交叉编译工具链与 Qemu 用户模式仿真实验

1、实验准备

在进行本次实验时，我们约定大家已经成功的安装了 ubuntu22.04LTS 桌面版系统，本实验指导默认的系统用户名为 ubuntu 密码默认 ubuntu@2023 自己实验的时候根据自己的情况更换主机名和用户名，比如以姓名缩写建立，工作目录为 /home/ubuntu/cross_qemu 如果你的路径不一样，需要根据自己的修改，比如不是 mkdir -p /home/ubuntu/cross_qemu 而是 mkdir -p /home/wgy/cross_qemu 。

实验过程可以参考视频：https://datav.bjtu.edu.cn/static/android_7/v7-1.mp4

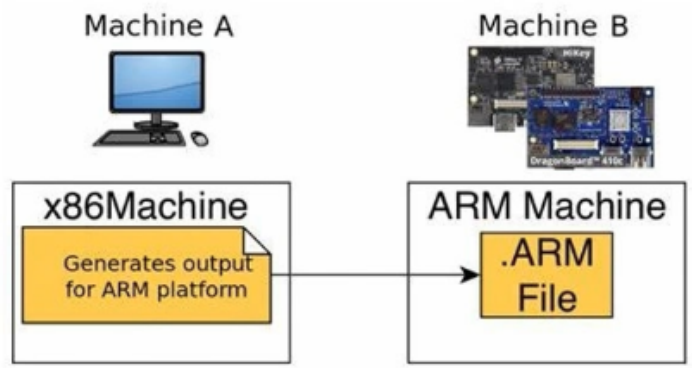
1.1 创建工作目录

Plain Text

```
# 创建工作目录
ubuntu@ubuntu-VirtualBox:~$ mkdir -p /home/ubuntu/cross_qemu
# 进入工作目录
ubuntu@ubuntu-VirtualBox:~$ cd /home/ubuntu/cross_qemu

```

2、构建交叉编译环境



交叉编译：在一个平台上生成另一个平台上的可执行代码。

本实验构建交叉编译环境是在 X86 架构的 Ubuntu22.04 上，构建能够被 ARM 开发板执行的 ARM 程序的环境。

2.1 环境准备

Plain Text

```
# 更新软件源列表并升级已安装的软件包
ubuntu@ubuntu-VirtualBox:~$ sudo apt update
ubuntu@ubuntu-VirtualBox:~$ sudo apt upgrade
# 安装依后续可能用到的依赖
ubuntu@ubuntu-VirtualBox:~$ sudo apt-get install vim make build-essenti
al openssh-server net-tools flex bison libncurses-dev libssl-dev u-boot
-tools
```

2.2 安装交叉编译工具链

选择常见的交叉编译工具链并安装

（1）arm-none-linux-gnueabi-gcc：是 Codesourcery 公司（目前已经被 Mentor 收购）基于 GCC 推出的的 ARM 交叉编译工具。可用于交叉编译 ARM（32 位）系统中所有环节的代码，包括裸机程序、u-boot、Linux kernel、filesystem 和 App 应用程序。

（2）arm-linux-gnueabihf-gcc：是由 Linaro 公司基于 GCC 推出的的 ARM 交叉编译工具。可用于交叉编译 ARM（32 位）系统中所有环节的代码，包括裸机程序、u-boot、Linux kernel、

filesystem 和 App 应用程序。

```
Plain Text
1 ubuntu@ubuntu-VirtualBox:~$ sudo apt install gcc-arm-linux-gnueabi
2 ubuntu@ubuntu-VirtualBox:~$ sudo apt install g++-arm-linux-gnueabi
3
4 ubuntu@ubuntu-VirtualBox:~$ sudo apt-get install gcc-arm-linux-gnueabih
  f
5 ubuntu@ubuntu-VirtualBox:~$ sudo apt-get install libc6-armhf-cross
```

2.3 交叉编译环境检查

```
Plain Text
1 # 检查Codesourcery交叉编译环境
2 ubuntu@ubuntu-VirtualBox:~$ arm-linux-gnueabi-gcc -v
3 # 或者Linaro交叉编译环境
4 ubuntu@ubuntu-VirtualBox:~$ arm-linux-gnueabihf-gcc -v
```

3.Qemu 用户仿真模式下练习

3.1 Qemu 安装检查

通过命令检查是否已经安装了 Qemu

```
Plain Text
1 ubuntu@ubuntu-VirtualBox:~$ qemu-img -V
```

如果没有安装好，需要用以下命令安装

```Plain Text
1 安装依赖
2 ubuntu@ubuntu-VirtualBox:~$ sudo apt-get install zlib1g-dev libglib2.0-
  0 libglib2.0-dev libtool libsdl1.2-dev libpixman-1-0 libpixman-1-dev au
  toconf
3 # 安装qemu
4 ubuntu@ubuntu-VirtualBox:~$ sudo apt-get install qemu-system qemu qemu-
  system-arm qemu-user
```

安装完成后再次重新检查，直到 Qemu 可用。

3.2 交叉编译练习

如果现在还没装好 vim 需要安装一下，安装命令如下，如果之前已经安装了，跳过此步骤

```C/C++
1 ubuntu@ubuntu-VirtualBox:~$sudo apt-get install vim
2 # 要求输入密码时，不会显示自己输入的内容或者*，这里盲输即可，输完按一下回车。
```

(1) 准备源文件 hello.c

(2) 准备 makefile

```Plain Text
1 ubuntu@ubuntu-VirtualBox:~$ cd /home/ubuntu/cross_qemu
2 ubuntu@ubuntu-VirtualBox:~$ vi hello.c
```

hello.c 内容如下：（要求 1：修改其中的 printf 语句，输出自己的姓名和学号，复制程序代码）

Plain Text

```
//hello.c
#include <stdio.h>
int main()
{
    printf("Welcome wgy@2024!\n");
    return 1;
}
```

C/C++

```
ubuntu@ubuntu-VirtualBox:~$ cd /home/ubuntu/cross_qemu
ubuntu@ubuntu-VirtualBox:~$ vi makefile
```

makefile 内容如下【特别注意命令前的格式是 TAB 键】

Plain Text

```
# makefile test for hello program
CC=gcc
CFLAGS=
all:hello
hello:hello.o
    $(CC) $(CFLAGS) hello.o -o hello
hello.o:hello.c
    $(CC) $(CFLAGS) -c hello.c -o hello.o
clean:
    rm -rf hello *.o
```

执行 makefile 生成 可执行文件，查看可执行文件属性

Plain Text

```
ubuntu@ubuntu-VirtualBox:~/cross_qemu$ make
ubuntu@ubuntu-VirtualBox:~/cross_qemu$ file hello

```

```
bjtu@bjtu-VirtualBox:~$ ls
公共的  视频  文档  音乐  back        mkfile       rootfs
模板    图片  下载  桌面  bootloader  mkfile_back  snap
bjtu@bjtu-VirtualBox:~$ cd mkfile
bjtu@bjtu-VirtualBox:~/mkfile$ ls
hello  hello.c  hello.o  makefile
bjtu@bjtu-VirtualBox:~/mkfile$ file hello
hello: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpre
ter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=0adfecf80b9d196b0802b957209abb6bba8294c9, for
 GNU/Linux 3.2.0, not stripped
bjtu@bjtu-VirtualBox:~/mkfile$
```

可以看到这个 hello 可执行程序是 X86-64 的，下面我们通过交叉编译工具链生成能在 arm 架构下执行的可执行文件。

修改 makefile【特别注意命令前的格式是 TAB 键】

通过交叉编译链工具，生成 ARM 架构下的可执行文件

C/C++

```
ubuntu@ubuntu-VirtualBox:~$ cd /home/ubuntu/cross_qemu
ubuntu@ubuntu-VirtualBox:~$ vi makefile
```

内容做如下修改，将 gcc 改成 arm-linux-gnueabi-gcc

Plain Text

```
# makefile test for hello program use arm gcc
CC=arm-linux-gnueabi-gcc
CFLAGS=
all:hello
hello:hello.o
    $(CC) $(CFLAGS) hello.o -o hello
hello.o:hello.c
    $(CC) $(CFLAGS) -c hello.c -o hello.o
clean:
    rm -rf hello *.o
```

将上面生成的 x86-64 下的可执行文件 hello 删除掉，重新运行 make

Shell

```
bjtu@bjtu-VirtualBox:~/mkfile$ make clean
rm -rf hello *.o
bjtu@bjtu-VirtualBox:~/mkfile$ make
```

会出现如下提示：表示执行的是 arm-linux-gnueabi-gcc

arm-linux-gnueabi-gcc -c hello.c -o hello.o

arm-linux-gnueabi-gcc hello.o -o hello

执行 makefile 后最终生成可执行文件，查看可执行文件属性

Plain Text

```
ubuntu@ubuntu-VirtualBox:~/cross_qemu$ file hello

hello: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), dynamica
  lly linked, interpreter /lib/ld-linux.so.3, BuildID[sha1]=b68bfc5f4bc7c
  4f9e145945d47f75806ab6c6e4e, for GNU/Linux 3.2.0, not stripped

```

3.3 Qemu 用户模式仿真调试

Plain Text

```
1 ubuntu@ubuntu-VirtualBox:~/cross_qemu$ qemu-arm -L /usr/arm-linux-gnuea
  bi/ ./hello
2
3 Welcome wgy@2024!
4
```

（要求 2：截取运行效果图，如下所示）