Universidade Federal de Uberlândia
FEELT – Faculdade de Engenharia Elétrica

Sistemas Embarcados II

Trabalho Semana 10 – Compilação Cruzada

Professor: Eder Alves de Moura

Engenharia de Controle e Automação

Neila Cristina Morais Silva 12011EAU015

29/02/2022

01. O que é uma ferramenta de compilação cruzada?

É um conjunto de ferramentas que permite construir código binário para uma plataforma de destino diferente de o que você está executando no momento.
Por exemplo, se você estiver em um Linux x86 e você deseja construir um binário x86 para Windows você está fazendo compilação cruzada mesmo que seja a mesma arquitetura da CPU.
Ou seja, ferramentas que produzem códigos para uma arquitetura de CPU diferente, para um sistema operacional diferente ou para uma ABI diferente.
Nesta construção de processo três máquinas estão envolvidas, a máquina de construção que é onde a construção ocorre, a máquina host que é onde ocorre a execução e máquina target para a qual os programas podem estar gerando código.

02. Cadeia de ferramentas

Autoconf define o conceito das Definições de sistema que são a arquitetura da CPU, o sistema operacional, vendor, ABI, biblioteca C.

- Different forms:
  - `<arch>-<vendor>-<os>-<libc/abi>`, full form
  - `<arch>-<os>-<libc/abi>`
- Components:
  - `<arch>`, the CPU architecture: arm, mips, powerpc, i386, i686, etc.
  - `<vendor>`, (mostly) free-form string, ignored by *autoconf*
  - `<os>`, the operating system. Either `none` or `linux` for the purpose of this talk.
  - `<libc/abi>`, combination of details on the C library and the ABI in use

Alguns exemplos podem ser vistos na imagem:

- `arm-foo-none-eabi`, bare-metal toolchain targeting the ARM architecture, from vendor *foo*
- `arm-unknown-linux-gnueabihf`, Linux toolchain targeting the ARM architecture, using the EABIhf ABI and the glibc C library, from an *unknown* vendor
- `armeb-linux-uclibcgnueabi`, Linux toolchain targeting the ARM big-endian architecture, using the EABI ABI and the uClibc C library
- `mips-img-linux-gnu`, Linux toolchain targeting the MIPS architecture, using the glibc C library, provided by Imagination Technologies.

03. Bare-metal vs. Linux toolchain

Bare-metal toolchain
São usados para o desenvolvimento sem um Sistema operacional. Você pode habilitar um número de tipos básicos de chamadas de sistema implementando em um hardware específico, com operações ou conectando-o ao seu pequeno sistema operacional em tempo real.
Linux toolchain
São usados para desenvolver aplicativos para usuários Linux e compartilhar bibliotecas.

04. Binutils

É uma coleção de ferramentas binárias possui algumas ferramentas de analises e debugs mas as principais ferramentas, são:

ld - link vários arquivos de objetos em uma biblioteca compartilhável, um executável ou em outro arquivo de objeto.

as – pega um código assembler de uma arquitetura especifica em forma de texto e produz um arquivo correspondente em código binário.

Uma característica é que precisa ser configurada para cada arquitetura da CPU.

Para fazer uma compilação cruzada, nada demais precisa ser feito, pode-se seguir o exemplo abaixo:

```
./configure --target=arm-buildroot-linux-gnueabihf --with-
sysroot=PATH
```

05. gcc

Coleção compilador GNU. É front-end para muitas linguagens fontes como: C, C++, Go, etc e back-end para muitas arquiteturas de CPU.

Fornece:

Um compilador para C e C++, que só gera código assembler em formato de texto.

O driver compilador;

Bibliotecas para a máquina target;

E cabeçalhos para bibliotecas C++ padrões.

Construir gcc é um pouco mais complexo que construir em Binutils, dois passos são necessários.

06. Linux Kernel headers

Para construir bibliotecas em C, faz-se necessário os cabeçalhos do Kernel do Linux: definição de números de chamadas de sistemas, tipos de estruturas e diversas outras definições.

No Kernel os headers são divididos nos headers do espaço do usuário, e os headers internos do kernel, os quais não são visíveis ao usuário.

É necessário fazer instalação, utilizando a seguinte linha de código:

```
make ARCH=.. INSTALL_HDR_PATH=... headers_install
```

07. C Library

Fornece a implementação das funções padrões do POSIX, mais vários outros standards e extensões.

A biblioteca C fornece um linker dinâmico que é ativo antes do arquivo linker da aplicação, a biblioteca em si e outros tipos de biblioteca, e os cabeçalhos padrões da biblioteca C, como stdio.h etc.

08. glibc

É o padrão das bibliotecas do Linux, usada virtualmente nas distribuições comuns de servidores/desktops.

Suporta uma quantidade enumerada de arquiteturas e sistemas operacionais.

Não suporta MMU plataformas.
Quase não tem configurabilidade e é considerada muito grande para embarcados.

09. uClibc/ uClibc-ng

Foi criada nos anos 2000's e o principal objetivo era fornecer uma biblioteca C menor, na época focados em embarcados.
Tem muitas possibilidades de configurabilidade, suporta muitas arquiteturas, por ser mais fácil em ser construía, mas o sistema operacional é focado somente em Linux.
Suporta numerosas arquiteturas não-MMUs e é apenas suportada para linker estático.
O proeto original uClib está fechado, mas uClib-ng está ativo.

10. Musl

Uma biblioteca mais recente do ano de 2011, tem licença MIT, existe muitos desenvolvedores ativos, suporta várias arquiteturas diferentes, recentemente noMMU suporte foi adicionado, não existe configurabilidade, consegue ser ainda menor que uClibc, especialmente para cenários com linker estáticos e sai muito bem se comparando as outras bibliotecas C citadas aqui.

11. Bibliotecas necessárias p gcc

Para construir gcc são necessárias algumas bibliotecas matemáticas, como mpfr, gmp e mpc. Um detalhe importante é que essas bibliotecas só são necessárias na máquina host.

12. Processo de Construção

Esse processo de construir um compilador cruzado regulador para Linux são definidos em poucos passos, como visto na imagem:

1. Build *binutils*
2. Build the dependencies of *gcc*: *mpfr*, *gmp*, *mpc*
3. Install the Linux kernel headers
4. Build a first stage *gcc*: no support for a C library, support only for static linking
5. Build the *C library* using the first stage *gcc*
6. Build the final *gcc*, with C library and support for dynamic linking

13. Sysroot

Sysroot é um argumento necessário ao fazer binults, que é basicamente o diretório virtual para cabeçalhos e bibliotecas, ou seja, é onde gcc automaticamente procura por headers e bibliotecas.
É onde o usuário colocar os headers e bibliotecas que serão lidos pelo arquivo.

14. Multilib toolchains

Multilib toolchains conténs múltiplos sysroots, cada um tendo uma versão para as bibliotecas target para diferentes arquiteturas/variantes ABI.
Ao usar uma multilib o compilador automaticamente escolhe o sysroot correto dependendo das flas do gcc.

15. Tuning de Arquitetura

A gcc fornece vários tiposi de tunnig para cada variância especifica de arquitetura/CPU, como na imagem:

Essa linha de comando define a variância especifica da arquitetura padrão que a gcc gerará o código.
Elas podem ser reescrevidas usando as opções:

-march, -mcpu, -mabi, -mfpu

16. Definição de ABI

ABI é a sigla para a definição Application Binary Interface. Do ponto de uma toolchain a ABI define:
Como as chamadas de funções são feitas;
O tamanho de tipos de dados básicos;
Alinhamento de membros em estruturas;
Quando tem um sistema operacional, como são feitas as chamadas de sistema.

17. ABI para ARM 32

OABI: obsoleto;
EABI: definido como padrão pela ARM, permite mix de código hard-float com código soft-float e os argumentos de ponto flutuante são passadas em instruções de registros inteiros.
EABIhf: também considerada padrão pela ARM, só código hard-float argumentos de ponto flutuante são passadas em instruções de registros floating.
ggc options:

- EABI soft-float: -mabi=aapcs-linux -mfloat-abi=soft
- EABI hard-float: -mabi=aapcs-linux -mfloat-abi=softfp
- EABIhf: -mabi=aapcs-linux -mfloat-abi=hard

18. Diferença entre toolchain e SDK

Toolchain: é só o compilador, binutils e biblioteca C.
SDK: a toolchain e mais um grande número de bibliotecas construídas para a arquitetura target e algumas ferramentas adicionas para a máquina nativa que ajudam quando construoindo softwares.