How to Replace the Kernel with DC-Linux on DC-ROMA RISC-V Mainboard for Framework Laptop 13

To assist developers who need to use the DC-Linux kernel for development, we’ve created this guide to help you implement the kernel in your DC-ROMA RISC-V Mainboard for Framework Laptop 13. Below are the steps to replace the default Ubuntu 24.04 kernel (based on RISC-V) with the DC-Linux kernel.

Source code download address

https://github.com/DC-DeepComputing/DC-linux.git

Prerequisites for kernel replacement

1、Linux development environment

- Download address of Ubuntu24.04 based on RISC-V：https://github.com/DC-DeepComputing/fml13v01/releases/tag/V1.0
- Development packages to download：

```
apt install -y libncurses-dev libssl-dev bc flex bison make gcc gcc-riscv64-linux-gnu git

apt install -y qemu-user-static binfmt-support debootstrap debian-ports-archive-keyring systemd-container rsync wget

apt install -y dpkg-dev kmod cpio fakeroot libelf-dev debhelper-compat python3 rsync
```

2、Need to Install libssl-dev:riscv64 for building deb

```
dpkg-buildpackage --build=binary --no-pre-clean --unsigned-changes -R'make -f
debian/rules' -j1 -a$(cat debian/arch)


dpkg-buildpackage: info: source package linux-upstream
dpkg-buildpackage: info: source version 6.13.0-rc1-24


dpkg-buildpackage: info: source distribution unstable
dpkg-buildpackage: info: source changed by root <root@roma-MacBookPro>
dpkg-buildpackage: info: host architecture riscv64
dpkg-source --before-build .
dpkg-checkbuilddeps: error: Unmet build dependencies: libssl-dev
dpkg-buildpackage: warning: build dependencies/conflicts unsatisfied; aborting

Need to install libssl-dev:riscv64
```

3、How to Install libssl-dev:riscv64

```
#ubuntu24.04
# add below into /etc/apt/sources.list.d/ubuntu.sources

Types: deb
URIs: http://ports.ubuntu.com/ubuntu-ports
Suites: noble noble-updates noble-backports
Components: main universe restricted multiverse
Signed-By: /usr/share/keyrings/ubuntu-archive-keyring.gpg
Architectures: riscv64

# exec below command to install

apt update
dpkg --add-architecture riscv64
apt install libssl:riscv64
```

The first way to replace the kernel

1、Build deb

```
export ARCH=riscv
export CROSS_COMPILE=riscv64-linux-gnu-

make defconfig


nice make -j 16 bindeb-pkg
#Files:
arch/riscv/boot/dts/starfive/jh7110-deepcomputing-fml13v01.dtb
../linux-image-6.13.0-rc1_6.13.0-rc1-25_riscv64.deb
```

2、Install deb --- for grub

```
sudo dpkg -i linux-image-6.13.0-rc1_6.13.0-rc1-25_riscv64.deb
sudo cp -r jh7110-deepcomputing-fml13v01.dtb /boot/


sudo vim /boot/grub/grub.cfg

#You can quickly locate line 131 and modify the sections highlighted in red

set linux_gfx_mode=
export linux_gfx_mode
menuentry 'Ubuntu' --class ubuntu --class gnu-linux --class gnu --class os
$menuentry_id_option 'gnulinux-simple-8b676d04-5fab-4c2b-9783-aaaa4e509e51' {
	load_video
	insmod gzio
	if [ x$grub_platform = xxen ]; then insmod xzio; insmod lzopio; fi
	insmod part_gpt
	insmod ext2
	search --no-floppy --fs-uuid --set=root 8b676d04-5fab-4c2b-9783-aaaa4e509e51
	echo	'Loading Linux 6.13.0-rc1 ...'
	linux	/boot/vmlinuz-6.13.0-rc1 root=UUID=8b676d04-5fab-4c2b-9783-
aaaa4e509e51 ro  efi=debug earlycon sysctl.kernel.watchdog_thresh=60
	echo	'Loading initial ramdisk ...'
	initrd	/boot/initrd.img-6.13.0-rc1
	echo	'Loading device tree blob...'
	devicetree	/boot/jh7110-deepcomputing-fml13v01.dtb
}
```

```
submenu 'Advanced options for Ubuntu' $menuentry_id_option 'gnulinux-advanced-8b676d04-5fab-4c2b-9783-aaaa4e509e51' {
	menuentry 'Ubuntu, with Linux 6.13.0-rc1' --class ubuntu --class gnu-linux --class gnu --class os $menuentry_id_option 'gnulinux-6.13.0-rc1-advanced-8b676d04-5fab-4c2b-9783-aaaa4e509e51' {
		load_video
		insmod gzio
		if [ x$grub_platform = xxen ]; then insmod xzio; insmod lzopio; fi
		insmod part_gpt
		insmod ext2


search --no-floppy --fs-uuid --set=root 8b676d04-5fab-4c2b-9783-aaaa4e509e51
		echo	'Loading Linux 6.13.0-rc1 ...'
		linux	/boot/vmlinuz-6.13.0-rc1 root=UUID=8b676d04-5fab-4c2b-9783-aaaa4e509e51 ro  efi=debug earlycon sysctl.kernel.watchdog_thresh=60
		echo	'Loading initial ramdisk ...'
		initrd	 /boot/initrd.img-6.13.0-rc1
		echo	'Loading device tree blob...'
		devicetree	/boot/jh7110-deepcomputing-fml13v01.dtb
	}
	menuentry 'Ubuntu, with Linux 6.13.0-rc1 (recovery mode)' --class ubuntu --class gnu-linux --class gnu --class os $menuentry_id_option 'gnulinux-6.13.0-rc1-recovery-8b676d04-5fab-4c2b-9783-aaaa4e509e51' {
		load_video
		insmod gzio
		if [ x$grub_platform = xxen ]; then insmod xzio; insmod lzopio; fi
		insmod part_gpt
		insmod ext2
		search --no-floppy --fs-uuid --set=root 8b676d04-5fab-4c2b-9783-aaaa4e509e51
		echo	'Loading Linux 6.13.0-rc1 ...'
		linux	/boot/vmlinuz-6.13.0-rc1 root=UUID=8b676d04-5fab-4c2b-9783-aaaa4e509e51 ro recovery nomodeset
		echo	'Loading initial ramdisk ...'
		initrd	 /boot/initrd.img-6.13.0-rc1
		echo	'Loading device tree blob...'
		devicetree	/boot/jh7110-deepcomputing-fml13v01.dtb
	}
''''''''''
```

The second way to replace the kernel

1、Build Image

```
export ARCH=riscv
export CROSS_COMPILE=riscv64-linux-gnu-

make defconfig
make



export INSTALL_PATH=output
export INSTALL_MOD_PATH=output

make install
make modules_install
```

2、Replace kernel

Kernel partition -- the 3rd partition.

Rootfs partition -- the 4th partition.

On fm7110, it supports loading two type kernel images.

- Image.fit , it's used to load buildroot mini rootfs which is built by SDK directly.

```
jh7110_uEnv.txt  starfiveu.fit  vf2_nvme_uEnv.txt  vf2_uEnv.txt
```

Generate image.fit: It includes dtb, Image, initramfs.

visionfive2-fit-image.its

```
mkimage -f visionfive2-fit-image.its -A riscv -O linux -T flat_dt starfiveu.fit
```

- vmlinuz, it's used to load debian / ubuntu rootfs.

```
config-5.15.0   dtbs   extlinux   initrd.img-5.15.0   System.map-5.15.0  'System Volume
Information'   vf2_uEnv.txt   vmlinuz-5.15.0
```

```
#Dtb:
cp arch/riscv/boot/dts/starfive/jh7110-framework.dtb to dtbs/starfive/



#Kernel:
cp arch/riscv/boot/Image.gz to vmlinuz-*



#Initramfs:
initrd.img-5.15.0

#Moudles:
Rootfs partition:  lib/modules

#Firmware:
Initramfs: lib/firmware

Rootfs partition:  lib/firmware/
```