LABORATÓRIO TPSE II

Lab 03: Compilando e Gravando o Linux

Prof. Francisco Helder

31 de agosto de 2023

Neste laboratório você irá aprender a configurar um ambiente de compilação cruzada do kernel para o Kit de desenvolvimento da Texas Instrument AM335x (BeagleBone Black), também o uso do U-Boot para baixar o kernel, além de verifique se o kernel que você compilou inicia o sistema. O kernel do linux é exatamente o que a tradução sugere, é o nucleo do sistema operacional, ou seja, a parte mais importante de um sistema embarcado. Para a disciplina, é bastante necessário que tenha esse conceito em mente.

1 Compilando e Gravando o Linux

Entre no diretório de exercícios do treinamento:

```
$ cd /opt/labs/ex/04
```

Baixe o código-fonte do kernel Linux:

```
$ git clone https://github.com/beagleboard/linux.git
$ cd linux
$ git checkout 5.10
```

Para fazer a compilação cruzada do Linux, você precisa ter um toolchain de compilação cruzada. Usaremos o toolchain de compilação cruzada que produzimos anteriormente, então precisamos apenas disponibilizá-la no PATH:

```
$ export PATH=$PATH:/home/heldercs/UFC/disciplinas/QXD0150/lab/
   toolchain/gcc-arm-linux-gnueabihf/bin/
```

Além disso, não se esqueça de:

- Defina o valor das variáveis ARCH e CROSS_COMPILE no ambiente (usando export)
- Ou especifique-os na linha de comando a cada chamada do make, ou seja: make ARCH=... CROSS_COMPILE=... <target>

Ao executar o **make help**, encontre o destino do Makefile adequado para configurar o kernel para a placa Beagle (dica: a configuração padrão não é nomeada com o nome da placa, mas com o nome do SoC). Uma vez encontrado, use este destino para configurar o kernel com a configuração pronta. Não hesite em visualizar as novas configurações executando **make menuconfig** depois!

Na configuração do kernel, como experiência, altere a compactação do kernel de Gzip para XZ. Esse algoritmo de compactação é muito mais eficiente que o Gzip, em termos de taxa de compactação, à custa de um tempo de descompactação maior.

Como o kernel Linux suporta diferentes plataformas de hardware, ele deve ser configurado antes de ser compilado. Portanto, execute o comando abaixo para configurar o kernel Linux para o target:

```
$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- bb.org_defconfig
```

1.1 Compilando o Kernel

Agora o kernel está pronto para ser compilado. Então execute o seguinte comando:

```
$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- -j4
```

Este processo pode demorar algum tempo, dependendo de computador para computador. Após o kernel ser compilado, gere uma versão de uma imagem com o device tree. A variavel de ambiente LOADADDR serve para indicar em qual posição de memória está o device tree.

```
$make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- uImage dtbs
   LOADADDR=0x80008000 -j4
```

Após gerar o arquivo **uImage**, veja usando o comando **ls**, se o arquivo realmente está no diretório **arch/arm/boot/**, o resultado esperado é o seguinte:

```
$ ls arch/arm/boot/uImage -lha
-rw-rw-r-- 1 heldercs heldercs 9,7M set  6 10:04 arch/arm/boot/uImage
```

Verifique também se o arquivo **am335x-boneblack.dtb** está no diretório **arch/arm/boot/dts**. Com esses dois arquivos gerados, copie eles para seu servidor TFTP.

```
$ ls arch/arm/boot/dts/am335x-boneblack.dtb -lha
-rw-rw-r-- 1 heldercs heldercs 59K set  6 09:28 arch/arm/boot/dts/
   am335x-boneblack.dtb
```

Com esses dois arquivos gerados, copie eles para seu servidor TFTP.

```
$ cp arch/arm/boot/dts/am335x-boneblack.dtb /tftpboot/
$ cp arch/arm/boot/uImage /tftpboot/
```

1.2 Carregue e Inicialize o kernel Usando o U-Boot

Após a geração das imagens e a copia para o servidor TFTP, podemos então rodar essas imagens usando a placa. Use o bootloader no cartão SD, como foi feito na ultima prática, pois vai lhe poupar bastante tempo. Inicie o bootloader pelo cartão SD e insira os seguintes comandos:

```
=> set ipaddr 10.4.1.2
=> set serverip 10.4.1.1
```

Feito isso, o próximo passo é configurar a variável “bootargs”, que é a responsável por passar informações para o Linux. Seu conteúdo é automaticamente passado para o kernel Linux como argumentos de inicialização. Como vamos usar a porta serial como console, então a configuração da variável é a seguinte:

```
=> setenv bootargs "console=ttyS0,115200,n8"
```

E por fim, é preciso baixar as imagens que estão no servidor TFTP para regiões de memória da placa. Então, para não precisarmos fazer as mesmas configurações mais de uma vez, podemos configurar a variável “bootcmd”, que é basicamente a variável que guarda o comando que a placa vai executar caso nenhuma tecla seja pressionada para parar a inicialização, então basicamente essa variável deve ser configurada da seguinte forma:

```
=> tftpboot 0x80F80000 am335x-boneblack.dtb
=> tftpboot 0x80007FC0 uImage
=> bootm 0x80007FC0 - 0x80F80000
```

E assim o resultado esperado deve ser como mostrado na Figura 1.2. Perceba que o kernel retorna uma mensagem de erro com a seguinte descrição “ end Kernel panic - not syncing: VFS: Unable to mount root fs”, isso ocorreu porque ele não encontrou o sistema de arquivo para executar. Veja a próxima seção para corrigir esse erro.

```
[   10.239388] cpu cpu0: Dropping the link to regulator.3
[   10.245228] cpu cpu0: Linked as a consumer to regulator.3
[   10.254217] omap_rtc 44e3e000.rtc: setting system clock to 2000-01-01 00:00:00 UTC (946684800)
[   10.263964] ALSA device list:
[   10.267039]   No soundcards found.
[   10.271362] VFS: Cannot open root device "(null)" or unknown-block(0,0): error -6
[   10.279038] Please append a correct "root=" boot option; here are the available partitions:
[   10.287517] b300         3817472 mmcblk1
[   10.287523]  driver: mmcblk
[   10.294447]   b301         3816448 mmcblk1p1 00000000-01
[   10.294450]
[   10.301325] Kernel panic - not syncing: VFS: Unable to mount root fs on unknown-block(0,0)
[   10.309633] CPU: 0 PID: 1 Comm: swapper/0 Not tainted 4.19.94+ #1
[   10.315752] Hardware name: Generic AM33XX (Flattened Device Tree)
[   10.321915] [<c0113ec4>] (unwind_backtrace) from [<c010e1fc>] (show_stack+0x20/0x24)
[   10.329708] [<c010e1fc>] (show_stack) from [<c0d72abc>] (dump_stack+0x88/0x9c)
[   10.336971] [<c0d72abc>] (dump_stack) from [<c013d668>] (panic+0x108/0x278)
[   10.343977] [<c013d668>] (panic) from [<c14018c0>] (mount_block_root+0x1e0/0x294)
[   10.351498] [<c14018c0>] (mount_block_root) from [<c1401aa0>] (mount_root+0x12c/0x154)
[   10.359455] [<c1401aa0>] (mount_root) from [<c1401c28>] (prepare_namespace+0x160/0x1a4)
[   10.367498] [<c1401c28>] (prepare_namespace) from [<c14013cc>] (kernel_init_freeable+0x3ac/0x3c0)
[   10.376414] [<c14013cc>] (kernel_init_freeable) from [<c0d88648>] (kernel_init+0x18/0x124)
[   10.384719] [<c0d88648>] (kernel_init) from [<c01010e8>] (ret_from_fork+0x14/0x2c)
[   10.392321] Exception stack(0xdc145fb0 to 0xdc145ff8)
[   10.397397] 5fa0:                                     00000000 00000000 00000000 00000000
[   10.405613] 5fc0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
[   10.413826] 5fe0: 00000000 00000000 00000000 00000000 00000013 00000000
[   10.420490] ---[ end Kernel panic - not syncing: VFS: Unable to mount root fs on unknown-block(0,0) ]---
```

2 Atividades Práticas

pratica 1

Use o sistema de arquivo existente na flash, passando os argumentos no “bootargs”, para iniciar o Linux que você gerou com o FileSystem já existente na flash.

pratica 2

Altere as variáveis de ambiente para que o bootcmd carregue o device tree, o kernel e inicie o linux, usando o rootfs (sistema de arquivo) na flash.

pratica 3

Grave a imagem do kernel e o arquivo device tree no sdCard e altere o bootcmd para carregar dos sdCard

pratica 4

Ligar e desligar os 4 LEDs da placa via Linux