**QUESTÃO 1**

**O que é compilação cruzada?**

Um conjunto de ferramentas que permite construir código fonte em código binário para uma plataforma de destino diferente daquela onde a construção ocorre

Três máquinas envolvidas no processo de construção

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

toolchain de compilação cruzada : build == host != target

**Tupla Toolchain**

autoconf define o conceito de definições do sistema, representadas como tuplas

Uma definição de sistema descreve um sistema: arquitetura de CPU, sistema operacional, vendor, ABI, biblioteca C

Formas diferentes:

* » <arch>-<vendor>-<os>-<libc/abi>, forma completa
* » <arch>-<os>-<libc/abi>

Componentes:

* <arch>, a arquitetura da CPU: arm, mips, powerpc, e 1686, etc.
* <vendor>, (principalmente) string de formato livre. ignorado pelo autoconf
* <os> o sistema operacional. Ou None ou Linux.
* <libc/abi>, combinação de detalhes sobre a biblioteca C e a ABL em uso

Há 4 componentes base numa toolchain de compilação cruzada Linux:

* binutils (Coleção de ferramentas binárias)
* gcc (Coleção de compiladores GNU)
* Linux kernel headers
* Biblioteca C

**gcc :**

* Front end para várias linguagens fonte: C, C++, Fortran, Go etc
* Back end para muitas arquiteturas CPU
* Mais envolvido que a construção com binutils

Muitas bibliotecas matemáticas são necessárias para construir com gcc

Elas são compiladas na máquina host, não são necessárias na máquina alvo.

**Linux kernel headers:**

Para construir uma biblioteca C, são necessários os cabeçalhos do kernel do Linux: definições de números de chamadas do sistema, vários tipos de estrutura e definições.

**Biblioteca C:**

Fornece a implementação das funções do padrão POSIX, além de vários outros padrões e extensões

Tem base nas chamadas do sistema Linux

**QUESTÃO 2**

**1. Para identificar as Toolchains do GNU projeto use o seguinte comando no terminal**:

gcc -dumpmachine

Você deve receber algo como:



* Onde x86\_64 é a CPU;
* linux é o Kernel;
* gnu é o sistema operacional

**2. Escolhendo uma toolchain**

Temos 3 opções:

* Uma pré-pronta;
* Uma criada do zero antes da instalação;
* Uma gerada usando a ferramenta de contrução embarcada.

Uma pré-pronta é a opção mais simples, mesmo que não flexível. Esteja certo que a toolchain escolhida possua:

* Sua biblioteca C preferida;
* Seja fácil de atualizar.

Uma aproximação simplificada consiste em utilizar a “crosstool NG” que possui um monte de scripts úteis conduzido por um front-end.

Para instalar Crosstool NG siga os passos:

1. git clone https//github.com/crosstool-ng/crosstool-ng.git
2. mova-o para a pasta “/testing/docker”
3. Ache o diretório de distribuição e mova para dentro dele
4. Abra o “Dockerfile” com um editor de texto
5. Verifique a lista de dependências.
6. Vá para a pasta “crosstool-ng”
7. Faça: git checkout crosstool-ng-<LATEST-NUM>
8. ./bootstrap
9. ./configure –enable-local
10. make
11. make install
12. ./ct-ng (Inicia o menu para testar a instalação)

**3. Construindo uma toolchain para QEMU**

Pegue uma lista de amostras de configurações que são conhecidas para construir e trabalhar

./ct-ng list-samples

Escolha sua configuração, se você já tem uma

./ct-ng <configuration-name>

Abra o menu de configuração:

./ct-ng manuconfig

Remova a flag de read-only, na “Paths and misc options” desmarque a opção “Render the toolchain only”

Comece a construir:

./ct-ng build

A toolchain deve então ficar localizada em:

~/x-tools/arm-unknown-linux-gnueabi/bin

Aqui, ferramentas como compilador, debugger, linker, todas renomeadas com a identidade da toolchain, estarão localizadas.

ld=arm-unknown-linux-gnueabi-ld

**4. Como usar a toolchain**

**4.1. Analisar a configuração do compilador**

Substitua o ponto de suspesão com o nome completo do seu compilador

...-gcc -v

Outros comandos:

Une image contenant texte, intérieur, capture d’écran

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

**4.2. Static Linking**

Para criar bibliotecas estáticas e ligar elas em um arquivo executável:

... -gcc -c mystaticlib1.c

... -gcc -c mystaticlib2.c

… -ar rc libmystaticlib.a mystaticlib1.o mystaticlib2.o

... -gcc myprog.c -lmystaticlib -l../libs -L../libs -o myprog

**4.3. Dynamic linking**

Para criar bibliotecas dinâmicas e ligar elas à um arquivo executável:

... -gcc -fPIC -c mydynlib1.c

... -gcc -fPIC -c mydynlib2.c

… -gcc -shared -o libmydynlib.so mydynlib1.o mydynlib2.o

... -gcc myprog.c -lmydynlib -l../libs -L../libs -o myprog