Kernel Upgrade

Author:

Armando Emmanuel Correa Amorelli

Zapopan, Jalisco, June 20, 2023

Table of Contents

Introduction

To develop the kernel, it is necessary to understand the sequence of processes that generate the Linux executable binary. This will facilitate adding new modules or device drivers, as well as customizing the features and/or parameters used by the kernel.

Objective

Configure, compile and install a personalized kernel to the BeagleBone

Kernel Code

Cloning the repo

First of all we need to clone the source code for the kernel. To do this we need to use the git command an clone the repo from the stable branch to get all the tags and branches of all the stable releases, otherwise we only have access to the main branch.

```
➜  modulo4 git:(main) ✗ git clone git://git.kernel.org/pub/scm/linux/kernel/git/stable/linux.git OGkernel
Cloning into 'OGkernel'...
remote: Enumerating objects: 11482776, done.
remote: Counting objects: 100% (119/119), done.
remote: Compressing objects: 100% (52/52), done.
remote: Total 11482776 (delta 83), reused 79 (delta 67), pack-reused 11482657
Receiving objects: 100% (11482776/11482776), 4.44 GiB | 879.00 KiB/s, done.
Resolving deltas: 100% (9171667/9171667), done.
Updating files: 100% (80340/80340), done.
```

Choosing the kernel version

The next step is to choose the version that we want to use, and change to the corresponding branch with the git checkout command.

```
➜  OGkernel git:(master) git checkout linux-5.10.y
Updating files: 100% (60672/60672), done.
Branch 'linux-5.10.y' set up to track remote branch 'linux-5.10.y' from 'origin'.
Switched to a new branch 'linux-5.10.y'
```

Kernel Configuration

Making output directory

To keep our code clean and organized we need to make a specific directory for the output files of the configuration and the building. For this we used the mkdir commands

```
➜  modulo4 git:(main) ✗ mkdir kernel_build
```

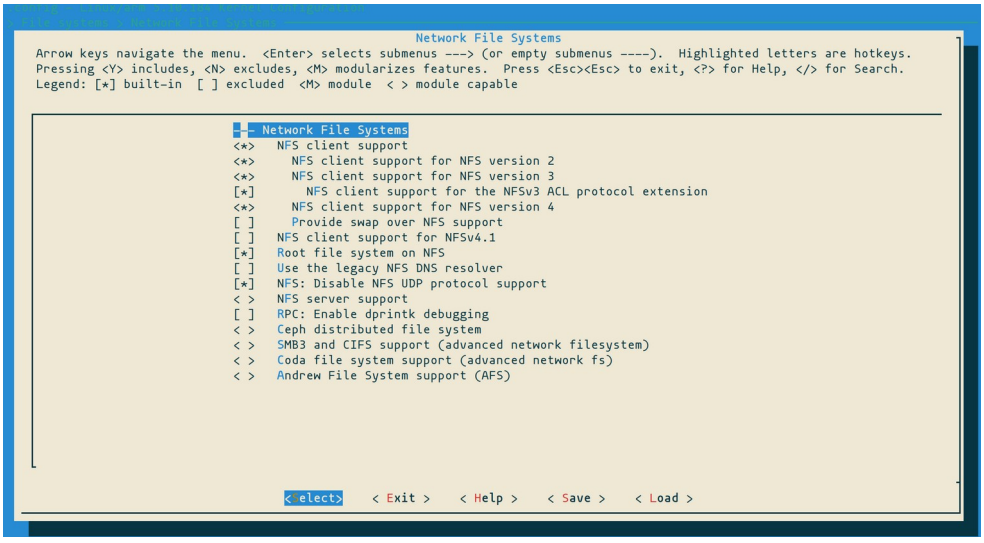
Creating configuration file

For this step we can use a multiple ways to create the config file. In this case we decided to use the omap2plus_defconfig option in combination with the make command, to start from a base configuration that we know will work on this hardware.

```
➜  OGkernel git:(linux-5.10.y) make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- omap2plus_defconfig -j4 O=../kernel_build/
```

Adding extra configuration

Because we are going to use the USB port as a Ethernet interface, and mount a NFS file system, we need to make sure to enable both in the configuration. For this we used the same command as in the previous step, but changing the omap2plus_defconfig with menuconfig.



Kernel Compilation

For compiling the kernel we use the make command, specifying the architecture, cross-compiler, output directory, and for making it faster the number of jobs with the -jx flag.

```
➜  OGkernel git:(linux-5.10.y) make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- O=../kernel_build/$ -j4
```

TFTP files preparation

To be able to load the new device tree and zImage, we need to copy them to the tftp directory where our BBB will look for them.

```
➜  boot git:(main) ✗ sudo cp ./dts/am335x-boneblack.dtb /srv/tftp/
➜  boot git:(main) ✗ sudo cp zImage /srv/tftp
```

Kernel loading

To loading the kernel we are taking advantage of the previous environment that we have saved on our SD card, and just tweak tftp_custom command that we already have.

```
tftp_custom=tftp 0x81000000 encoded_zImage.bin; decode_kernel 0x81000000 0x81000000 10; tftp 0x82000000 am335x-boneblack.dtb; bootz 0x81000000 - 0x82000000;
update_to_fit=setenv loadaddr ${fit_loadaddr}; setenv bootfile ${fit_bootfile}
usb_boot=usb start; if usb dev ${devnum}; then setenv devtype usb; run scan_dev_for_boot_part; fi
usbnet_devaddr=f8:dc:7a:00:00:02
usbnet_hostaddr=f8:dc:7a:00:00:01
vendor=ti
ver=U-Boot 2018.05-00001-g7be1a198c8-dirty (Jun 18 2023 - 18:15:12 -0600)

Environment size: 9666/131067 bytes
=> tftp_custom "tftp 0x81000000 zImage; tftp 0x82000000 am335x-boneblack.dtb; bootz 0x81000000 - 0x82000000;"
CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.7.1 | VT102 | Offline | ttyUSB0
```

Booting process

Once we set everything to pull the new images and use them to boot, we can restart our board and check if, in fact, the kernel version changed

Results

As we can see on Image 1, the kernel version change from the one on image 2

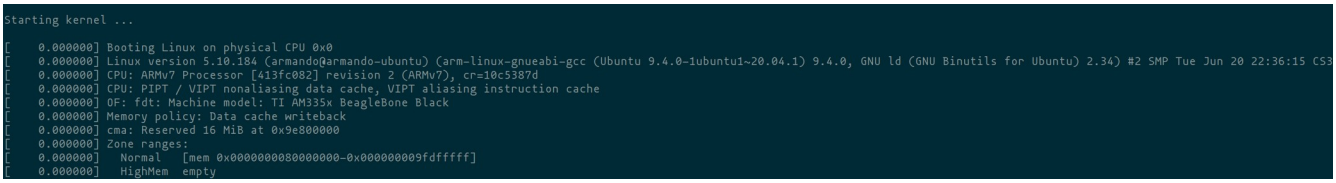


Image 1: New kernel

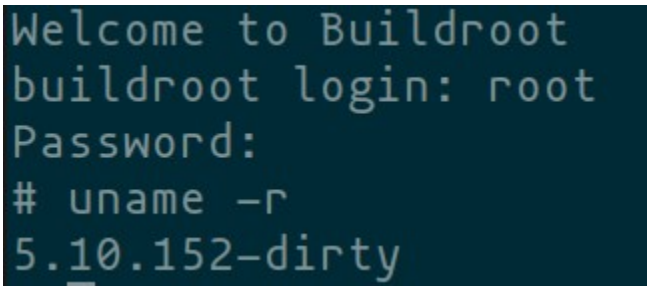
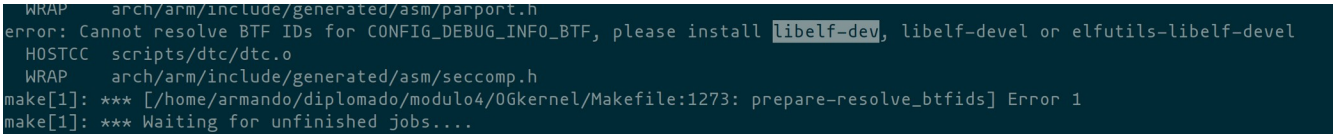


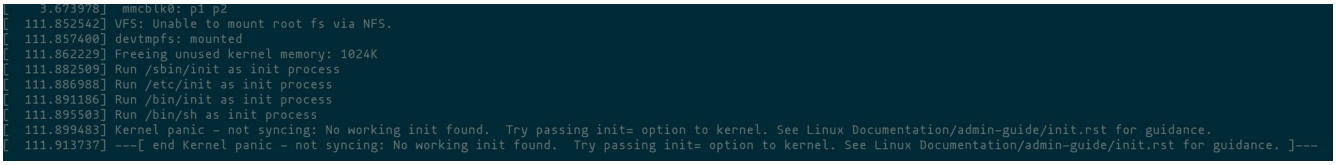
Image 2: Original kernel

Issues encountered

For the compilation process we encounter the necessity to install either libelf-dev, libelf-devel or elfutils-libelf-devel, otherwise we couldn't continue with the compilation process.



While booting we encounter that the file system that we are using dosen't count with a boot directory and also, for some reason, the kernel couldn't sync the init processes.



Conclusions

Through this activity I got familiar with the process of compiling a kernel and also realized that it might broke some thing if upgraded carelessly. For example, we are no longer able to run from the nfs system. Although I might have make some kind of error in the configuration of the kernel or the bootloader.

Bibliography

[1] “Linux kernel stable tree ,” Kernel/Git/Stable/linux.git - linux kernel stable Tree, https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux.git/ (accessed Jun. 20, 2023).