Modify, rebuild and reload the Linux® kernel

Modify, rebuild and reload the Linux® kernel

Stable: 24.06.2020 - 15:45 / Revision: 22.06.2020 - 13:16

1 Overview

This stage explains how modify, rebuild and reload the Linux® kernel.
You will first be guided to install the Linux® kernel source code in the Developer Package directory. Then step by step you will execute procedures to modify, rebuild and reload the Linux® kernel.

2 Download the the Linux® kernel source code

The software package is provided AS IS, and by downloading it, you agree to be bound to the terms of the software license agreement (SLA). The detailed content licenses can be found here.

To download a package, it is recommended to be logged in to your "myst" account [1]. If, trying to download, you encounter a “403 error”, you could try to empty your browser cache to workaround the problem. We are working on the resolution of this problem.
We apologize for this inconvenience

2.1 For ecosystem release v1.2.0

- Download the STM32MP15-Ecosystem-v1.2.0 Developer Package Sources to the following directory:
  $HOME/STM32MPU_workspace/STM32MP15-Ecosystem-v1.2.0/Developer-Package
- Uncompress the tarball file to get the Linux® kernel tarball, the ST patches and the ST configuration fragments

```
PC $> cd $HOME/STM32MPU_workspace/STM32MP15-Ecosystem-v1.2.0/Developer-Package
PC $> tar xvf en.SOURCES-kernel-stm32mp1-openstlinux-20-02-19.tar.xz
```

2.2 For ecosystem release v1.1.0

- Download the STM32MP15-Ecosystem-v1.1.0 Developer Package Sources to the following directory:
  $HOME/STM32MPU_workspace/STM32MP15-Ecosystem-v1.1.0/Developer-Package
- Uncompress the tarball file to get the Linux® kernel tarball, the ST patches and the ST configuration fragments

```
PC $> cd $HOME/STM32MPU_workspace/STM32MP15-Ecosystem-v1.1.0/Developer-Package
PC $> tar xvf en.SOURCES-kernel-stm32mp1-openstlinux-4.19-thud-mp1-19-10-09.tar.xz
```

2.3 For ecosystem release v1.0.0 ⚠

- Download the STM32MP15-Ecosystem-v1.0.0 Developer Package Sources to the following directory:
  $HOME/STM32MPU_workspace/STM32MP15-Ecosystem-v1.0.0/Developer-Package
- Uncompress the tarball file to get the Linux® kernel tarball, the ST patches and the ST configuration fragments

```
PC $> cd $HOME/STM32MPU_workspace/STM32MP15-Ecosystem-v1.0.0/Developer-Package
PC $> tar xvf en.SOURCES-kernel-stm32mp1-openstlinux-4.19-thud-mp1-19-02-20.tar.xz
```

3 Prepare the Linux® kernel source code

- Extract the Linux® kernel source

```
For ecosystem release v1.2.0 ⚠
PC $> cd stm32mp1-openstlinux-20-02-19/sources/arm-ostl-linux-gnueabi/linux-stm32mp-
4.19-r0
PC $> tar xvf linux-4.19.94.tar.xz

For ecosystem release v1.1.0 ⚠
PC $> cd stm32mp1-openstlinux-4.19-thud-mp1-19-10-09/sources/arm-openstlinux_weston-
linux-gnueabi/linux-stm32mp-4.19-r0
PC $> tar xvf linux-4.19.49.tar.xz

For ecosystem release v1.0.0 ⚠
PC $> cd stm32mp1-openstlinux-4.19-thud-mp1-19-02-20/sources/arm-openstlinux_weston-
linux-gnueabi/linux-stm32mp-4.19-r0
PC $> tar xvf linux-4.19.9.tar.xz
```

- Apply the ST patches

```
PC $> cd linux-4.19.94/ /* For ecosystem release v1.2.0 ⚠ */
PC $> cd linux-4.19.49/ /* For ecosystem release v1.1.0 ⚠ */PC $> cd linux-4.19.9//*Fo
r ecosystem release v1.0.0 ⚠  */

PC $> for p in `ls -1 ../*.patch`; do patch -p1 < $p; done
```

- Apply fragments

```
PC $> make ARCH=arm multi_v7_defconfig "fragment*.config"
PC $> for f in `ls -1 ../fragment*.config`; do scripts/kconfig/merge_config.sh -m -r .
config $f; done
PC $> yes '' | make ARCH=arm oldconfig
```

4 Build the Linux® kernel source code for the first time

The first time the kernel is build it could take several minutes.

- Build kernel images (uImage and vmlinux) and device tree (dtbs)

```
PC $> make ARCH=arm uImage vmlinux dtbs LOADADDR=0xC2000040
```

- Build kernel module

```
PC $> make ARCH=arm modules
```

- Generate output build artifacts

```
PC $> mkdir -p $PWD/install_artifact/
PC $> make ARCH=arm INSTALL_MOD_PATH="$PWD/install_artifact" modules_install
```

5 Deploy the Linux® kernel on the board

5.1 Push the Linux® kernel into the board

```
PC $> scp arch/arm/boot/uImage root@<board ip address>:/boot
```

5.2 Push the devicetree into the board

```
PC $> scp arch/arm/boot/dts/stm32mp157*.dtb root@<board ip address>:/boot
```

5.3 Push the kernel modules into the board

For ecosystem release v1.2.0 ⚠:

- Remove the link created inside the **install_artifact/lib/modules/4.19.94** directory

```
PC $> rm install_artifact/lib/modules/4.19.94/build install_artifact/lib/modules/4.
19.94/source
```

For ecosystem release v1.1.0 ⚠:

- Remove the link created inside the **install_artifact/lib/modules/4.19.49** directory

```
PC $> rm install_artifact/lib/modules/4.19.49/build install_artifact/lib/modules/4.
19.49/source
```

For ecosystem release v1.0.0 ⚠:

- Remove the link created inside the **install_artifact/lib/modules/4.19.9** directory

```
PC $> rm install_artifact/lib/modules/4.19.9/build install_artifact/lib/modules/4.19.9
/source
```

- Optionally, strip kernel modules (to reduce the size of each kernel modules)

```
PC $> find install_artifact/ -name "*.ko" | xargs $STRIP --strip-debug --remove-
section=.comment --remove-section=.note --preserve-dates
```

- Copy Kernel modules

```
PC $> scp -r install_artifact/lib/modules/* root@<ip of board>:/lib/modules
```

- Using the Linux console, re-generate the list of module dependencies (modules.dep) and the list of symbols provided by modules (modules.symbols)

```
Board $> /sbin/depmod -a
```

- Synchronize data on disk with memory

```
Board $> sync
```

5.4 Reboot the board

```
Board $> reboot
```

6 Modifying a built-in Linux kernel device driver

This simple example adds unconditional log information when the display driver is probed.

- Using the Linux console, check that there is no log information when the display driver is probed

```
Board $> dmesg | grep -i stm_drm_platform_probe
Board $>
```

- Go to the Linux® kernel source directory

```
For ecosystem release v1.2.0 ⚠:
PC $> cd $HOME/STM32MPU_workspace/STM32MP15-Ecosystem-v1.2.0/Developer-Package
/stm32mp1-openstlinux-20-02-19/sources/arm-ostl-linux-gnueabi/linux-stm32mp-4.19-r0
/linux-4.19.94

For ecosystem release v1.1.0 ⚠:
PC $> cd $HOME/STM32MPU_workspace/STM32MP15-Ecosystem-v1.1.0/Developer-Package
/stm32mp1-openstlinux-4.19-thud-mp1-19-10-09/sources/arm-openstlinux_weston-linux-
gnueabi/linux-stm32mp-4.19-r0/linux-4.19.49

For ecosystem release v1.0.0 ⚠:
PC $> cd $HOME/STM32MPU_workspace/STM32MP15-Ecosystem-v1.0.0/Developer-Package
/stm32mp1-openstlinux-4.19-thud-mp1-19-02-20/sources/arm-openstlinux_weston-linux-
gnueabi/linux-stm32mp-4.19- r0/linux-4.19.9
```

- Edit the *./drivers/gpu/drm/stm/drv.c* source file
- Add a log information in the *stm_drm_platform_probe* function as follow

```
static int stm_drm_platform_probe(struct platform_device *pdev)
{
        struct device *dev = &pdev->dev;
        struct drm_device *ddev;
        int ret;
        [...]

        DRM_INFO("Simple example - %s\n", __func__);

        return 0;
        [...]
}
```

- Save the file
- Rebuitd the Linux® kernel

```
PC $> make uImage LOADADDR=0xC2000040
```

- Update the Linux kernel image into board

```
PC $> scp arch/arm/boot/uImage root@<board ip address>:/boot
```

- Reboot the board

```
Board $> reboot
```

- Check that there is now log information when the display driver is probed

```
Board $> dmesg | grep -i stm_drm_platform_probe
[    2.764080] [drm] Simple example - stm_drm_platform_probe
```

Direct Rendering Manager (kernel module that gives direct hardware access to DRI clients, find more information on official DRI web site http://dri.freedesktop.org/wiki/DRM)