INSTITUTO DE PESQUISAS TECNOLÓGICAS

COMUNICAÇÃO TÉCNICA

N° 180060

Fazendo uma distro Linux mínima: pré-requisitos

Douglas Bellomo Cavalcante
Rodrigo Dias Garcia
Bruno Gabriel da Fonseca

Palestra apresentada no curso no IPT/SITEC. 4 slides.

Instituto de Pesquisas Tecnológicas do Estado de São Paulo S/A - IPT
Av. Prof. Almeida Prado, 532 | Cidade Universitária ou
Caixa Postal 0141 | CEP 01064-970
São Paulo | SP | Brasil | CEP 05508-901
Tel 11 3767 4374/4000 | Fax 11 3767-4099

www.ipt.br

INSTITUTO DE PESQUISAS TECNOLÓGICAS

TREINAMENTO: CONSTRUINDO UM LINUX EMBARCADO DO ZERO (5 DIAS)

Dia 1: Pré-requisitos e Toolchain

Público-alvo: Estudantes/Profissionais com conhecimentos básicos de terminal Linux.

Hardware necessário: Computador Host (Linux/WSL) e Raspberry Pi 4 (Target).

Ministrado por: Douglas Bellomo Cavalcante, Rodrigo Dias Garcia e Bruno Gabriel da Fonseca

Objetivo

Preparar o ambiente de desenvolvimento, compreender a arquitetura do Linux e realizar a configuração da Toolchain (Compilador Cruzado).

1. Teoria: O Ecossistema Linux Embarcado

Para construir um sistema embarcado, é necessário entender seus três componentes fundamentais:

- **Bootloader:** O software responsável pela inicialização do hardware. No PC utilizamos BIOS/UEFI/GRUB; no Raspberry Pi, quem realiza este papel é o Firmware da GPU.
- **Kernel:** O núcleo do sistema operacional, responsável pelo gerenciamento de CPU, memória e drivers de hardware.
- **Userland (Rootfs):** O espaço do usuário, onde residem os binários, bibliotecas e configurações (ex: Shell, `ls`, `cd`).

Compilação Cruzada (Cross-Compilation):

Diferente do desenvolvimento desktop, aqui trabalhamos com duas arquiteturas:

- **Host (PC):** Arquitetura x86_64.
- **Target (RPi)**: Arquitetura ARM64 (AArch64).

É necessário utilizar uma Toolchain para traduzir o código compilado no PC para que ele seja executável no Raspberry Pi.

2. Prática: Configuração do Host

Execute os comandos abaixo no terminal do sistema Host (Ubuntu/Debian/WSL):

2.1. Instalação de dependências

Instale os pacotes necessários para compilação e download de fontes:

```
Bash
sudo apt update
sudo apt install -y build-essential git bison flex libssl-dev ncurses-dev bc wget
```

2.2. Instalação do Cross-Compiler

Instale o compilador GCC para a arquitetura ARM64:

```
Bash
sudo apt install -y gcc-aarch64-linux-gnu
```

2.3. Criação do Workspace

Crie a estrutura de diretórios para organizar o projeto:

```
Bash
```

```
mkdir -p ~/linux-distro/{sources,build,output}
cd ~/linux-distro
```

- sources: Armazenará o código fonte (Kernel, Busybox).
- output: Armazenará os arquivos finais para o cartão SD.

3. Prática: Teste de Compilação Cruzada

3.1. Código de teste

Crie um arquivo chamado teste.c:

```
C
#include <stdio.h>
int main() { printf("Ola do ARM64!\n"); return 0; }
```

3.2. Compilação e Verificação

Compile o código utilizando o cross-compiler e verifique a arquitetura do binário gerado:

```
Bash
aarch64-linux-gnu-gcc teste.c -o teste-arm
file teste-arm
```

A saída do comando file deve indicar: ***ARM aarch64****.*

3.3. Configuração de Variáveis de Ambiente

Para facilitar os comandos nos próximos passos, adicione as seguintes variáveis ao final do seu arquivo ~/.bashrc:

```
Bash
```

```
export ARCH=arm64
export CROSS_COMPILE=aarch64-linux-gnu-
export WORK_DIR=~/linux-distro
```

Recarregue as configurações do terminal:

```
Bash
source ~/.bashrc
```