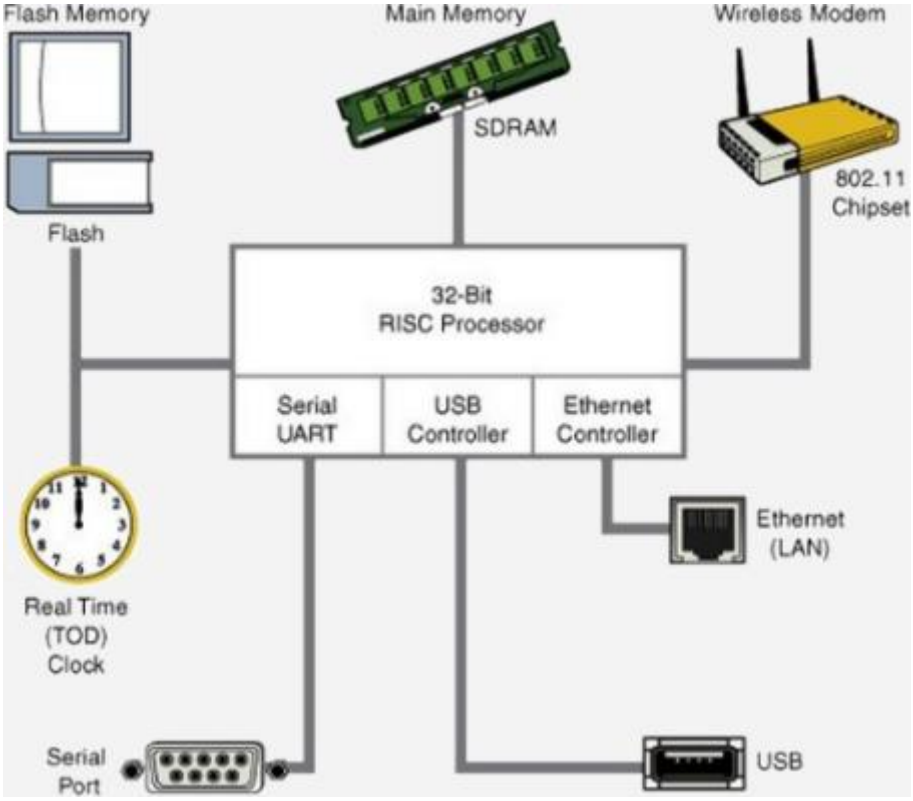


1. ***Hardware***: o seu produto;
2. ***Bootloader***: iniciado pelo *hardware*, responsável pela inicialização básica, carregamento e execução do *kernel* Linux;
3. ***Kernel* Linux**: núcleo do sistema operacional. Gerencia CPU, memória e I/O, exportando serviços para as aplicações do usuário;
4. **Rootfs**: sistema de arquivos principal. Possui as bibliotecas do sistema para uso dos serviços exportados pelo *kernel*, além das bibliotecas e aplicações do usuário;
5. ***Toolchain***: conjunto de ferramentas para gerar os artefatos de *software*do sistema.

.

Hardware

Vamos considerar um roteador Wi-Fi doméstico como exemplo de plataforma de*hardware*:



.

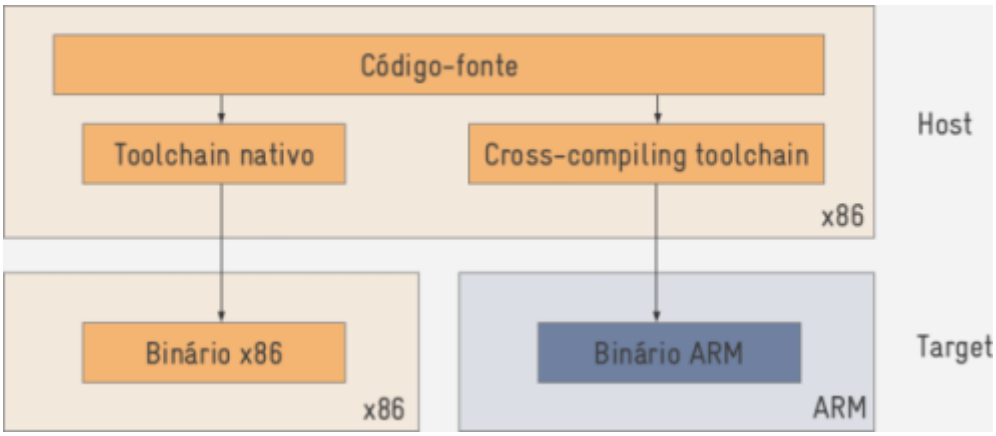
O Linux *kernel* é capaz de rodar em mais de 20 arquiteturas de CPU diferentes. Como por exemplo: x86, ia64, ARM, PowerPC, MIPS, SuperH, Blackfin, Coldfire, Microbalze.
Tem suporte à arquiteturas de 32 e 64 bits e arquiteturas que não possuem MMU (*Memory Management Unit*).
Suporta armazenamento em memórias NAND, FLASH, MMC e hard disks.

No nosso exemplo, um sistema básico pode funcionar com até 8MB de RAM e 2MB de armazenamento, uma vez que, poucos *drivers*, funcionalidades do *kernel*e bibliotecas e aplicativos são necessárias para que o sistema desempenhe as funções desejadas.

.

Toolchain

Conjunto de ferramentas de programação usadas para gerar determinado produto, seja um *software* ou mesmo um sistema completo. Quando a plataforma de desenvolvimento (*host*) é diferente da plataforma alvo (*target*), chamamos o *toolchain* de *cross-compiling toolchain*.



Componentes principais:

- Compilador (gcc);
- *Assembler* e *Linker* (binutils);
- Biblioteca C padrão (glibc, uclibc, dietlibc, musl, etc).
  Algumas opções de toolchains prontas;

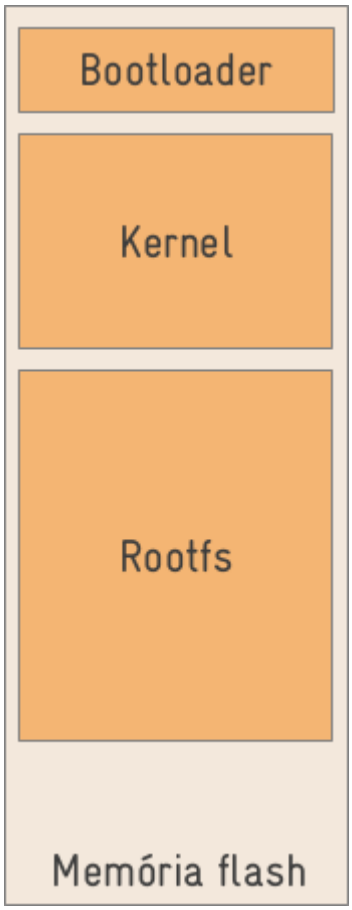
- **GNU Toolchain**: Suporte para até 39 arquiteturas ;

- **Code Sourcery**: ARM ;
- **Linaro**: ARM ;
- **MIPS** .

.

Artefatos de *software* de um Sistema Linux embarcado

Os artefatos de *software* principais de um sistema Linux embarcado são: *Bootloader*, *Kernel* e Rootfs. Tomando-se como base o *hardware* proposto, estes artefatos estão organizados na memória *flash* da seguinte maneira:



.

Bootloader

Toda CPU possui um mecanismo de inicialização, que é responsável por carregar e executar o *bootloader*. Em algumas arquiteturas de CPU é necessário o uso de um *bootloader* de primeiro estágio, no qual é carregado pelo processador e executado a partir de sua memória interna. Em alguns casos esse *bootloader* de primeiro estágio está concatenado junto ao *bootaloader*.

As principais funcionalidades do *bootloader* são:

- Inicializar o *hardware* antes de executar o *kernel* como, por exemplo, configurar a controladora de SDRAM;
- Passar parâmetros para o *kernel*;
- Prover mecanismos para carregar e gravar o *kernel* e o sistema de arquivos na memória *flash* ou cartão SD;
- Inicializar via rede ou pelo cartão SD;
- Rotinas de diagnóstico de *hardware*.

Principais *bootloaders* utilizados em sistemas embarcados:

- x86:
  - LILO;
  - Grub;
  - Syslinux.
- ARM, MIPS, PPC e outras arquiteturas:
  - U-Boot;

- Barebox;
- Redboot.

.

Kernel

O *Kernel* é o coração do nosso sistema. No *boot* ele é responsável por:

- Inicializar CPU, memória e barramentos;
- Configurar a memória virtual (se tiver MMU);
- Inicializar os *device drivers*;
- Iniciar o escalonador de tarefas;
- Iniciar *threads* do *kernel*;
- Montar sistema de arquivos principal (rootfs) e chamar o processo init.

Como principais características podemos apontar:

- Gerencia execução de processos e controla acesso à memória e I/O;
- Gerenciamento do *kernel* space X user space;
- Interface de *user space* com *kernel space* via chamadas do sistema (*system calls*);
- Acesso ao *hardware* via arquivos de dispositivo;
- Gerenciamento dinâmico dos módulos do *kernel*.

.

Rootfs

Após ter sido montado pelo *kernel* e ter o processo init (PID = 1) iniciado, o rootfs utiliza seu mecanismo de inicialização (ex.: **SysVinit**, **Systemd** etc) para inicializar os processos e aplicações do sistema. É responsável por prover as bibliotecas de sistema e de usuário. Alguns exemplos de aplicações para sistemas embarcados:

- Dropbear: cliente e servidor SSH;
- Thttpd: servidor web;
- DirectFB: biblioteca gráfica;
- SQLite: banco de dados;
- Busybox: o canivete suíço de sistemas embarcados com Linux.

.

Referência

Material do Treinamento **Desenvolvendo Sistemas Linux Embarcado** da Embedded Labworks.

Share this:

- Twitter

- Facebook
- Google
-

Relacionado

O que é um sistema embarcado, qual a sua importância?Em "Assuntos Técnicos"
Estréia do Engº Puhlmann no site EmbarcadosEm "Assuntos Técnicos"
OPORTUNIDADE - ESC BRASIL 2012 - Curso gratuito!Em "Notícias"

Tags:Bootloader, electronics, eletrônica, eletrônico, embarcado, Free, grátis,hardware, kernel, Linux, Rootfs, técnico, technical, Toolchain, treinamento,virtual