UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA MECÂNICA
GRADUAÇÃO EM ENGENHARIA MECATRÔNICA

FEELT49081 – SISTEMAS DIGITAIS PARA MECATRÔNICA

Prática de Sistemas Digitais para Mecatrônica

Prof. Éder Alves de Moura

SEMANA 10:

RESUMO

ANATOMY OF CROSS-COMPILATION TOOLCHAINS

MATHEUS ALVES DE PAULA 11521EMT008

Uberlândia

Fevereiro de 2022

ANATOMY OF CROSS-COMPILATION TOOLCHAINS

O presente resumo trata-se sobre cadeias de ferramentas de compilação cruzada. Inicialmente, vamos abordar a definição de cadeia de ferramentas de compilação cruzada. Definição: É um conjunto de ferramentas que permite construir código-fonte em código binário para uma plataforma de destino diferente daquela em que a compilação ocorre. Alguns exemplos são: diferentes arquiteturas de CPU; ABI diferente; sistema operacional diferente; biblioteca C diferente.

Para ser possível construir código-fonte em código binário, é preciso de três máquinas envolvidas no processo:

- máquina de construção (*build machine*), onde ocorre a construção;
- máquina host (*host machine*), onde ocorre a execução;
- máquina de destino (*target machine*), para a qual os programas geram código.

Vale ressaltar que na cadeia de ferramentas nativa, temos: *build* == *host* == *target*. Já na cadeia de ferramentas de compilação cruzada, temos: *build* == *host* != *target*. Sendo que, a correspondência aos argumentos do script de configuração são: *--build, --host e --target autoconf.* Por padrão, automaticamente adivinhado pelo autoconf para o autoconf da máquina atual define o conceito de definições do sistema, representadas como tuplas.

Agora, falando mais sobre as tuplas da cadeia de ferramentas, temos que a definição de sistema descreve um sistema como: arquitetura de CPU, sistema operacional, fornecedor, ABI e biblioteca C. Existem formas diferentes para representar o sistema:

- <arch>-<vendor>-<os>-<libc/abi>, formulário completo;
- <arch>-<os>-<libc/abi>.

Sendo que cada componente tem um significado na estrutura do sistema:

- <arch>, a arquitetura da CPU: arm, mips, powerpc, i386, i686, etc.
- <vendor>, (principalmente) string de forma livre, ignorada pelo autoconf.
- <os>, o sistema operacional. Ou nenhum ou linux para o propósito desta palestra.
- <libc/abi>, combinação de detalhes da biblioteca C e da ABI em uso.

Agora, uma breve comparação entre a cadeia de ferramentas *bare-metal* com o *Linux*. Sendo que existem dois valores principais para <os>:

- *none* para cadeias de ferramentas de metal nu;
- *linux* para cadeias de ferramentas Linux.

Existem quatro componentes principais em uma cadeia de ferramentas de compilação cruzada do Linux:

1. binutils;
2. gcc;
3. cabeçalhos do kernel Linux;
4. biblioteca C.

Além disso, algumas dependências são necessárias para construir o próprio gcc. Vamos abordar brevemente sobre cada um dos componentes citados.

binutils: "coleção de ferramentas binárias" e suas principais ferramentas são: *Id*, o vinculador e *how*, o montador. Ademais, tem-se as ferramentas de depuração/análise e outras ferramentas: *addr2line, ar, c++filt, gold, gprof, nm, objcopy, objdump, ranlib, readelf, size, strings, strip*. Precisa ser configurado para cada arquitetura de CPU: seus binutils x86 nativos não podem produzir código ARM. Além disso, é bastante simples de compilação cruzada, sendo que não são necessárias dependências especiais.

gcc: coleção do compilador GNU. Front-ends para muitas linguagens de origem: C, C++, Fortran, Go, etc. Back-ends para muitas arquiteturas de CPU. Construir gcc é um pouco mais complicado do que construir binutils, pois são necessários dois passos.

cabeçalhos do kernel Linux: para construir uma biblioteca C, são necessários os cabeçalhos do kernel do Linux: definições de números de chamada do sistema, vários tipos de estrutura e definições. No kernel, os cabeçalhos são divididos entre:

- Cabeçalhos visíveis no espaço do usuário, armazenados em diretórios uapi: include/uapi/, arch/<ARCH>/include/uapi/asm;
- Cabeçalhos de kernel internos.

biblioteca C: fornece a implementação das funções padrão POSIX, além de vários outros padrões e extensões. Baseado nas chamadas do sistema Linux.

Agora, vamos abordar sobre o processo geral de criação (*Overall build process*). O processo de compilação para uma cadeia de ferramentas de compilação cruzada regular do Linux é, na verdade, bastante fácil e segue as seguintes etapas:

1. Construir binutils;
2. Construir as dependências do gcc: mpfr, gmp, mpc;
3. Instalar os cabeçalhos do kernel Linux;

4. Construir um gcc de primeiro estágio: sem suporte para uma biblioteca C, suporte apenas para links estáticos;

5. Construir a biblioteca C usando o primeiro estágio gcc;

6. Construir o gcc final, com biblioteca C e suporte para vinculação dinâmica.

DIFERENÇA ENTRE CADEIA DE FERRAMENTAS (TOOLCHAIN) E SDK

Na cadeia de ferramentas (*toolchain*), temos apenas o compilador, binutils e biblioteca C. Já no SDK, temos uma cadeia de ferramentas, além de um número (potencialmente grande) de bibliotecas criadas para a arquitetura de destino e ferramentas nativas adicionais úteis na construção de software.

Para finalizar, vamos falar sobre como obter uma cadeia de ferramentas de compilação cruzada dividindo em dois critérios:

1. Pré-construído:

- Da sua distribuição. Ubuntu e Debian têm vários compiladores cruzados prontamente disponíveis.

- De várias organizações: Linaro fornece cadeias de ferramentas ARM e AArch64.

2. Construído por você mesmo:

- Crosstool-NG, ferramenta especializada na construção de uma cadeia de ferramentas de compilação cruzada. De longe o mais configurável/versátil.

- Os sistemas de compilação Linux embarcados geralmente sabem como construir uma cadeia de ferramentas de compilação cruzada: *Yocto/OpenEmbedded, Buildroot, OpenWRT*, etc.