***Environment setting:***

sudo apt-get install qemu-system

sudo apt-get install gcc-arm-linux-gnueabi

sudo apt-get install gdb-multiarch

sudo apt-get install bridge-utils

**Linux kernel**

make ARCH=arm CROSS\_COMPILE=arm-linux-gnueabi- vexpress\_defconfig

make ARCH=arm CROSS\_COMPILE=arm-linux-gnueabi- -j 4 all

make ARCH=arm CROSS\_COMPILE=arm-linux-gnueabi- menuconfig

make ARCH=arm CROSS\_COMPILE=arm-linux-gnueabi- clean

make ARCH=arm CROSS\_COMPILE=arm-linux-gnueabi- mrproper

~/work $ mkdir arm-image

~/work $ cp ./linux-3.10.61/arch/arm/boot/zImage ./arm-image

~/work $ cp ./linux-3.10.61/arch/arm/boot/dts/\*.dtb ./arm-image

~/work $ mkdir -p ./arm-linux/lib

~/work $ cp -a /usr/arm-linux-gnueabi/lib/\* ./arm-linux/lib

**Busybox**

make ARCH=arm defconfig

make ARCH=arm CROSS\_COMPILE=arm-linux-gnueabi- install

cd \_install

mkdir proc sys dev etc etc/init.d

vi etc/init.d/rcS

#! /bin/sh

mount -t proc none /proc

mount -t sysfs none /sys

mount -t devtmpfs devtmpfs /dev

/sbin/mdev -s

chmod +x ./etc/init.d/rcS

find . | cpio -o --format=newc > ../rootfs.img

1.

qemu-system-arm -M vexpress-a9 -cpu cortex-a9 -m 1024M -kernel linux-3.10/arch/arm/boot/zImage -initrd busybox-1.21.1/rootfs.img -append "root=/dev/ram rdinit=/sbin/init"

2.

qemu-system-arm -M vexpress-a9 -cpu cortex-a9 -m 1024M -net nic -net tap -kernel ./arm-image/kernel/zImage -dtb arm-image/kernel/vexpress-v2p-ca9.dtb -initrd arm-image/rootfs.img -serial stdio -append " root=/dev/ram rdinit=/sbin/init"

注：实验环境是 Ubuntu 16.04.1 gnome

安装交叉编译工具、GDB 和 QEMU

~/work $ sudo apt-get install gcc-arm-linux-gnueabi

~/work $ sudo apt-get install gdb-multiarch

~/work $ sudo apt-get install qemu-system-arm

配置编译内核

~/work/linux-3.10.61 $ make CROSS\_COMPILE=arm-linux-gnueabi- ARCH=arm vexpress\_defconfig

~/work/linux-3.10.61 $ make CROSS\_COMPILE=arm-linux-gnueabi- ARCH=arm

创建目录 arm-image 目录，将 zImage 和 dtb（device tree binary）文件复制到此目录

~/work $ mkdir arm-image

~/work $ cp ./linux-3.10.61/arch/arm/boot/zImage ./arm-image

~/work $ cp ./linux-3.10.61/arch/arm/boot/dts/\*.dtb ./arm-image

复制编译器提供的 C 库和 ld 库

~/work $ mkdir -p ./arm-linux/lib

~/work $ cp -a /usr/arm-linux-gnueabi/lib/\* ./arm-linux/lib

BusyBox

(1) 将下载好的 BusyBox 解压在 /home/user/work/busybox-1.25.1 中，进行配置

~/work/busybox-1.25.1 $ make ARCH=arm defconfig

~/work/busybox-1.25.1 $ sudo apt-get install libncurses5-dev

~/work/busybox-1.25.1 $ make menuconfig

(2) 然后 ：

Busybox Settings -> Build Options -> [\*] Build BusyBox as a static Binary (no shared libs)

保存退出

(3) 编译和安装:

~/work/busybox-1.25.1 $ make ARCH=arm CROSS\_COMPILE=arm-linux-gnueabi-

~/work/busybox-1.25.1 $ make ARCH=arm CROSS\_COMPILE=arm-linux-gnueabi- install

(4) 复制 \_install 目录下的内容

~/work/busybox-1.25.1 $ ./\_install/\* ../arm-linux/

新建系统启动脚本

~/work/arm-linux $ mkdir -p proc sys dev etc/init.d

~/work/arm-linux $ vim etc/init.d/rcS

rcS 的内容：

#! /bin/sh

mount -t proc none /proc

mount -t sysfs none /sys

mount -t devtmpfs devtmpfs /dev

/sbin/mdev -s

分配运行权限：

~/work/arm-linux $ chmod +x ./etc/init.d/rcS

根文件系统通过 nfs 的方式提供给 ARM Linux 使用

~/work $ sudo apt-get install nfs-kernel-server

~/work $ sudo vim /etc/exports

加入一行规则，nfs 将合适的路径提供给 ARM Linux 使用。示例：

/home/user/work/arm-linux \*(rw,sync,no\_subtree\_check,no\_root\_squash,insecure)

重新启动 nfs 服务：

~/work $ sudo service nfs-kernel-server restart

用 qemu-system-arm 启动 ARM linux 系统

~/work $ vim ./start.sh

start.sh （注意修改对应的 IP 地址和 nfsroot 的位置）：

qemu-system-arm \

-M vexpress-a9 -m 1024M \

-kernel ./arm-image/zImage \

-dtb arm-image/vexpress-v2p-ca9.dtb \

-serial stdio \

-append "console=ttyAMA0 nfsroot=192.168.118.133:/home/user/work/arm-linux

rw ip=dhcp"

使用 GDB 调试内核

~/work $ vim ./debug.sh

debug.sh:

#! /bin/sh

gnome-terminal -x gdb-multiarch; \

qemu-system-arm \

-S -s \

-M vexpress-a9 -m 1024M \

-kernel ./arm-image/zImage \

-dtb arm-image/vexpress-v2p-ca9.dtb \

-serial stdio \

-append "console=ttyAMA0 nfsroot=192.168.118.133:/home/user/work/arm-linux

rw ip=dhcp"

使用 .gdbinit

(1)

~ $ vim .gdbinit

.gdbinit 的内容：

add-auto-load-safe-path /home/user/work/.gdbinit

(2)

~/work $ vim .gdbinit

.gdbinit 的内容：

set architecture arm

target remote localhost:1234

(3) 不同阶段调试内核 gdb 需要输入的命令

zImage 重定位之前：

~/work $ ./debug.sh

(gdb) set disassemble-next on

(gdb) add-symbol-file ./linux-3.10.61/arch/arm/boot/compressed/vmlinux 0x60010000

(gdb) b \* 0x60010000

(gdb) c

(gdb) si

zImage 重定位之后：

~/work $ ./debug.sh

(gdb) set disassemble-next on

(gdb) add-symbol-file ./linux-3.10.61/arch/arm/boot/compressed/vmlinux 0x604E4160

(gdb) b \* 0x604E4210

(gdb) c

内核解压后：

~/work $ ./debug.sh

(gdb) set disassemble-next on

(gdb) add-symbol-file ./linux-3.10.61/vmlinux 0x60008240 -s .head.text 0x60008000 -s .rodata 0x60479000

(gdb) b stext

(gdb) c

start\_kernel 及之后：

~/work $ ./debug.sh

(gdb) file ./linux-3.10.61/vmlinux

(gdb) b start\_kernel

(gdb) c

文件结构概览

/home/user

.gdbinit

./work

./linux-3.10.61

./busybox-1.25.1

./arm-image

./arm-linux

.gdbinit

start.sh

debug.sh

注： 使用 ctags 和 cscope

~/work/linux-3.10.61 $ make tags ARCH=arm

~/work/linux-3.10.61 $ make cscope ARCH=arm

全文完

Reference:

http://xda.com.cn/archives/type/image

http://blog.csdn.net/linyt/article/details/42504975

http://blog.chinaunix.net/uid-26009923-id-3825761.html

http://zhiyisun.github.io/2015/04/22/how-to-use-gdb-to-debug-ARM-kernel-in-Qemu.html

http://www.alivepea.me/kernel/gdb-kernel-startup/

作者：Twistack

鏈接：http://www.jianshu.com/p/980b4ae4b900

來源：簡書

著作權歸作者所有。商業轉載請聯繫作者獲得授權，非商業轉載請註明出處。

RUNNING QEMU IN -NOGRAPHIC MODE

When qemu is run in th -nographic mode, like this :

> qemu-system-arm -M versatilepb -m 128M -nographic -kernel u-boot.bin

qemu monitor can be obtained through <Ctrl-a foo\_key>, similar to Minicom or GNU Screen.

To see the help use <Ctrl-a h> or <Ctrl-a ?> :

VersatilePB #

C-a h print this help

C-a x exit emulator

C-a s save disk data back to file (if -snapshot)

C-a t toggle console timestamps

C-a b send break (magic sysrq)

C-a c switch between console and monitor

C-a C-a sends C-a

<Ctrl-a x> toggles between qemu monitor (internal to qemu) and console in which u-boot is run :

VersatilePB # QEMU 0.12.5 monitor – type ‘help’ for more information

(qemu)